ITC Dissertation Number 79
The research presented in this thesis was financially supported by the International Institute of Aerospace Survey and Earth Sciences (ITC), P. O. Box 6, 7500 AA Enschede, The Netherlands.

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Printed by FEBODRUK BV, Enschede.
Cover printed by Taib El Ghazi.

Copyright © 2000 by Ale Raza
ISBN 90-3651-540-8
Object-Oriented Temporal GIS for Urban Applications
PhD thesis University of Twente and ITC,
with reference abstract in Dutch.
OBJECT-ORIENTED TEMPORAL GIS FOR URBAN APPLICATIONS

DISSERTATION

to obtain the doctor’s degree at the University of Twente, on the authority of the rector magnificus, Prof. Dr. F. A. van Vught, on account of the decision of the graduation committee, to be publicly defended on Thursday 18th January 2001 at 13:15 hrs.

by

Ale Raza
Born on 5th January 1965
In Karachi, Pakistan.
This thesis is approved by

Professor Dr. Wolfgang Kainz, Promotor.
Geoinformatics, Spatial Information Theory and Applied Computer Science
International Institute of Aerospace Survey and Earth Sciences (ITC)
P.O. Box 6, 7500 AA Enschede, The Netherlands.
If the ocean were ink (wherewith to write out) the words of my Load, sooner would the ocean be exhausted than would the words of my Load, even if we added another ocean like it, for its aid.

To my parents, brothers and sisters for their love and gratitude.
Abstract


Where to get data for urban planning and management is one issue, but how then to manage this data is still a challenging task. This research focuses mainly on the latter (technical issue). The key data needed for urban planning and management is spatio-temporal (ST) parcel and land use data, which if updated regularly, raises the issue of structuring, storing, querying and displaying this data. Modeling, querying and displaying changes in land use and associated data has been a challenging task for many designers and urban planners. A number of issues have been identified by many researchers in the development of a TGIS.

One of the fundamental issues in designing a generic TGIS is the spatio-temporal data model. Integrating time in object-based data models increases the complexity of the data structure. This thesis mainly addresses two issues, i.e., the development of an object-based spatio-temporal data model and the modeling of parcel and urban land use changes. To solve these issues, a novel approach has been adopted and a simple data structure is presented. This approach is based on object-oriented (OO) concepts and the mathematical theory of cell complexes in order to distil the complexity. The mathematical concept of cell complexes has been extended to temporal cell complexes in order to provide a sound basis for defining space and time. This may provide an unambiguous definition of the data model for implementation and a basis for further development of a spatio-temporal query language. This temporal cell complex (TCC) is the extension of ST-simplicial complexes. The cell tuple structure has been extended to temporal cell tuple (TCT) structure. This TCC is called ‘a unified cell tuple-based spatio-temporal data model’ (CTSTDM). The CTSTDM preserves the topology of spatio-temporal objects in an implicit manner. This approach has an advantage as the topology of an object may change without any change in the object itself.

The data model consists of a data structure, operators and consistency rules. The conceptual schema (data structure) has been devised in UML (unified modeling language) by aggregating the three components of reality, i.e., space, time and attribute (each is considered as a class). Two types of operators have been contemplated for the CTSTDM, i.e., dynamic and static operators. Dynamic operators change the system status, while the static are query operators and do not change the system status. Four dynamic operators (Create, Kill, Destroy and Reincarnate) governing the various classes of the CTSTDM have been defined. Topological notions of the point-set approach have been employed to analyze the various intersections of objects of these classes. It has been demonstrated how these operators can be constructed in a consistent fashion. The static operators associated with TCT class have been formulated. It has been shown that almost all spatio-temporal relations can be extracted from a TCT structure. The spatio-temporal consistency rules have been defined at conceptual and logical levels. To fully utilize the power and functionalities of RDBMS and SQL, an object-relational approach has been employed. The CTSTDM has been realized in Oracle. A three-tier (layer) client / server system architecture has been devised for cell tuple-based OO TGIS. The first layer provides the OO graphical user interface (GUI), the second layer consists of the GUI controller and the third is the database layer.

Two types of land use have been investigated, i.e., general and detailed (parcel-level) land use change. Changes are modeled by considering each as an object. By developing a prototype for time series data, it has been demonstrated that the CTSTDM can be implemented and can be extended to other applications. Using SQL, many queries are formulated, to demonstrate that not only can the history of an object be scanned but the state, change and topology of the object can also be determined. Parcel-level land use data is indispensable for many urban planning exercises. However, sharing this data is one of the main problems in many planning agencies. The client / server approach adopted here may alleviate the data accessibility problem in the urban environment and may facilitate
the designing of an integrated urban information systems - and thus can improve the coordination among these agencies. The approach adopted in this research for the CTSTDM may pave the way for designing and implementing a generic TGIS.

**Keywords:** Object-orientation, temporal GIS, CTSTDM, temporal cell complex, temporal cell tuple, active and in-active objects, static and dynamic operators, essential and non-essential changes, and client / server architecture.
Samenvatting


Naast het verkrijgen van gegevens voor stedelijke planning en stadsbeheer ligt een tweede probleem in het beheren en manipuleren van deze gegevens. Dit onderzoek richt zich vooral op de technische kant van het laatste. De sleutelgegevens voor stedelijke planning en stadsbeheer zijn spatio-temporele (ST) perceelsgegevens en grondgebruikgegevens. Omdat deze gegevens regelmatig veranderen moet er aandacht worden besteed aan het structureren, opslaan, raadplegen en weergeven ervan. Modelleren, bevragen en weergeven van veranderingen in grondgebruikgegevens en gerelateerde gegevens is voor veel ontwerpers en planologen een moeilijke taak. Bij de ontwikkeling van temporele GIS (TGIS) is door menig onderzoeker een aantal aandachtspunten naar voren gebracht.

Een van de fundamentele kwesties bij het ontwerp van een generiek TGIS is het spatio-temporele gegevensmodel. Het integreren van tijd in object-georiënteerde gegevensmodellen verhoogt de complexiteit van de gegevensstructuur. Dit proefschrift behandelt voornamelijk twee vraagstukken, te weten de ontwikkeling van een object-gebaseerd spatio-temporele gegevensmodel en de modellering van veranderingen van perceelen en grondgebruik. Om deze vraagstukken op te lossen is een nieuwe benadering toegepast en een simpele gegevensstructuur gepresenteerd. Deze benadering is gebaseerd op object-georiënteerde (OO) concepten en de mathematische theorie van cell-complexes met als doel het probleem te vereenvoudigen.

Het mathematisch concept van cell-complexes is uitgebreid tot temporele cell-complexes om zodoende een deugdelijke basis te vormen voor het definieren van ruimte en tijd. Dit kan resulteren in een ondubbelzinnige definitie van het gegevensmodel voor implementatie en een basis voor verdere ontwikkeling van een spatio-temporele ondervragingstaal.

Deze temporele cell-complex (TCC) is de uitbreiding van ST-simplicial complexes. De cell-tuple structuur is uitgebreid tot een temporele cell-tuple (TCT) structuur. Deze TCC wordt aangeduid als ‘een verenigd cell-tuple gebaseerd spatio-temporele gegevensmodel’ (CTSTDM). Het CTSTDM behoudt de topologie van de spatio-temporele objecten op een impliciete manier. Deze benadering heeft als voordeel dat de topologie van een object kan veranderen zonder dat het object zelf verandert.

Het gegevensmodel bestaat uit een gegevensstructuur, operatoren en consistentieregels. De conceptuele gegevensstructuur is ontworpen in UML (Unified Modelling Language) door het samenvoegen van de componenten van de werkelijkheid, zijnde ruimte, tijd en attribuut (alle worden behandeld als een class). Twee types operatoren zijn beschouwd voor het CTSTDM, namelijk dynamische en statische operatoren. Dynamische operatoren veranderen de systeemstatus, terwijl de statische operatoren bevragingsoperatoren zijn en de systeemstatus niet veranderen. Vier dynamische operatoren (Create, Kill, Destroy en Reincarnate) die bepalend zijn voor de diverse classes in het CTSTDM, zijn gedefinieerd. Topologische concepten van de point-set benadering zijn toegepast om de verschillende intersecties van objecten van deze classes te analyseren. Aangetoond is hoe deze operatoren op een consistente manier geconstrueerd kunnen worden. De statische operatoren gerelateerd aan de TCT-class zijn gedefinieerd. Het is aangetoond dat bijna alle spatio-temporele relaties kunnen worden afgeleid van een TCT structuur. De spatio-temporele consistentieregels zijn gedefinieerd op conceptuele en logische niveaus. Een object-relatieloop reeks is toegepast om te komen tot het volledig gebruik van de kracht en functionaliteit van RDBMS en SQL. Het CTSTDM is gerealiseerd in Oracle. Een three-tier (drie-laags) client/server systeemarchitectuur is ontworpen voor cell-tuple-gebaseerd OO TGIS. De eerste laag vertegenwoordigt de OO-grafische gebruikersinterface (GUI), de tweede laag omvat de GUI bestuurder en de derde is de database laag.

Twee soorten grondgebruik zijn onderzocht; te weten algemeen en gedetailleerde verandering van grondgebruik (de laatste op perceelsniveau). Veranderingen zijn gomodelleerd door elk te beschouwen als objecten. Door het ontwikkelen van een prototype voor tijdkrommegegevens is aangetoond dat het CTSTDM kan worden geïmplementeerd en uitgebreid tot andere applicaties. Met het gebruik van SQL zijn vele queries geformuleerd om zo te demonstreren dat niet alleen de
geschiedenis van een object kan worden vastgesteld maar dat ook de staat, verandering en topologie van het object kan worden bepaald. Grondgebruikgegevens op perceelsniveau zijn onmisbaar voor vele stedelijke-planologische exercities. Desalniettemin is het uitwisselen van deze gegevens een van de grootste problemen van vele planologische instanties. De client/server benadering die hier is toegepast kan het probleem van gegevenstoegang in de stedelijke omgeving verlichten en maakt het ontwerp van een geïntegreerd stedelijk informatiesysteem mogelijk. Dit kan de coördinatie tussen deze organisaties verbeteren. De methode die is gevolgd in dit onderzoek voor het CTSTDM creëert de mogelijkheid voor het ontwerp en implementatie van een generiek TGIS.

Acknowledgments

This research would not have been possible without the help, encouragement and appreciation of many individuals and organizations. Naming all is not possible. PhD research is a life-time experience and I would not say that it is an easy task. A PhD research solves one scientific problem but opens many others for future research. This is the result of hard work to meet deadlines, and often restless nights, uncertainties, ambitions etc. To attain a successful research and well balanced life at ITC, one must keep a balance between work and social life. This is not an easy task and priorities must be set for each. At this stage, I am glad to say that I was able to keep this balance. Apart from being involved in the research, I was one of the PhD representatives. Together with other representatives, we solved a number of problems faced by many PhD students at ITC and, at the same time, I was active in other social activities as well.

I would like to express my sincere gratitude to my supervisor, Professor Dr. Wolfgang Kainz (Division of Spatial Information Theory and Applied Computer Sciences, ITC). This research would not be in its current shape without his encouragement and continuous support. He has provided me with many insights into geo-information theories, particularly the mathematical concepts of topological approaches. We had a number of stimulating scientific discussions. I am very much impressed by his scientific knowledge and personal dealings. Sometimes, I could go to him without any appointment to discuss the problems. I am indebted to Professor Dr. Ian Masser (Division of Urban Planning and Management, ITC). I had the honor to work with him from the middle of my research, when he joined ITC. His support and guidance were extremely useful and instrumental to my fieldwork in Karachi. It enabled me to better prepare myself for interviewing many urban planners in a relatively short span of time. I deeply acknowledge the interest of Dr. Rolf de By in my research. His critical comments throughout my research, particularly on the database design approach have been greatly valued. From the beginning of this research, I occasionally had discussions with Professor Dr. Werner Kuhn, Institute of Geoinformatics (IfGI), University of Munster, Germany, on my research directions. I appreciate his concern regarding the direction and status of my research, which helped me to narrow down my research focus.

This acknowledgment would be incomplete without acknowledging Research Coordinator Dr. Liesbeth Kosters. Her continuous support helped me to keep concentrating on my research without bothering about the finances. Similarly, the administrative support of Loes Colenbrander is highly appreciated. I had a number of discussions with Richard Sluizas on various aspects of urban planning, particularly with respect to less developed countries. I deeply acknowledge his expert opinions, advice and comments on these issues. I heartily appreciate the discussions, meetings and support from ITC colleagues and staff: Klaas Jan Beek, F. van de Bosch, Mark Brussel, Theo Boulooucos, Martin Ellis, Corné van Elzakker, Karl Grabmaier, Paul Hofstee, John Horn, Hilde Hulsman, Menno J. Kraak, Wim Koolhoven, Rob Lemmens, Martien Molenaar, Sherif Amer, Jan Turkstra, Arbid Tuladhar, Yuxian Sun, and François de Sousa. I appreciate the day-to-day help and assistance of Ard Blenke in keeping the hardware and software operational, especially his support in setting up the Oracle server and making regular backups of my data. I am thankful to Marga Koelen and Carla Gerritsen for their support from the library and to the secretaries of the Geoinformation Division, Saskia Tempelman, Donny Scholten and Erica Weijer. I am indebted to Ms. Janice Collins for editing this thesis.

I am thankful to many PhD colleagues and friends for their company, help and discussions on scientific and social issues: Maryory Arango, Dr. Ali Akbar, Narcissso Barrera, Dr. Maria Bastidas, Citlalli L. Binnquist, Dr. Yaser Bishr, Dr. Tao Cheng, Dr. Rudiger Genz, Dr. Wilson Khaemba, Yao lin Liu, Xue Hua Liu, Saadi Mesgari, Lorena Montoya, Dr. Wenning Peng, Patrick Ogao, Dr. Wilber Ottichilo, Qingming Zhan, Ding Zheng and Dr. Siyka Zlatanova.

I am indebted to the ITC Directorate for various meetings to discuss a number of problems associated
with PhD students at ITC, to the former Rector and Vice Rector, Prof. Dr. Karl Harmsen and Dr. Niek Rengers, respectively, and to the head of Student Affairs, Fred Paats. My thanks to other staff, Riet Allessie, Teresa Brefeld, Andre Klijnstra and Jan de Ruiter, for their support and assistance. Many thanks to the Dutch Government and ITC for providing me with a DGIS fellowship to achieve this highest academic degree.

I have been fortunate in having a number of friends in Enschede, who have contributed to my balanced life by providing a home-like environment. I am thankful to Letteke van Nispan and her family. She cared for me like her son. She was always there in my good and bad times. I owe a lot to Dr. Anupma Prakash, Anneke Homan and Wilma Grootenboer for their love, moral support and encouragement.

Many people assisted me during my fieldwork in Karachi. The help and assistance from Shahab A. Alvi, Aurangzeb Haque, Fazal K. Khatri, Iftikhar A. Kaimkhani, M. Qayoom and the field staff of Master Plan Department was indispensible. Without their cooperation, I might not have been able to collect the information within the limited time.

I owe a great deal to my friends Nasreen Islam Khan and Imtiaz Ahmed for their support and care. Although very far from here, they have been in contact with me during my studies. Imtiaz regularly sent emails to me and gave me a lot of support through encouragement and appreciation.

I am indebted to my Ammi (mother), Papa, brother and sisters. I do not have words for their love, support, encouragement, and patience to wait for me for six years. This research would not have been possible without their prayers and good wishes. They supported me by all possible means. At the final stage of my thesis, I appreciate the support and love of my wife Dr. Farwa. My thanks to all!

Ale Raza
January, 2001
# Table of Contents

Abstract .................................................................................................................................................. i
Samenvatting ........................................................................................................................................ iii
Acknowledgments ...................................................................................................................................... v
Table of Contents ..................................................................................................................................... vii
List of Figures .......................................................................................................................................... xii
List of Tables .......................................................................................................................................... xvii

Chapter-1 ............................................................................................................................................... 1
Introduction and Framework .................................................................................................................... 1
1.1 MOTIVATION .................................................................................................................................... 1
1.2 TEMPORAL GIS PROBLEM AREAS .................................................................................................. 3
   1.2.1 Application problem .................................................................................................................. 4
   1.2.2 Technological problem .............................................................................................................. 5
1.3 SCOPE OF THE RESEARCH .......................................................................................................... 5
1.4 OBJECTIVES OF THE STUDY ....................................................................................................... 7
1.5 RESEARCH APPROACH AND STRUCTURE OF THE THESIS .................................................... 7
   1.5.1 Part-I: Introduction and fundamental concepts ......................................................................... 7
   1.5.2 Part-II: System development ................................................................................................. 8
   1.5.3 Part-III: Conclusions and recommendations ........................................................................... 9

Chapter-2 ............................................................................................................................................... 11
Concepts of Space and Time ................................................................................................................... 11
2.1 INTRODUCTION ............................................................................................................................. 11
2.2 ABSTRACTION PROCESS OF REALITY ....................................................................................... 11
2.3 CONCEPTUAL ISSUES ................................................................................................................... 14
   2.3.1 Space ....................................................................................................................................... 14
   2.3.2 Spaces and manifolds .......................................................................................................... 15
   2.3.3 Functions ............................................................................................................................ 15
   2.3.4 Time ...................................................................................................................................... 16
   2.3.5 Space and time .................................................................................................................... 17
      2.3.5.1 Similarities .................................................................................................................... 17
      2.3.5.2 Differences .................................................................................................................. 18
   2.3.6 State, event and evidence ........................................................................................................ 18
   2.3.7 What is a change? .................................................................................................................. 19
2.4 COMPONENTS OF SPATIO-TEMPORAL MODELING AND MATHEMATICAL FOUNDATION .... 20
   2.4.1 Spatial component ............................................................................................................... 20
      2.4.1.1 State of spatial object .................................................................................................... 21
      2.4.1.2 Representation and relationships of spatial objects ..................................................... 22
   2.4.2 Spatial operations .................................................................................................................. 24
      2.4.2.1 Spatial metric relations ............................................................................................... 25
      2.4.2.2 Spatial topological relations ....................................................................................... 26
   2.4.3 Temporal component ............................................................................................................. 35
3.4 DESIGNING THE SPACE-TIME MODEL ..................................................................................... 52
3.3 SPATIO-TEMPORAL DATA MODELING .................................................................................... 46
3.1 INTRODUCTION.............................................................................................................................. 43

Chapter-3 ........................................................................................................................................ 43

Spatio-Temporal Data Modeling: Conceptual Schema ...................................................................... 43

2.4.5 Aspatial component ................................................................................................................ 42
2.4.5.1 Aspatial object and operations ......................................................................................... 42
2.4.4 Temporal operations .............................................................................................................. 38
2.4.3.8 Dimensions of the time .................................................................................................... 36
2.4.3.7 Calendars ......................................................................................................................... 37
2.4.3.6 Chronon and time granularity ......................................................................................... 37
2.4.3.5 Type of time .................................................................................................................... 37
2.4.3.4 Representation of time .................................................................................................... 36
2.4.3.3 Continuous time ............................................................................................................. 36
2.4.3.2 Discrete time .................................................................................................................... 36
2.4.3.1 Structural perspective ..................................................................................................... 35
2.4.2 Integration perspective ......................................................................................................... 54
2.4.1 Time modeling .................................................................................................................... 53
2.4.1.1 Process modeling ........................................................................................................... 53
2.4.1.2 Time modeling .............................................................................................................. 53
2.4.1 Modeling perspective .......................................................................................................... 53
2.4.3 Design issues ....................................................................................................................... 54
2.4.4 UML presentation, notation and organization ...................................................................... 56

3.5 SPATIO-TEMPORAL DATA MODEL: CONCEPTUAL SCHEMA .............................................. 57
3.5.1 TemporalClass ...................................................................................................................... 60
3.5.2 SpatialClass ........................................................................................................................... 62
3.5.2.1 PointClass .................................................................................................................... 62
3.5.3 TemporalCellTupleClass ..................................................................................................... 70
3.5.3.1 ZeroTCellClass .............................................................................................................. 65
3.5.3.2 OneTCellClass .............................................................................................................. 65
3.5.3.3 TwoTCellClass .............................................................................................................. 66
3.5.3.4 Object and versions ....................................................................................................... 66
3.5.3.5 Spatio-temporal consistency rules .................................................................................. 66
3.5.1.1 Hierarchical data structure for US state boundaries ......................................................... 46
3.5.1.2 Snapshot approach ........................................................................................................ 47
3.5.1.3 Base state with overlay approach .................................................................................. 47
3.5.1.4 Space-time composite approach .................................................................................... 47
3.5.1.5 Space-time cube approach ............................................................................................ 48
3.5.1.6 Event-based approach .................................................................................................. 48
3.5.1.7 Three-domain model .................................................................................................... 49
3.5.1.8 4D GIS .......................................................................................................................... 50
3.5.1.9 Object-oriented approach .............................................................................................. 50
3.5.1.10 Summary ...................................................................................................................... 52
3.3.1 SPATIAL DATA MODELING..................................................................................................... 43
3.3.2.2 Time as an attribute ...................................................................................................... 54
3.3.2.1 Time as a separate dimension ......................................................................................... 54
3.3.2 Integration perspective ........................................................................................................ 54
3.3.1 Review of spatial / topological data models .......................................................................... 44
3.3.1.1 Summary ....................................................................................................................... 45
3.3.1.10 Summary ...................................................................................................................... 52

2.4.2.1 Time as an attribute ........................................................................................................ 54
2.4.2.2 Time as a separate dimension ......................................................................................... 54
2.4.2 Time modeling ..................................................................................................................... 54
2.4.1 Modeling perspective .......................................................................................................... 53
2.3.5.4 TemporalCellTupleClass................................................................................................... 70
7.4.3 Algorithm: direction of OTC and centroid of TTC ................................................................. 179
7.5 PROTOTYPE................................................................................................................................. 180
  7.5.1 Data transfer...................................................................................................................... 180
  7.5.2 Spatio-temporal data: land use and parcel records ......................................................... 180
  7.5.3 Updating land use objects................................................................................................. 180
  7.5.4 Updating parcel objects................................................................................................. 182
  7.5.5 Accessing spatio-temporal objects .................................................................................. 185
    7.5.5.1 Scanning history of objects ....................................................................................... 185
    7.5.5.2 Simple temporal query .............................................................................................. 187
    7.5.5.3 Temporal range query ............................................................................................... 188
    7.5.5.4 Spatio-temporal topology ......................................................................................... 193
7.6 PERFORMANCE ISSUES OF CTSTDM ................................................................................. 195
7.7 DISCUSSION AND CONCLUSIONS ...................................................................................... 197

Chapter-8 ........................................................................................................................................ 201

Conclusions and Recommendations............................................................................................ 201
  8.1 INTRODUCTION.................................................................................................................... 201
  8.2 SUMMARY ............................................................................................................................ 201
  8.3 CONCLUSIONS ..................................................................................................................... 203
  8.4 RECOMMENDATIONS .......................................................................................................... 206

Bibliography ................................................................................................................................. 209
  Appendix-1 ................................................................................................................................ 217
  Urban planning and service agencies in Karachi ...................................................................... 217
  Appendix-2 ................................................................................................................................ 218
  Spatio-temporal data in planning agencies .............................................................................. 218
  Appendix-3 ................................................................................................................................ 220
  Definition of relations (tables) in Oracle using DDL ............................................................... 220
  Appendix-4 ................................................................................................................................ 223
  Some illustration of implementation in VC++ ......................................................................... 223
  Appendix-5 ................................................................................................................................ 227
  Algorithms for Point and ZeroTCell objects ............................................................................ 227
  Appendix-6 ................................................................................................................................ 231
  Implementation of algorithm for OTC-OTC intersection in VC++ ....................................... 231
  Appendix-7 ................................................................................................................................ 236
  AML routines for overlay process ............................................................................................ 236
  Appendix-8 ................................................................................................................................ 237
  Land use maps .......................................................................................................................... 237
  Appendix-9 ................................................................................................................................ 240
  Land use codes .......................................................................................................................... 240
  Completed PhD studies at ITC ................................................................................................. 241
  Curriculum Vitae ...................................................................................................................... 245
List of Figures

Figure 1.1 A spatio-temporal map of the bus trip (adapted from Wood and Fels, 1986) .......................................................... 2
Figure 1.2 Land use change from T1 to T3. ................................................................. 4

Figure 2.1 Modeling reality (adapted from Jackson, 1983). ........................................... 12
Figure 2.2 Four levels of abstractions (adapted from Peuquet, 1984) .................................. 12
Figure 2.3 System design and abstraction process for spatio-temporal data modeling. .......... 13
Figure 2.4 Relation between space and time: (A) impossible configuration (B) possible configuration. ............................................. 17
Figure 2.5 Essential and non-essential change model. ..................................................... 19
Figure 2.6 Extreme level of abstraction for reality: spatio-temporal attribute object (STAO). .............................................................. 20
Figure 2.7 Description of spatial component: ................................................................. 21
Figure 2.8 Configuration of simplex and simplicial complex. ........................................... 22
Figure 2.9 Cell and cell complex configuration. ............................................................. 23
Figure 2.10 Boundary and co-boundary relations in a cell complex. ................................. 24
Figure 2.11 Spatial operations. ............................................................... 25
Figure 2.12 Examples of transformation: (A) intersections, (B) incidences, (C) adjacencies and (D) inclusions. .............................. 27
Figure 2.13 Two branches of topology. ................................................................. 28
Figure 2.14 Three topological primitives of a spatial object. ........................................ 28
Figure 2.15 Spatial topological relations between two objects in $\mathbb{R}^2$. ......................... 29
Figure 2.16 Examples of graph. ................................................................. 30
Figure 2.17 Types of graph. ................................................................. 31
Figure 2.18 Spatial order. ................................................................. 32
Figure 2.19 Spatial order: ................................................................. 33
Figure 2.20 Example of poset: ................................................................. 34
Figure 2.21 (A) A poset which is not a lattice. (B) A complete lattice after normal completion. .......................................................... 34
Figure 2.22 Description of temporal component. .......................................................... 35
Figure 2.23 Three time models (adapted from Worboys, 1995). .................................... 35
Figure 2.24 Relationships between world and database time. ........................................ 38
Figure 2.25 Temporal existence of a spatio-temporal object in single time dimension. ........ 39
Figure 2.26 Spatial and temporal equivalent of a STAO. ................................................. 39
Figure 2.27 Reincarnation of an Object O1, example of land use change. ......................... 40
Figure 2.28 Event-event and event-state relationships. ................................................. 40
Figure 2.29 Bounded and unbounded temporal intervals. .............................................. 41
Figure 2.30 Thirteen possible temporal relations for bounded interval (adapted from Allen, 1984) ......................................................... 41
Figure 2.31 Description of aspatial component. ............................................................ 42

Figure 3.1 Categories of spatial data models. ............................................................ 43
Figure 3.2 Evolution of urban land use change from 1980 to 1995. ................................. 47
Figure 3.3 Sequence snapshot of Figure 3.2 ............................................................... 47
Figure 3.4 Spatial-time composite technique. .............................................................. 48
Figure 3.5 Illustration of space-time cube technique. ..................................................... 48
Figure 3.6 The TRIAD model (adapted from Peuquet and Qian, 1996) ......................... 49
Figure 3.7 Three-domain model (adapted from Yuan, 1994) .......................................... 50
Figure 3.8 Change in administrative boundaries. ....................................................... 50
Figure 3.9 Decomposition of object into atoms. .......................................................... 51
FIGURE 7.17  POINT IN TTC TEST. ILLUSTRATION OF HALF-LINE THEOREM.......................................................... 179
FIGURE 7.18  TRANSLATOR TO READ DATA FROM ARC/INFO TO CTSTDM........................................................... 180
FIGURE 7.19  THE PARCEL LAND USE FORM........................................................................................................ 181
FIGURE 7.20  UML ACTIVITY DIAGRAM: PROCESS OF UPDATING LAND USE OBJECTS...................................... 182
FIGURE 7.21  THE PARCEL FORM........................................................................................................................ 183
FIGURE 7.22  UML ACTIVITY DIAGRAM: PROCESS OF UPDATING PARCEL OBJECTS........................................ 184
FIGURE 7.23  TIME POINT BETWEEN TIME INTERVAL......................................................................................... 187
FIGURE 7.24  RESIDENTIAL IN 1990 .................................................................................................................. 189
FIGURE 7.25  RESIDENTIAL IN 1995 .................................................................................................................. 189
FIGURE 7.26  RESIDENTIAL IN 1999.................................................................................................................... 190
FIGURE 7.27  CHANGE TO RESIDENTIAL IN 1998 ............................................................................................ 190
FIGURE 7.28  INTERVAL-INTERVAL RELATIONS................................................................................................. 191
FIGURE 7.29  RESIDENTIAL LAND USE FROM 1990 TO 1992 ......................................................................... 192
FIGURE 7.30  RESIDENTIAL LAND USE FROM 1994 TILL NOW ........................................................................ 193
FIGURE 7.31  PARCELS CHANGE PER YEAR ...................................................................................................... 195
FIGURE 7.32  NUMBER (CUMULATIVE) OF ST-OBJECTS BY SYSTEM TIME................................................... 196
FIGURE 7.33  COMPARATIVE CHANGES (%) IN ST-OBJECTS DURING FEB-MAR.............................................. 197
FIGURE 7.34  COMPARATIVE CHANGES (%) IN ST-OBJECTS DURING MAR-APR.......................................... 197
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Equivalence of space and time (adapted from Langran, 1992a)</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>State, event and evidence (adapted from Langran, 1992b)</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Attribute description of state, event and evidence</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Types of database (adapted from Snodgrass, 1992)</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>Temporal cell tuple configuration at T1</td>
<td>72</td>
</tr>
<tr>
<td>3.2</td>
<td>Temporal cell tuple configuration at T2</td>
<td>73</td>
</tr>
<tr>
<td>3.3</td>
<td>Land use classification</td>
<td>74</td>
</tr>
<tr>
<td>4.1</td>
<td>OTC’s life span and parents</td>
<td>86</td>
</tr>
<tr>
<td>4.2</td>
<td>Spatio-temporal relations for CTSTDM</td>
<td>110</td>
</tr>
<tr>
<td>5.1</td>
<td>Users and owners of urban spatio-temporal data</td>
<td>120</td>
</tr>
<tr>
<td>5.2</td>
<td>Use and update frequency of urban spatio-temporal data</td>
<td>120</td>
</tr>
<tr>
<td>5.3</td>
<td>Data sources and resolution (adapted from Rhind and Hudson, 1980)</td>
<td>125</td>
</tr>
<tr>
<td>5.4</td>
<td>Temporal characteristics of data (adapted from Rhind and Hudson, 1980)</td>
<td>125</td>
</tr>
<tr>
<td>5.5</td>
<td>Uses and users of urban land use data (adapted from Rhind and Hudson, 1980)</td>
<td>126</td>
</tr>
<tr>
<td>6.1</td>
<td>Relational versus OO databases: some advantages and disadvantages</td>
<td>148</td>
</tr>
<tr>
<td>7.1</td>
<td>History of parcel D-139</td>
<td>185</td>
</tr>
<tr>
<td>7.2</td>
<td>List of parcel D-139 with spatial information</td>
<td>186</td>
</tr>
<tr>
<td>7.3</td>
<td>Previous land use of parcel C-38</td>
<td>186</td>
</tr>
<tr>
<td>7.4</td>
<td>Land use history of parcel ST-O</td>
<td>187</td>
</tr>
<tr>
<td>7.5</td>
<td>Residential land use in 1990</td>
<td>188</td>
</tr>
<tr>
<td>7.6</td>
<td>Residential land use in 1995</td>
<td>188</td>
</tr>
<tr>
<td>7.7</td>
<td>Land use change to residential in 1998</td>
<td>188</td>
</tr>
<tr>
<td>7.8</td>
<td>Residential land use from 1990 to 1992</td>
<td>192</td>
</tr>
<tr>
<td>7.9</td>
<td>Residential land use from 1994 till now</td>
<td>192</td>
</tr>
<tr>
<td>7.10</td>
<td>Spatio-temporal topology of parcel F-29</td>
<td>194</td>
</tr>
<tr>
<td>7.11</td>
<td>Illustration of bounded time-interval</td>
<td>195</td>
</tr>
<tr>
<td>7.12</td>
<td>Number of parcels change per year</td>
<td>195</td>
</tr>
<tr>
<td>7.13</td>
<td>Number of ST-objects by system time</td>
<td>196</td>
</tr>
</tbody>
</table>
Chapter-1
Introduction and Framework

1.1 MOTIVATION

Cities or urban areas are the symbol of modernization. Urban activities are the wheel of a country’s economy. The city is a magnet; it is a home and recreational center for city residents; it is a market place and shopping center for rural dwellers; it offers work and job opportunities to the unemployed and criminals; it is a source of souvenirs for tourists, visitors and foreigners. If it is a capital, major national policies are made and decisions are taken there; it is the home of major financial and economic institutes; it is a transportation hub etc. On the other hand cities are the source of complex problems. They need special attention from the whole fabric of society, particularly urban planners and decision-makers, who are responsible for the planning and management of the cities.

Urban areas cover a relatively small part of the earth’s surface. Recent estimates suggest that urban areas containing 43% of the world’s population occupy only 1% of the total land surface (Miller, 1988, as cited by Douglas, 1994). It was forecast that globally 24 million ha of cropland would be transformed into urban-industrial land use by the year 2000. This is only about 2% of the world’s total land. The conversion of land to urban use is most severe in the less developed countries, some 17 to 25 million ha of urban land by the year 2000 (Douglas, 1994). In 1970, the world’s urban population was 1.4 billion. It is expected to reach 2.9 billion by the year 2000 and 5.1 billion by the year 2025. The share of urban population living in the less developed countries increased from 49% in 1979 to 58% in 1985 and is expected to increase by 70% by the year 2025 (Shabbir, 1993).

This rapid urbanization has created tremendous pressure on urban services and created major socio-economic and physical problems. These issues have been challenging urban planners. There are two approaches to deal with this challenge (Shabbir, 1993): first, by controlling the population pressure by reducing the urban-rural population and encouraging a lower birth rate; second, by improving urban management. The first alternative has been found extremely difficult (Shabbir, 1993). Urban management consists of two planning processes, i.e., strategic and operational planning processes. Strategic planning pertains to long- or medium-term planning, such as development plans, while operational planning pertains to the daily activities of the urban planning agencies. To cope with this ever-increasing population pressure on urban areas and associated dynamic and complex aspects of urban issues, the role of urban planning (strategic) is becoming increasingly prominent. Strategic planning can be strengthened through operational planning. Urban planning is a continuous and cyclic process in which the formulation of initial goals is followed by the monitoring of change and the redefinition of goals and strategies. To review the information about change, continuous monitoring of relevant data sets, e.g., land use data, is vital. Monitoring trends demands time series data. Land use patterns are the setting for the activity. As creators of new activity settings and conservers of existing settings, land use planners need accurate information on activity status and trends (Kaiser et al., 1995). If the land is to be managed efficiently in urban areas, then accurate information on current land use has to be collected regularly to forecast the impact of land use change (Bird, 1991). Two types of land uses can be considered, i.e., detailed and general land use. The former is recorded at parcel level whereas the latter does not require parcel information. A parcel-level land use is indispensable for many operational and strategic planning exercises.

Where to get data for urban planning and management is one issue but how to manage this data is still a challenging task (Raza and Kainz, 2000b). This research focuses mainly on the latter, i.e. the technical issue. The core data for urban planning is land use and parcel data, which if updated frequently, raises the issue of structuring, storing, querying and displaying this data. Modeling,
querying and displaying changes in land use data has been a challenging task for many designers and urban planners.

Problems arising from conventional GIS from the data management and query processing perspective can be classified as:

- Data duplication: all data is stored irrespective of whether there is any change or not.
- Data redundancy: due to inappropriate conceptual data model.
- Accuracy: non-coincidence of spatial data (mismatching).
- Query process: temporal data is stored as snapshots in different files, which makes spatio-temporal queries difficult.
- Data volume: duplication and redundancy increase the data volume.
- Spatial and temporal consistency: data is neither spatially consistent, because every time new data is stored (digitized) from scratch, nor temporally consistent, because it destroys the previous data at every update.

Van Heldon (1994) proposed an integrated information system for urban land use management for efficient urban planning. The core of this integrated system is a dynamic database for urban land use management. Current (atemporal) GIS fails to accommodate the dynamic aspects of land use change because the time component is ignored. An atemporal GIS provides a snapshot approach to store the land use data. It describes only one data state (Langran, 1992a). This snapshot approach certainly has limitations when it comes to answering spatio-temporal queries. A system that explicitly incorporates time as one of its structural elements, i.e., temporal GIS (TGIS), is suggested as more appropriate for monitoring land use change. Unfortunately, no such commercial or operational system is available.

Although time is an inherent component of GIS, it has been ignored for various reasons (explained in § 1.2). Its neglect in GIS can lead to the misinterpretation of manifestations of reality. Wood and Fels (1986) provided an excellent example of a spatio-temporal map of a bus trip (Figure 1.1[a]), where by ignoring the time dimension, time collapses to space. A planar projection in which the temporal dimension has been collapsed to zero thickness is shown in Figure 1.1[b]. Space emerges as a product of synchronization (temporal flattening) and the closure of movement. The starting and finishing points of the bus trip become the same, although in the space-time context they are different. A similar example is presented by Parkes and Thrift (1980).

Integrating time and space is not as simple as adding one more dimension in space. This introduces many more problems, from the conceptual design to implementation. For example, at conceptual level one has to decide what type of time model (linear, branching or cyclic) is suitable for TGIS, and what causes change in the feature object. For example, in a very simple case, land use change is normally represented by a spatial part (polygon) and has many attributes (land use, density, number of buildings etc). Change may occur in the spatial or attribute part independently or simultaneously. One has to investigate which change creates a new feature (land use) and which just creates a new version
Technology and development of TGIS

When space and time are integrated, it starts the debate on the fundamental question of how space and time are related. Separately, each is well understood. Many operational or commercial systems are available, i.e., spatial and temporal databases. Technologically, there used to be a lack of appropriate tools for designing and implementing the space-time concepts. These tools are used at various stages of system design and implementation. Advancement in computer science leads us to address/tackle the complexity when space and time are integrated. These tools are available on two fronts, i.e., software and hardware. On the software frontier, object-orientation (OO) is one of the best tools to deal with complexity. It has proven its power and capabilities in complex applications, e.g., computer-aided drafting (CAD), computer-aided manufacturing (CAM), computer-assisted software engineering (CASE) tools, office automation etc. Its importance in spatial data modeling has been recognized by many researchers. Some OO GIS has started to emerge now, e.g., Smallworld GIS or Laser-Scan GIS (Laser-Scan, 1999). Many vendors are now incorporating OO features in their traditional relational database products (e.g., Oracle or Informix) and call this an object-relational approach. Object-orientation has many branches. The important branches are: OO modeling (design and analysis), OO programming languages (C++, Java++, Smalltalk, Object Pascal, Eiffel etc.), OO databases (ONTOS, OpenODB, ObjectStore, Object design, Objectivity, Versant, GemStone etc.), OO user interfaces, artificial intelligence and operating systems. One does not have to rely on different tools available at different stages of designing and implementing the system. There are four stages in OO system development, i.e., conceptualization, analysis, design and implementation (Blaha and Premerlani, 1998). With the aid of OO techniques, a computer scientist or developer can use OO concepts seamlessly from analysis through design to implementation. Better computer networking is available, e.g., LAN (local area network), which is the backbone of client/server architecture. Distributed or client/server architecture can be exploited to boost system performance. Hardware performance is getting better. More space (hard disk) and memory (RAM) and faster processors (single or multi-processors) are available.

1.2 TEMPORAL GIS PROBLEM AREAS

The importance of time in GIS is indisputable. The last two decades have witnessed the development of Temporal GIS (TGIS) from various perspectives. But there are a number of problems associated with the design and implementation of a TGIS. Many researchers have endeavoured to provide insight into the various issues associated with TGIS. Many issues and barriers have been identified in the design and implementation of a TGIS, such as the concept of time and change (Hazelton, 1992; Hazelton et al., 1990a, 1990b; Nunes, 1995; Hermosilla, 1994; Kemp and Kowalczyk, 1994; Wachowicz and Healey, 1994; Yeh and Viémont, 1992; Barrera et al., 1991; Urban and Delcambre, 1986 etc.), temporal relations (Allen, 1983, 1984), spatio-temporal modeling (Langran and Chrisman, 1988; Langran, 1992a; Worboys, 1992a, 1992b, 1994c etc.), architecture, system design, clustering of data (Langran, 1992a), algorithms, technological development, process-oriented (Cheng and Molenaar, 1998) and event-oriented spatio-temporal modeling (Claramunt and Thériault, 1995; Peuquet and Duan, 1995), spatio-temporal query language (Gadia and Nair, 1993; Sarda, 1993; Snodgrass, 1993). Recently, Abraham and Roddick (1999) have provided a survey on spatio-temporal databases.

Encompassing this wide spectrum of issues is beyond the scope of this research. Some issues are application-dependent such as the concepts of time and change, while others are more fundamental and are relevant to any generic TGIS. Langran (1993b, 1992a) pointed out five barriers in the implementation of a TGIS, i.e., architecture, spatio-temporal modeling, spatial changes, clustering of data, algorithms and the individuality of system design. The problems are not limited to these issues. Inadequacy of technological support is also a chief factor (Worboys, 1995). One of the fundamental impediments in designing a generic TGIS is the spatio-temporal data model, which is the focus of this research. The spatio-temporal data model has a profound effect on querying and visualization.
Application-specific modeling will be more efficient if it is based on a generic model (Worboys, 1992a). This generic model could provide a launching pad for many applications, such as urban applications, flood mitigation, environmental impact studies etc. Technologically, incorporating the time component in spatial databases is a challenging task and an active area of research.

### 1.2.1 Application problem

While from the urban planning perspective, time series data is important, its management and display is challenging from a technological point of view. “So far, most GISs have been designed and marked as providing software and functionality for comparative-static analysis where the time frame is indefinite” (Batty, 1992). Conventional GIS represent the real world with only a tenseless snapshot that is inadequate for many urban applications, where facts and data need to be interpreted in the context of time. Conventional GIS (atemporal) assumes a world that exists only in the present. The sense of change or dynamics through time is not maintained while updating the spatial database (Peuquet and Wentz, 1996). A GIS that can store the data with time stamps is more appropriate not only for urban applications but also for many others, ranging from global applications such as environmental monitoring, agricultural applications, forest management, research and development, electronic navigation charts, map and chart production to local applications such as land registration or building permit issue, infrastructure management, transportation etc.

The current systems (GIS) are atemporal, i.e., these systems cannot store information (spatial or non-spatial) varying through time. Every update creates a new data set (map). For example, every update of land use creates a new database or map, which makes the management, query and display of data difficult. Figure 1.2 is a simple example of land use change at times T1, T2 and T3. In current GIS, this information is stored in three different databases (e.g., three coverages in ARC/INFO).

![Figure 1.2 Land use change from T1 to T3.](image)

Any single database (at T1, T2 or T3) may not provide answers to simple queries such as:

- What were the boundaries of the residential area (land use) at Ti (i=1, 2, 3)?
- What spatial or non-spatial change(s) occurred during T1 and T3?
- When was the residential land use converted to industrial?

Land use data is inherently spatio-temporal in nature, and monitoring and planning requires queries such as:

- What was the land use of parcel X in 1990?
- What were the boundaries of the urban area in 1970?
- What was the previous land use of parcel X?
- What land use preceded the current use?
- What were the neighboring land uses of the current parcel in 1960?
- What were the neighboring land uses of the current land use in 1960?
- Has the current land use or parcel ever been an industrial land use?

Conventional (atemporal) GIS may not answer queries like these, nor many others, because space and time are not integrated in a way to provide answers to these queries. So, what are the problems of integrating space and time?
1.2.2 Technological problem

Spatio-temporal data models
As mentioned earlier, the core of the problem is the spatio-temporal data model. Research in spatio-temporal data modeling is scattered. Few integrated approaches exist that can treat the spatio-temporal data in a unified manner. Past research provides a basis but not a complete design of a spatio-temporal model. There are three stages in spatio-temporal modeling, i.e., which comprise the designing of a conceptual, a logical and a physical schema. Moving from the conceptual to the physical schema means more details to ease implementation. At the conceptual level, the prime semantics of the real world objects (feature) are captured by defining the class/object relationships and their behavior. Objects are bound to follow certain rules, which are enforced as consistency constraints. These are the three fundamental requirements of any data model, and naturally of a spatio-temporal data model as well. The objects and their relationships form what we call a data structure or schema.

In the past, two bewildering arrays of approaches have been followed by researchers. The first approach provides a temporal skin on top of existing spatial data models (GIS) and the second, in-depth approach comes up with a relatively complete spatio-temporal data structure. The first approach is sometimes called a 'dual-architecture' (Van Oosterom, 1993). Research by Peuquet and Qian (1996), Peuquet (1994), Yuan (1994), Langran (1992a), Claramunt and Thériault (1995), Raza et al. (1998), Raafat et al., (1994) etc. falls in the first category of models. The difficulties in this approach are noted by van Oosterom (1997). One of the problems in adopting the proposed models is that they are based on existing commercial GIS. Most of the commercial systems are closed systems, which cannot be extended or modified (Van Oosterom, 1993 as cited by Mioc et al., 1998). Wachowicz (1999) shared the same experience and reported the limitations and difficulties of adopting Smallworld GIS while implementing the spatio-temporal public boundaries of Great Britain. In Smallworld GIS, the definition of object properties and their relationships for application is loaded into the version-managed data store (VMDS), using the CASE tool. Although Magik is an OO language (used in Smallworld GIS), there is a mismatch of functionalities between Magik and VMDS (Wachowicz, 1999) and Smallworld is not a fully OO GIS. The second approach is generic, e.g., research by Pigot and Hazelton (1992), Hazelton (1991), Kelmelis (1991) and the OO approach by Worboys (1992a, 1992b, 1992c, 1994c). Some of the prominent spatio-temporal data models and their shortcomings are discussed in Chapter-3.

Langran (1993a) and Peuquet and Duan (1995) have focused on the management and querying of spatio-temporal information within a database, without discussing the spatial relations (Gold, 1996). The approach proposed by Worboys (1992b and 1994c) is based on the concept of simplicial complexes, which is one of the fundamental approaches for any generic TGIS. Although the simplicial complexes are the simplest forms of spaces and their manipulation is comparatively easy, their adoption results in voluminous data. This may increase the system response time.

The models listed are incomplete, as none of them provides the relevant operators and consistency rules. Much of the work in spatio-temporal modeling is restricted to the data structure. An exception is Worboys (1994c), where some possible operators (equal, subset, boundary, union, intersect etc) for the model presented in Worboys (1992b) are defined. However, none of the models explains how the changing topology (spatio-temporal) is stored.

How can these problems be solved? The object-oriented concepts are a promising solution for integrating space, time and attributes for design and implementation of a TGIS. In this research, the OO approach will be used for the design and implementation of TGIS for urban applications.

1.3 SCOPE OF THE RESEARCH

This research addresses two problems, emerging from the application (urban planning and
management) and the technological front (design of spatio-temporal data model). From the application perspective, spatio-temporal issues of parcel and land use change in urban planning and management are investigated. Such technological aspects are dealt with as the design of a uniform spatio-temporal data model and the architecture for a TGIS. The research focuses on the core issue in TGIS and urban planning and management, i.e., the spatio-temporal data model and land use and parcel data. To cover these two aspects, the following fundamental considerations are taken into account:

- The research aims to design the conceptual and logical schemas for a TGIS by taking into account the critical factors associated with each level of modeling, i.e., a unified data structure to treat all objects (feature, spatial and attribute) uniformly, support two spatial and at least two temporal dimensions (linear time), support spatio-temporal topology etc. The spatio-temporal topology is an integral part of TGIS, which is missing in existing models.

- Spatio-temporal modeling in object- (vector-) based systems is the focus of this research, where the incorporation of time increases the complexity of the data structure. The object-based approach is more appropriate in operational planning processes as parcel and land use data is usually recorded using this approach. Most of the research (Al-Taha, 1992, 1993; Al-Taha and Barrera, 1990, 1994; Langran, 1992a; Worboys, 1992b etc.) utilize this approach to describe the spatio-temporal characteristics of parcel and land use change. The use of object-based GIS (3D) has been found more appropriate for the urban environment (Pilouk, 1996; Tempfli, 1998; Zlatanova, 2000; Mesgari et al., 1998).

- The aim is to design and implement each schema by employing standard, yet powerful tools at various stages of modeling, such as object-orientation techniques. At the conceptual level, it is the aim to utilize solid mathematical theory and OO techniques. The mathematical concepts provide a sound basis for defining space and time, which may provide an unambiguous definition of a data model for implementation and a basis for further development of a spatio-temporal query language. OO design facilitates the modeling of space, time and attributes (three components of real world feature objects) in a modular and systematic manner.

- At the logical level, the goal is to design and implement a spatio-temporal database by employing existing commercial database management systems and exploiting the strength and functionalities of these systems.

- Since the TGIS is subject to frequent change, an updating procedure is indispensable. This procedure demands that operators update the database. The aim is to identify, design and implement the fundamental operators needed for updating the spatio-temporal database.

- The aim is to support the client / server system architecture for an integrated urban information system, such that each urban planning department can access parcel and land use data. This architecture permits data to be stored on the server, where each client can access and / or update (depending upon the access rights) parcel and land use data.

- It is important to see the issue of spatio-temporal data in the urban environment from the urban planner’s perspective. The aim is to construct a data flow matrix that identifies the major spatio-temporal data; the data owner and data user; and the use and frequency of updating urban spatio-temporal data. This would also identify the core data needed in urban planning and management.

- The data collection is not primarily the aim of this research. However, time series data collected from the study area is utilized to demonstrate the applicability of the model by developing a prototype system.

It is not the aim of the research to address in detail:

- Spatial indexing to optimize the performance.
- Performance issues including the performance of algorithms in terms of space and time.
- 3D visualization of space and time.
- Philosophical discussion of space and time.
- Spatio-temporal query languages.
- Any other point not listed explicitly in the scope of the research.
Although the research covers two issues (urban planning and technological aspects), its prime focus is on the technological issues. Karachi is selected as a study area because of its complexity from the urban planning and management perspective; it has the characteristics of a typical mega city.

1.4 OBJECTIVES OF THE STUDY

The main objective of the research is to design and implement an object-oriented spatio-temporal data model based on sound mathematics and state-of-the-art object-oriented technology, where time is a structural component. The secondary objective is to document the need of urban planning agencies for spatio-temporal data. The model will be exemplified by a prototype, which will demonstrate how the parcel and land use objects can be updated in a consistent manner, by preserving the spatio-temporal topology of each object. Therefore, the detailed objectives can be outlined as follows:

1] Review and relate the important theoretical foundations of spatio-temporal data modeling.
2] Document the requirements of various service agencies and / or development authorities for spatio-temporal data.
3] Investigate the concept of change at various levels, i.e., generic (node, arc, polygon) and application (parcel and land use) levels and model these changes using OO concepts.
4] Analyze, design and implement a uniform spatio-temporal data model, associated fundamental operators and consistency rules for a TGIS, using an OO approach for consistent updating.
5] Preserve the spatio-temporal topology of changing objects over time.
6] Evaluate the proposed spatio-temporal data model by developing a prototype system for parcel and land use applications.

1.5 RESEARCH APPROACH AND STRUCTURE OF THE THESIS

This research is conducted with the belief that employing state-of-the-art OO concepts (tools) and relying on a solid mathematical basis may reduce the complexity of spatio-temporal data models. In this respect, the core concepts of simplicial complexes (Worboys, 1994c), the generalized cell complex (Pigot, 1995) and the cell tuple structure (Brisson, 1990) will be utilized for further development of spatio-temporal data models.

The research is not intended to develop a fully functional TGIS; rather it focuses on the core problem in the development of TGIS, i.e., spatio-temporal data modeling and system architecture. The approach is to use OO concepts to systematically identify the

• classes (objects) at various stages of the modeling,
• attributes and responsibilities (operations / behavior) of these objects,
• relations between these objects,
• consistency constraints on these objects,
• other related issues.

Although the thesis is organized into eight chapters, it can be broadly divided into three parts: introduction and fundamental concepts (Chapter-1 and Chapter-2), system development (Chapter-3 to Chapter-7) and conclusions and recommendations (Chapter-8).

1.5.1 Part-I: Introduction and fundamental concepts

Chapter-1 (this chapter) provides the motivation behind this research. In a broader sense, it highlights the problem areas in the development of a TGIS and the core problems in urban planning and management. It defines the scope of the research and formulates the objectives of the research.

The objective of Chapter-2 is to discuss the fundamental concepts of space and time. It covers and identifies the components of TGIS (space, time and attribute), associated mathematical terms and
concepts. It provides the definition of space and time, and discusses the spatial and temporal relations. The topics discussed in this chapter provide a basis for further development of the system in part-II.

### 1.5.2 Part-II: System development

This part provides the various phases of OO system development, which consist of conceptualization, analysis, design and implementation.

**Conceptualization phase**

The conceptualization phase conceives an application and formulates application requirements. The application developed here is general and any urban planning agency can make use of it. The system will solve the problem of spatio-temporal (parcel and land use) data management. It will provide the tools to query and visualize not only the state of any particular object but also the change in the object. The system needs tools (designing, database and programming) and time series parcel and land use data. Spatio-temporal data needs to be organized in an efficient manner that can facilitate not only the operational planning process but also strategic planning by monitoring the changes in urban areas. The system will decouple program functionality from the user interface, therefore requiring two layers: one for the user interface and the other for connecting the user interface to the database. This phase, as discussed, is covered in the current chapter.

**Analysis phase**

The analysis phase lists classes and associations, including generalizations between classes, attributes, operations and consistency constraints. Hence, this analysis provides a conceptual schema. Chapter-3 and Chapter-5 define a conceptual schema (with consistency constraints) for a generic and for a parcel-level land use change model, respectively. Chapter-4 provides a conceptual framework for operators.

Chapter-3 provides an overview of existing spatial / topological and spatio-temporal data models (relevant to this research) and identifies their potential role as well as limitations in adopting these models for a TGIS. The modeling of time (i.e., whether time should be considered as either an attribute or as a separate dimension) is discussed in this chapter. Design issues are taken into account where the OO design approach has been followed rather than conventional top-down structured design or data-driven design. Here the Unified Modeling Language (UML) is used to define the conceptual schema. Three fundamental classes, SpatialClass, AttributeClass and TemporalClass, their subclasses and aggregated classes, along with attributes and operations, are discussed in detail. Here, the concepts borrowed from cell tuple structure, simplicial complexes and the generalized cell complex are extended and called the temporal cell complex. This leads to the development of the unified cell tuple-based spatio-temporal data model (CTSTDM).

Chapter-4 elaborates the operators (static and dynamic) identified in Chapter-3. The conceptual design of fundamental dynamic operators for the generic part of the model is provided. Here the construction of CTSTDM is demonstrated, using these fundamental dynamic operators. With the aid of the point-set topology approach, various intersections between CTSTDM objects have been analyzed. All valid intersections are reported in this chapter. The chapter proceeds by defining the static operators by formalizing the spatio-temporal relations between CTSTDM objects.

The need of urban planning agencies for spatio-temporal data is documented in Chapter-5. After a brief introduction to the study area, the chapter discusses the role of land use data in urban planning, why monitoring of land use is significant and the types of land uses. The concept of change in parcel and land use is introduced in this chapter. Based on OO concepts, general and detailed land use changes are modeled. As the detailed land use changes are parcel-based, spatio-temporal aspects of parcel objects are also modeled. The conceptual schema for general and detailed land use change is devised by integrating the conceptual schema for CTSTDM (as appears in Chapter-3).
Design and implementation phase
The design phase deals mainly with the refinement of the analysis phase, architecture, the selection of a data management approach and algorithms for the methods. Decisions taken here are platform-independent. The implementation translates the design into a database schema and programming codes. These two phases are covered in Chapter-6 and Chapter-7.

The conceptual schema presented in Chapter-3 and Chapter-5 is mapped to a relational schema in Chapter-6. Functional dependencies are checked to achieve the normalized relations.

A three-tier (or three-layer) client/server architecture for OO TGIS is devised in Chapter-7. The first layer consists of an OO graphical user interface, the second of a graphical user interface controller and the third of a database. Various possibilities regarding how to access the data on the server are discussed in this chapter. Some problems of geometric intersection and algorithms associated with the fundamental operators are presented in this chapter. In the prototype, first the procedure of data transfer from ARC/INFO to CTSTDM is explained, followed by the procedure for updating the land use and parcel objects. Using SQL, not only are simple temporal and temporal range queries to retrieve the spatio-temporal objects demonstrated, but also how to scan the object history and how to retrieve temporal topology. Further, the applicability of the cell-tuple structure is shown for retrieving the objects based on spatio-temporal topology. A graphical interface is developed to display some of the query results. A short discussion on the CTSTDM performance concludes this chapter.

1.5.3 Part-III: Conclusions and recommendations
Chapter-8 summarizes the major findings of the research and provides recommendations for future work.
Chapter-2

Concepts of Space and Time

2.1 INTRODUCTION

We are living in a dynamic world in which the time element plays a vital role. A data model helps us to understand the dynamic processes of this real world. Current data models (atemporal) are not sufficient to address the dynamic nature of the world because they lack the time component. Spatio-temporal data modeling demands different techniques to capture and model the real world phenomena. There are various issues in respect to omitting the temporal element in spatial information systems. These issues range from fundamental conceptual issues of space, time, and space and time, to the complex data structure at the implementation level (physical design).

This chapter deals with the fundamental constituents of spatio-temporal data modeling by summarizing the concepts pertaining to space, time, space-time, similarities and differences between space and time, change etc. Further components of Temporal GIS (TGIS), i.e., spatial, aspatial (attribute) and temporal components, are discussed in their own domains. The spatial component describes the space and spatial relations, the temporal component recapitulates the time and temporal operations, and the aspatial component defines the attributes and attribute relations.

2.2 ABSTRACTION PROCESS OF REALITY

The dynamic aspects of the world are described by events, processes, actions, activities and accomplishments, e.g., population of a city at time T (Allen, 1984). In describing the dynamic aspects of the real world, the time dimension plays a vital role. The real world (reality) may be described as ‘static’ when the time dimension is ignored or less important, and ‘dynamic’ when the time dimension is an inherent component. Data modeling helps us to understand the static and dynamic properties of the world to the desired level of precision, which would otherwise not be possible. The expressiveness is the power of any data model. Modeling means the abstraction of reality and it is the core of an information system. An abstraction is a simplified description of reality. A good abstraction is that in which information significant to the user is emphasized, and details that are immaterial or diversionary, at least for the time being, are suppressed. In computer science, modeling means the formulation or representation of reality, using various tools and disciplines for the system under consideration. Like in any other information science, such as spatial information science, modeling attempts to define the real life phenomena through objects and their relationships and constraints. Modeling of space and time is known as spatio-temporal modeling. An information system based on space-time modeling is called the Temporal Geographic Information System (TGIS). There is no data model that can claim to be a 100% representation of reality, due to the fact that there are several levels of modeling this reality (discussed in next section). Information is lost in mapping from one level to another, or because of the concepts, definitions and semantics used in different disciplines and in different societies for the same phenomenon or application.

Static and dynamic real worlds need different techniques for description and modeling (Jackson, 1983). Various attempts have been made to define the number of levels required to model reality. The JSD (Jackson system development) model, proposed by Jackson (1983), suggests two distinct levels for modeling the real world (Figure 2.1). Realization in the computer consists of a conceptual, logical and physical data model. The handling of the time dimension in JSD is of central importance, because of its emphasis on the time component, JSD is widely used in applications such as banking, airlines, elevator control systems, on-line and batch processing systems etc. (Jackson, 1983). Peuquet (1984) defined four levels of abstraction. In Figure 2.2, the last three views of the data corresponds to
the major steps involved in database design and implementation (Peuquet, 1984). Pilouk (1996) elaborated on the process of abstraction proposed by Peuquet by introducing the construction phase. Molenaar (1995) proposed the involvement of various disciplines while modeling reality. The core of these approaches, i.e., in a conceptual, logical or physical level design, is based on ANSI/SPARC architecture for the abstraction of reality.

![Figure 2.1 Modeling reality (adapted from Jackson, 1983).](image1)

![Figure 2.2 Four levels of abstractions (adapted from Peuquet, 1984).](image2)

The overall system design and modeling process is depicted in Figure 2.3. In the dynamic world, objects and their relationships are changing because of human intervention and natural processes. At the external level, realities are perceived/viewed differently by different disciplines/users. The difference between spatio-temporal and aspatio-temporal data modeling is that, in spatio-temporal data modeling, the relevant part of reality is mapped in the database with a time stamp. Usually, only that part which is relevant to the application is modeled. For example, in the building control application, a building at time T1 may be modeled as the location of the building, relative position, owner, land use, shape, size, height, or perimeter; if there is any change, this is recorded subsequently at time T2, T3 etc. Other applications might look at the materials of which the walls are built, the water, electrical, and telephone networks, or the architecture of the building.

The design phase deals with the abstraction of reality, i.e., the representation of objects and their relations, at three levels: the conceptual, logical and physical levels. At the conceptual level, components of reality are defined as an object, attributes and relationships in a more abstract form. For example, a building would be represented by an area or a point feature; (what would be) the attributes, i.e. owner’s name, road number, land use; and (what would be) the nature of time (continuous or discrete), and how time would be represented as days or years. At this level, semantic data modeling such as the Entity-Relationship (ER) approach is used for mapping reality. Objects are then mapped to the logical level and stored in a database. Four major choices are available, i.e. relational, network, hierarchical and object-oriented (OO) data models. At this level, the application developer has to decide how the geometry will be stored, the type of numeric variable, the width of string for the attribute, and the way in which time will be represented (year, month or day). At the physical level, data are stored on hardware. Application discipline, computer science and geo-information theory act as guiding rules and tools to bridge the gap between the representation of reality and the implementation of the physical model. The design phase embedded the first two parts of data modeling, i.e., the object and their relationships.
The third part of the data modeling, 'constraints', is partially defined in the design phase and in a database management system (DBMS), in our case a spatio-temporal information system or Temporal Geographic Information System (TGIS). Originally, these constraints were introduced by Date (1990) as database integrity rules. Constraints are employed to enforce the logical consistency in the database. Logical consistency deals with the logical rules of the structure and the attribute rules for spatial data (Kainz, 1995). There are several levels of logical consistency, ranging from simple attribute, spatial (geometric and topological) and temporal constraints, to specific consistency rules for spatial and temporal relationships and particular applications. These constraints are necessary as real world events may occur in strict order, and attribute values lie within a certain range. For example, a person (owner) must have a parcel prior to applying for a loan from the bank for house construction. There is no property tax on parcels smaller than 80 m². An owner must pay non-utilization fees (penalty) if he/she fails to construct the house within five years of allotment. The value of a day ranges from 1 to 31 (in Gregorian calendar) and areas cannot be negative. An information system is called consistent when it makes sure that, whenever change occurs in an object or its components, all other objects or components whose values are derived from that object or component will be updated automatically. Kainz (1995) discussed consistency at three levels: database consistency, topological consistency and scene consistency. An added difficulty in modeling reality in TGISs is maintaining the consistency or logical link with past and future data, as the updating process either creates new objects and relations or modifies existing objects or relations. Therefore, temporal consistency is important in a spatio-temporal database. For example, avoiding the re-write of unnecessary information, a system whose temporal topology is linear can force a date to have only one active version of an object at a time; this is preceded and succeeded by a single active version (Langran, 1992a).

A user interface (graphic or textual/alphanumeric) is provided in the information system to assist the users in creating, updating, processing and presenting the spatio-temporal data. Every system has a certain life, which is called the 'life cycle' of the system. This life cycle varies according to application and state-of-the-art technology. After a certain time, the system has to be upgraded or replaced to accommodate the requirements of the user. A good system is the one designed in modular fashion. For example, a system can be upgraded by just replacing the relational data model by the OO data model (at the logical level).
From the above discussion, four major parts of any data model can be identified, i.e., a] object, b] relations, c] attributes and d] constraints. To continue our discussion on the modeling process, we shall now discuss the conceptual issues pertaining to the representation of reality. We shall start with the concepts of space, time, space and time, and change.

2.3 CONCEPTUAL ISSUES

In the following section, we shall discuss the conceptual issues pertaining to spatio-temporal data modeling, i.e., space, time and change. Emphasis has been given to discussing these conceptual issues in a more structured way. Therefore, the concept of mathematics has been employed because mathematics is the study of the properties of mathematical structures.

2.3.1 Space

In the past, many philosophers and scientists have tried to define space. There have been two broad trains of thought on this issue, i.e., relative and absolute space. In relative conceptions, space is a property of things, meaning that things have spatial attributes; in absolute conceptions, space is a real entity, the container or the medium. Leibnitz (1646-1716) considered that space was only a system of relationships between things (Nunes, 1995). The first approach is based on the concept of considering space as a set of objects with spatial properties (relative space), and the second as a set of locations with properties (absolute space). These approaches have resulted in two types of implementation, commonly known as the vector- or object-based, and the raster- or field-based approach, respectively. Compared with time, the research and discussion on space in GIS is now quite mature. We shall restrict our discussion to the mathematical concept of space, rather than the philosophical, by considering space as a system of relationships, as defined by Newton. These relationships exist between the objects of real life. Therefore, space may be defined as a set of objects and relations between them. In GIS, we deal with objects and their relationships in Euclidean space. Objects in this space are portrayed by nodes, arcs, polygons and volumes. Different types of relations define different types of spaces; for example, distance relations define metric space and topological relations define topological space. These spatial relations are discussed in § 2.4.2.

Any mathematical theory is based on some primitive notions. Set theory is based on the primitive notions of ‘set’ and ‘relations’. Mathematically, we can define a space using set theory, which was first introduced by Cantor (1845-1918).

Let \( O = \{ O_1, O_2, O_3, \ldots, O_n \} \) be a set of objects. \( R \) is a binary relation on \( O \), if \( R \subseteq O \times O \). If \( R \) is a relation on \( O \), then the relationship \((O_i, O_j)\) can be represented as \( R(O_i, O_j) \). We can define space \( S = (O, R) \), which is a collection of subsets \((O, R)\). In reality, space is unbounded and consists of large numbers of objects and relations. The space \( S \) (finite) is a view of reality from the discipline perspective.

Some basic properties of relations \( R \) are as follows; \( R \) is:

- **Reflexive:** A relation \( R \) on \( O \) is reflexive iff \( (O_i, O_i) \in R \), \( \forall O_i \in O \), i.e., if every element of \( O \) is in relation with itself. For example, point A is equal to itself and point time \( T \) is equal to itself.

- **Symmetric:** \( R \) is symmetric iff \( R(O_1, O_2) \Rightarrow R(O_2, O_1) \), \( \forall O_1, O_2 \in O \). For example, in space, parcel A is adjacent to parcel B implies that parcel B is adjacent to parcel A.

- **Antisymmetric:** \( R \) is antisymmetric iff \( R(O_1, O_2) \) and \( R(O_2, O_1) \Rightarrow (O_1 = O_2) \), \( \forall O_1, O_2 \in O \), i.e., \( R(O_1, O_2) \) does not imply \( R(O_2, O_1) \).

- **Transitive:** \( R \) is transitive iff \( R(O_1, O_2) \) and \( R(O_2, O_3) \Rightarrow R(O_1, O_3) \), \( \forall O_1, O_2, O_3 \in O \). For
example, in space, parcel A is greater than parcel B and parcel B is greater than parcel C, then parcel A is greater than parcel C; and in time, A happens before B and B happens before C, then A happens before C.

For example, \(<\) is a transitive relation on \(\mathbb{R}\), \(\leq\) is a reflexive, antisymmetric and transitive relations on \(\mathbb{R}\) and \(\neq\) is a symmetric relation on \(\mathbb{R}\). Binary relations which are reflexive, symmetric and transitive are known as equivalence relations. Detailed discussion on relations can be found in Moise (1977) and Willard (1970).

### 2.3.2 Spaces and manifolds

Three general types of spaces (manifolds) can be defined (Hazelton, 1991):

- infinite spaces (open manifold)
- finite space (closed manifold)
- infinite space with boundaries (manifold with boundary).

The open manifold is a (space) spatial object that is infinite and has no limits (i.e., spatial object without boundaries or edges), for example, an infinite line (1-manifold) or an infinite plane (2-manifold). Open manifolds are not useful for describing objects in CAD or GIS. Closed manifolds are (spaces) spatial objects that are closed or have finite extent (cubes or spheres). Closed manifolds have finite area and do not have any boundaries or edges. In fact, a manifold with a boundary is derived from the open manifold and categorized as an open manifold with boundaries. Euclidean manifolds fall in this category. They are the simplest of all manifolds. More often, spatial objects in CAD and GIS are embedded in Euclidean manifolds. Closed and open manifolds can be defined as follows (Pigot, 1995):

An \(n\)-manifold (closed) can be defined as a topological space \(X\) in which every point \(x \in X\) has a neighborhood homeomorphic to the \(n\)-dimensional disk.

An \(n\)-manifold (open) with boundary can be defined as a topological space \(X\) in which (other than the boundary) every point \(x \in X\) has a neighborhood homeomorphic to the \(n\)-dimensional disk and every point on the boundary has a neighborhood homeomorphic to the \(n\)-dimensional hemi-disk.

### 2.3.3 Functions

Function is a special type of relation, where every member of the first set relates to exactly one member of the second set. Each member of the first set (domain) is transformed into a member of the second set (co-domain) through a mapping rule. Formally, function could be defined as:

\[
f: D \rightarrow C
\]

where \(f\) is a function, \(D\) is domain and \(C\) is co-domain. Functions too have a number of properties.

- **Injective function**: \(f\) is injective (one-to-one) if \(d \neq d' \implies f(d) \neq f(d')\), i.e., if two different members of the domain are transformed into two distinct members in the co-domain, then the function is called injection. For example, \((-2)^3 = -8\) and \((2)^3 = 8\) results in two different values; therefore, cube is an injective function.

- **Surjective function**: \(f\) is surjective (onto) if \(f(D) = C\). An image is a subset of the co-domain. If the image is equal to the co-domain, then the function is called a surjective function.

- **Bijective function**: A function that is both injective and surjective is termed a bijective function.
• **Inverse function:** An injective function whose co-domain becomes the domain of a function is known as an inverse function. The inverse function of \( f \), denoted by \( f^{-1} \), is the converse relation of \( f \).

### 2.3.4 Time

Time, a phenomenon that can be perceived only by its effects, has stymied philosophers since its beginning. Many philosophers have made endeavors to define time but none of their definitions has been universally accepted. Time is the single most controversial issue in a generic TGIS. Understanding the enigma of time has been an intriguing problem ever since the beginning of human civilization (Ramachandran *et al.*, 1994). Newton viewed time as a dimension that is separate from but similar to spatial dimensions, in that all are containers for occurrences. The turn of the 19th century witnessed a popular fascination with the nature of the fourth dimension. The fourth dimension was presumed to be a spatial one – a sentiment that gave rise to non-Euclidean geometries. By the 1920s, most scientists had accepted Einstein's premise that time is the fourth dimension that interacts with space (Freeman and Sellons; 1971, Morris, 1984; as cited by Langran, 1992a). Time only seems to pass because our memories work backwards. Thus, while we are ever present at every instant of our life, at each instant we can remember only what has passed, not what is yet to come. Indeed, "Time definition is an ambitious exercise to which many scientists or philosophers have tried to give an answer. However, none of these is universally accepted" (Joerin and Claramunt, 1994). The nature of a complication is rooted in the way we understand and deal with the real world. Al-Taha (1992) summarized the factors contributing to the complications as:

- **Imprecision of Expression / Lack of Semantics of a language:** For example, in English *now* means today, this week, this year, or this century; *today* means at present or this day; *future* means tomorrow, after a week or after a year; and *past* means an hour ago, yesterday, or a week ago.

- **Diversity of Interpretation:** A *year* might be very small for the historian but a *second* could be large enough for a certain nuclear process.

- **Diversity of Events or Processes:** Events have effects and order.
  - **Effects:** Events or processes may have distinct effects, for example instantaneous, duration, delay, or no effects.
    - Instantaneous: turn on the light.
    - Duration: processes that last over a period of time.
    - Delay: a building permit will be issued a week from the date of submitting the application.

- **Order:** Events may have a certain order. For example, in an online banking system, the customer inserts his bank card before pressing a PIN code; if the PIN code is correct, he then keys in the amount to be drawn and withdraws the card after receiving the cash. In agricultural applications, events also occur in a particular sequence, i.e., crop cycles (Joerin and Claramunt, 1994).

Events themselves can be divided into two categories, external and internal (man-made events). External processes are natural processes, such as flood, temperature, rainfall etc. Internal events are non-natural processes, such as change of ownership or mortgage in a cadastral system, change of land use in a planning system etc.

The issue is not restricted to the aforementioned complications; several other issues have to be considered while designing a temporal database (Al-Taha, 1992). These are related to the conceptual, logical and structural issues of the time. These are discussed in §2.4.4 (Temporal operations). Time seems to be simple in our daily life but its integration in databases is complex and challenging. By sidestepping the philosophical strife of time, we can define it as a phenomenon that can be perceived...
only by its effects (Langran, 1992a). Sharing the Isard’s comments (1970), as cited by (Langran (1992a):

“I do not wish to think about time in any philosophical or metaphysical sense. I am searching for operational concepts of time - concepts that will lead to its direct measurement, to better theory, and to a rich set of models using time as a variable”.

Users may refer to Nunes (1995) for a detailed discussion on the concepts of time, and time and space. Some history of the philosophical thoughts about time are discussed by Roshannejad (1996).

2.3.5 Space and time

The thorny problem of TGIS is the integration of space and time. The concept of space and time is known to everybody in daily life. Although space and time are two fundamental notions learned by human beings in their childhood, consensus about them is difficult because of different cultures, professions and disciplines (Nunes, 1995).

The discussion on the concept of space-time covers a wide period of history. Many philosophers, for example Pythagoras, Plato, Aristotle, Euclid, Newton and Einstein have investigated and presented the concepts of space and time, but none of them is universally accepted. Newton conceived time as a separate dimension, although similar to the spatial dimension because both contain occurrences (Langran, 1992a). Except for point representation of time, which does not have a sense of containment, it has a magnitude rather than an extent. Based on Einstein’s famous theory of special relativity, time has been considered as a forth dimension because of its similarities with space. Here, we shall succinctly highlight some analogies between space and time and continue our discussion of a conception of change.

Let \( O = \{O_1, O_2, \ldots, O_n\} \) be a set of objects, \( S = \{S_1, S_2, \ldots, S_n\} \) is space (place) and \( T = \{T_1, T_2, \ldots, T_n\} \) is a time. While correlating space and time, an interesting scenario could be drawn.

- An object cannot be at different places at one time (Figure 2.4a).
- An object can be at one place more than one time (Figure 2.4b).

The situation in Figure 2.4b is isomorphic to the spatial dimension, where the two or more spatial dimensions are independent; in the case in Figure 2.4a, space and time are dependent. Some more differences and similarities can also be observed from the following considerations.

2.3.5.1 Similarities

- Space and time have a sense of containment (space filling).
- Space and time have more than one dimension.
- Both are metric, because both have a distance function.
- Space and time may be regarded as discrete, dense and continuous.
Chapter-2: Concepts of Space and Time

- Space and time may be considered as bounded.
- Both have a sense of absolute (at 5 meters and at 9:00 a.m.) and relative (10 meters away and five hours later) value.

2.3.5.2 Differences

- Space is limited to three dimensions while time may have more than three dimensions.
- Walk-back is not possible in time; a space may be walked through many times.
- Time is dynamic, as it is moving constantly; a space is static, as places remain unchanged.
- The range of time granularity (temporal resolution) is very high; the range of scale (spatial resolution) is confined.
- More often, time is open-ended (unbounded) but space is almost closed or bounded (Easterfield et al., 1991). In GIS, we do not need to count values greater than approximately 70 million meters (Snodgrass, 1992).
- Time has a natural order and it provides the sense of past, present and future.

Some equivalents of space and time are illustrated in Table 2.1.

<table>
<thead>
<tr>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>State</td>
</tr>
<tr>
<td>Configuration separated by</td>
<td>Events</td>
</tr>
<tr>
<td>Units</td>
<td>Versions</td>
</tr>
<tr>
<td>Separators between units</td>
<td>Mutations</td>
</tr>
<tr>
<td>Size measured by</td>
<td>Date</td>
</tr>
<tr>
<td>Position described by</td>
<td>Duration</td>
</tr>
<tr>
<td>Neighbors</td>
<td>Previous and next version</td>
</tr>
<tr>
<td>Maximum number of neighbors</td>
<td>Two</td>
</tr>
<tr>
<td>Map</td>
<td>Length, area</td>
</tr>
<tr>
<td>Sheet lines</td>
<td>Co-ordinates</td>
</tr>
<tr>
<td>Objects</td>
<td>Adjacent objects</td>
</tr>
<tr>
<td>Boundaries</td>
<td>Infinite</td>
</tr>
<tr>
<td>Units</td>
<td></td>
</tr>
<tr>
<td>Separators between units</td>
<td></td>
</tr>
<tr>
<td>Size measured by</td>
<td></td>
</tr>
<tr>
<td>Position described by</td>
<td></td>
</tr>
<tr>
<td>Neighbors</td>
<td></td>
</tr>
<tr>
<td>Maximum number of neighbors</td>
<td></td>
</tr>
</tbody>
</table>

*Table 2.1 Equivalence of space and time (adapted from Langran, 1992a).*

Because of these analogies, the treatment of time in spatio-temporal databases requires careful analysis of space and time, compared with dealing with time in non-spatial databases. There is a great deal of work on temporal databases, but not much on spatio-temporal databases. If we consider time as a fourth dimension (three spatial and one time), then space and time should be modeled using the same modeling techniques. If we consider time as a separate dimension, different techniques would be required to model each dimension in its own domain.

If we consider time as a phenomenon that can be perceived by its effects, then the question of what a change is becomes more crucial. Time and change are strongly related. The sense of change is meaningless without time. Change indicates that there is a change in the state of an object at a particular time, but a change in time does not imply that there is a change in a object's state. We may say that time is a function of change or of an object's state \(T \rightarrow C\). Understanding the concept of change would help us to delineate the boundaries of event, evidence, versions and mutations.

2.3.6 State, event and evidence

Atemporal GIS represents the 'state' of the geographic object, but TGIS needs three types of data, i] state, ii] events, and iii] evidence (Langran, 1992b). State is comprised of feature and attribute. Events are the occurrence that causes a given state to change, making various versions of a particular feature. Evidence is the source document that informs us that an event has occurred or that a new state exists.
Together state, event, and evidence may be described in terms of spatial extent and attributes. It is possible to treat all three types of data in TGIS identically, as a set of features with spatial extent and descriptive attributes. Normally, the state in the GIS is represented by point, line or area, and then described by attributes. Table 2.2 and Table 2.3 illustrate the data needed in the case of state, event and evidence with attribute descriptions in urban applications. By maintaining state, event and evidence with a time stamp (world time and database time) in the database, TGIS can help us to establish relationships between state and event. Taking into account lag time, it is possible to track in the database, e.g., when changes occur in the world, when they are measured, and when they were entered in the database.

### 2.3.7 What is a change?

The concept of change is still fundamental to philosophy and science, and can be defined as follows: if object O changed, then it cannot be considered the object O (Nunes, 1995). Therefore, object O is no longer in the same state. The real world is dynamic, everything is changing, either slowly or very rapidly. It is not possible to capture every change. Therefore, these changes are characterized by a set of properties (Roshannejad, 1996). As discussed, in modeling reality the essence of any application is the identification of its objects and the relations between them.

Properties of an object are defined by its application. Categorizing the changes on the basis of properties helps us to identify the new object(s) and its new versions. For example, 1] in the urban environment, a building remains a building in spite of its change of use from residential to commercial or change of owner or change in color; 2] in land use applications, change of land use is more important than change
in shape or boundary, 3] in legal cadastral systems ownership is more important than change in boundary, and 4] in building control applications, alteration in the building is more important than change in ownership.

The categorization of changes on the basis of properties could be classified as 1] essential change and 2] non-essential change (Raza, 1996). Roshannejad, 1996 termed these two changes as constructive and non-constructive changes, while Raafat et al. (1994) referred to these changes on the basis of essential and non-essential properties. In the case of land use applications, land use is the essence of the applications and, therefore, it is an essential property. If the land use changes, then it is an essential change; change in shape or size is a non-essential change. Essential change means the creation of a new object and non-essential change means a new version of the same object. This process is illustrated in Figure 2.5. This change model consists of three layers: 1] object, 2] properties and 3] change. Detailed discussions on the issue of ‘what makes a change’ are presented in spatio-temporal data modeling (Chapter-3 and Chapter-5).

2.4 COMPONENTS OF SPATIO-TEMPORAL MODELING AND MATHEMATICAL FOUNDATION

We are living in a three-dimensional world. Facts and processes, being real and concrete, are defined by three components, i.e., space, time and attribute (Figure 2.6a). For modeling purposes a reality or terrain feature could be considered an object. Irrespective of the application (except a few), this object comprises three components, the spatial, temporal and attribute (aspatial) components, and may be called a spatio-temporal-attribute object (STAO). Based on OO concepts, these three components could be further divided into their respective domains, describing classes and operations (relations). The spatial component consists of geometry and spatial operations; the temporal component is composed of time and temporal operations; and the attribute component describes the attribute of the object. The attribute component is sometimes referred to as a thematic component. We shall now highlight the spatial and temporal components, based on pure spatial and pure temporal concepts, and later (Chapter-3) integrate the three components for spatio-temporal modeling. We can define STAO, which possesses state and behavior.

2.4.1 Spatial component

The spatial component pertains to space. The state of a spatial object is described by its position, dimension and geometry, and its behavior by spatial relations or operations over time (Figure 2.7).

---

Figure 2.6 Extreme level of abstraction for reality: spatio-temporal attribute object (STAO).
2.4.1.1 State of spatial object

Position
The positioning of objects in space is a fundamental requirement in any spatial information system. The positioning of geo-objects requires a referencing system. Referencing is a method of describing the absolute or relative placement of an object or its representation. In two- or three-dimensional space, referencing may be a Cartesian or polar system. Polar is based on distance and extend direction from a specific point and Cartesian has two values specifying distances in two perpendicular directions. The Cartesian reference system mostly uses Euclidean geometry for positioning the objects in space to measure the distances from a specified origin (local or global) (Laurini and Thompson, 1996).

Dimension
One of the criteria for classifying an aspatio- or spatio-temporal information system is the 'multidimensionality' of the system, i.e., the number of spatial dimensions supported by the system. The n-dimensional (nD) system supports all spatial objects of dimension less than or equal to n-dimension. The n-dimension usually refers to the spatial dimension. The dimensions of Euclidean space are expressed through the number of referential axes (independent). Mathematically, the distance from the origin of each axis is represented by an ordered n-tuple $(a_1, a_2, a_3, ..., a_n)$, which is a sequence of n-real numbers. The set of all ordered n-tuples is called nD-space and denoted by $\mathbb{R}^n$ (Anton, 1994).

Usually, 2D- and 3D-GIS refers to the geographic information systems that support two (x and y) or three (x, y and z) spatial dimensions, respectively. 4D-GIS (x, y, z and t) is a geographic information system that incorporates three (x, y and z) spatial dimensions and at least one time dimension. However, most of the commercial GIS (ARC/INFO, SmallWorld, ILWIS, Intergraph etc.) provide two spatial dimensions, which means all spatial objects of 0-, 1- and 2-D can be described. We are not going to argue about 'what is a 4D- or 3D-GIS'. Our focus is to incorporate a time in a 2-dimensional space. As discussed earlier, because of the differences and similarities between space and time, we cannot treat time as a purely spatial dimension. At the same time, we cannot say that the time is an attribute of a spatial information system. We will leave this paradox here (this issue will be discussed in Chapter-3) and simply consider a TGIS, which supports two spatial dimensions and at least one time dimension.

Geometry
Geometry is a property of space or a branch of mathematics that deals with the quantities and shapes of real world objects. It is a mathematical study of the properties and relations of objects in space. Objects are either man-made or natural. Man-made objects include buildings, national boundaries, planning zone boundaries, roads, railways, communication networks, streetlights etc., while natural objects are rivers, mountains, environmental zones, land cover boundaries etc. Irrespective of their nature, these objects are geometrically portrayed as point, line, area or volume. Geometrically, these objects can be specified by a set of coordinate tuples, or computable equations and by a set of primitive objects (Worboys, 1992c).
2.4.1.2 Representation and relationships of spatial objects

Spaces are represented as objects, called spatial objects. More often, in GIS (vector), two types of objects are used to represent the space, i.e., simplices and cells. The aim of dividing the space into simplices and cells is to simplify the calculation of topological properties of space. Therefore, these objects not only represent the space but carry the topological information as well, which can be used to enforce the topological consistency in the spatial data model.

2.4.1.2.1 Simplex and simplicial complex

Objects in space are classified according to their spatial dimension. For each dimension, there is a minimal object, called simplex (Egenhofer, 1993). A simplex is the elementary geometric object in a given dimension (Frank and Kuhn, 1986). It is a building block from which other objects can be constructed; therefore, a simplex is a primitive level of representing the geometry. A closed simplex is a closed subset of $\mathbb{R}^n$, i.e., it contains all its limit points (Giblin, 1977). The simplex (closed) of dimension $n$ is called an $n$-simplex. A 0-simplex is a node or vertex, a 1-simplex is an edge, a 2-simplex is a triangle and a 3-simplex is a tetrahedron (Figure 2.8).

![Figure 2.8 Configuration of simplex and simplicial complex.](image)

Mathematically, we can define a simplex as: let there be $n+1$ linearly independent points ($v_0, v_1, ...., v_n$), the vertices in $\mathbb{R}^n$. An affine simplex $S_n$ of dimension $n$ or $n$-simplex in $\mathbb{R}^n$ is defined as:

$$S = \left\{ \sum_{i=0}^{n} \lambda_i v_i \right\} \text{ with } \sum_{i=0}^{n} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \text{ for all } i = 0,1,...,n \right\}$$

Therefore, an $n$-simplex is a convex hull of $n+1$ geometrically independent points, called vertices ($v_0, v_1, ...., v_n$). For example, any $n$-simplex is bounded by $n+1$ geometrically independent simplices of dimension $n-1$. A triangle, a 2-simplex, is bounded by 3 1-simplices and a line, a 1-simplex, is bounded by 2 0-simplices. The orientation of the simplex is defined by the ordering of the vertices ($v_0, v_1, ...., v_n$). The orientation of a 0-simplex is unique; a 1-simplex may be oriented from first to second vertex or inversely from second to first vertex; and a 2-simplex may be oriented clockwise or anti-clockwise. A face of a $k$-simplex is a simplex of dimension less than $k$, forming part of the simplex (Kainz, 1995). An $n$-simplex has \( \binom{n+1}{i} \) faces of dimension $i$. For example, a 2-simplex has \( \binom{3+1}{1} \) three 1-dimensional faces and \( \binom{3+1}{0} \) three 0-dimensional faces. The 0- and 1-dimensional faces of a simplex construct a complete graph of $n+1$ vertices (see § 2.4.2.2.1).

**Simplicial Complex**

A simplicial complex $S$ is a set of affine simplices in $\mathbb{R}^n$ that fulfill the following conditions:

- If the simplex $S_0$ is an element of $S$, then each face of $S_0$ belongs to $S$.
- For any two simplices in $S$, the intersection is either empty or a common face (Figure 2.8).

Simplicial complexes are useful for representing more complicated geometry. There are various applications of simplicial complexes in spatial data modeling; examples include 3D GIS, triangular-
irregular-network (TIN) and network analysis. The major disadvantage of simplicial complexes lies in the fact that large numbers of triangles are generated to represent the terrain feature, because every curve has to be approximated to a straight line. The second major factor for rejecting simplicial complexes is to control the data volume, as spatio-temporal data grow exponentially. Therefore, all techniques should be exploited to keep the data volume low. Cells can be used as much more general building blocks than simplexes. The advantage of cell is that every point, line, polygon and volume can be topologically mapped to a cell of respective dimension through a homeomorphism (Kainz, 1995). A bijective, continuous mapping with continuous inverse between two spaces is called a homeomorphism.

2.4.1.2.2 Cell and cell complex

An (open) \( n \)-cell is a topological space homeomorphic to an open ball \( E^n \) of \( R^n \). Simply, a cell is a space homeomorphic to a disk. Every \( n \)-simplex is an \( n \)-cell, but the converse is not true. A 0-cell is a node, a 1-cell is an arc, a 2-cell is an area and a 3-cell is a volume (Figure 2.9).

In Figure 2.9, every 1-cell is bounded by two 0-cells and every 2-cell is bounded by a closed cycle of 0- and 1-cells. Orientation in the cells is maintained in the same fashion as in a simplex.

Cell complex

A finite collection \( K \) of cells is a cell complex if (Kainz, 1989, 1995 and Corbett, 1979):

1- Different elements of \( K \) (cell complex) have a disjoint interior.
2- For each \( n \) (\( n \)-cell) in \( K \) (cell complex) the boundary of \( n \) is a union of elements of \( K \).
3- If \( n, m \in K \), and \( n \cap m \neq \emptyset \), then \( n \cap m \) is a cell, and is a union of elements of \( K \).

A cell complex in Figure 2.9 has a disjoint interior of cells (0-cell, 1-cell and 2-cell). The proof of these conditions can be found in (Corbett, 1979 and Giblin, 1977). The advantage of cell complexes lies in the general shapes of its cells, which permit spatial data to be treated as a cell complex. Cell complexes are a generalization of simplicial complexes. Therefore, cells are the most generalized form representing the geometry of an object.

Cells are natural cartographic objects and may be manipulated algebraically (Corbett, 1979). The calculations with cells are easier than with simplices because they are less in number (Giblin, 1977) and fewer cells are required to represent spatial objects (Pigot, 1995). But technically, the definitions of the cell complex are not as simple as those of the simplicial complex (Giblin, 1977).

We can formally define the topological consistency rules in a cell complex.

1- Every 1-cell is bounded by two 0-cells.
2- For every 1-cell, there are two 2-cells (left and right polygon).
3- Every 2-cell is bounded by a closed cycle of 0- and 1-cells.
4- 1-cells intersect only in 0-cells.
5- Every 0-cell is surrounded by a closed cycle of 1- and 2-cells.
Therefore, cells inherently carry geometric constraints in their structure. These constraints may be used to test the consistency of the model, in terms of properties of individual cells and in terms of the global relational scheme (Corbett, 1979). For example:

- Rule 1 ensures that every arc is bounded by one start and one end node.
- Rule 2 preserves the left and right area of each arc for a given orientation.
- Rule 3 guarantees the closedness of area features.
- Rule 4 enforces the planar topology.
- All five rules are obligatory for the partition of space into area features.
- Rule 1 is necessary for network analysis.


Simplices, cells and their complexes are simple kinds of spaces that serve as the topological equivalent of more complicated subsets of Euclidean space and topological relations can also be embedded in the structure of complexes. 2-D cell and simplicial complexes have an additional advantage, i.e., the Euler equality can be used to check the invariant relationships between the 0-, 1- and 2-cells or simplices of a complex.

In cell complex structures, spatial relations can be classified into two major groups (Pigot, 1995):

- *intra* cell complex, i.e., relations between the cells in the cell complex and
- *inter* cell complex, i.e., relations between different cell complexes.

*Intra* cell complex relations can be described using boundary and co-boundary relations and are briefly discussed in the following section, while *inter* cell complex relations are discussed in § 2.4.2 (spatial operations / relations).

### 2.4.1.2.3 Boundary and co-boundary

The boundary of an $n$-cell are the $(n-1)$ cells incident with it. The co-boundary of an $n$-cell are the $(n+1)$ cells incident with the $n$-cell. The boundary and co-boundary relations capture two types of topological relationships, i.e., adjacency and containment, known as the topology of a spatial object (cell) and first introduced by Corbett (1985, 1979) and White (1978), as cited by Pigot (1995). For example, the boundary of 1-cell \((a_1)\) is 0-cells \(<n_1,n_2>\) and the co-boundary of 1-cell \((a_2)\) is 2-cells \(<A_1,A_2>\) (Figure 2.10).

#### Figure 2.10 Boundary and co-boundary relations in a cell complex.

### 2.4.2 Spatial operations

The spatial relations refer to the relations between spatial objects (simple or complex) and the geometric elements of a spatial object. These relations can be described using a quantitative and qualitative approach. They can therefore, be measured through two major approaches based on:

- geo-metric information and
non-metric information.

Geo-metric uses Cartesian coordinates to measure relations such as distance between two objects. The non-metric approach is based on non-geometric information and captures the qualitative properties of space, such as a connection between two objects. The first approach is called the metric approach and the second is called the topological approach. The topological approach addresses the topological relations, i.e., orientation, connectivity, containment and adjacency.

Kainz (1989) provided an overview of the mathematical foundations of spatial relations and the new applications of order structures to GIS-related problems. The mathematical concepts of topology, order theory and metric space are useful to formulate the spatial relations between real-world objects. Relations of location, neighborhood and inclusion can be defined by these mathematical disciplines. Treating these spatial relations mathematically will provide better understanding of what is achievable with GIS and how it can be designed (Kainz, 1989). Different sets of relations may be defined in different types of space; for example, the metric relation 'distance' is defined in metric space and topological space is based on topological relationships. Therefore, spatial relations (operations) could be categorized as shown in Figure 2.11.

1- spatial metric relations
2- spatial topological relations
3- spatial order relations.

2.4.2.1 Spatial metric relations

Spatial metric relations are defined in metric space. Metric space was first introduced by Frechet (1906) to define the concept of continuous topology. For example, the relation 'distance' between two objects could be defined in metric space by using Euclidean distance. Basic relations regarding distance, direction and location could be addressed using metric space (Figure 2.11).

Metric space

Let M be a non-empty set and \( f : M \times M \rightarrow \mathbb{R} \) a function, the metric on M. \((M, f)\) is called a metric space subject to the following conditions:

1- \( \forall p, q \in M \Rightarrow f(p, q) \geq 0; \) distance from p to q is greater or equal to zero
2- \( \forall p, q \in M \Rightarrow f(p, q) = 0, \) if and only if \( p = q; \) distance from p to q is zero, if \( p \) equal to \( q \)
3- \( \forall p, q \in M \Rightarrow f(p, q) = f(q, p); \) distance from p to q equal to distance from q to p
4- \( \forall p, q, r \in M \Rightarrow f(p, q) + f(q, r) \geq f(p, r); \) distance from p to q plus distance from q to r is greater than or equal to the distance from p to r.

where \( \mathbb{R} \) is the set of all real numbers. \( \mathbb{R}^n \) is a Cartesian n-space. \( f \) is called the distance function for \( M \) and \( M \) could be referred to as metric space (Moise, 1977). Any function that satisfies these axioms...
Chapter-2: Concepts of Space and Time

is a metric. We can also define an open disk using a metric space. An open disk of radius $r$ around $x$ of $M$ is defined as

$$K(x, r) = \{ y \in M | f(x, y) < r \}$$

A set $N$ is called an open neighborhood of a point $x$ of $M$, if an open disk $K(x, r)$ exists around $x$ such that $K(x, r) \subseteq N$. A set $N \subseteq M$ is called an open set if $N$ is an open neighborhood of each of its points.

A (closed) $k$-manifold is a topological space $T$ in which every point $t \in T$ has a neighborhood homeomorphic to the $k$-dimensional disk.

There are many metrics that can be used in GIS to express the distances. For example, distance calculated by Euclidean distance is the shortest distance between two points. In Euclidean space, we can define the distance as

$$f((x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n)) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

The metric topology generated by the usual metric on any subset of $\mathbb{R}^n$ is called the usual topology. However, topology derived from metric space is conceptually doubtful and leads to an implementation problem in the computer due to the finiteness of the number system (Egenhofer and Herring, 1990) and with metric space and analytical geometry, the relations of neighborhood cannot be addressed. Therefore, topological space can be used to introduce the concept of neighborhood. It is more appropriate given the current nature of computer technology, which is based on the arithmetic computation of a finite number system. As topology is a study of continuity, current computer technology cannot be used to completely define the continuity based on Euclidean distance, and results in the deduction of inconsistent topology. For example, to conclude that a point lies to one side of line might not be true after scaling or rotation (Franklin, 1984 as cited by Frank and Kuhn, 1986).

2.4.2.2 Spatial topological relations

Topological relations are defined in topological space. Topological spaces were first introduced by Hausdorff in 1914 and are based on the concept of ‘open set’ (Kainz, 1995) or neighborhood. The purpose of introducing the topology is to define any continuous function without mentioning distance (Willard, 1970). Formally, we can define the topological space (open set) as follows:

**Topological space**

A topology on a set $T$ is a collection of $\tau$ subsets of $T$, called the open sets. This topology should satisfy the following assertions:

1- $\forall x \subset \tau \Rightarrow \bigcup_{o \in x} o \in \tau$, i.e., any union of elements of $\tau$ belongs to $\tau$

2- $\forall O_1, O_2 \in \tau \Rightarrow O_1 \cap O_2 \in \tau$, i.e., any finite intersection of elements of $\tau$ belongs to $\tau$

3- $\emptyset \in \tau, T \in \tau$, i.e., $\emptyset$ and $\tau$ belongs $\tau$

$(T, \tau)$ is called a topological space or simply $T$ is a topological space. $\tau$ is called topology on $T$. The elements of $\tau$ are called open sets, those of $T$ are called points. Succinctly, we may say that, given a set of points, the topology is defined by associating a set of subsets with the set. The original set and subsets form a topological space. These subsets are closed because union and intersection of any two are also an element of the set (Hanrahan, 1985). Every metric space is a topological space (Kainz, 1995). Topological space can also be defined using neighborhoods instead of open sets.

26
Chapter-2: Concepts of Space and Time

A transformation is a mapping of all the points in the space to a new set of points. A transformation is said to be a continuous function if it preserves the neighborhood sets of a topological space. A homeomorphism is a bijective and bicontinuous function. A function is called bijective and bicontinuous if the function and its inverse are unique and continuous. Homeomorphism is the most general transformation that preserves topological properties. Two objects are topologically equivalent if a homeomorphism exists between them. Map projections such as Mercator, Transverse Mercator, Polyconic, and similar are examples of homeomorphism (Corbett, 1979). Other examples include the correction of distortion resulting from paper/film shrinking or enlargement in digitizing.

Topology

Topology is the study of certain invariants of structured spaces. Structuring of spaces can be performed through:

- generalized notion of distance (metric space)
- abstract notion of neighborhood and
- pasting together elementary objects (simplices or cells) to complexes (simplicial or cell complex).

Spatial topology is a coordinate-free geometry (Herring, 1987) or topology is simply a ‘most general’ geometry. It is a general study of continuity (Frank and Kuhn, 1986). Topology investigates the characteristics of geometry that remain invariant under certain transformations (topological mapping or homeomorphisms). Under such transformations, the following relations remain unchanged:

- intersections, e.g., intersection of two 1-cells in a 0-cell
- incidences, e.g., two 0-cells for a 1-cell
- adjacencies, e.g., neighbors of 2-cell
- inclusions, e.g., 0-cell in a 2-cell.

Therefore, topological mappings are elastic transformations that keep topological incidence relations unchanged (Kainz, 1995). Examples of such relations are scale, rotation and shift, which result in unchanged connectivity and neighborhood (Figure 2.12).

Spatial topology is a power tool for spatial analysis and the backbone of any spatial information system. Because of this analytical power (topological relations), GISs are different from any CAD or CAM systems. Spatial topology investigates the topological structures of point-sets (Frank and Kuhn, 1986). A modern spatial data handling system (or GIS) exploits the two major branches of topology, i.e., point-set (analytical) and algebraic (combinatorial or geometric) topology (Figure 2.13).

Analytical topology is independent of any algebraic structure of point-sets. It focuses on sets of points and is based on real analysis, using concepts such as open sets, neighborhood and convergence (Frank and Kuhn, 1986). It uses the set-theoretical concepts like equal, joint, disjoint, boundary, inside, overlap etc., to define the spatial relations.
Algebraic topology uses algebraic means to describe the spatial relations. The algebraic approach is based on the concept of a finite number of points (nodes) and their connecting lines (edges), for example, simplicial and cell complexes, orientable closed surfaces and graph theory. It addresses the relations concerning neighborhoods (Kainz, 1995; Frank and Kuhn, 1986; Herring, 1987; Corbett, 1979 and Hanrahan, 1985).

### 2.4.2.2.1 Topological relations

**Point-set approach**

Topology describes the relationships between objects and their neighbors. The point-set approach is the most general model for topological spatial relations. Other methods, such as intervals or simplicial complexes, are generalized by this point-set approach (Egenhofer and Franzosa, 1991). Topological concepts such as continuity, boundary, closure and interior can be defined in terms of neighborhood relations. The basic topological properties of a point-set are based on neighborhoods. A point is in the complement ($A^{-1}$) of set $A$ if it is not in $A$. A point is said to be in the interior ($°$) of $A$ if $A$ is a neighborhood of the point. A point is said to be in the closure ($ζ$) of $A$ if every neighborhood of the point intersects $A$. A point is on a boundary ($∂$) of $A$ if it lies in the intersection of the closure of $A$ and the closure of the complement of $A$.

If $A$ is a subset (or object) of a topological space, then using point-set topological notions three fundamental primitives of topology could be defined as (Figure 2.14):

- **Closure ($ζ$):** intersection of all closed sets that contain $A$
- **Interior ($°$):** union of all open sets that are contained in $A$
- **Boundary ($∂$):** intersection of $ζA$ and $ζ(A^{-1})$.

The binary topological relations between two objects (regions), $A$ and $B$, in $R^2$ can be defined in terms of intersection (set theoretical operation) of $A$’s closure ($ζA$), interior ($°A$) and boundary ($∂A$) with $B$’s closure ($ζB$), interior ($°B$) and boundary ($∂B$). For example:

- if the boundary of object $A$ intersects with the boundary of $B$, then the relation is meet;
- if the boundary and interior of $A$ and boundary and interior of $B$ yield a non-empty intersection, then the relation between $A$ and $B$ is overlap; and
- if the intersection between the three parts of the object is empty, then the relation is disjoint.

To derive all possible topological relations between two objects, Egenhofer et al. (1993) introduced 9-intersections between six object parts. Each of the three object parts, i.e., closure, interior and boundary, is non-empty and connected. This 9-intersection model, represented by a 3x3 matrix describes topological relations between two objects in $R^2$. 

---

**Figure 2.13 Two branches of topology.**

**Figure 2.14 Three topological primitives of a spatial object.**
The 9-intersection model is the extension of the 4-intersection model of Egenhofer and Franzosa (1991). The 4-intersection model considers intersection between the interior and boundary of two spatial objects, while the 9-intersection model considers intersection of the interior, closure and boundary of two spatial objects.

The content invariant of 9-intersections is characterized by empty (⊘) and non-empty (¬⊘) values. On the basis of empty and non-empty values, \(2^9 = 512\) different binary topological relations can be derived between \(A\) and \(B\). Not all 512 mutually exclusive binary relations are topologically valid. Only eight relations make sense for two objects in 2-D space or two lines in 1-D space (Egenhofer and Sharma, 1993b; and Kainz, 1995). Like the 4-intersection model, the same number of relations (eight) makes sense in the 9-intersection model (Figure 2.15). No additional relations due to the consideration of closure intersections are possible (Egenhofer et al., 1993). However, the 9-intersection model with content invariant provides more detail than the 4-intersection model. Details can be found in Egenhofer et al. (1993).

### Table 2.15

<table>
<thead>
<tr>
<th>Set Theoretical Relations</th>
<th>9-Intersection</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disjoint</td>
<td>⊘</td>
<td><img src="#" alt="Disjoint Illustration" /></td>
</tr>
<tr>
<td>Contains</td>
<td>¬⊘</td>
<td><img src="#" alt="Contains Illustration" /></td>
</tr>
<tr>
<td>Inside</td>
<td>¬⊘</td>
<td><img src="#" alt="Inside Illustration" /></td>
</tr>
<tr>
<td>Equal</td>
<td>¬⊘</td>
<td><img src="#" alt="Equal Illustration" /></td>
</tr>
<tr>
<td>Meet</td>
<td>¬⊘</td>
<td><img src="#" alt="Meet Illustration" /></td>
</tr>
<tr>
<td>Covers</td>
<td>¬⊘</td>
<td><img src="#" alt="Covers Illustration" /></td>
</tr>
<tr>
<td>CoveredBy</td>
<td>¬⊘</td>
<td><img src="#" alt="CoveredBy Illustration" /></td>
</tr>
<tr>
<td>Overlap</td>
<td>¬⊘</td>
<td><img src="#" alt="Overlap Illustration" /></td>
</tr>
</tbody>
</table>

*Figure 2.15 Spatial topological relations between two objects in \(R^2\).*


### Analytical approach (graphs)

The analytical approach of topology is based on graph theory. The founder of the systematic study of topology is the mathematician Leonard Euler, who in 1736 solved the famous (Königsberg bridge) problem of that time, using the graph theoretical model. Therefore, topology is rooted in graph theory, which investigates the problems that can be formulated by a set of nodes and the connections between
them, i.e., edges (Kainz, 1995). In modern computer science, graph theory has a strong influence on data structuring, data modeling, engineering applications and computer implementation.

A graph is a finite set of points in space. Such a point is called a vertex (node) of the graph. An edge of a graph is the connected pair of vertices. Mathematically, a graph $G$ could be defined by a set $N(G)$ of nodes and a set $E(G)$ of edges, where each edge is the connection between two points of $G$ (Kirschenhofer, 1995; Giblin, 1977).

$$G = \{N, E\}$$

$N = \{n_1, n_2, n_3, \ldots, n_N\}$ and

$$E = \{e_1, e_2, e_3, \ldots, e_N\}$$

Separate single edges or disconnected sets of edges are called subgraph. In the case of a planar graph, all crossing (intersection) of edges occurs in the plane; in the non-planar case, edges may cross without creating intersection. Both are applicable in different disciplines. In the case of utility networks and ground transportation (over/underpasses), edges do not need to be connected. The water supply line could not be intersected by power or gas lines. A planar graph is topologically equivalent to a 1D cell complex.

A graph can be directed or undirected (Figure 2.16a and Figure 2.16b). If an edge is connected by an identical node, then the graph is called a loop (Figure 2.16c). A graph without any loop is called a simple graph. A path or chain is a walk where all nodes are different. If the first and last nodes are the same in a path, then the walk is cyclic. In Figure 2.16d, the path $(e_1, e_7, e_8, e_9)$ is cyclic, while the path $(e_2, e_6, e_7)$ is only a path.

The sequence of the nodes is of importance, as $(n_1, n_2) \neq (n_2, n_1)$. This direction of edge is vital for establishing the topological relations. It maintains the adjacency relation and left and right polygon topology. Two edges are adjacent if, and only if, they have a common node. It is further helpful to describe the structure of planar graphs and their faces by means of cell complexes. An undirected connected graph $G$ which contains no cycles is called a tree. A binary tree is an example of a tree graph (Figure 2.16e). The degree $d(N)$ of a node $N$ in a graph $G$ is the number of vertices connected at node $N$. In Figure 2.16d, $d(n_1) = 2$ and $d(n_2) = 3$ for the nodes $n_1$ and $n_2$, respectively. A graph $G$ is called complete if every node $N$ of the graph has the same degree $d(N)$. A polygon is a closed chain where $d(n) \geq 2$ for all nodes.

A graph has various properties; some graph examples are illustrated in Figure 2.17. One of the graph's special properties is that it has a fixed relationship between its number of nodes, number of edges and number of faces. In a two-dimensional configuration this could be proved mathematically using Euler's formula. Euler's equation also checks the planarity of a graph. It is important to check the internal geometric consistency in 2D spatial models; for example, an edge must have two different nodes and a triangle must have three edges.

Euler’s Formula:
\[ N + F = E + 2 \]  \[ \text{[1]} \]

where

- \( N \) = Number of nodes
- \( F \) = Number of faces
- \( E \) = Number of edges and
- \( 2 \) = Equation constant (if outer region is considered as a face, if not then 1).

Initially, Euler’s equation-[1] is only valid for connected graphs. However, for non-connected graphs this equation could be modified as

\[ N + F = E + (c+1) \]  \[ \text{[2]} \]

where

- \( c \) = Number of unconnected subgraphs.

Note that if the counting of the outer region is taken as a face, then the constant is \((c+1)\), otherwise this constant is \((c)\).

Figure 2.16 and Figure 2.17 could be verified for geometric correction using Euler’s equation:

Geometry verification of Figure 2.16:
- (a): \( N = 2, F = 1, E = 1, c = 1 \) \( \Rightarrow \) \( 2 + 1 = 1 + (1 + 1) = 3 \)
- (c): \( N = 1, F = 2, E = 1, c = 1, \) \( \Rightarrow \) \( 1 + 2 = 1 + (1 + 1) = 3 \)
- (d): \( N = 7, F = 4, E = 9, c = 1, \) \( \Rightarrow \) \( 7 + 4 = 9 + (1 + 1) = 11 \)

Geometry verification of Figure 2.17:
- (a): \( N = 8, F = 1, E = 6, c = 2 \) \( \Rightarrow \) \( 8 + 1 = 6 + (2 + 1) = 9 \)
- (b): \( N = 6, F = 3, E = 7, c = 1 \) \( \Rightarrow \) \( 6 + 3 = 7 + (1 + 1) = 9 \)
- (c): \( N = 6, F = 3, E = 6, c = 2 \) \( \Rightarrow \) \( 6 + 3 = 6 + (2 + 1) = 9 \)
- (d): \( N = 7, F = 2, E = 5, c = 3 \) \( \Rightarrow \) \( 7 + 2 = 5 + (3 + 1) = 9 \)

Although Euler’s equation provides an internal consistency check for the topology, it could be misleading when the numbers compensate the miscount.

Graphs can be used to build chains and polygons, but they do not directly describe the geometric and topological structure in 2D plane, and the concept of faces needs to be added. Spatial data modeling often use planar graphs with nodes, edges and faces, although faces do not directly belong to graphs (Kainz, 1995). However, most of the current commercial GIS use this approach, known as the arc-node approach. The concepts of cell complexes have been proposed by many researchers to address...
more complicated spatial objects in terms of geometry, shape and topology (Corbett, 1979, 1985; Frank and Kuhn, 1986; Giblin, 1977).

2.4.2.3 Spatial order relations

Spatial order relations pertain to the concept of comparing two or more spatial objects. Order relations are based on well-established mathematical theories. Order relations in GIS are important as they represent the concept of inclusion and containment, which is difficult to answer through metric and topological concepts. For example, if we take two regions, Europe and Austria (Figure 2.18), based on spatially ordered concepts Austria is contained in Europe (Kainz, 1989). Based on ordered sets, two types of relations can be distinguished: 1] totally ordered relations and 2] partially ordered sets (poset) or relations. A set is said to be partially ordered when an order relation is defined between its members, making them comparable.

2.4.2.3.1 Totally ordered relations

A set $T$ with a binary relation $<$ is called a totally ordered set $(T, <)$, if for all $x, y, z$ in $T$.

\[
(x < y) \implies \neg (y < x): \text{ asymmetric}
\]

\[(x < y) \land (y < z) \implies (x < z): \text{ transitive}
\]

A totally ordered relation establishes a hierarchical structure where any two objects can be compared with each other. A man-made spatial structure often follows the total order, i.e., continents are divided into countries, and countries into cities, and so on (Figure 2.18). But sometimes this strict order does not exist for natural or man-made subdivisions when boundaries of spatial regions overlap. For example, consider two crop zones: A (potato) and B (onion), and fields a, b and c in these zones (Figure 2.19a). Figure 2.19b clearly demonstrates that the hierarchical structure is not suitable for such a situation because it models part of the information. Field b can belong to either zone A or B, although field b is a part of zone A and B. Queries such as:

“What can be grown in field b?”
“Where can potato and onion be grown?”

cannot be answered in a totally ordered set. Partially ordered relations or sets (posets) overcome this problem.
A set is called partially ordered when an order relation is defined between its elements. A set \( P \) with a binary relation \( \leq \) is called a poset \((P, \leq)\), if all \( x, y, z \) are in \( P \)

\[
\begin{align*}
    x \leq y &: \text{ reflexive} \\
    x \leq y \text{ and } y \leq x \Rightarrow x = y &: \text{ anti-symmetric} \\
    x \leq y \text{ and } y \leq z \Rightarrow x \leq z &: \text{ transitive}
\end{align*}
\]

Therefore, a set \( P \) with reflexive, anti-symmetric and transitive relations (partial order relations) \( \leq \) is called a partially ordered set or poset \((P, \leq)\). Both totally ordered and poset offer inverse relations.

The relation \( x \leq y \) conversely implies that \( y \geq x \). Relation \( \leq \) means 'is contained in' while \( \geq \) means 'contains'. For every poset \( P \), we can define a dual of \( P \); for example the relation \( x \leq y \) is a dual of \( y \geq x \) (Kainz, 1989). Each finite poset can be represented by a ‘Hasse’ diagram based on \( x \leq y \) relation (x covers y), i.e., y is greater than x and no element exists between x and y. A set of all the elements that cover x is called cover of x (\( x_\uparrow \)) and, conversely, a set of all elements that are covered by x is called co-cover of x (\( x_\downarrow \)). Figure 2.20 illustrates the poset of Figure 2.19a.

**Upper and lower bounds**

If \( P \) is a poset and \( S \subseteq P \) a subset of the poset, then an element \( x \in P \) is an upper bound of \( S \) if \( S \leq x \), for all \( s \in S \). A lower bound can be defined using duality.

\[
\begin{align*}
    S^\uparrow = S \text{ upper is the set of all upper bounds of } S. \\
    S^\downarrow = S \text{ lower is the set of all lower bounds of } S.
\end{align*}
\]

The upper bounds of \( \{b\} \) in Figure 2.20a are \( \{A,B\} \), while \( \{b\} \) is the lower bound for \( \{b\} \). The upper and lower bounds of \( \{E, F\} \) in Figure 2.20b are \( \{A,B,C,D\} \) and \( \{G\} \), respectively. If \( S^\uparrow \) has a least element, then it is called the least upper bound (LUB) of \( S \). If \( S^\downarrow \) has a greatest element, then it is called greatest lower bound (GLB) of \( S \). We denote:

\[
\begin{align*}
    x \lor y &= \text{LUB (supremum or join) of two elements } x \text{ and } y \\
    x \land y &= \text{GLB (infimum or meet) of two elements } x \text{ and } y.
\end{align*}
\]
Chapter 2: Concepts of Space and Time

An LUB or GLB is always unique. In Figure 2.20b, the LUB of {E,F} is {D} and GLB of {B,C} is {D}. Various operations can be performed on a poset:
- cover / co-cover
- upper bound / lower bound
- least upper bound / greatest lower bound.

These operations can be exploited to extract spatial relations (topology). Kainz (1995, 1990, 1989), Kainz et al. (1993) and Greasley (1990) have demonstrated how these relations can be used to express topological relations in GIS, for example:
- intersection
- boundary
- touch
- containment
- neighborhood.

The concept of poset is useful where the spatial hierarchical structure is insufficient; however, it lacks the capability to answer all queries regarding the spatial regions. Unfortunately, there is not always a greatest lower bound (GLB) or a least upper bound (LUB) in a poset (Kainz, 1990, 1989). Therefore, a more specific structure is needed. This specific structure is called a lattice.

2.4.2.3.3 Lattice

A lattice \((L, \leq)\) is a special poset that for every two elements always has GLB and LUB. A complete lattice \((L, \leq)\) is a poset in which every subset of the poset has a GLB and an LUB. All the lattices are posets but the converse is not true (Figure 2.21). Every poset can be transformed into a lattice by adding some elements to the poset. The normal completion process is to find the smallest number of elements necessary to add to a poset to create a lattice, i.e., to build the minimal containing lattice of a poset. Figure 2.21(a) is a poset but not a lattice. Through normal completion (by introducing U, X and E), we can achieve a lattice (Figure 2.21b).

The application of posets and lattices, particularly in spatial data handling, are important and can be found in (Kainz, 1989, 1990, 1995; Kainz et al., 1993; Greasley, 1990; Perry, 1990). Kainz et al. (1993) applied the theory of simplicial complexes to derive the spatial topologies.
2.4.3 Temporal component

The temporal component is associated with time as a temporal object and relations between it, i.e., temporal operations (Figure 2.22). We have already discussed some problems of interpreting time (§ 2.3.4). Here we shall briefly highlight some associated terms and some more issues of time from the structural, conceptual and representation perspective. Different disciplines demand a distinct structure of time; in the following sections, examples are cited where it can be applied.

2.4.3.1 Structural perspective

The structural issues are associated with the mathematical modeling of time. Three time models have dominated in temporal modeling: linear time, branching time, and cyclic time (Figure 2.23). A ‘time model’ is a mathematical abstraction that can be expressed by axioms (Barrera et al., 1991). In the linear time model (single axes), time advances from past to future in a totally ordered fashion. In the branching model (multiple axes), time advances from past to present, where it then splits into several time lines; or it may have many pasts and a single future. The linear time model is similar to the concepts of totally ordered sets, and the branching model is comparable to partially ordered sets in mathematics. In the cyclic time model, time repeats itself after some time.

In most applications, particularly GIS and DBMS, time is considered linear (Al-Taha, 1992). Snodgrass (1992) argued that the linear time model is quite adequate in spatial modeling. However, it is claimed here that all three time models are needed in any generic TGIS. Monitoring land use change, cadastral systems, land registration systems, building control systems etc. are examples where linear time is quite suitable. Langran (1992a), Hermosilla (1994) and Raza (1996) cited some examples of applications in TGIS. Applications such as an electronic navigation chart, urban ecology management, urban planning, urban infrastructure management, global change management, transportation, map and chart production etc. are bound to make forecasts to see the impact of change. Therefore, TGIS should have a capability to handle many scenarios simultaneously. In these applications, branching time is an appropriate approach (Hermosilla, 1994). Joerin and Claramunt (1994) implemented a cyclic time in the application of ‘flood assessment impacts on agricultural land’. In these time models, time may be measured as discrete, continuous, bounded, unbounded, dense and complete. Only discrete and
Chapter-2: Concepts of Space and Time

Continuous time is discussed here; for further details on bounded, unbounded, dense and complete, users may refer to Jensen (1994), Al-Taha (1992) and Dutta (1991).

2.4.3.2 Discrete time

There is a finite number of points between any two points, and points are isomorphic to natural numbers, i.e., there is the notion that every point has a unique successor. Time is considered as a sequence of event-punctuated intervals, and time's properties are the consequences of a set of axioms (Barrera et al., 1991). This measurement of time is useful in applications that are discrete in nature, such as cadastral, land use, building control, administrative boundaries, election zones, planning zones etc. Changes in land use boundaries, for example, do not occur continuously; it does not make sense to say that an intermediate boundary exists between T₁ and T₂. This type of measurement is change-dependent, i.e., when change occurs in the real world, it is triggered by the next tick of clock.

2.4.3.3 Continuous time

There are an infinite number of points between two points, and points are isomorphic to the real numbers. Continuous time is used to measure the changes which are occurring continuously, and require an arbitrary level of precision for measurements, for example, global change management, electric navigation charts etc. (Barrera et al., 1991; Worboys, 1995). This measurement is independent of change; whether change occurs in the real world or not, the database keeps recording the state regularly.

However, there have been arguments about the implementation of these two types of time in computer systems. Due to the digital nature of computer systems, a continuous phenomenon has to be treated in a discrete fashion. Therefore, while modeling time in spatial information systems, it is less likely to be implemented as a continuous variable. Several practical arguments have been given in favor of the discrete time model over the continuous time model (Snodgrass, 1992); measuring time is inherently imprecise and most natural language references to time are compatible with a discrete model. Whether time is contemplated as a discrete or continuous variable, the issue of the representation of time, type of time, time granularity (temporal resolution), calendrical system and time dimension has a prime impact on the performance of spatio-temporal data modeling and system performance.

2.4.3.4 Representation of time

Time may be represented as a Point (instant), Interval (duration), or Span. Point time holds a single value, interval pairs of values or time between two instants (points) and span is a special type of interval. For example, point time recorded is only the time the event occurred (12/01/96 or 1998) and interval reflects the lifetime or duration of an event represented by a start and end instant (10/01/96 to 10/06/96 or 1990 to 2000). Span time representation uses the duration without start and end instant, i.e., 10 days or five years. Therefore, the amount of time length is known. This length can be fixed (one week or one hour) or variable (one month or one year). A span can be positive or negative, i.e., one week earlier or one week later. An interval is of anchored duration, while a span is of unanchored duration (Bagg and Ryan, 1996). Easterfield et al., (1991) termed point and interval types of representation as implied time and fixed time, and proposed a third, i.e., augmenting either of the two with some form of temporal index, similar to the spatial index. Data are broadly associated with interval time in two ways: valid throughout the interval, i.e., land use of parcel ‘X’ was commercial during 1990 and 1995; and valid for some time, i.e., land use of parcel ‘X’ was commercial sometime during 1990 and 1995. The point and interval representation form are slightly different. The choice depends upon the particular application and trade-off between volume of data and processing speed. Most temporal models in theoretical computer science and artificial intelligence adopted the point-based approach. Gadia (1986) proposed the interval-based time approach, Allen (1984) introduced the interval-based approach for deriving temporal relations in temporal logic, Loucopoulos et al. (1991)
proposed the interval-based approach for effective information systems, Easterfield et al. (1991) suggested the interval representation, while Dutta (1991) considered the point-based approach for topological reasoning. Both approaches have their own advantages and disadvantages, but they are equal in representation power (Tsang, 1983 and Ladkin, 1987; as cited by Dutta, 1991).

### 2.4.3.5 Type of time

Time can be portrayed as **Absolute** time or **Relative** time. Absolute time indicates that a specific world-time (§ 2.4.3.8) at a given time-stamp granularity is associated with a fact. For example, “John was born on 5th January, 1965”. Relative time indicates that the world time of a fact is related to either the valid time of another fact or the current time (now). Relative time can be qualitative or quantitative, i.e., ‘after’, ‘before’, ‘during’ etc. and ‘two years ago’, ‘ten hours late’ etc. are examples of qualitative- and quantitative-relative time, respectively (Jensen, 1994).

### 2.4.3.6 Chronon and time granularity

The non-decomposable unit of time is known as the chronon. A one-dimensional chronon is the shortest time that can be represented in a data model. It is not a point, but a line fragment on the time line (Snodgrass, 1992). Time granularity refers to the basic unit of time. This temporal resolution is like a spatial resolution. A map of 1:25,000 scale means 25 m on the ground equals 1 mm on the map; similarly 1:1440 temporal-scale means the ratio of 1 minute to 24 hours. If the land use (general) is recorded every five years, then the annual land use changes cannot be detected, which means that some information will be lost. On the other hand recording land use every month would result in wasted computer memory. Time granularity varies from one application to another. In banking, building permit and cadastral systems, transactions can be associated with day granularity, but representation of general land use changes may require 'years' as a unit of time.

### 2.4.3.7 Calendars

Calendric systems are the collection of calendars and operators such as meets, overlap, after etc. We are using many calendar systems, i.e., the Islamic calendar (consists of months in relation to the appearance of the moon), or the Chinese calendar, the Greek calendar, the academic calendar (consisting of semesters), the fiscal calendar, the Gregorian or Julian calendar etc. Usage depends on the application and culture. Applications relating to global change management or electronic navigation etc., demand knowledge of different days and hours at different locations on the earth. Generally, the Gregorian calendar is used in DBMSs, which is composed of 12 months and 12 months form a year. The time granularity is represented by seconds, minutes, hours, days, weeks, months etc. Bagg and Ryan (1996) proposed user-defined calendars in TGIS. Therefore, any generic TGIS should support multiple calendars, and functions should be provided to convert time between these calendars.

### 2.4.3.8 Dimensions of the time

Barrera et al. (1991) termed this ‘types of time’. Unlike the spatial dimension (limited to 3D), time is multidimensional. Some authors considered time as a single dimension (Lum et al., 1984), but others considered it as multidimensional (Snodgrass, 1992). The time dimension is associated with the occurrence of real world events: when the event occurred, when it was observed, when it was measured, when it was reported, when it was keyed into the database, when it was processed etc. In a land use application, in the real world land use might change at time $T_1$, be surveyed (by fieldwork, or by remote sensing) at time $T_2$, processed (pre-digitization) at time $T_3$, digitized at time $T_4$, processed and analyzed (post-digitization) at time $T_5$ etc. The required number of dimensions are determined by the users of a particular application. But careful examination of these dimensions is important because spatio-temporal data models are already complex; adding more time dimensions may increase complexity and may negatively effect the system’s performance. Historically, two time dimensions have been suggested in spatial and non-spatial data models, world time (WT) and database time.
(DBT). Some authors use valid and transaction time (Snodgrass, 1992), some physical and logical time (Lum et al., 1984). WT refers to the time the event actually occurred in the real world and DBT refers to the time at which this was recorded in the database.

Based on these two distinct time dimensions, databases have been classified into four categories as presented in Table 2.4:

- snapshot database: supports no time
- rollback database: only DBT is incorporated
- historic database: only WT is incorporated
- temporal database: both WT and DBT are incorporated.

The two time dimensions may relate to one another in three different ways (Figure 2.24). In a perfect system where events and states are monitored in real time (WT = DBT), there is no need to refer to DBT (Sarda, 1993).

2.4.3.9 Indeterminacy

Information that is time-indeterminate may be characterized as ‘do not know exactly when’ information (Jensen, 1994). It is known that an event occurred, but not when; then the event is historically indeterminate (Snodgrass, 1992). This type of indeterminacy is possible in the case of:

- finer system granularity (e.g., when database time granularity may be in seconds while world time granularity is in days),
- uncertainty in urban planning (e.g., the low-cost housing project will be finished within six months),
- unknown event time (e.g., last year, land use of parcel ‘X’ changed to residential from commercial. It is known that the land use has changed but the time is not known.)

etc. Mostly time indeterminacy occurs in world time. Indeterminacy in database time is rare as database time is almost always known, i.e., when a transaction takes place.

2.4.4 Temporal operations

Temporal operations refer to temporal relations. Temporal operations are isomorphic to the spatial relations. These relations could be defined as using metric, topological and order theory concepts. For example, ‘two hours’ is a metric relationship, ‘one hour later’ is a topological temporal relationship and ‘four weeks in a month’ is an order relationship (Worboys, 1995). In reality STAOs start their life (are born) at Tᵣ, exist for a period of time (life span or duration) Tₑ and cease to exist (die or finish) at Tᵣ. But during their life they may perform various activities, i.e., spatially they may move, change
size or change shape, and aspatially they may change attribute(s). Fundamentally, in OO concepts, an object can be born, die, be reincarnated or go through various mutations (versions). As discussed earlier, a particular discipline decides which part of a spatial or an aspatial component triggers birth, death, reincarnation or mutation of a spatio-temporal object.

Figure 2.25 Temporal existence of a spatio-temporal object in single time dimension.

Simply speaking, we can say that the life of the spatio-temporal object is similar to an edge where two nodes start and end, representing the birth and death of an object, respectively, and that mutations are the number of vertices between two nodes (Figure 2.25).

As discussed in § 2.3.4, the effect of a process is instantaneous, has duration or delay. Temporal events are associated with time points, intervals (state) or disjoint union along the time line (Worboys, 1995). The existence of a STAO may be defined as a point existence, linear existence or disjoint union (Figure 2.26).

Objects whose existence can be expressed as a point (non-extending state) along the time line are known as event or temporal instant. Objects whose existence can be expressed as an interval (extending state) along the time line are known as state or temporal duration (TS, TF). In the case of an event, the object’s times of birth and death are the same and mutation is not possible (TS = TF). In the case of state, the object’s times of birth and death are different, and may have many mutations. While in the case of disjoint union (where the object can be reincarnated) the object has multiple birth and death times, and may have many mutations (TS1, TF1, TS2, TF2). An example of reincarnation of an object is a land use change. A land use reserved for ‘recreational facilities’ at time T1 may have been permitted to be used for ‘car parking’ until time T2 and again be used for ‘recreational facilities’ at time T3 (Figure 2.27).
It is of prime importance at what level of temporal granularity we are observing these temporal activities of an object. An object whose lifetime is an hour may not be represented by a day granularity. Therefore, in this case, a day would be represented by a point time. The existence of an object depends upon the temporal resolution or granularity (Roshannejad, 1996). If the granularity is lower (coarse) than the life of an object, then the life of the object could not be expressed as state. In fact, the issue of representation of objects in temporal dimensions is the same as issues of generalization in spatial dimensions (Worboys, 1995).

There are three types of fundamental relations between event and state. Event-event, event-state and state-state. We consider event as a point and state as an interval. Figure 2.28 portrays point-point and point-interval relations, while an interval-interval relationship is illustrated in Figure 2.30.

Four types of interval can be defined in mathematics:
- unbounded or open both ends \( (T_S, T_F) \)
- left bounded \( [T_S, T_F) \)
- right bounded \( (T_S, T_F] \)
- bounded or closed both ends \( [T_S, T_F] \).

Figure 2.29 is the illustration of these relations. The inclusion of bounded and unbounded intervals has an implicit effect on the interpretation of the existence of an object. If the interval is bounded, then the state of an object shares the same point on the time line. For example, in Figure 2.27, object \( O_1 \) finished at \( T_2 \), but \( T_2 \) is the starting point for object \( O_2 \). Therefore, \( O_1 \) and \( O_2 \) share the same time point \( T_2 \). There is no time gap between the finish \( (T_F) \) of \( O_1 \) and the start \( (T_S) \) of \( O_2 \). Allen (1984) considered right bounded temporal intervals for temporal logic, while Lorentzos (1993) proposed left bounded temporal intervals. Both came up with 13 mutually exclusive temporal relations for interval. Figure 2.30 is the demonstration of 13 temporal relations in 1D space (interval), which are compared.
with eight spatial relations in 2D space (Figure 2.30), proposed by Egenhofer and Franzosa (1991) and presented in Figure 2.15.

\[
\begin{align*}
\text{Unbounded} & \quad T_S \quad T_F \quad (T_S, T_F) \\
\text{Left bounded} & \quad T_S \quad \bullet \quad [T_S, T_F) \\
\text{Right bounded} & \quad \bullet \quad T_F \quad (T_S, T_F] \\
\text{Bounded} & \quad T_S \quad T_F \quad [T_S, T_F]
\end{align*}
\]

**Figure 2.29 Bounded and unbounded temporal intervals.**

<table>
<thead>
<tr>
<th>Spatial Relations</th>
<th>Illustration in 2D Space</th>
<th>Temporal Relations</th>
<th>Illustration in 1D Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Disjoint</td>
<td><img src="image" alt="Disjoint Illustration" /></td>
<td>I₁ Before I₂</td>
<td><img src="image" alt="1D Disjoint Illustration" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Disjoint Illustration" /></td>
<td>I₁ After I₂</td>
<td><img src="image" alt="1D Disjoint Illustration" /></td>
</tr>
<tr>
<td>3 Equal</td>
<td><img src="image" alt="Equal Illustration" /></td>
<td>I₁ Equal I₂</td>
<td><img src="image" alt="1D Equal Illustration" /></td>
</tr>
<tr>
<td>4 Meet</td>
<td><img src="image" alt="Meet Illustration" /></td>
<td>I₁ Meets I₂</td>
<td><img src="image" alt="1D Meet Illustration" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="Meet Illustration" /></td>
<td>I₁ Met I₂</td>
<td><img src="image" alt="1D Meet Illustration" /></td>
</tr>
<tr>
<td>6 Overlap</td>
<td><img src="image" alt="Overlap Illustration" /></td>
<td>I₁ Overlaps I₂</td>
<td><img src="image" alt="1D Overlap Illustration" /></td>
</tr>
<tr>
<td>7</td>
<td><img src="image" alt="Overlap Illustration" /></td>
<td>I₁ Overlapped I₂</td>
<td><img src="image" alt="1D Overlap Illustration" /></td>
</tr>
<tr>
<td>8 Contains</td>
<td><img src="image" alt="Contains Illustration" /></td>
<td>I₁ Covers I₂</td>
<td><img src="image" alt="1D Contains Illustration" /></td>
</tr>
<tr>
<td>9 Inside</td>
<td><img src="image" alt="Inside Illustration" /></td>
<td>I₁ During I₂</td>
<td><img src="image" alt="1D Inside Illustration" /></td>
</tr>
<tr>
<td>10 CoveredBy</td>
<td><img src="image" alt="CoveredBy Illustration" /></td>
<td>I₁ Started I₂</td>
<td><img src="image" alt="1D CoveredBy Illustration" /></td>
</tr>
<tr>
<td>11</td>
<td><img src="image" alt="CoveredBy Illustration" /></td>
<td>I₁ Finishes I₂</td>
<td><img src="image" alt="1D CoveredBy Illustration" /></td>
</tr>
<tr>
<td>12 Covers</td>
<td><img src="image" alt="Covers Illustration" /></td>
<td>I₁ Starts I₂</td>
<td><img src="image" alt="1D Covers Illustration" /></td>
</tr>
<tr>
<td>13</td>
<td><img src="image" alt="Covers Illustration" /></td>
<td>I₁ Finished I₂</td>
<td><img src="image" alt="1D Covers Illustration" /></td>
</tr>
</tbody>
</table>

**Figure 2.30 Thirteen possible temporal relations for bounded interval (adapted from Allen, 1984).**
Chapter-2: Concepts of Space and Time

I \text{ Before } I_2: \text{ Time interval } I_1 \text{ is before } I_2.

I \text{ After } I_2: \text{ Time interval } I_1 \text{ is after } I_2.

I \text{ Equal } I_2: \text{ Time interval } I_1 \text{ and } I_2 \text{ are equal. } I_1 \text{ and } I_2 \text{ start and finish at the same time. Both have complete overlapping.}

I \text{ Meets } I_2: \text{ Finish time of } I_1 \text{ is equal to start time of } I_2. \text{ The two intervals do not overlap, but meet.}

I \text{ Met } I_2: \text{ Start time of } I_1 \text{ equal to finish time of } I_2, \text{ the two intervals do not overlap.}

I \text{ Overlaps } I_2: \text{ Finish time of } I_1 \text{ is greater than start time of } I_2. \text{ Both intervals have some overlapping.}

I \text{ Overlapped } I_2: \text{ Start time of } I_1 \text{ is less than finish time of } I_2.

I \text{ Covers } I_2: \text{ Start and finish time of } I_1 \text{ is greater than } I_2 \text{ and } I_1 \text{ completely covers } I_2.

I \text{ During } I_2: \text{ Start and finish time of } I_2 \text{ is less than } I_1 \text{ and } I_1 \text{ is completely covered by } I_2.

I \text{ Starts } I_2: \text{ Time interval } I_1 \text{ starts with } I_2 \text{ but finishes before } I_2.

I \text{ Started } I_2: \text{ Time interval } I_1 \text{ started with } I_2 \text{ but finished after } I_2.

I \text{ Finishes } I_2: \text{ Time interval } I_1 \text{ finishes with } I_2 \text{ but started earlier than } I_2.

I \text{ Finished } I_2: \text{ Time interval } I_1 \text{ finished with } I_2 \text{ but started later than } I_2.

Roshannejad (1996) categorized the event and state relationships as “Type One Relationships”, and the non-connected state relationships as “Type Two Relationships”. No further discussion on non-connected state relationship time is conducted here. Users may refer to Roshannejad for this relationship.

2.4.5 Aspatial component

The aspatial component is the attribute part of the spatial component. It is associated with the relevant part of the spatial data. The aspatial component consists of two parts, i.e., aspatial (attribute) objects and operations between objects (Figure 2.31). For example, objects are names like owner, color, address, land use etc, and operations are relationships between objects such as father of, citizen of, owner of etc.

2.4.5.1 Aspatial object and operations

Aspatial objects are attached to the relevant part of the spatial component, i.e., volume, polygon, arc, node etc., to represent real life objects such as land use (usually by polygon), roads, river boundaries, utility lines (by arc), and street lights, trees etc. (by node). All these representations need attribute descriptions: in the case of a land parcel, the owner-name of the parcel, address, area etc. As explained earlier, as a result of a real world event, change may occur in a spatial or aspatial component of STAO. The changes in an aspatial component are not as complex as in a spatial component. Therefore, no further elaboration on an aspatial component is considered here.
3.1 INTRODUCTION

This chapter reviews the limitations of existing spatio-temporal data models and defines and designs a uniform spatio-temporal data model. The motivation behind designing such a model is based on the assumption that mathematical theory and OO concepts can be employed to reduce the complexity of spatio-temporal data models. First, the major spatial data models are discussed, followed by a review of the main spatio-temporal data models and a conceptual schema for the unified cell tuple-based spatio-temporal data model (CTSTDM). CTSTDM is one of the chief contributions to this research. To have a better understanding of spatio-temporal models, it is imperative to highlight the spatial counterpart first and then the spatio-temporal part. Past research in TGIS could be viewed from two perspectives: research in spatial data modeling and spatio-temporal data modeling. Moreover, the established theories for spatial data modeling can be utilized for designing a spatio-temporal data model.

3.2 SPATIAL DATA MODELING

Traditionally, the spatial component of atemporal GIS is formalized using a raster- or vector-based approach for spatial data modeling. These two modes of representation are sometimes known as the field (raster)- and object (vector)-based approach (Figure 3.1).

The raster mode of representation uses tessellation (or meshes in computer graphics) for geometric modeling. Tessellation is the partition of space as a union of disjoint areal objects, or we can say that the basic data unit is a unit of space for which object information is recorded. This type of representation can be broadly grouped into regular and irregular tessellation.

- A regular tessellation (grid square) is a repeatable pattern of a regular polygon or polyhedron.
- An irregular tessellation is a repeatable pattern of a polygon or polyhedron of varying size and shape. Irregular planar tessellation is equivalent to a 2-cell complex.

Armstrong (1988) noticed that complexity in spatio-temporal modeling increases when dealing with the temporality in vector structure, as compared with raster structure. He identified the absence of spatial uniformity as a source of complexity. As the focus of the study is vector-based spatio-temporal data modeling, no further discussion is considered on the raster-based approach. For further details, user may see Laurini and Thompson (1996).

In the vector-based approach, an individual object is the basic data unit for which spatial information is explicitly recorded. The objects are compact in terms of computer storage, as only relevant spatial objects are recorded rather than entire space. This approach is adopted in this research. These object-based data models can be categorized into unstructured and structured data models. The first is known as a spaghetti data model and the second as a topological data model. Spaghetti data models are
simple in structure but have limited analytical power to store topological information. On the other hand, topological data models may not be simple in structure but they contain rich topological information.

Some hybrid systems also exist, where attempts have been made to integrate these two modes of representation, e.g., the TRIAD model by Peuquet and Wentz (1996). Each have inherent advantages and disadvantages in terms of data storage, processing speed and spatio-temporal query processing. Most of the research in spatio-temporal modeling has focused upon the extension of individual models (object- or field-based) to incorporate the temporal dimension (Peuquet and Qian, 1996). Only major spatial / topological models are discussed in the following section.

3.2.1 Review of spatial / topological data models

Topological data models explicitly store the spatial relations called topology. In most of the topological models, the basic logical object corresponds to a line. Each individual line segment is stored with two end points (coordinates). Various topological models have been proposed to date, examples include:

- Geographic Base File / Dual Independent Map Encoding (GBF/DIME): In the 1970s, the census bureau of the United States developed the DIME model for line features (street, roads, rivers etc.). Each line feature is represented by a line segment. The end of each line segment is an intersection of two lines or where the line changes direction. The main drawback of DIME is that individual line segments are stored in non-sequential order, therefore a search requires exhaustive scanning of all individual segments in sequential order.

- Digital Line Graph (DLG): The concepts of DLG were devised by the United States Geological Survey (USGS) during late 1970. Various point, line and areal features can be modeled using DLG format. For example, ARC/INFO is a hybrid data model that consists of topological and relational models. Geometry and topology are broken down into nodes, lines, area and coordinates (Laurini and Thompson, 1996). Attributes are kept in a separate file. DLG is less convenient for users but suitable for mapping and data exchange.

- Topologically Integrated Geographic Information System (TIGRIS): Developed by Intergraph Corporation; unlike ARC/INFO, TIGRIS, integrates spatial and attribute data into a single database, i.e., there is no thematic layer. Points, lines and area features are created from the basis of a topological structuring of primitive-level directed edges, start and end nodes, the left and right faces. The topological structure is based on a generalization of 2D cellular topology (Herring, 1987).

- Geographic Information System Retrieval and Analysis (GIRAS): Developed by USGS in the early 1970s. Elementary units are nodes, arcs, polygons, islands and polygon labels. The problem of containment (holes) is solved in this model.

- Topologically Integrated Geographic Encoding and Referencing (TIGER), Census Bureau of USA. The TIGER structure is an outgrowth of the idea of DIME (Laurini and Thompson, 1996). Basic element, i.e., topological 1-cell related to 0- and 2-cells. All lines were straight. No explicit polygon encoding.

- Cell tuple structure: Brisson (1990) devised a simple cell-tuple data structure for the representation of n-dimensional geometric objects. Spatial objects are defined as subdivided n-manifolds. Let $X$ be a CW complex, $\chi$ be a topological space, $\chi^c$ be the closure and $\partial \chi$ be the boundary of $\chi$. If $M$ is an $n$-manifold and $C = \{c_a\}_{a \in I}$ is a finite collection of disjoint open cells whose union is $M$, then the pair $(M, C)$ is a subdivided $n$-manifold if $\forall m$-cell $c_a$ of $C$, there exists a map $f_a: m^w \chi \rightarrow X$ such that:
• $f_a$ maps $\chi^m$ homeomorphically onto $c_a$.
• $f_b$ maps $\chi^m$ homeomorphically onto $c_b$.
• $f_c(\alpha^m)$ is a union of open cells of $C$, each of dimension $< m$.

The cell tuple structure is a collection of cells of different dimensions that can be defined as

$$T_M = \{(c_0, c_1, c_2, \ldots, c_n) \mid c_i \in C, c_0 < c_1 < \ldots < c_n\}$$

The $(n+1)$-tuple of $T_M$ is called cell tuple. Ordering of any $m$-cells ($0 \leq m \leq n$) is preserved in this structure.

3.2.1.1 Summary

The vast diversity in ways of organizing the topological data results in bewildering arrays of alternatives. The aforementioned models are examples of these alternatives. Some are implemented and commercially available (public domain), while others are still in the academic (private) domain. Research carried out by Corbett and others on cell complexes has been the basis of many successful implementations. In short, most of the commercial systems are successfully implemented using cell complex structures. ARC/INFO and TIGRIS are examples of such systems. Recent research into topological relations for higher dimensions (3D topology) also suggests the use of cell complexes, for example Pigot (1995). Pigot introduced the generalized regular cell complex to reduce the number of cells required to represent an individual spatial object. The purpose of this discussion is not to evaluate these topological data models, but to have an understanding of these models to provide the basis for spatio-temporal modeling. In summary, the concepts of spatial objects or spatial data modeling are quite well understood and significant research has been done, some from the purely geometric perspective and some from the topological perspective. The main problem is the standardization of terms in this field. In the next section, some spatio-temporal models that are
relevant to this research are discussed.

3.3 SPATIO-TEMPORAL DATA MODELING

Research into spatio-temporal data modeling is scattered. Unfortunately, few integrated approaches exist that can treat the spatio-temporal data in a unified manner. Past research provides a basis for spatio-temporal models but not a complete design. As mentioned earlier (Chapter 1), Langran (1992a and 1989) indicated five barriers to the implementation of a TGIS:

1] architecture
2] modeling spatial change
3] clustering of data
4] algorithms
5] individuality of system design.

Bagg and Ryan (1996) divided temporal issues in GIS into three parts: data management, display and analysis. The core of all these barriers and issues is the spatio-temporal data model. The unified spatio-temporal data model may facilitate the solving of other issues. Ideally, a TGIS should be able to deal with spatial, aspatial and temporal data as components of a single coherent data model. According to Langran (1989), “how to represent spatial change conceptually and in data structure is an equally troubling issue”. We would extend the problem to include the attribute-temporal component of TGIS, i.e., how to represent spatial (geometric and topological) and attribute change conceptually, and in the data structure.

This chapter focuses on spatio-temporal modeling. Most of the research in spatio-temporal modeling focuses only on the temporality of the spatial component, i.e., it addresses spatial change (more often geometric). Some authors emphasize that the type of spatial change must be treated but do not suggest a treatment. For example, Armstrong (1988) discussed the temporality of the area object and identified the types of change that could be associated with this object. As argued earlier (Chapter 2), a generic TGIS should be able to treat the temporality of the spatial and attribute components concurrently.

The objective of this section is to highlight different models proposed in the spatio-temporal domain and to briefly describe their main features. These features are studied from the conceptual or semantic perspective, i.e., how reality is captured, how time is modeled (attribute or separate dimension), how space is structured (object- or field-based), whether models are formalized using a mathematical basis, what are the primitive data types (spatial and temporal), how many time dimensions are incorporated etc. In some cases, limitations of these models are also identified.

3.3.1 Review of spatio-temporal data models

Various spatio-temporal data models have been proposed in the past two decades. Perhaps, the snapshot model is the earliest spatio-temporal data model in TGIS, followed by hierarchical data structure for US state boundaries, base state with overlay, space-time composite, space-time cube, event-based (including three-domain (TRIAD) and time-based models), process-oriented, OO models, 4D-GIS etc. Some of these are covered in the following discussion. Langran (1992a) provided comprehensive details on some of these models. Modeling techniques in the object-based approach are discussed in this research. Ideally, a single conceptual model should be able to handle all geographic data elements (node, arc and polygon).

3.3.1.1 Hierarchical data structure for US state boundaries

Perhaps the first attempt to address temporality in spatial databases is the data structure devised by Basoglu and Morrison (1978, as cited by Langran, 1992a). This data structure was designed to produce the snapshot of US state boundaries. The drawback is that neither temporal nor spatial
topology is addressed. Moreover, it cannot respond to queries as to how the two time slices differ (Langran, 1992a). Like most of the topological models in the spatial domain, this model is also arc-centric, i.e., the temporal information is attached to individual line segments. The structure is not intelligent enough to know that a certain line segment might no longer be linked to a particular county although remaining in use as the boundary of different county through historical subdivision (Langran and Chrisman, 1988).

### 3.3.1.2 Snapshot approach

A series of snapshots record only a fixed phenomenon at a given time. Such snapshots capture all phenomena whose granularity of change approximates its exposure time (Langran, 1992a). The intervals between time slices are not necessarily equal. This approach is rooted in traditional mapping (GIS) and mimics the progressive nature of the slow-motion video (Dangermond, 1984; Ross 1985, as cited by Langran, 1992a). This approach is commonly employed in the current GIS, which are non-temporal in nature.

Figure 3.2 is an example of urban land use evolution over the past 20 years, where change has been observed every five years. In the year 1980, there were two land uses, i.e., residential and recreational. Recreational changed to commercial use in 1990. In 1990, the commercial area expanded in size by reducing the residential area. Finally, in 1995, all residential changed to industrial use.

This snapshot approach represents a state but does not represent the events that change one state to another. Change boundaries are not visible. Instead of storing change, it stores a state. Thus it provides a crude way of representing change, which is the important component of time (Langran, 1992a).

### 3.3.1.3 Base state with overlay approach

The model base state with overlay provides a clear and concise description of the fundamental component of time, i.e., change. Figure 3.3 illustrates the base state with subsequent amendments. Only the amendments from time T1 to T2, T2 to T3, and T3 to T4 are recorded. It is constructed by flattening the three-dimensional space-time cube (Figure 3.4) into two dimensions. The difference in the time dimension is shown by a new object in two-dimensional space.

Compared with sequential snapshots, the volume of data is reduced, because only the changes are recorded instead of a completely new snapshot. Storing the change data rather than a complete sequence of snapshots makes more sense. It represents the change as the boundaries of both states and versions; therefore base state with overlay is better than the snapshot approach.

### 3.3.1.4 Space-time composite approach

Chrisman (1983) originally suggested this technique. The previous method, base state with overlay, provides a starting point for constructing the space-time composite. The space-time composite is generated by overlapping all time-stamped layers to produce a space-time composite layer (Langran
and Chrisman, 1988). For the purpose of modeling spatial change, the space-time composite is based on the base state within amendment to accumulate the geometry and spatial topology of the region, i.e., a space-time composite is a single polygon coverage generated from a snapshot series by overlaying the snapshot in order to define the greatest common spatio-temporal object that shares the histories of attribute change.

Figure 3.4 is the space-time composite of the snapshot demonstrated in Figure 3.2. Each polygon is represented by a different hatch pattern and has a distinct attribute history, which makes it different from its neighbors. Accessing temporal information is conceptually straightforward. To compile the single time-slice attribute the history of each polygon can be scanned to locate the attribute that was current at the desired time slice.

3.3.1.5 Space-time cube approach

This model depicts the process of two-dimensional space that is extended to only the third temporal dimension. Spatio-temporal objects can be treated as solids within a space-time cube. The space-time cube can be constructed by adding the third dimension (time) in two-dimensional space, based on base state with overlay. "This is the model of Hagelstrand, 1970; Ruker, 1977; Szego, 1987; and many others" (Langran, 1992a). Figure 3.5 is the space-time cube of sequential snapshots illustrated in Figure 3.5. Each cube has an attribute history distinct from that of its neighbors; it keeps the prior state until a certain time point, and then changes to another state.

Snapshot is not suitable because it does not keep track of version and mutation. Base state with overlay is not space filling and querying is not straightforward. The space-time cube still needs a lot of research in order to answer questions related to 3D-GIS. Moreover, dealing with 3D is beyond the scope of this research. Detailed discussions comparing these approaches can be found in Langran and Chrisman (1988) and Langran (1992a). An approach to integrate the space-time composite and OO technique has been designed and implemented in existing commercial software (ARC/INFO and ArcView) as a prototype. Part of that technique can be found in (Raza, 1996; Raza et al., 1998; Raza and Kainz, 2000c).

3.3.1.6 Event-based approach

Triad model:

Peuquet and Qian (1996) argued that the object- or feature-based models are more effective for the query and retrieval of information about objects and field- or location-based models are more effective for the query and retrieval of information regarding locations. Similarly, time-based models are more effective than either feature-based or location-based models for retrieving information about specific time or changes through time. Integration of these three (location-, feature- and time-based) components is provided through the TRIAD model (Figure 3.6). Only the feature- and time-based view approach is briefly discussed here. Details of the TRIAD model can be found in Peuquet and

Object or feature view encapsulates the feature characteristics, such as the unique feature ID, name, feature class, spatial delimiters, temporal delimiter etc. Two feature positions are stored in a spatial delimiter: a) the starting position and b) the latest position. These positions can be associated with point, line or polygon features. The temporal delimiter stores two times: a) the starting time of a feature (T_s) and b) the finishing time of a feature (T_f). Therefore the age of the feature can be determined. The TRIAD model suffers from the following drawbacks:

- Descriptions of geometric changes are not explained. For example, if there is a change in the size of the polygon, then the new state of the polygon will record the whole polygon or only the change boundaries.
- Topological changes are not incorporated.
- A single time dimension is considered, i.e., world time.

Important to note is that there is no room for the version; only the starting and latest positions of the feature are stored. Similarly, the starting (T_s) and finishing times (T_f) are incorporated. There is no clear picture of feature mutations. Therefore, spatial change detection is possible only between the initial and latest state of an object or feature.

The event-based spatio-temporal data model (ESTDM) approach advocated by Peuquet and Duan (1995) is basically an event-oriented approach, where location in time becomes the primary organization for recording change. The sequence of events or processes is recorded via a single timeline. This approach has been applied to location-based GIS and is not discussed further.

Claramunt and Thériault (1995) model events as a set of processes that transform entities. The link between thematic and spatial data has been established through a versioning mechanism. Three version tables, past, present and future are provided for temporal information. Each entity is time-stamped at tuple level (relational form). However, it is not clear what the spatial entities are or how they are evolved.

### 3.3.1.7 Three-domain model

Yuan (1994) proposed a conceptual design for a spatio-temporal model in wildlife application called the three-domain model, i.e., semantics, time and space domains (Figure 3.7). The semantics domain consists of abstract concepts of aspatial and atemporal properties. The temporal objects such as instances and interval time are defined in the temporal domain. The spatial domain comprises spatial objects such as point, line, polygon, cells and volumes; each of them represents 0-, 1-, 2- or 3-dimensional spatial objects. The three-domain model supports the location-centred and entity-centred approaches. The location-centred approach is used in conventional GIS layer-view models as reality is viewed as layers of themes. The entity-centred approach describes temporal and spatial properties as attributes of entities.
Each semantic object (geographic entity or event) can be associated with many temporal and spatial objects. Each temporal object can be linked with many spatial objects in order to convey that the semantic object appears in many places at one time, e.g., more than one land-use-residential change at time T1. Similarly, the spatial object can also be linked to many temporal objects. The major drawback in Yuan’s work is the missing identifier for spatial and temporal objects. This identifier is very important in a TGIS to keep track of the history of the objects and for the scenario-modeling exercise. But this work highlights the importance of semantics in TGIS. This semantics approach can be used in multiple urban applications, as there are many-to-many links between semantic, spatial and temporal objects.

3.3.1.8 4D GIS

Pigot and Hazelton (1992) defined two manifolds. The first manifold is called spatial Mn and the second temporal M1. The topological product of these two is considered as spatio-temporal objects. The domain of n-dimensional spatial objects (n ≤ 3) forms the topological spaces of the n-cell complexes of dimension ≥ 2, forming n-manifolds, where n-manifold Mn (without boundary) is connected and homogeneously polytope (each point has a neighborhood homeomorphic to an n-dimensional disk). An n-cell ∈ n-cell complex is homeomorphic to flat n-manifold Mn with (n-1)-manifold Mn-1 boundaries. The time is considered as 1-manifold M1 with boundary points (t1 and t2). The product of M1 with i-cell Ei at time t1 generates an (i+1)-cell Ei+1 terminated by Ei at time t2. The sum of all products forms an (n+1)-cell complex Cn+1 and (n+1)-manifold Mn+1 (spatio-temporal). In this approach, spatial objects are defined in modeling space R3 and time is considered as an extra spatial dimension. Further, the modeling of the aspatial change (component) is ignored.

3.3.1.9 Object-oriented approach

Worboys (1992a) proposed an OO approach to handle the spatio-temporal objects. He references the object in geographic space to spatio-temporal (ST-) and non-spatio-temporal (attribute) objects. The ST-object is defined in space-time, while the attribute is assumed to be constant during the spatio-temporal extents. Worboys considered the ST-object as a finite collection of disjoint atoms (ST-atoms). An ST-atom is an ordered pair <S,T>, where S is the spatial object and T is an extended temporal interval.

To elaborate the modeling concept, we cite the example considered by Worboys. Figure 3.8, shows the change in administrative boundaries. At time T1, there were two objects, O1 and O2. At time T2, the spatial extent of O1 was reduced which resulted in the enlargement of O2. At time T3, a new object O3 was created by reducing the spatial extent of O2. Figure 3.9 shows the decomposition of ST-objects into ST-atoms.
• $O_1$ is represented by two ST-atoms $<A_1, [T_1, T_2]>$ and $<A_2, [T_2, T_\infty]>$.
• $O_2$ is decomposed into three ST-atoms $<B_1, [T_1, T_2]>, <B_2, [T_2, T_3]>$ and $<B_3, [T_3, T_\infty]>$.
• $O_3$ is represented by a single atom $<C_1, [T_3, T_\infty]>$.

From the TGIS perspective, Worboys' model has suffered from at least the following drawbacks:

• Temporal treatment of the attribute component is not modeled.
• No link is provided to scan the history of objects, i.e., which was the parent object of $O_3$.
• Links between atoms are missing, i.e., $B_3$ is a new version of $B_2$, similarly $B_2$ is a new version of $B_1$, but the spatial extent of $B_3$ does not lead to the spatial extent of $B_2$ (Roshannejad, 1996).
• Repeating the unchanged information (Yuan, 1994).

Another approach introduced by Worboys (1992b and 1992c) is based on the concepts of simplicial complexes. In this approach, only 1D valid- or world-time (WT) intervals are associated with spatial objects (simplicial objects). However, in (1994c), he proposed a unified model for spatial and temporal information by combining the 2D space and 2D time (bi-temporal element, i.e., WT and DBT). In this research, he extended his work of (1992c), which was based on combinatorial topology, where spatial objects were represented as simplicial complexes. A spatio-temporal object $<S, T>$ is defined as a unified object with spatial and temporal extents. The object is called ST-simplex and the collection of ST-simplex is the ST-complex. Some spatial-temporal relations are also defined. See Worboys (1994c) for more details.

Worboys (1994a) presented some potential options for the integration of space and time ($S \times T$). In this approach, space and time are fused to construct a spatio-temporal model. He discussed the two options for time-stamping spatio-temporal objects, i.e. 1] the entire geographic object and 2] the spatial primitives of an object. He proceeded with the second option as the first approach has limited expressive power in terms of temporal properties. In the second approach, two major choices are available for fusing time in space: 1] only at point level or 2] at spatial primitive level (point, node, chain and polygon). However, he emphasized the point-level approach for fusing space and time. Three object classes have been outlined:

![Figure 3.9 Decomposition of object into atoms.](image-url)
• spatio-temporal atom: a two-dimensional temporal object associated with a point (spatial)
• tpoint: an object that stipulates a point in space-bi-time
• tstring: an object that specifies a string in space-bi-time.

This approach is promising as the topological classes node, chain, and polygons are generalized to incorporate time. \textit{Tnode} is a subclass of \textit{tpoint}. Classes \textit{tchain} and \textit{tpolygon} are built from \textit{tnode}. But, like previous approaches of Worboys, this model, too, fails to deal with the attribute component. Applications, such as land use change, cadastral systems, building control systems etc. are examples where change in the attribute component is equally vital. By ignoring the temporality of the attribute component, one cannot achieve the general unified model for spatio-temporal processes. As discussed earlier, although simplicial complexes are the simplest forms of spaces and manipulations of these complexes are comparatively easy, adoption of simplicial complexes results in voluminous data (see simplicial complexes, Chapter-2). These bulky data may increase the system response time.


\textbf{3.3.1.10 Summary}

Designing a unified spatio-temporal data model is like a puzzle. The aforementioned models are examples of the bewildering array of alternative solutions to this puzzle. The pieces of the puzzle have to be placed properly to solve this puzzle. The research in space-time modeling is not as mature as in spatial or time modeling. Not much has been done in dealing with topology and time. Most of the models (vector-based) deal with time, but ignore topology, except Worboys (1994c) and Pigot and Hazelton (1992). Several facets (attribute or dimension, discrete or continuous, linear or branching, single- versus multiple-dimension etc.) of time have been tackled in numerous fashions, e.g., in snapshot, layers are time-stamped; in space-time composite, attributes are time-stamped; in event-oriented, events get the time-stamp; in object-oriented, objects are time-stamped etc. Whether the issue of time should be considered as an attribute or dimension is discussed in § 3.4.2. Most of the models acknowledged time as discrete, such as snapshot, base state with overlay, space-time composite, space-time cube, event-oriented, object-oriented, the three domain model, and the TRIAD model. One may consider time as continuous, but it boils down to the fact that processors used in machines (computer) are discrete. Even if time is considered as continuous, at physical level it is treated as discrete. Some models consider a single time dimension (snapshot, event-oriented model etc), others two dimensions (object-oriented model). The purpose of discussion is not to evaluate these models but rather to distil the complex issues in space-time modeling and to pave the way for designing the unified space-time model.

As stated at the beginning of this chapter, the aim of this research is to design a spatio-temporal data model that is based on a sound mathematical (topological structure) basis. The approaches adopted by Worboys (1994c) and Pigot and Hazelton (1992) are based on the mathematical theory of simplicial and cell complexes, respectively. A limitation of the Worboys (1994c) approach has already been identified in § 3.3.1.9. The approach adopted in this research is an extension of the Worboys (1994c) approach, in the sense that it is extended to cell complexes. This approach is different from the one of Pigot and Hazelton (1992), because in this research time is considered as a separate dimension, which is similar to, but different from the spatial dimension (see Chapter-2, Similarities and difference between space and time).

\textbf{3.4 DESIGNING THE SPACE-TIME MODEL}

The concepts of time, space and space-time have been discussed in Chapter-2. Two issues are vital when considering designing a space-time model, i.e., the modeling and integration perspectives.
3.4.1 Modeling perspective

Modeling time in GIS can be performed in two ways (Langran, 1989):

- process modeling
- time modeling.

3.4.1.1 Process modeling

Process modeling can be further divided into two approaches:

- Event-based approach
- Time- or trigger-based approach.

The event-based approach is also known as the change-based approach. In this approach, the database is updated as soon as a change or an event occurs in the real world. Therefore, the main source of updating is an event or change, i.e., the database has to be or will be updated when the event/change occurs or is expected to occur, respectively. For example, the land use data will not be updated until land use is changed in reality or it will not be updated until land use is expected (forecast) to change in reality (according to plans). This is applicable when updating the land use data through the operational planning process, such as building permits. The land use changes when the building permit is issued. In this case, the evidence (evidence is a result of an event) is the issuance of the building permit. Process- or event-based modeling allows trend analysis to be performed, such as

- where was the land use (residential) location five years ago?
- did a particular feature move?
- how has land use (residential) changed over the last five years.

In the time- or trigger based approach, the source of change is time not an event. In this approach, the system triggers another process after the elapse of a certain time (application-specific). For example, in building control application, if the building is not constructed after T1 (say three years) then a notice will be served on the owner to pay non-utilisation fees (NUF). This type of approach is important for urban planning agencies, especially development authorities (such as Karachi Development Authority), as a large number of owners are due to pay NUF. Currently, this is being handled manually, which is a time-consuming process.

3.4.1.2 Time modeling

Each moment in time (of interest) captures the reality and stores what exists at that time. It recognizes only one basic change, i.e., the passage of time, which flows constantly, unaffected by anything else (Shoham and Goyal, 1988 as cited by Al-Taha and Barrera, 1990). This approach is independent of any change or triggers. It mimics the concept of a movie camera, where each scene is disjoint from each other. According to Hazelton (1992), the time-based approach is merely a data collection technique. In this technique, time is considered as a parameter, rather than an attribute or dimension. Some research into temporal databases and GIS is conducted along this line with time as a parameter (Hazelton, 1991). Data collected via satellites or land use data collected through field surveys fall into this category. The moment (resolution) could vary from a day to a year(s). In this technique, data are recorded irrespective of any change or event.

In the first approach, i.e., process-based modeling, the time element is not a structural component of the model. In the object (vector)-based model (GIS), the major difficulty is to store the changing topology over time. This approach has limited capabilities to store information for analysis because there is no structure for this information (Hazelton, 1992). The second approach is a time-based approach. Each moment in time (of interest) captures reality and stores what exists at that time.

In this research both approaches are adopted: the event-based approach for updating land use data
Chapter 3: Spatio-Temporal Data Modeling: Conceptual Schema

through the operational process (parcel and building permits) and the time-based approach for recording land use data through field surveys of any temporal resolution, i.e., the minimum is a second and the maximum a year(s). This time modeling is associated with the dimension (database and world time) of time. These dimensions are discussed in a conceptual schema (§ 3.5).

3.4.2 Integration perspective

The integration of time and space is another important issue. How to incorporate time in space (GIS) is still a matter of debate (Hazelton, 1992). There are at least two choices to consider:

- time as an attribute or
- time as a separate dimension.

3.4.2.1 Time as an attribute

Historically, time in aspatial and spatial databases has been considered as an attribute rather than a structural component of modeling. In this approach time is an attribute of an object (Hazelton, 1991). Langran (1992a) proposed a space-time composite model and considered time as an attribute of spatial data. Each change (spatial or aspatial) creates a new tuple (relational structure). This approach provides a quicker solution to the problem but does not provide an elegant way of dealing with time and space in an integrated manner. Models that support time as an attribute have been indicated in § 3.3.1.9.

3.4.2.2 Time as a separate dimension

Dimensions and attributes are two different concepts. When we talk about dimension, we have the notion of direction, i.e., the direction of a vector (v). However, when we consider attribute, we may have a sense of magnitude, i.e., the magnitude of a vector (although it is not a vector). Time always has a direction, e.g., past to present or present to future; it always has a sense of direction (bottom to top or left to right etc.). This approach places time outside the limits of a particular object, and places it together with the space (space-time) in which an object resides (Hazelton, 1991).

As mentioned in § 3.3.1, recently a few attempts have been made to consider time as a structure component of space-time modeling. For example:

- Pigot and Hazelton (1992) and Hazelton et al. (1990b) proposed a topological model for a 4D-GIS and represented time as an extra spatial dimension so that spatio-temporal objects were embedded in an open 4-manifold R^4.
- Worboys (1992b) considered time as a structural component based on the concepts of simplicial complexes. Each primitive object node, arc and triangle is treated as a 0-T-simplex, a 1-T-simplex and a 2-T-simplex, respectively.

Some limitations of these approaches were identified earlier. Moreover, no implementation of these approaches yet exists. For a unified spatio-temporal data model, time should be modeled as a structure component of a TGIS. This research proceeds with a process modeling approach, where we consider time as a structural component of space or as a dimension.

3.4.3 Design issues

Two approaches can be considered in designing a spatio-temporal data model, i.e., 1] design a new data model by using the concepts proposed in the spatial domain and the temporal domain or 2] create an additional layer (temporal) on top of existing spatial models (GISs). The first approach provides an efficient way of dealing with the dynamic world but needs substantial efforts. The second approach provides comparatively quick and easy development but with limited capabilities. One of the problems in adopting this approach is that most of the commercial systems (existing GISs) are closed systems, which cannot be extended or modified (van Oosterom and Bos, 1989 as cited by Mioc et al.,
This approach was adopted by Raza and Kainz (2000c). In that approach, an attempt was made to model the land use changes by integrating the space-time composite and the OO approach using ARC/INFO and ArcView. Some shortcomings have been reported. In the non-spatial domain, Snodgrass and Ahn (1986) proposed a similar approach for the development of temporal databases.

The first approach is followed to integrate the spatial and temporal components. In the spatial domain some models have already been discussed in § 3.2. The approach for integrating of space and time is an extension of Worboys’ (1992b, 1994c) work. The approach proposed by Worboys is based on the concept of simplicial complexes (as discussed in § 3.3), which is one of the fundamental approaches for any generic TGIS. As mentioned earlier, the simplicial complexes are the simplest forms of spaces and their manipulation is comparatively easy but their adoption results in voluminous data. This may increase the system response time (Raza and Kainz, 1999).

Most methods in computer science for designing the model can be categorized as one of three kinds (Booch, 1994):
- top-down structured design
- data-driven design
- object-oriented (OO) design.

Traditionally, most of the software has been written using a top-down approach, but it does not address the issues of data abstraction, information hiding, or design for extremely complex systems; nor is it appropriate for OO programming language. The data-driven approach has been successfully applied to complex systems, e.g., information management systems. However, it pays little attention to time-critical events. Modern and complex engineering applications such as computer-aided design (CAD), computer-aided software engineering (CASE), computer-aided manufacturing (CAM) and intelligent offices (office automation) have been the prime target of the OO technique (design and databases). In spatial and particularly spatio-temporal databases, the amount of information and complex relationships between objects make the OO technique a good candidate for designing, programming, databases and user interfaces. It helps to alleviate the semantic gap between various stages of modeling. For example, the classes defined at the conceptual level can be directly programmed in high-order programming language (C++, Java++, Smalltalk, Object Pascal, Ada, Common Lisp Object System (CLOS) etc.) and mapped to tables (in OO or relational databases). In the object-relational approach, however, some normalization may be required.

The OO technique is used to design the model: it is a natural way of designing a model and its importance in spatial and temporal databases has been acknowledged by many researchers (Booch, 1994; Egenhofer and Frank, 1989; Kösters and Pagel, 1997; Maguire et al., 1990; Ramachandran et al., 1994; van Oosterom and Bos, 1989; Wachowicz, 1999; Worboys, 1992a etc.).

OO is centered around three notions, i.e., object ID, inheritance and abstract data type (classes). Inheritance is an abstraction mechanism. Four well-known abstractions are the core of the OO design technique, i.e., classification, generalization, aggregation and association. Generalization/specialization is an ‘is-a’ relationship, aggregation is a ‘part-of’ relationship and association is a ‘member-of’ relationship. Generalization/specialization provides the inheritance hierarchy. Aggregation combines different classes to form a new class. The classes have association. This association is defined through cardinalities (1:1, 1:n, m:n etc.). Two types of classes are defined, i.e., abstract classes and concrete classes. An abstract class has no instance; it is assumed that any of its concrete subclasses will add to its data and operation by implementing its abstract operation. The implementation of the concrete class is complete and it may have instances (Booch, 1994).

The data model consists of data structure, operators and consistency rules. A class is a set of objects with common properties. A class consists of data members and member functions (operations); the first is called the static and the latter the dynamic aspects of the class. The structure of the data model is defined by data members (data), while operators and consistency rules are defined through
operations. Emphasis in the following sections lies on data structure and consistency rules for the model. The operators are discussed in Chapter-4.

The conceptual schema for spatio-temporal data modeling is presented here. An OO visual-modeling language, i.e., Unified Modeling Language (UML), is used to define the data structure and operations. First, UML is based on solid semantics and notation definition, which is necessary for interoperability. Secondly, UML has evolved from the Booch (Booch, 1994) object modeling technique (OMT), OO software engineering (OOSE) and other OO methods, which have a proven track record of successful implementation in complex software design. It supports the definition of interfaces to objects as data (attributes), operations and association. First, a brief introduction of the UML presentation, notation and organization is presented, followed by the conceptual schema.

3.4.4 UML presentation, notation and organization

UML is a language for designing a conceptual schema, using OO concepts. The vocabulary used in the design of the model is presented in Figure 3.10. It is based on UML notations. The use of UML is increasing steadily; one reason is standardization. For example, the International Standard Organization (ISO) uses UML for quality principles in geographic information (ISO-TC-211, 1998) and to define the geometric model for open GIS (Open GIS Project Document 98-072, 1998).

OO modeling is graphical and, although it contains some inherent constraints, sometimes it may not be sufficient for a precise and unambiguous specification and may need additional constraints regarding the objects (UML 1.3, 1999). These constraints are normally defined by natural language, which may result in ambiguities. The Object Constraints Language (OCL) were developed by the IBM Insurance Division and incorporated in UML to contribute additional constraints. It is a formal, pure expression and type language that remains easy to read and write. However, it is not a programming language; therefore, program logic, flow control and all implementation issues are beyond its scope. The constraints in the model are defined in OCL.

![Figure 3.10 Icons used to represent the model.](image)

The following conventions are used to define the data members and member functions (operations) of the classes. They are in compliance with the UML notation guide (UML 1.3).
Chapter-3: Spatio-Temporal Data Modeling: Conceptual Schema

Data members:

<<stereotype>> visibility name: type-expression = initial-value

Operations:

<<stereotype>> visibility name (parameter-list): return-type-expression

parameter-list : comma-separated list of formal parameters with following syntax
kind name : type-expression = default-value

<<stereotype>> : tag for data member (attribute) or operations
visibility : + public visibility
           : - private visibility
           : # protected visibility
/ : derived data member, such as calculating area
name : identifier string that represents the name of the attribute (data type)
type-expression : language-dependent specification of the implementation of the attribute
initial-value : initial value of newly created object
kind : in, out, or inout (default is in)
nname : name of formal parameter
default-value : optional value expression for parameter.

The abstract class, attribute or operations are defined in italic font. In addition to UML types, stereotype can be defined in UML to add extra semantics. They are defined within guillemets "<< >>". The following is the list of stereotypes used in designing the model.

<<position>> : the object is defined in space and has position
<<time>> : the object is defined in time; this is assumed to be linear time
<<dynamic>> : the kind of operation create, kill, destroyed or reincarnate (discussed later); this operation may modify the state of the system
<<query>> : the nature of the operation is query; this operation does not modify the system’s state
<<index>> : information can be used in indexing.

The following data types correspond to the collection. The type collection is predefined in OCL. There are three collection types in OCL, i.e., set, bag and sequence. Set is the mathematical set; a bag is like set; a sequence is like a bag in which the elements are ordered. Collection is a set as defined in the OCL and can be of following types:

Set{ } : a normal set; members or objects of the set appeared only once and order is not important
Sequence{ } : an ordered set, with no repetition
Bag{ } : a collection of objects, where repetition is allowed.

UML organizes the classes as packages. A package is a set of related types and interfaces that form a consistent component of a software system design.

3.5 SPATIO-TEMPORAL DATA MODEL: CONCEPTUAL SCHEMA

For modeling purposes phenomena of reality are perceived as objects and represented by three components, i.e., space, time and attribute (Raza and Kainz, 1999 and Raza et al., 1998). This object is called the Spatio-Temporal-Attribute Object (STAO). The three components form three classes: SpatialClass, TemporalClass and AttributeClass. These classes are aggregated to create SpatioTemporalAttributeClass, commonly known as FeatureClass. Figure 3.11 is a view of this reality forming the class hierarchies. The three classes that are aggregated to generate
SpatioTemporalAttributeClass provide a basis for modeling STAO, which is the backbone of any TGIS.

As time is associated with spatial and attribute classes, it can be integrated to form two classes, i.e., SpatioTemporalClass and AttributeTemporalClass (Figure 3.12). As mentioned earlier the problem peculiar to TGIS is the identification of change. These classifications facilitate the analyses and identification of change in STAO. A general change model has already been discussed in Chapter-2. Now this essential and non-essential change model is applied to the different components of STAO. At any given time T, there are seven possibilities that can propagate the change in STAO (Figure 3.13).

This categorization based on the changes is SpatioTemporalClass (STClass) and / or AttributeTemporalClass (ATClass). The STClass can further be classified into the geometric and topological classes. Geometric and topological changes are associated with the STClass and attribute/thematic changes are associated with the ATClass of STA-Class. At time T, these changes can be categorized into eight basic types (Figure 3.13):

1] all three change
2] geometry and topology change
3] geometry and attribute change
4] topology and attribute change
5] only geometry changes
6] only topology changes
7] only feature changes
8] none of them change.

The complexity in the vector (object-based) data structure is increased due to the absence of spatial uniformity (Armstrong, 1988). A similar approach for the classification of change as presented by Armstrong is shown in Figure 3.14.
Note that spatial dimension change is not considered here. For example, a city at time $T_1$ represented by a point may be represented by a polygon at time $T_2$ as a result of city expansion at a particular scale. Changes in (Figure 3.13) are mutually exclusive, i.e., they can occur independently or simultaneously. For example, consider three parcels at time $T_1$, namely A, B and C representing residential, commercial and industrial land use, respectively. At time $T_2$, residential changed to recreational (Figure 3.15). Although there is no change in parcel B, its neighbor residential is changed to recreational. Should one consider this change as merely a change of parcel A, or incorporate this change in parcel B as well? This is an example of attribute change. Consider another example of geometric change (Figure 3.16) relating to two parcels, A (residential) and B (commercial), at time $T_1$.

Figure 3.14 Type of changes in STAO.

Figure 3.15 Attribute and topological change.

Figure 3.16 Attribute and geometric change.

Parcel A is bounded by arcs $a_1$ and $a_2$ and parcel B is bounded by arcs $a_3$ and $a_5$. At time $T_2$, parcel B
splits into two parcels: parcel B (commercial) and C (industrial). Apart from the change in parcel B, there is also a change in parcel A. Now parcel A is bounded by three arcs, namely \(a_1\), \(a_3\) and \(a_4\), (although its shape and area are the same). There is a geometric change in parcel A but no attribute change. The question is again: should one consider this type of geometric change as a change in parcel A? The answer lies in the particular application or implementation strategy. Based on OO concepts, we consider this type of geometric change in parcel A as a new polygon object, because A is defined by a new set of arcs \((a_1, a_3, \text{and } a_4)\). Spatial (geometric and topological) changes are discussed in SpatioTemporal- and TemporalCellTuple-Class. These changes are generic in nature and non-application-specific. The attribute changes are application-dependent and are discussed in Chapter-5, where the modeling of parcel and land use change is discussed.

Figure 3.13 is the general configuration of changes in STAO, where the spatial dimension is not considered. If we consider the spatial dimension when dealing with the changes in STAO, a careful analysis is necessary, as they are represented by primitive objects, i.e., node (0-cell), arc (1-cell), polygon (2-cell). Evolution of these primitive objects constitutes an indispensable foundation for TGIS (Claramunt and Thériault 1995). Three classes, i.e., TemporalClass, AttributeClass and SpatialClass, and their subclasses, are discussed in sequel. Only data members and constraints are defined here. Fundamental operations are discussed in Chapter-4; other operations governing the classes are discussed in Chapter-7 (implementation).

3.5.1 TemporalClass

The TemporalClass deals with the modeling of time. The structural aspects of time are defined by three subclasses, i.e., linear, branching and cyclic time, each with a number of temporal operations such as before, after, equal, meets, met, overlaps, overlapped, covers, during, starts, started, finishes, finished (Allen, 1983). Various facets of time have been discussed in Chapter-2; therefore, structural aspects, type, representation, granularity of time etc., are not discussed here. As mentioned earlier, we have sidestepped the philosophical strife over time and consider time as a phenomenon that can only be perceived by its effects. We consider time as a function of change \((T \rightarrow C)\), where change can occur in the spatial or attribute component independently or simultaneously.

In LinearTimeClass (single axis), time advances from past to future in a totally ordered fashion. In branching time (multiple axes), time advances from past to present, where it then splits into several time lines; it may also have many pasts and a single future. The linear time model is similar to the concept of totally ordered sets, and the branching model is comparable to partially ordered sets in mathematics. In the cyclic time model, time repeats itself after an interval. In most applications, particularly GIS and DBMS, time is considered linear. Snodgrass (1992) argued that the linear time model is quite adequate in spatial modeling. However, in any generic TGIS all three time classes are needed. Figure 3.17 illustrates the three subclasses of TemporalClass.

To simplify the model, the LinearTimeClass is considered. Time in LinearTimeClass is considered as an absolute time. Three dimensions of time are incorporated as three specialized classes of LinearTimeClass, i.e., DataBaseTimeClass (DBT), WorldTimeClass (WT) and SystemTimeClass (ST). Therefore, the model is tri-temporal. Figure 3.18 illustrates the class hierarchy of
LinearTimeClass. WT refers to the time the event actually occurred in the real world and DBT refers to the time at which this was recorded in the database. ST reflects the time at which spatial changes occur in the system. ST is explicitly associated with the spatial object and is independent of DBT and WT. It is different from DBT in the sense that the latter represents the updating of STAO in the database, while the former indicates the updating of the spatial object. In LinearTimeClass, three data types are defined, PointTime, IntervalTime and SpanTime. 0-T refers to PointTime [T_From], 1-T represents IntervalTime [T_From, T_Until] and SP-T refers to span time \( \{T_{\text{Until}} - T_{\text{From}} \} \). 0-T holds single values, and 1-T (duration) pairs of values of time between two instants (points) that determine the life of the object. 1-T is anchored duration and SP-T is unanchored duration. DBT and WT have one subclass, i.e., SpanTimeClass. PointTimeClass is the most primitive time class. This class defines the date and time up to the second granularity. ST has the finest time granularity (second), while DB and WT may have any granularity required by a particular application. DB and WT are dependent on IntervalTimeClass. IntervalTimeClass has association with PointTimeClass, and PointTimeClass has association with SpanTimeClass.

The temporal consistency rule governing this class can be defined as:

1) For all interval time 1-T, \( T_{\text{Until}} \) is greater or less than \( T_{\text{From}} \):
\[
\forall (1 - T)[T_{\text{From}} \leq T_{\text{Until}}]
\]

2) \( T_{\text{From}} \) is always non-null.
\[
\forall (1 - T)[T_{\text{From}} \neq \emptyset]
\]
3.5.2 SpatialClass

SpatialClass represents the space. The core of SpatialClass is the spatial objects and their relationships. Spatial objects are defined by a cell complex. A cell complex is a collection of cells of various dimensions. A cell complex has a sound mathematical basis, which facilitates modeling in a more structured fashion. Their advantage is that every point, line, and polygon can be mathematically mapped into cells of respective dimensions through homeomorphism (Kainz, 1995). Cells are the most generalized forms that represent the geometry of an object. SpatialClass has one point subclass (Figure 3.19).

The extended cell complex is called the temporal cell complex, where time is considered as a structured element. The time is fused at the basic cell level, which could be propagated to the cell complex level, forming SpatioTemporalClass. SpatioTemporalClass is an aggregated class of SpatialClass and TemporalClass (Figure 3.19). Data members and the operations define the responsibilities of the classes. Operations are the signatures and methods are the implementation of these signatures. The data structures (attributes) of these classes are discussed in the following sections; operations are dealt within Chapter-4.

3.5.2.1 PointClass

PointClass is one of the basic, i.e., primitive, classes of the metaclass SpatialClass (Figure 3.20). All the spatial and spatio-temporal objects are derived from this class. The object of this class itself is atemporal and does not require any time stamp (Figure 3.20). The instant of PointClass is point object (p) with unique PointID and a coordinate pair (x, y).

3.5.3 SpatioTemporalClass

The object of this class is defined in a space at time t. Formally, we can define these objects as follows:

An (open) \( m \)-tcell is a topological space homeomorphic to an open ball \( E^m \) of \( R^n \) (Euclidean n-space). A finite collection \( k \) of \( m \)-tcells is a TemporalCellComplex (TCC) if:

- different elements of \( k \) have disjoint interiors
- for each \( m \)-tcell in \( k \), the boundary of \( m \) is a union of elements of \( k \)
- if \( a, b \in k \), and \( a \cap b \neq \emptyset \), then \( a \cap b \) is a tcell, and is a union of elements of \( k \), where tcell is either \( a \) or \( b \) (of different dimension) or a common face.

This research is restricted to \( n = 2 \) (2D space-time), where \( m \in \{0,1,2\} \). Therefore, \( R^{m,n} \) represents the object in \( E^n \) at time \( t \), where \( m \) is the dimension of the object and \( n \) is the dimension of space where the object is located at time \( t \), such that \( n \geq m \).

Generally, in a cell complex no, cell intersects another except along their boundaries. Therefore, when
a cell intersects with a ‘world’ cell (say 2-cell), it subdivides the space. However, the \( i \)-cell \((i = 0,1)\) may intersect with the interior of a 2-cell (non-world cell) in three ways (Corbett, 1985 and 1979):

- 2-cell with cyclic singularity
- 2-cell with acyclic singularity
- 2-cell with interiors singularity.

These singularities are accommodated in TCC and are depicted in Figure 3.21.

TemporalCellComplex is considered as a SpatioTemporalClass (Figure 3.21). SpatioTemporalClass is the aggregation of SpatialClass and SystemTimeClass. Moreover, it is a metaclass of three classes, i.e., ZeroTCellClass, OneTCellClass and TwoTCellClass. The objects of ZeroTCellClass, OneTCellClass and TwoTCellClass are temporal node (ZTC), temporal arc (OTC) and temporal polygon/area (TTC). ZTC, OTC and TTC are members of the temporal cell complex (TCC). \( \forall iTC \in TCC, where i = \{0,1,2\} \).

The three subclasses (ZeroTCellClass, OneTCellClass and TwoTCellClass) of SpatioTemporalClass are described in Figure 3.22. The association between PointClass, ZeroTCellClass, OneTCellClass and TwoTCellClass is defined using association relations. Similarly, the association (Interval) between SystemTimeClass and ZeroTCellClass, OneTCellClass and TwoTCellClass is shown in Figure 3.22.

TemporalClass is already defined in Figure 3.18. Figure 3.22 is an abstraction of SpatioTemporalClass. To simplify, the content (data members and member functions) of these classes is suppressed. Emphasis is laid on describing the various associations, dependencies, generalization, aggregation etc. However, details of the fundamental operations within the respective classes are given in Chapter-4. The objects ZTC, OTC and TTC define their own spatial and temporal configurations and are discussed in sequel.
Figure 3.22 SpatioTemporalClass: the metaclass of ZeroTCellClass, OneTCellClass and TwoTCellClass.
### 3.5.3.1 ZeroTCellClass

ZeroTCellClass is the subclass of SpatioTemporalClass (Figure 3.22). ZTC is the object of this class; it is a zero-dimensional object, which has a position and is represented by one PointObject. A ZTC object must have a point object, while a point object may have zero or one ZTC object. ZTC’s life span is represented by interval time 1-T \([T_{\text{Start}}, T_{\text{End}}]\). This object can be born or die. The responsibilities of ZeroTCellClass are defined in terms of its member functions (attribute) and operations. The attributes of this class are shown in (Figure 3.23). Two data members of this class are PointID and system IntervalTime (1-T).

```
ZeroTCell

<<position>> + pointid: Point
<<time>> + systemtime: 1-T

<<dynamic>> + create(P: Point): Boolean
<<dynamic>> + kill(P: ZeroTCell): Boolean
<<dynamic>> + destroy(P: ZeroTCell): Boolean
<<dynamic>> + reincarnate(P: ZeroTCell): Boolean

......
```

---

### 3.5.3.2 OneTCellClass

OneTCellClass is the subclass of SpatioTemporalClass. OTC is the object of this class. It is a one-dimensional object, which is bounded by two ZTC. The OTC object is an ordered sequence of point objects. The \(j\)-th OTC\(_j\) = sequence \(\{p_1, p_2, p_3, \ldots, p_i, \ldots, p_n\}\), where \(i \geq 2\). In the case where \(i = 2\), i.e., two point objects, the last \(p_2\) and first points \(p_1\) are the first and last ZTC. In the case where \(i > 2\) (loop), the first \(p_1\) and last point \(p_n\) are a ZTC and are the same. In the parametric form, an OTC is an ordered sequence of points:

\[
\{x | x = p_1 + (p_2 - p_1) * z \\
\quad \text{where} \\
\quad p_1 \neq p_2 \text{ and } 0 < z < 1 \\
\quad x = p_1 + (p_2 - p_1) * z \\
\quad x = p_2 + (p_3 - p_2) * z \\
\quad \ldots \\
\quad x = p_{n-1} + (p_n - p_{n-1}) * z
\]

Points other than the first and the last points are called intermediate points; these form the shape of the OTC. A loop must have two intermediate points. There is no limit (depending upon implementation) on intermediate points for an OTC.

The object of this class has a unique OTC ID. The data members (attributes) of this class are pointsequence, 1-T (systemtime), length, parentonecell (Figure 3.24). The life of each OTC is represented by a 1-T \([T_{\text{Start}}, T_{\text{End}}]\). This object can either be born or die. Therefore, Point-Time \(T_{\text{Start}}\) represents the birth time and \(T_{\text{End}}\) represents the death time of this object. The OTC object must have two or more point objects and a point object may have zero or more OTC objects (Figure 3.22). Each OTC object may have one or more children and each child (OTC) must have a parent (OTC).

```
OneTCell

<<index>> + pointsequence: sequence(Point)
<<time>> + systemtime: 1-T
<<query>> + length: Real
<<index>> + parent: OneTCell

<<dynamic>> + create(P: Sequence(Point)): Boolean
<<dynamic>> + kill(P: OneTCell): Boolean
<<dynamic>> + destroy(P: OneTCell): Boolean
<<dynamic>> + reincarnate(P: OneTCell): Boolean

<<update>> + initialize(): Boolean

......
```

---
3.5.3.3 TwoTCellClass

TwoTCellClass is a subclass of SpatioTemporalClass (Figure 3.22). The object of this class is a TTC object. A TTC is a two-dimensional object bounded by a closed cycle of ZTCs and OTCs. Any \( j \)-th TTC = \( \text{set}\{\text{OTC}_1, \text{OTC}_2, \ldots, \text{OTC}_i, \ldots, \text{OTC}_n\} \), where \( i = \{1, 2, \ldots, n\} \). When \( i = 1 \), the first and last point object and the start and end ZTC are identical.

The data members (Figure 3.25) of TwoTCellClass are a set of onetcells, \( 1-T \) (systemtime), area, perimeter and parent (TwoTCell). The life of each TTC is depicted by \( 1-T [T_{\text{From}}, T_{\text{Until}}] \). Like OTC, this object too can either be born or die. The birth and death times are represented by two point times, \( T_{\text{From}} \) and \( T_{\text{Until}} \) respectively. A TTC must have one or more OTC(s) and an OTC may have zero or two TTC(s). Each TTC object may have one or more children and each child (TTC) must have a parent (TTC).

3.5.3.4 Object and versions

Access to the previous states of an object is an inherent part of many atemporal applications, but it is a fundamental requirement in TGIS. Applications such as CAD, CAM, CASE, intelligent office etc. are a few examples of applications where the object and its versions are maintained (Khoshafian, 1993). Objects in a TGIS are not destroyed, rather they are kept with a valid time stamp and tied with a version mechanism.

For any object ZTC, OTC or TTC, \( T_{\text{Until}} \) could be NULL, which shows the object is still alive (active). If \( T_{\text{Until}} \) is not-null, then it indicates that the object is dead (inactive). Therefore, all active objects are represented with time stamp \( [T_{\text{From}}, T_{\text{Until}}] \) and inactive with \( [T_{\text{From}}, T_{\text{Until}}] \). Partition is not possible for the ZTC object, while OTC and TTC can be partitioned and need a time period to define their lives. The objects OTC and TTC have a single parent, keeping track of parent, grandparent, great-grandparent etc., each with a different life period. The object ZTC does not have any parents.

Each object has a number of properties, and changes in the object can be characterized by a set of properties. For example, an \( n \)-tcell could be defined by a number of properties, i.e., ZTC has location and boundary; OTC has location, boundary, co-boundary and length; and TTC has location, area, co-boundary etc. We can classify these properties as essential and non-essential properties (discussed in Chapter-2). Change in an essential property generates a new object, while change in a non-essential property spawns a new version of the object. The essential property of an \( n \)-tcell is its space and geometry. Therefore, as long as a cell occupies the same space and has the same geometry, it is considered as the same object. To simplify the model, only the essential properties of an \( n \)-tcell are considered, while other non-essential properties (area, length, boundary etc.) are considered as an operation (behavior) of an \( n \)-tcell. Therefore, no mutation is possible for any \( n \)-tcell. An \( n \)-tcell can be born, die and/or be reincarnated. The mutation (the process of generating versions) of objects is discussed in Chapter-5. In spatio-temporal databases, a die or kill operation differs from a delete operation, as the latter is merely a purge operation. These operations are dealt with in Chapter-4.

3.5.3.5 Spatio-temporal consistency rules

The following are the spatio-temporal topological consistency rules for the objects of SpatioTemporalClass:
1] Every OTC is bounded by two ZTCs.
2] For every OTC there are two cells (left and right TTC).
3] Every TTC is bounded by a closed cycle of ZTCs and OTCs.
4] OTCs intersect only in ZTCs.
5] Every ZTC is surrounded by a closed cycle of OTCs and TTCs.
6] If ZTC and OTC are the members of TTC, then the time \((T_{\text{From}})\) of TTC must be equal to the maximum time \((T_{\text{From}})\) of OTC. Similarly, the time \((T_{\text{From}})\) of OTC must be equal to the maximum time \((T_{\text{From}})\) of ZTC.

\[
\prod_{i=2}^{\infty} T_{\text{From}}(i - tcell) = \text{Max}(T_{\text{From}}((i - 1) - tcell)) \land T_{\text{From}}((i - 1) - tcell) = \text{Max}(T_{\text{From}}((i - 2) - tcell))
\]

where \(\forall (i-1)-tcell \land (i-2)-tcell \in (i-tcell)\)

7] The \(T_{\text{Until}}\) of OTC\_parent must be less than or equal to the \(T_{\text{From}}\) of OTC\_child. The same condition is valid for TTC.

\[
\prod_{i=1}^{\infty} (T_{\text{Until}}(i - tcellparent)) \leq (T_{\text{From}}(i - tcellchild))
\]

8] The \(T_{\text{Until}}\) of any \(i-tcell\) is greater than or equal to the \(T_{\text{From}}\) of the same \(i-tcell\).

\[
\prod_{i=1}^{\infty} (T_{\text{Until}}(i - tcell)) \geq (T_{\text{From}}(i - tcell))
\]

9] Every ZTC, OTC and TTC is bounded by interval time \(1 - T\) \([T_{\text{From}}, T_{\text{Until}}]\). \(\forall n - tcell \in [1 - T][T_{\text{From}} \leq T_{\text{Until}}]\)

10] Any inactive \(i-tcell\) cannot be a boundary of an \(i+1\)-tcell.

\(\forall i-tcell \notin (i+1)-tcell\)

where \(i-tcell\) is an inactive cell and \((i+1)-tcell\) is an active cell.

11] Any inactive \(i-tcell\) cannot be a co-boundary of an \(i-1\)-tcell.

\(\forall i-tcell \notin (i-1)-tcell\)

where \(i-tcell\) is an inactive cell and \((i-1)-tcell\) is an active cell.

An active \(n\)-tcell is not allowed to have an inactive boundary \((n-1)\)-tcell, whereas an inactive \(n\)-tcell can have an active boundary of \((n-1)\)-tcell. For example, an active OTC cannot have inactive ZTC(s), whereas, an inactive OTC can be defined by active ZTC(s). Figure 3.26 is the manifestation of some simple valid and invalid spatio-temporal configurations for ZTC, OTC and TTC. Active (alive) objects are indicated in black shade and dead (inactive) by gray shade and/or dashed line. This imposes consistency constraints while constructing or updating the spatio-temporal database. Some of the singularities of cells are depicted in Figure 3.26, e.g., a TTC (A) has one ZTC (n), a TTC (B) has one OTC (a1) and a TTC (D) has one OTC (a2).

The cells forming a higher-dimensional object do not necessarily have the same time stamp. For example, 1-tcell could be defined with two 0-tcells of different time stamps. Inherently, cells carry geometric constraints in their structures. These constraints are used to test the consistency of the model in terms of the properties of the individual cells and of the global relational scheme (Corbett, 1979 and Kainz, 1995). For example:

- Rule 1 ensures that every OTC is bounded by one start and one end ZTC.
- Rule 2 preserves the left and right areas of each OTC for a given orientation.
- Rule 3 guarantees that the TTCs have a closed boundary.
- Rules 4 and 5 enforce the planar topology.
- Rules 1 to 5 are obligatory for the partition of space into area features.
- Rule 1 is necessary for network analysis.
Rules 3 and 5 retain the ordering information about ZTC and TTC. This ordering of cells is maintained over time; for example consider Figure 3.27. Two types of ordering can be defined in TCC: ordering of ZTC and TTC. Ordering of ZTC ($n_2$) and TTC ($B$) at times $T_1$ and $T_2$ can be defined as:

\[
\begin{align*}
\text{ZTCT}_1(n_2) &= \{ a_2, B, a_1, \emptyset, a_3, C, a_2 \} \\
\text{ZTCT}_2(n_2) &= \{ a_2, B, a_1, \emptyset, a_4, D, a_2 \} \\
\text{TTC}_1(B) &= \{ n_2, a_2, n_1, a_1, n_2 \} \\
\text{TTC}_2(B) &= \{ n_2, a_2, n_1, a_1, n_2 \}
\end{align*}
\]

The ordering of TTC is time-invariant, while the ordering of ZTC is not (Figure 3.27). If C is a tcell, then the ordering (O) of ZTC and TTC at time t can be expressed as:

\[
\begin{align*}
O(ZTC) &= \{ C'_i, C''_i, C'''_i, \ldots, C'_i \mid i = 0 \text{ and } t = 0-T \} \\
O(TTC) &= \{ C'_i, C''_i, C'''_i, \ldots, C'_i \mid i = 0 \text{ and } t = 0-T \}
\end{align*}
\]

Rules 6 to 10 ensure the temporal and spatio-temporal consistency between cells. In the spatial domain rules 1 to 5 are well understood. We elaborate rules 6 to 11 and associated assertions. Rule 8 is derived from temporal consistency rule 1 (§ 3.5.1). Rule 9 is straightforward and does not need any explanation.

Rule 7: We first demonstrate the process of creating the parents of OTC and TTC. As mentioned earlier, there is no partition for ZTC and therefore there is no parent of ZTC. Figure 3.28 illustrates this process. At time $T_1$, there was one OTC ($a$), with parent null ($\emptyset$), consisting of two ZTCs ($n_1$ and $n_2$). At time $T_2$, $a$ split into two parts, $a_1$ and $a_2$, both with parents $a$. At time $T_3$, $a_1$ became the parent of $a_3$ and $a_4$. At time $T_1$, there was one TTC ($A$) with parent null. At $T_2$, $A$ split into two parts, making a pavement for TTC B and C. As the object A is the ancestor of B and C, the parent of B and C is A. At $T_3$, the object B remains unchanged, while two new objects are spawned, i.e., D and E, with C as a parent. The generation of $n$-tcell’s parent is based on the earlier definition of $n$-tcell. The life of each object is indicated as follows:

\[
\begin{align*}
\text{OTC: } a &\to [T_1, T_2], \quad a_1 \to [T_2, T_3], \quad a_2 \to [T_2, \ast], \quad a_3 \to [T_3, \ast] \text{ and } a_4 \to [T_3, \ast]. \\
\text{TTC: } A &\to [T_1, T_2], \quad B \to [T_2, \ast], \quad C \to [T_2, T_3], \quad D \to [T_3, \ast] \text{ and } E \to [T_3, \ast].
\end{align*}
\]

Figure 3.26 Valid and invalid spatio-temporal configurations.

Figure 3.27 Ordering of ZTC and TTC over time.
In the case of OTC, a is the parent of a₁ and a₂. The T₂ of parent (a) is T₂, which is equal to the T₁ of children a₁ and a₂. Similarly, C (TTC) is the parent of children D and E. The T₂ of C is T₃, which is equal to the T₁ of D and E. This object-parent relationship establishes the hierarchy structure.

**Figure 3.29** is a demonstration of Figure 3.28 with 3D (x, y and t) and 2D (x, y) perspectives. It shows the configuration of the temporal cell complex or the evolution of OTC and TTC as viewed from database time (T₁ to T₃) and system time (t₁ to t₃).

Rule 10 sets the necessary conditions for the valid configuration of any n-cell. Figure 3.26 depicts the valid and invalid cell configurations for ZTCs, OTCs and TTCs. An inactive ZTC cannot be an incident of an active OTC. Similarly, an inactive ZTC or OTC cannot be a member of an active TTC. An active cell must have all active lower order cells and an inactive cell must have inactive lower order cells. Rule 11 is similar to rule 10, except it includes the co-boundary condition.
3.5.4 TemporalCellTupleClass

The spatial and temporal relations are briefly discussed in the Chapter-2. Spatio-temporal relations in the cell tuple-based spatio-temporal data model (CTSTDM) are discussed here. As the object is defined as a spatio-temporal object, the topological relations could be defined as spatio-temporal topological relations, i.e., the spatial relations which are valid over time. In the temporal cell complex, Intra cell complex relations, i.e., relations between the cells in the cell complex, can be described using boundary and co-boundary relations. The boundary (\(\partial\)) of an n-tcell are its (n-1) faces at time t. The co-boundary (\(\Phi\)) of an n-tcell produces the (n+1) cells incident with n-tcell at time t. The boundary and co-boundary relations capture two types of topological relationships, i.e., adjacency and containment. Relations between spatial objects can be found based on boundary/co-boundary relations between cells. The boundary and co-boundary relations are encapsulated in a simple temporal cell tuple structure, which is an extension of the cell tuple structure of Brisson (1990). A cell tuple T is an \((n+1)\)-tuple of cells \(\{c_0, c_1, c_2, \ldots, c_n\}\), where any i-cell is incident with a \((i+1)\)-cell.

TemporalCellTupleClass preserves the temporal cell tuple structure (Figure 3.30) and is the aggregation of ZeroTCellClass, OneTCellClass and TwoTCellClass (Figure 3.22). The object of TemporalCellTupleClass has a unique tuple-ID and a unique combination of ZTC, OTC and TTC. Each tuple must have a ZTC, zero or one OTC and zero or one TTC. Therefore, a temporal cell tuple structure encapsulates the spatio-temporal topology of each spatio-temporal object. A temporal cell tuple (TCT) is a set of C and T, i.e.:

\[
\text{TCT} = \{C, T\}
\]

where C is a set of cells

\[
C = \{c_0, c_1, c_2, \ldots, c_n | c_i \in \text{TCC}\}
\]

and

\[
T \text{ is a time interval } (1-T)
\]

\[
T = \{T_{\text{From}}, T_{\text{Until}} | (T_{\text{From}} < T_{\text{Until}}) \land (T_{\text{From}}, T_{\text{Until}} \in \text{ST})\}
\]

Therefore,

\[
\text{TCT} = \{c_0, c_1, c_2, \ldots, c_n, T_{\text{From}}, T_{\text{Until}}\}
\]

Any i-tcell \((c_i)\) is incident with an \((i+1)\)-tcell \((c_{i+1})\). Every \(c_i\) cell is a boundary of a \(c_{i+1}\) cell, where \(0 \leq i \leq n\) and \(n+1\) is the maximum number of cells in each tuple. For \(n = m\), the first cell \(c_0\) is a ZTC, the second cell \(c_1\) is an OTC, the third cell \(c_2\) is a TTC, and \(m\)-cell \(c_m\) is a \(m\)TC. In \(c_0, c_1, c_2, \ldots, c_m, T_{\text{From}}, T_{\text{Until}}\), at time t any i-tcell \((c_i)\) is either a boundary of an \((i+1)\)-tcell \((c_{i+1})\) or co-boundary of an \((i-1)\)-tcell \((c_{i-1})\). The advantage of TCT is that it stores topology implicitly. It is dimension-independent, i.e., it can accommodate objects of dimension \(k (k \geq 1)\), and it encapsulates boundary and co-boundary and order relations over time. We can formalize the spatio-temporal relations (\(\Phi\) and \(\partial\)) history of \(k\)-tcell at time \(T_i\) as:

\[
\Phi(k\text{-tcell})_{T_i} = \{\forall(k+1)\text{tcell} | T_{\text{From}} \leq T_i\}
\]

\[
\partial(k\text{-tcell})_{T_i} = \{\forall(k-1)\text{tcell} | T_{\text{From}} \leq T_i\}
\]

The process of assigning the cell tuples to a ZTC is illustrated in Figure 3.30. Temporal cell tuple is a unique combination of ZTC, OTC and TTC. The world TTC \(W = \{0\}\) is defined NULL. In principle every ZTC gets a TCT. If this OTC is a member of \(W\), then there is only one TCT; if it belongs to OTC or TTC, the TCT are assigned accordingly. Every OTC has two TCTs if it is not a boundary of TTC (except \(W\)), and it gets four TCTs if it is a boundary of TTC (except \(W\)). The number of TCTs for TTC depends upon the number of TTC boundaries.

If the ZTC \(\in \{W\} \land \notin \{\text{OTC, TTC}\}\), where ZTC, OTC and TTC \(\in \{W\}\), then Figure 3.30[a] shows this configuration for ZTC \((n)\), i.e., \(c(n, 0, 0, 1-T)\). The duration or lifetime of this relation is indicated by time interval 1-T. Figure 3.30[b] shows the configuration when the ZTC \(n1\) and \(n2\) are the boundary of OTC \((a1)\). In other words, \(\{n1, n2\} \in \{a1\}\) and \(\{a1\} \in \{W\}\). The tuple \(c1(n1, a1, 0, 1-T)\).
1-T) shows that this tuple belongs to ZTC (n1), OTC (a1), and TTC (0). The c2 shows that it belongs to ZTC (n2), OTC (a1), and TTC (0).

3.5.4.1 Direction and orientation

The cell tuple structure originally proposed by Brisson (1990) does address the direction and orientation of cells (OTC). The TCT structure proposed here extends not only the temporal dimension but also the direction and orientation. The panacea to direction and orientation is provided in the following fashion:

The direction, i.e., start and end (ZTC) of OTC, is maintained in the temporal cell tuple number (ID). The ZTC in the lower tuple number indicates the start of OTC, and in the higher tuple number the end of OTC. Similarly, orientation (left and right TTC) of OTC is preserved in the temporal cell tuple. The lower-order temporal cell tuple indicates the left TTC and the higher-order temporal cell tuple captures the right TTC. This can be formalized as follows.

If $c = \{c_1, c_2, \ldots, c_m, 1-T\}$ is a set of TCT. Let $c_i = \{n, a, A, 1-T\}$ and $c_j = \{n', a, A', 1-T\}$ are the subset of $c$, such that $i < j$. For any OTC $a$:

- start ZTC = $\{n \mid n \in c_i\}$
- end ZTC = $\{n' \mid n' \in c_j\}$
- left TTC = $\{A \mid A \in c_i\}$
- right TTC = $\{A' \mid A' \in c_j\}$

The TCT configuration of Figure 3.30(c] is based on the same principle. Figure 3.31 shows the objects of SpatioTemporalClass, i.e., the temporal cell complex. The TCC consists of the Point object, the ZTC object, the OTC object and the TTC object; to simplify, they are denoted as point (P), node (n), arc (a) and polygon (poly), respectively. Each of the objects has a unique ID and life span. At T1, there were two TTCs, i.e., poly-1 and poly-2; three OTCs, i.e., a1, a2, and a3; and two ZTCs, i.e., n1,
and n2. At T2, two new TTCs (poly-2 and poly-3) are created and poly-1 ceases to exist. Four new OTCs (a4, a5, a6, a7) are born and a1 dies at T2. At the same time, two new ZTCs (n3 and n4) appear. At time T1, the boundary of poly-1 (TTC) was <a1, a3> (OTC) and the boundary of a1 (OTC) was <n1, n2> (ZTC).

Figure 3.31 demonstrates the process of assigning the tuple to the cells of the complex. Cell tuple is denoted by the symbol ‘o’. The symbol ‘o’ is not part of the structure but is merely shown to demonstrate the concepts. Each unique combination gets a single tuple. Therefore, each ZTC of OTC is assigned two tuples (Table 3.1). At base state (T1), there were 12 (no. of OTC+4) temporal cell tuples (c1, c2,……., c12). At time T1, the tuple c1 has the following configuration:

\[
c1 = (1, 1, 0, T1, *)
\]

![Figure 3.31 Spatial configuration at T1 and T2 and corresponding temporal cell tuple structure.](image)

Table 3.1 Temporal cell tuple configuration at T1.

<table>
<thead>
<tr>
<th>TCTID</th>
<th>ZTCID</th>
<th>OTCID</th>
<th>TTCID</th>
<th>ST-From</th>
<th>ST-Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
</tbody>
</table>
3.5.5 AttributeClass

Depending upon the spatial resolution, real life objects are perceived as point, line and area objects. More often in spatial information systems, objects such as trees, poles, wells etc., are regarded as point objects; rivers, roads, railway-tracks etc., are considered line objects; and parcels, land use boundaries, countries, cities etc., are captured as area objects. Information regarding the attributes of these objects are dealt within AttributeClass.

Most of the information has a temporal nature, but not all, at least in the context of the system life cycle or where capturing time is less significant from the application-user perspective. The attribute class contains data that are non-temporal in nature or where time becomes less significant. AttributeClass has three subclasses, i.e., AttributePointClass, AttributeLineClass and AttributeAreaClass (Figure 3.32).

### Table 3.2 Temporal cell tuple configuration at T2.

<table>
<thead>
<tr>
<th>TCTID</th>
<th>ZTCID</th>
<th>OTCID</th>
<th>TTCID</th>
<th>ST-From</th>
<th>ST-Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>T1</td>
<td>*</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>26</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>27</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>T2</td>
<td>*</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>T2</td>
<td>*</td>
</tr>
</tbody>
</table>

![Figure 3.32 AttributeClass hierarchy.](image-url)
Data members (attributes) in these classes are application-dependent and do not require a time stamp. For example, land use classification may not require the time coding of classification (Table 3.3); in most cases the name of the owners or cities or countries remain unchanged (at least during the system lifetime). Where time is indispensable, the attribute data must be coded with a time stamp. AttributeTemporalClass deals with this type of information.

### 3.5.6 AttributeTemporalClass

AttributeClass and TemporalClass are aggregated to form AttributeTemporalClass. AttributeTemporalClass is a meta class of AttributeTemporalPoint-, AttributeTemporalLine- and AttributeTemporalArea-Class (Figure 3.33). As mentioned earlier, data members of AttributeClass are application-dependent; this is also the case with AttributeTemporalClass. For example, tree height, or a person’s salary or address may change with the passage of time. This classification of AttributeTemporalClass into three classes provides a semantic link between point, line and area objects (features / real life objects).

The data members and operations are also application-specific and are discussed in more detail in Chapter-5 (modeling urban applications). There it is shown how the different applications can be modeled, what role these classes (AttributeClass and AttributeTemporalClass) play in modeling these applications, and what the ingredients of these classes are in terms of attributes and operation.

### 3.5.7 SpatioTemporalAttributeClass

SpatioTemporalAttributeClass is an aggregation of SpatialClass, TemporalClass (WorldTimeClass and DataBaseTimeClass) and AttributeClass. This class too has three subclasses, i.e., SpatioTemporalAttributePointClass, SpatioTemporalAttributeLineClass and SpatioTemporalAttributeAreaClass (Figure 3.34). The objects of these classes are the abstraction and manifestation of reality, i.e., point (well locations, telephone poles, trees, mountain peaks etc.), line (streets, streams, elevation contours etc.) and area (states, counties, parcels, soil types, land use zones etc.). The objects (features) belong to SpatioTemporalAttributePointClass, SpatioTemporalAttributeLineClass and SpatioTemporalAttributeAreaClass, respectively. The object of each (SpatioTemporalAttributePointClass, SpatioTemporalAttributeLineClass and SpatioTemporalAttributeAreaClass) class has a predecessor (parent) and each object (predecessor / parent) may have a number of successors (children). The object of each class may have a number of versions. The generation of successors and versions is dictated by the particular application. This process is discussed in Chapter-5 (modeling urban applications). In principle, a parent (object) can

<table>
<thead>
<tr>
<th>Land use level-1</th>
<th>Land use level-2</th>
<th>Land use level-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Residential</td>
<td>1.1 Apartment</td>
<td>1.1.1 5 stories</td>
</tr>
<tr>
<td></td>
<td>1.2 Houses</td>
<td>1.1.2 10 stories</td>
</tr>
<tr>
<td>2- Commercial</td>
<td>2.1 Shops</td>
<td>1.1.3 10 + stories</td>
</tr>
<tr>
<td></td>
<td>2.2 Offices</td>
<td>1.2.1 Bungalow</td>
</tr>
<tr>
<td></td>
<td>2.3 Restaurants</td>
<td>1.2.2 Double stories dwelling</td>
</tr>
</tbody>
</table>

Table 3.3 Land use classification.
have many successors (objects) but a successor must have a single parent. An object may have many versions and a version must have an object.

The overall conceptual schema for the unified cell tuple-based spatio-temporal data model (CTSTDM) is presented in Figure 3.35. Most of the classes and various associations among them have already been discussed earlier in this chapter. In Chapter-4, the operations pertaining to SpatioTemporalClass and its subclasses are discussed.

3.6 CONCLUSIONS

Various issues pertaining to the design of spatio-temporal data modeling have been encompassed in this chapter. A simple data structure is provided to address the complexity of spatio-temporal data. Some spatial / topological models are reviewed. The transition from spatial data model to spatio-temporal data model is not straightforward, as many obstacles have to be overcome. Two guiding factors motivate the outgrowth of the unified cell tuple-based spatio-temporal data model: first, the sound mathematical basis (concepts of topology and the concept of cell complexes) and, second, the semantic power of the OO paradigm.

The concept of topology is quite general and can accommodate spatial objects of various dimensions. Earlier work of Worboys on ST-simplicial complexes has been extended to cell complexes and cell tuple structure by unifying the temporal dimensions. The extended cell complex (called temporal cell complexes) is the union of various temporal cells of dimensions 0, 1 and 2. These cells are considered as an object, i.e., ZTC, OTC and TTC, and are treated uniformly. Cyclic, acyclic and interior singularities of cells are incorporated in the temporal cell complex. The spatio-temporal topology is all the time preserved in an implicit manner in a simple temporal cell tuple structure. The structure has been extended to accommodate the direction and orientation of OTC. The original cell tuple structure of Brisson ignored these two spatial relations. Based on boundary and co-boundary relations, the adjacency and containment relations can be discerned at any given time (shown in Chapter-4).
Unlike many spatio-temporal data models, time is considered as the structure component of space. It is treated as a separate dimension, which is similar to but different from the spatial dimension. It is linear and discrete, with three dimensions: WT, DBT and ST. The objects ZTC, OTC, TTC and TCT are recorded at the ST axis, while other objects are synchronized with WT and DBT.

The OO design technique facilitates the modeling of the three components: space, attribute and time in a modular fashion. These components are treated as classes. The aggregation of these classes forms the cell tuple-based spatio-temporal model. One of the great advantages of this model is that spatio-temporal topology is all the time stored in an implicit manner and spatial objects are treated uniformly. The model is dimension-independent. Therefore, the approach can be extended to higher spatial dimensional objects. The cell tuple structure may generate a large number of tuples, but this is not expected to affect the system performance because of its simple structure.

Figure 3.35 Conceptual schema for cell tuple-based spatio-temporal data modeling.
Chapter-4

Operators for Cell Tuple-Based Spatio-Temporal Data Model

4.1 INTRODUCTION

As pointed out earlier, a data model consists of data members (attribute), member functions (operations) and consistency constraints. Chapter-3 dealt with attributes and consistency rules associated with fundamental classes of unified CTSTDM. Various classes and consistency rules have been identified and explained. A conceptual schema, the outgrowth of these classes (hierarchy, association and generalization) and constraints are presented in Figure 3.35. Fundamental operations associated with these classes at abstract level are encompassed in this chapter.

Operations in spatial databases can be categorized as static, e.g., calculating area, length, orientation etc., and dynamic, e.g., adding new node, arc, polygon etc. Normally in atemporal GIS, three fundamental dynamic operations are applied, i.e., insert, delete and update operations. Unlike atemporal GIS, in spatio-temporal databases (TGIS) objects may die or be killed, but they remain in the database with a certain time stamp indicating their life span.

First, the major operators (static and dynamic) will be identified, followed by a detailed discussion of these operators for PointClass, ZeroTCellClass, OneTCellClass, TwoTCellClass and TemporalCellTupleClass. As a result of any operators, the spatio-temporal topology of any \( n \)-tcell may be modified; therefore, the corresponding cell tuple structure will also be discussed. Some of the work on these operators has been presented in Raza and Kainz (2000a).

4.2 FUNDAMENTAL OPERATIONS

Two types of operators can be defined, i.e., static and dynamic operators. Static operators do not affect the system’s state or the status of spatio-temporal objects, e.g., query operators (calculating the length, area, time period, boundary or co-boundary etc). These operators are associated with TemporalCellTupleClass. On the other hand, dynamic operators change the state of the system or the status of the spatio-temporal objects, e.g., creating, deleting or updating an \( n \)-tcell. Normally in atemporal GIS, three fundamental dynamic operations are performed, i.e., insert, delete and update. These operators are associated with PointClass, ZeroTCellClass, OneTCellClass and TwoTCellClass. All dynamic operators are the variation of one of these operations (Worboys, 1995). Unlike in atemporal GIS, in a TGIS objects may die or be killed, but they remain in the database with a certain time stamp indicating their life span. As mentioned earlier, any \( n \)-tcell object can be born or die. Therefore, four fundamental dynamic operators can be distinguished in spatio-temporal databases, i.e., Create, Kill, Reincarnate or Delete (Destroy). These operators are associated with objects (ZTC, OTC or TTC). In spatio-temporal databases, the kill operation is different from the delete operation, as the latter is merely a purge operation. Updating spatio-temporal objects is complex; any update operation affects the other objects, particularly in the unified approach. Any spatial change is the result of the creation (birth) and / or destruction (death) of an \( n \)-tcell. Kill is a protected operation, while the others are public or private.

- The Create operator is equivalent to the usual insert operators. The task of this operator is to create a new and / or update an existing object. This operator specifies the time stamp \([\text{start, }^*]\) of each spatial object, where the upper bound of the time interval is undefined (*). All objects with \([\text{start, }^*]\) time stamps are called active objects.
- The Kill operator kills the spatio-temporal objects by defining the upper bound of the time interval. After being killed objects are called inactive objects. These objects remain in the database only for the query purpose or Reincarnate operator. Therefore, the upper bound (*) is
Chapter-4: Logical Data Modeling

replaced by current system time.

- The operator **Destroy** or **Delete** permanently deletes the spatio-temporal objects from the database. Therefore, they are no longer available for any types of operations (static or dynamic).
- The operator **Reincarnate** turns an inactive object into an active object by replacing the upper bound of time interval to (*).

Generally in a spatial data model, the Create (update) operation is performed by checking the intersection of spatial objects (nodes, arcs or polygons). Spatio-temporal data models demand different treatment of spatial objects, because the existing ones are not thrown out, they are preserved with a valid time stamp. This Create operation is basically an overlay operation. The overlay operation in spatio-temporal databases is much more complex than its spatial counterpart. Computationally, polygon-polygon operations are the most challenging task in vector-based spatial databases (Laurini and Thompson, 1996).

Existing algorithms (e.g., polygon-polygon overlay in spatial domain) cannot be used in the spatio-temporal domain. The traditional overlay algorithms in the literature (Wise, 1997 and 1996; Laurini and Thompson, 1996; Preparata and Shamos, 1990; Sedgewick 1988 etc.) do not consider the time element. Moreover, most of the algorithms are for line-line intersection, instead of line-line overlay. To demonstrate the limitations of existing (in spatial domain) algorithms, two examples are given to show their limitations of in spatio-temporal databases. The first example shows the insertion of OTC when it intersects with OTC, and the second, the insertion of TTC when it intersects with TTC.

**OTC intersects with OTC:**

Let \( a1 \) be an OTC at time \( T1 \) and \( a1' \) at \( T2 \); then as a result of overlay, existing algorithms produce the result shown in following Figure 4.1. In case [a] object \( a1 \) is lost and in case[b] object \( a1' \) is lost. Therefore, we need an algorithm to produce the geometric output as shown in Figure 4.2.

![Figure 4.1 Overlay operation in spatial database (line-line intersection)](image)

Figure 4.2 shows the configuration needed for the spatio-temporal objects where no object is lost as a result of the overlay operation. All active and inactive (discussed earlier in Chapter-3) objects are kept with a valid time stamp. Object \( a1 \), shown in a dashed line, is inactive because the life span of this object is \((T1,T2)\), whereas the other objects are active because their time stamp is \((T2,*\)). No information is lost in this configuration. This configuration or partition is used in this research.

![Figure 4.2 Overlay operation required for spatio-temporal databases (OTC-OTC intersection)](image)

**TTC intersects with TTC:**
Consider two TTCs, A at time T1 and B at time T2. Existing algorithms produce three TTCs, i.e., A’, B’ and AB. The original TTC, i.e., A, is lost during the overlay process (Figure 4.3). In Figure 4.4, the original TTC A is preserved with time stamp (T1,T2), whereas the other objects (A’, B’ AB) are active with time stamp (T2,*).

![Figure 4.3 Overlay operation in spatial database (polygon-polygon intersection).](image)

![Figure 4.4 Overlay operation in spatio-temporal database (TTC-TTC intersection).](image)

Why is the structure depicted in Figure 4.1 and Figure 4.2 needed? This can be explained by the following example. Figure 4.5, shows the evolution of change associated with a STAO and its spatial component (geometric part). At time T1, a school, an object O1 (STAO), is represented by a point (ZTC) located at n1. At time T2, O1 moves to another location n2. At time T1, the road is represented by OTC (a1); the same road moves to another location and is represented by a2. The ITC building at time T1 is represented by TTC (poly-1), which is shifted to a new location at time T2. What needs to be stored are the previous locations of these objects. Any update (insert or delete) operation must preserve the earlier location of the feature object. Therefore, in TGIS, a feature object may be born, die, be reincarnated, move, change shape or change size, but not all these mutations are possible for STO or the geometric part of an n-tcell. An STO or n-tcell can only be born or die.

![Figure 4.5 Feature and geometric change.](image)

When an n-tcell can be born or die is an important decision. It is logical to investigate the situations where an n-tcell is changed as a result of the updating (insert or destroy) process in the spatio-temporal database. These operations are necessary for designing the algorithms of various operators.
In the unified spatio-temporal data model, when a ZTC, OTC or TTC is inserted, the following scenario can be expected:
- A ZTC may intersect with ZTC, OTC or TTC
- An OTC may intersect with ZTC, OTC or TTC and
- A TTC may intersect with ZTC, OTC or TTC.

Figure 4.6 shows all nine possibilities when an n-tcell at time T1 may intersect with an n-tcell at time T2. The cross (X) indicates invalid intersection or overlay, i.e., geometrically ZTC cannot intersect with a ZTC object (a node cannot intersect a node). The tick (✓) is a valid overlay operation, e.g., a ZTC can intersect with an OTC or TTC object. In each case, there are various possibilities, e.g., ZTC may intersect at the boundary of OTC or the interior of OTC. The three topological invariants of spatial objects (n-tcells) are boundary, interior and exterior.

This point-set topology approach is employed to analyze these intersections. Only the boundary (∂) and interior (°) of OTC and TTC are considered in order to investigate these intersections. The intersection at the exterior of any n-tcell is straightforward. The boundary of ZTC is empty (∅). A similar approach, i.e., point-set topology, has been employed by Egenhofer et al. (1994); Egenhofer (1993); Egenhofer et al. (1993); Egenhofer and Franzosa (1991); Egenhofer and Herring (1990); Egenhofer (1989); Egenhofer et al. (1989); Pullar and Egenhofer (1988) etc. to identify and/or compare topological relationships between n-dimensional objects embedded in \( \mathbb{R}^n \). Create and Kill operators needed for ZeroTCellClass, OneTCell-Class and TwoTCellClass for 2-TemporalCellComplex are discussed. These operations are closed as an intersection of two n-tcells (\( 0 \leq n \leq 2 \)) always produces an n-TemporalCellComplex. It is assumed that all n-tcells are inside the universal (void) TTC. The syntax for each operation of the ZeroTCellClass, OneTCellClass and TwoTCellClass is given in Figure 4.7, Figure 4.12 and Figure 4.35, respectively. Any birth or death of the n-tcell is dictated by the definition of the n-tcell (see Chapter-3).

4.3 ZERO TCELLCLASS

Four fundamental operations for ZeroTCellClass are shown in Figure 4.7 and are discussed in the following sections.

4.3.1 ZTC insertion (Create operation)

A ZTC is inserted (in world TTC) along with a point object when it does not intersect with any other spatio-temporal objects of \( 0 \leq n \leq 2 \). Whenever a ZTC coincides with an intermediate point (of OTC), no new point object is created. Various possible scenarios and actions that may occur when a ZTC intersects (while inserting) with spatio-temporal objects of \( 0 \leq n \leq 2 \) are discussed in the following sections.

4.3.1.1 ZTC coincidence

This case is trivial because the ZTC cannot be added to an existing one.
4.3.1.2 ZTC with OTC

There are four possibilities when ZTC can intersect with OTC (Figure 4.8).

a] At a boundary of OTC: operation is rejected, as it is similar to ZTC to ZTC insertion.

b] At the intermediate point of OTC: Kill OTC (a1) and TCTs (c1 and c2); Create ZTC (n3), two OTCs (a2 and a3) and TCTs (c3, c4, c5 and c6).

c] At a point between the intermediate points of OTC: Create new point object (p’), new ZTC (n3) and new TCT (c3, c4, c5 and c6).

d] At a point between of two ZTC: same as # c].

The configuration formed in Figure 4.8[b], [c] and [d] is called a 1-temporal complex (1-TC) or 1D spatio-temporal object (OSTO). A 1-TC is a collection of OTCs, such that each OTC has a common face.

4.3.1.3 ZTC with TTC

A ZTC may intersect with TTC at the boundary or in the interior.

1] At the boundary of TTC:
   The boundary of TTC is OTC; therefore it is similar to the ZTC to OTC intersection, where we had three possibilities. ZTC can intersect at:
   a) the boundary of OTC: operation is rejected, as it is similar to ZTC to ZTC insertion.
   b) the point between the intermediate points or between ZTC of OTC (Figure 4.9): Kill TTC (A), OTC (a2) and TCTs (c1, c3, c5… c8); Create a point object (p’), a ZTC (n3), OTCs (a3, a4), a TTC (A’) and TCTs (c9, c10… c18).
   c) the intermediate point of OTC: same as [b], except no point object is created.

2] The interior of TTC:
   The process is shown in Figure 4.10. Only new ZTC and associated TCT are added.
4.3.2 ZTC Kill operator (↓)

A ZTC (isolated) can be killed when it is not a boundary or face of OTC by simply closing the upper bound of the time interval. The case of ZTC being a face of OTC is discussed below.

1] ZTC associated with OTC:
   A ZTC can be associated with OTC in one of the following ways (Figure 4.11).
   
   a) A ZTC can be a boundary of a single OTC (Figure 4.11[a]).
   b) A ZTC can be a boundary of two OTCs (Figure 4.11[b]).
   c) A ZTC can be a boundary of more than two OTCs (Figure 4.11[c]).

   In all cases, the operation is discarded, as an active OTC cannot have an inactive ZTC.

2] ZTC associated with TTC:
   The same convention applies as in the case of ZTC associated with OTC. Therefore, no further elaboration is considered.

![Figure 4.9 Create ZTC: at the boundary of TTC.](image)

![Figure 4.10 Create ZTC: at interior of TTC.](image)
4.4 ONETCELLCLASS

Four fundamental operations associated with this class are depicted in Figure 4.12 and are discussed in following sections.

4.4.1 OTC insertion (Create operator)

Insertion of OTC is a recursive operation, which starts by inserting the boundary of OTC, i.e., zero, one or two ZTCs. Like ZTC, this operation can also be viewed from three perspectives, i.e., when OTC intersects with ZTC, OTC and TTC. The intersection of ZTC with OTC and OTC with ZTC is not symmetric, like in the spatial data structure. In the spatio-temporal case, it is vital what exists first, i.e., ZTC or OTC.

4.4.1.1 OTC with ZTC

There are three possibilities when OTC can intersect with ZTC:

a] ZTC intersects at boundary of OTC (Figure 4.13[a]):
   If one of the boundaries of OTC coincides with ZTC (n1), then n1 becomes a boundary of the new OTC (a1). A new ZTC (n2) is added, which becomes a second boundary of OTC (a1). The TCT c1 is killed and new TCTs (c2 and c3) are added.

b] ZTC intersects at intermediate point of OTC (Figure 4.13[b]):
   Two new ZTCs (n2 and n3) are added. Two new OTCs are added, i.e., a1 and a2, with boundaries <n2,n1> and <n1,n3>, respectively. TCT c1 is killed. New TCTs (c2, c3, c4 and c5) are added.

c] ZTC intersects at point between intermediate or ZTC of OTC (Figure 4.13[c]):
   Two new ZTCs (n2 and n3) are added. Two new OTCs (a1 and a2) are added, with boundaries <n2,n1> and <n1,n3>, respectively. TCT c1 is killed. New TCTs (c2, c3, c4 and c5) are added.
4.4.1.2 OTC intersects with OTC

This operation could be very complex, i.e., there are a number of possibilities when OTC at T1 can intersect with OTC at T2. To simplify, we use a point-set approach. Let \( a_1 \) be an OTC (with no intermediate points) with \( \partial a_1 \) and \( ° a_1 \) at time T1 and \( a_1' \) be an OTC with \( \partial a_1' \) and \( ° a_1' \) at time T2. A 2x2-intersection can be defined as:

\[
\begin{align*}
\text{OO} &= \begin{pmatrix}
\tilde{a} a_1 \cap \tilde{a} a_1' & \tilde{a} a_1 \cap ° a_1' \\
° a_1 \cap \tilde{a} a_1' & ° a_1 \cap ° a_1'
\end{pmatrix}
\end{align*}
\]

When empty \([\emptyset]\) and non-empty \([\neg\emptyset]\) intersections (single) of two objects are considered, then there are \(2^4 = 16\) possible intersections between OTC at time T1 and OTC at time T2. Not all are applicable or valid. Some of them are symmetric. The valid and invalid cases are when:

1. \( \tilde{a} a_1 \) intersects with \( \tilde{a} a_1' \),
2. \( \tilde{a} a_1 \) intersects with \( ° a_1' \),
3. \( ° a_1 \) intersects with \( \tilde{a} a_1 \),
4. \( ° a_1 \) intersect with \( ° a_1' \),
5. \( \tilde{a} a_1 \) intersects with \( \tilde{a} a_1' \) and \( \tilde{a} a_1 \) intersects with \( ° a_1' \),
6. \( \tilde{a} a_1 \) intersects with \( ° a_1' \) and \( ° a_1 \) intersects with \( \tilde{a} a_1 \),
7. \( ° a_1 \) intersects with \( ° a_1' \),
8. \( \tilde{a} a_1 \) intersects with \( \tilde{a} a_1' \) and \( ° a_1 \) intersects with \( \tilde{a} a_1 \),
9. \( ° a_1 \) intersects with \( ° a_1' \),
10. \( \tilde{a} a_1 \) intersects with \( \tilde{a} a_1' \) and \( ° a_1 \) intersects with \( ° a_1' \),
11. \( ° a_1 \) intersects with \( \tilde{a} a_1 \), \( \tilde{a} a_1 \) intersects with \( ° a_1' \) and \( ° a_1 \) intersects with \( \tilde{a} a_1 \),
12. \( ° a_1' \) intersects with \( ° a_1' \) and \( \tilde{a} a_1 \) intersects with \( ° a_1' \),
13. \( ° a_1 \) intersects with \( ° a_1' \) and \( \tilde{a} a_1 \) intersects with \( ° a_1' \),
14. \( \tilde{a} a_1 \) intersects with \( ° a_1' \) and \( ° a_1' \) intersects with \( ° a_1 \),
15. \( \tilde{a} a_1 \) intersects with \( \tilde{a} a_1' \), \( \tilde{a} a_1 \) intersects with \( ° a_1' \), \( ° a_1 \) intersects with \( \tilde{a} a_1 \) and \( ° a_1 \) intersects with \( ° a_1' \),

Figure 4.14 is the illustration of these intersections, where the number of dimension is ignored. This is a simplified case of the line-line intersection problem. The existing algorithm does not provide solutions to these types of cases. For example, line-line (LL-) intersection no. 4 is a very simple case where the interior of arc / line A intersects with the interior of arc / line B. The existing algorithm generates four arcs as a result of two intersecting arcs, because temporal elements are not taken into consideration. What is expected in spatio-temporal databases is an intersection of the spatio-temporal arc / line (OTC), where original arc(s) / line(s) are preserved with valid time stamps, indicating its (arc) life span.
<table>
<thead>
<tr>
<th>Intersection</th>
<th>Illustration</th>
<th>Spatio-Temporal Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset \cap \emptyset$</td>
<td>$\emptyset \cap \emptyset$</td>
<td>$\emptyset \cap \emptyset$</td>
</tr>
<tr>
<td>$\emptyset \cap \emptyset'$</td>
<td>$\emptyset \cap \emptyset'$</td>
<td>$\emptyset \cap \emptyset'$</td>
</tr>
<tr>
<td>$\emptyset \cap \emptyset''$</td>
<td>$\emptyset \cap \emptyset''$</td>
<td>$\emptyset \cap \emptyset''$</td>
</tr>
<tr>
<td>$\emptyset \cap \emptyset'''$</td>
<td>$\emptyset \cap \emptyset'''$</td>
<td>$\emptyset \cap \emptyset'''$</td>
</tr>
</tbody>
</table>

**Figure 4.14 OTC-OTC intersection (single segment).**
Therefore, based on the definition of the temporal cell complex, the intersection of spatio-temporal objects (OTC in this case) will produce four active (alive) arcs (a2, a3, a4 and a5) and one inactive (dead) arc (a1), as shown in Figure 4.15. Only active objects can participate in the intersection process, which means dead objects are by-passed while detecting the intersection.

**Figure 4.15 Interior of OTC intersects with interior of OTC.**

Figure 4.14 is the list of all possible cases for OTC-OTC intersection. Not all are valid, e.g., (5, 6, 8, 11 and 15). Table 4.1 is the list of valid OTC-OTC intersections. It shows the time (Æ) associated with old and new objects and the parent (Å) of each object. For example, in the case of intersection 4, the lifetime of OTC a2 is (T1,*) and the parent of this object is a1.

![Diagram](image-url)

<table>
<thead>
<tr>
<th>Intersection</th>
<th>Life Span</th>
<th>Parent</th>
<th>Intersection</th>
<th>Life Span</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a1Æ(T1,<em>) a2Æ(T2,</em>)</td>
<td>0Æ-a1 0Æ-a2</td>
<td>10</td>
<td>A1Æ(T1,*)</td>
<td>0Æ-a1</td>
</tr>
<tr>
<td>2</td>
<td>a1Æ(T1,<em>) a2Æ(T2,</em>) a3Æ(T2,*)</td>
<td>0Æ-a1 0Æ-a2 0Æ-a3</td>
<td>12</td>
<td>A1Æ(T1,T2) A2Æ(T2,<em>) A3Æ(T2,</em>) A4Æ(T2,*)</td>
<td>0Æ-a1 a1Æ-a2 a1Æ-a3 0Æ-a4</td>
</tr>
<tr>
<td>3</td>
<td>a1Æ(T1,T2) a2Æ(T2,<em>) a3Æ(T2,</em>) a4Æ(T2,*)</td>
<td>0Æ-a1 a1Æ-a2 a1Æ-a3 0Æ-a4</td>
<td>13</td>
<td>A1Æ(T1,T2) A2Æ(T2,<em>) A3Æ(T2,</em>)</td>
<td>0Æ-a1 a1Æ-a2 0Æ-a3</td>
</tr>
<tr>
<td>4</td>
<td>a1Æ(T1,T2) a2Æ(T2,<em>) a3Æ(T2,</em>) a4Æ(T2,<em>) a5Æ(T2,</em>)</td>
<td>0Æ-a1 a1Æ-a2 a1Æ-a3 0Æ-a4 0Æ-a5</td>
<td>14</td>
<td>A1Æ(T1,<em>) A2Æ(T2,</em>)</td>
<td>0Æ-a1 0Æ-a2</td>
</tr>
<tr>
<td>7</td>
<td>a1Æ(T1,T2) a2Æ(T2,<em>) a3Æ(T2,</em>) a4Æ(T2,*)</td>
<td>0Æ-a1 a1Æ-a2 a1Æ-a3 a1Æ-a4</td>
<td>16</td>
<td>A1Æ(T1,<em>) A2Æ(T2,</em>)</td>
<td>0Æ-a1 0Æ-a2</td>
</tr>
<tr>
<td>9</td>
<td>a1Æ(T1,<em>) a2Æ(T2,</em>) a3Æ(T2,*)</td>
<td>0Æ-a1 0Æ-a2 0Æ-a3</td>
<td>17</td>
<td>A1Æ(T1,<em>) A2Æ(T2,</em>)</td>
<td>0Æ-a1 0Æ-a2</td>
</tr>
</tbody>
</table>

**Table 4.1 OTC’s life span and parents.**

Figure 4.14 shows the simplified cases of OTC-OTC intersection, where orientation is assumed to be clockwise direction (ni → nj, where i < j) and it consists of a single segment. This can be generalized for other orientations and n-segments.

**Orientation:**
Orientation can be generalized by considering all possible orientations for OTC at T1 and T2. OTC at time T1 can have an orientation of clockwise or anti-clockwise direction, as, similarly, can OTC at time T2. Therefore, it leads to four possibilities for the intersection of OTC at times T1 and T2.
- OTC at T1 is clockwise and at T2 anti-clockwise.
• OTC at T1 is anti-clockwise and at T2 clockwise.
• Both are anti-clockwise.

<table>
<thead>
<tr>
<th>OO</th>
<th>a1 at T1</th>
<th>a2 at T2</th>
<th>a1 at T1</th>
<th>a2 at T2</th>
<th>a1 at T1</th>
<th>a2 at T2</th>
<th>a1 at T1</th>
<th>a2 at T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n1</td>
<td>n2</td>
<td>n3</td>
<td>a1</td>
<td>a2</td>
<td>n1</td>
<td>n2</td>
<td>n3</td>
</tr>
<tr>
<td>2</td>
<td>n1</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
<td>n2</td>
<td>a2</td>
<td>n3</td>
<td>a3</td>
</tr>
<tr>
<td>3</td>
<td>n2</td>
<td>a4</td>
<td>n3</td>
<td>n4</td>
<td>n2</td>
<td>a4</td>
<td>n3</td>
<td>n4</td>
</tr>
<tr>
<td>4</td>
<td>n1</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
<td>n2</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
</tr>
<tr>
<td>5</td>
<td>n1</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
<td>n2</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
</tr>
<tr>
<td>6</td>
<td>n1</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
<td>n2</td>
<td>a2</td>
<td>a3</td>
<td>n4</td>
</tr>
</tbody>
</table>

Figure 4.16 Result of overlay with possible orientations of OTC at times T1 and T2.

This orientation has obvious effects on the boundary of OTC, i.e., start and end ZTC (Figure 4.16). For example, in the case of intersection 4, the OTC a2 and a3 can be defined by <n1, n5> and <n5,
n2>, respectively, when it is clockwise. The same are defined by <n5, n1> and <n2, n5>, when it is anti-clockwise. All these possible orientations have to be taken into account when updating the OTC. Figure 4.16 illustrates these orientations of valid OTC-OTC intersections.

**n-segment:**
Figure 4.14 shows the OTC-OTC intersection, where each OTC consists of a single segment. This can be generalized n-segments. Intersection 1 is straightforward and does not require any generalization. The rest of the cases are described below, and are based on the valid cases / intersections (1, 2, 3, 4, 7, 9, 10, 12, 13, 14, 16) of Figure 4.14. The ZTC is represented by symbol (●) and the intermediate point by (○). To simplify, the orientation of OTC at T1 and T2 is assumed to be the same. However, OTC is extended to n-segments. To simplify, the orientation of OTC at T1 and T2 is assumed to be the same. Only major combinations are discussed here; all other combinations are variants of these intersections.

### 4.4.1.2.1 Boundary of OTC Intersects with boundary of OTC'

A ZTC (n3), an OTC (a2) and TCTs (c3 and c4) are added (Figure 4.17).

### 4.4.1.2.2 Boundary of OTC Intersects with interior of OTC'

Two new ZTCs (n3 and n4) and two new OTC (a2 and a3) are added by defining boundaries as <n3, n2> and <n2, n4>, respectively. The existing ZTC (n2) is used in defining the boundaries of OTC. New TCTs (c3, c4, c5 and c6) are added (Figure 4.18).

### 4.4.1.2.3 Interior of OTC Intersects with boundary of OTC'

The boundary of OTC (a1' at time T2) can intersect with the interior of OTC (a1 at time T1) in two ways:

a] when it intersects at the intermediate point, a new ZTC is generated and the existing OTC (a1) is divided into two parts (a2 and a3), while a1' remains unchanged (Figure 4.19[a]). OTC a1 and TCT c1 and c2 are killed. ZTC (n3), three new OTCs (a2, a3 and a4) and new TCTs (c3..c6) are added.

b] when a1' intersects at any point between the intermediate point or ZTC of a1 (Figure 4.19[b]). A new point and ZTC are created and a1' is divided into two parts (a2 and a3). OTC (a1) and

88
TCTs (c1 and c2) are killed. A new point, ZTC (n3), three new OTCs (a2, a3 and a4) and new TCTs (c3, .. c6) are added.

4.4.1.2.4 Interior of OTC intersects with interior of OTC

The interior of OTC (a1) at time T1 can intersect with interior of OTC (a’) at time T2 in the following ways:

a] both intersect at the intermediate point (Figure 4.20[a]).
b] intermediate point of a’ intersects at a point between the intermediate points of a1 (Figure 4.20[b]).
c] intermediate point of a1 intersects at a point between the intermediate points of a’ (Figure 4.20[c]).
d] at the intermediate point of both a1 and a’ (Figure 4.20[d]).

The first three cases (a, b and c) have the same steps for inserting OTC, except in the third case where a new point is calculated as an intersection. All result in four new OTCs. Kill OTC (a1) and TCTs (c1 and c2). Add ZTCs (n3, n4 and n5), OTCs a2, a3, a4 and a5 with boundaries <n1,n4>, <n4,n2>, <n3,n4> and <n4,n5>, respectively and TCTs (c3, c4, ... c10). In the fourth case [d], five new OTCs are generated as shown in Figure 4.20[d].

4.4.1.2.5 Interior of OTC intersects with boundary and interior of OTC

The interior of OTC (a1 at time T1) can intersect with the interior and boundary of OTC (a’ at time T2) in several ways (Figure 4.21). In all cases, a1 is killed.
a] Boundary of $a_1'$ intersects with the intermediate points of $a_1$ (Figure 4.21[a]):
Create new ZTCs ($n_3$ and $n_4$), OTCs ($a_2$, $a_3$ and $a_4$) and TCTs ($c_3$, $c_4$, $c_5$, $c_6$, $c_7$ and $c_8$). Kill OTC ($a_1$) and TCTs ($c_1$ and $c_2$).

b] Boundary of $a_1'$ intersects with the interior of $a_1$ (Figure 4.21[b]):
Create new ZTCs ($n_3$ and $n_4$), OTCs ($a_2$, $a_3$ and $a_4$) and TCTs ($c_3$, $c_4$, $c_5$, $c_6$, $c_7$ and $c_8$). Kill OTC ($a_1$) and TCTs ($c_1$ and $c_2$).

c] Boundary of $a_1'$ intersects with the intermediate points of $a_1$, and interior of $a_1$ intersects with the interior of $a_1'$ (Figure 4.21[c]):
Create new ZTCs ($n_3$, $n_4$ and $n_5$), OTCs ($a_2$, $a_3$, $a_4$, $a_5$, $a_6$ and $a_7$) and TCTs ($c_3$, $c_4$, ..., $c_{14}$). Kill OTC ($a_1$) and TCTs ($c_1$ and $c_2$).

### 4.4.1.2.6 Interior of OTC' Intersects with boundary and interior of OTC

The interior of OTC ($a_1'$) can intersect with the boundary and interior of OTC ($a_1$) in many ways; three major combinations are shown here, others are variants of these intersections:

a] Boundary of $a_1$ intersects with intermediate points of $a_1'$ (Figure 4.22[a]):
Create new ZTCs ($n_3$ and $n_4$), OTCs ($a_2$ and $a_3$) and TCTs ($c_3$, $c_4$, $c_5$, $c_6$, $c_7$ and $c_8$). Kill TCTs ($c_1$ and $c_2$).

b] Boundary of $a_1$ intersects with the interior of $a_1$ (Figure 4.22[b]):
Create new ZTCs ($n_3$ and $n_4$), OTCs ($a_2$, $a_3$ and $a_4$) and TCTs ($c_3$, $c_4$, $c_5$, $c_6$, $c_7$ and $c_8$). Kill TCTs ($c_1$ and $c_2$).
Chapter 4: Logical Data Modeling

4.4.1.2.7 Interior of OTC intersects with boundary and interior of OTC'

Two cases can be contemplated; the only difference is the insertion of the point object (associated with \( n_3 \) and \( n_4 \)) in the second case (Figure 4.23).

a] Boundary of \( a1 \) can intersect with the intermediate point of \( a1' \) (Figure 4.23[a]).

b] Boundary of \( a1 \) can intersect with the point between ZTC and the intermediate point of \( a1' \) (Figure 4.23[b]).

In both cases [a] and b), operators create new ZTCs \( n_3 \) and \( n_4 \), OTCs \( a2, a3, a4, a5, a6, a7 \) and TCTs \( c3, c4, \ldots, c14 \); and kill OTC \( a1 \) and TCTs \( c1 \) and \( c2 \).

The other intersections when:

- \( \partial a1 \) intersects with \( \partial a1' \) and \( \partial a1' \) intersects with \( \partial a1 \)
- \( \bar{\partial} a1 \) intersects with \( \bar{\partial} a1' \) and \( \bar{\partial} a1' \) intersects with \( \bar{\partial} a1 \)

are not elaborated on but are depicted in Figure 4.24 and Figure 4.25, respectively.
4.4.1.3 OTC with TTC (OT)

The same conventions are employed as in the case of OTC-OTC intersection. OTC (a) at time T1 can intersect with TTC (A) at time T2 in many ways. This can be expressed by a 2x2 intersection matrix:

\[
\begin{bmatrix}
\partial a \cap \partial A & \partial a \cap \partial A' \\
\partial a \cap \partial A' & \partial a \cap \partial A
\end{bmatrix}
\]

Like OTC, this too provides 16 possible intersections, but not all are valid (Figure 4.26). Eleven valid intersections exist when:

1. \( \partial a \) intersects with \( \partial A \),
2. \( \partial a \) intersects with \( \partial A' \),
3. \( \partial a \) intersects with \( \partial A' \),
4. \( \partial a \) intersects with \( \partial A \) and \( \partial a \) intersects with \( \partial A' \),
5. \( \partial a \) intersects with \( \partial A' \),
6. \( \partial a \) intersects with \( \partial A \) and \( \partial a \) intersects with \( \partial A' \),
7. \( \partial a \) intersects with \( \partial A' \),
8. \( \partial a \) intersects with \( \partial A \),
9. \( \partial a \) intersects with \( \partial A' \),
10. \( \partial a \) intersects with \( \partial A' \),
11. \( \partial a \) intersects with \( \partial A \) and \( \partial a \) intersects with \( \partial A' \),
12. \( \partial a \) intersects with \( \partial A' \),
13. \( \partial a \) intersects with \( \partial A \),
14. \( \partial a \) intersects with \( \partial A' \),
15. \( \partial a \) intersects with \( \partial A \),
16. \( \partial a \) intersects with \( \partial A' \).
9] \( A \) intersects with \( \partial A \),

10] \( \partial A \) intersects with \( A \) and \( \partial A \) intersects with \( A \),

13] \( \partial A \) intersects with \( \partial A \) and \( \partial A \) intersects with \( \partial A \),

14] \( \partial A \) intersects with \( \partial A \) and \( \partial A \) intersects with \( A \).

<table>
<thead>
<tr>
<th>OT</th>
<th>Intersection</th>
<th>Illustration</th>
<th>Spatio-Temporal Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>2</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>3</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>4</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>5</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>6</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>7</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>8</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>9</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>10</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>11</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>12</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>13</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>14</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>15</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
<td>( \neg \emptyset )</td>
</tr>
<tr>
<td>16</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

*Figure 4.26 OTC-TTC (OT) intersection.*

OT 16 is the case where none of them intersects. Some of the cases are symmetric, e.g., OT 14 and 13
are symmetric to OT 5 and 8, respectively. Therefore, a total of eight (OT 1, 3, 5, 6, 7, 8, 9 and 10) valid OT intersections are applicable.

In Figure 4.26, it is assumed that OTC has a single segment and TTC has no holes. The first three cases are discussed here, the rest are variants of these cases or the same concept is applied while inserting OTC when it intersects with TTC.

4.4.1.3.1 Boundary of OTC Intersects with Boundary of TTC

Two scenarios can be realized in this case, i.e., when a single boundary of OTC intersects with the boundary of TTC and both boundaries of OTC intersect with the boundary of TTC.

1] A single boundary of OTC intersects with boundary of TTC:

A single boundary of OTC can intersect with the boundary of TTC in two ways, i.e.,

a) intersects at boundary of TTC; Figure 4.27[a]:

Kill TTC (A1, OTC (a1) and TCTs (c1, c2, c3, c4, c6 and c8). Create ZTCs (n3 and n4), OTC (a3, a4 and a5), TTC (A2) and TCTs (c7, c9, c10, ..., c20).

b) intersects at boundary of OTC of TTC; Figure 4.27[b]:
Create ZTC (n3), OTC (a3) and TCTs (c9 and c10).

In the first case, the creation of OTC yields a new TTC, while in the second no new TTC is created.
Both boundaries of OTC intersect with the boundary of TTC:
Both boundaries of OTC can intersect in two fashions, i.e.,

a) both intersect at the boundary of TTC; Figure 4.28[a]:
Kill TTC (A1), OTC (a1) and TCTs (c1, c2, c3, c4, c6 and c8). Create ZTCs (n3 and n4), OTCs (a3, a4, a5 and a6), TTC (A2) and TCTs (c7, c9, c10, .. c24).

b) one intersects at the boundary of TTC and the other at the boundary of OTC of TTC; Figure 4.28[b]:
Kill TTC (A1), OTC (a1) and TCTs (c1, c2, c3, c4, c6 and c8). Create ZTC (n3), OTCs (a3, a4 and a5), TTC (A2) and TCTs (c7, c9, c10, .. c20).

In all cases, the creation of OTC yields a new TTC, except Figure 4.28[b].

4.4.1.3.2 Boundary of OTC intersects with interior of TTC

In this case, the existing TTC remains unchanged (geometrically), while its spatio-temporal topology is changed (Figure 4.29). A new OTC (a3) is inserted as a singular tcell with the usual TCTs (c9 and c10).

Figure 4.28 Create OTC: both boundaries of OTC intersect with boundary of TTC.

4.4.1.3.2 Boundary of OTC intersects with interior of TTC

In this case, the existing TTC remains unchanged (geometrically), while its spatio-temporal topology is changed (Figure 4.29). A new OTC (a3) is inserted as a singular tcell with the usual TCTs (c9 and c10).
The boundary of TTC is its face, i.e., OTC. This intersection is similar to intersections OO 3, 7, 9, 10, 12, 13 and 14 (§ 4.4.1.2). These cases are shown in Figure 4.30. Various intersections in Figure 4.30[a], [b], [c], [d], [e], [f] and [g] reflect the intersections OO 3, 7, 9, 10, 12, 13 and 14, respectively. Details of intersection [a] are given here, the rest are based on the same concepts.

**Figure 4.30 Create OTC: interior of OTC intersects with boundary of TTC.**
Kill TTC (A1), OTC (a1) and TCTs (c1, c2, c3, c4, c6 and c8).
Create ZTCs (n3, n4 and n5), OTCs (a4, a5, a6 and a7), TTC (A2) and TCTs (c9, …c22).

In all cases, a new TTC (A2) is spawned and the existing one (A1) ceases to exist because the geometry of TTC (A1) is modified.

Figure 4.30 shows the intersections when OTC intersects with the interior of OTC (boundary of TTC). However, the situation would be different if OTC intersected at the boundary of OTC (where OTC is the boundary of TTC). Only the first case is shown in Figure 4.31, which is similar to OO 3. Topologically, this is homeomorphic (Figure 4.30[a]); geometrically, both are different. In this case, the existing TTC (A1) remains unchanged because its geometry is not altered. Any insertion of OTC that modifies the TTC geometry will generate a new TTC and corresponding TCTs.

4.4.1.3.4 Boundary of OTC intersects with boundary-interior of TTC

If the boundary of OTC intersects with OTC (boundary of TTC) at ZTC (boundary of OTC) then no new TTC is created (Figure 4.32[a]); otherwise, if it intersects at the interior of OTC, then the existing TTC (A1) is killed and a new TTC is created (Figure 4.32[b]).

The rest of the cases, i.e., $\partial A$ intersects with $\partial A$; $\partial A$ intersects with $\partial A$; $\partial A$ intersects with $\partial A$; and $\partial A$ intersects with $\partial A$; follow the same conventions and are not elaborated on.

Figure 4.31 Create OTC: interior of OTC intersects with boundary of TTC.

Figure 4.32 Create OTC: boundary of OTC intersects with boundary-interior of TTC.
4.4.2 OTC Kill operator (↓)

Killing an isolated OTC is straightforward, as the null (*) value of the upper bound of the time interval (system time) is replaced by current time. Cases where the boundary (ZTC) of OTC belongs to another OTC can be complex. An OTC itself can be a boundary of TTC. These two cases are discussed below.

4.4.2.1 Boundary of OTC shared by another OTC

a] Shared by one OTC (Figure 4.33[a]):
   Kill ZTC (n1), OTC (a1) and TCTs (c1 and c2).
b] Shared by two OTCs (Figure 4.33[b]):
   Kill OTC (a2) and TCTs (c3 and c4).
c] Shared by two or more OTC (Figure 4.33[c]):
   Kill ZTC (n1), OTC (a1) and TCTs (c1 and c2).

4.4.2.2 OTC is a boundary of TTC

a] TTC is defined by a single OTC.
b] TTC is defined by more than one OTC.
c] OTC is shared by two TTCs.

In all three cases (Figure 4.34), the Kill operation is discarded (enforce spatio-temporal consistency rule) as the OTC is the boundary of TTC and TTC cannot have an inactive boundary.

Figure 4.33 Kill OTC: boundary of OTC shared by another OTC.

Figure 4.34 Kill OTC: TTC is a boundary of another OTC.
4.5 TWOTCELLCLASS

Four fundamental operations associated with this class are depicted in Figure 4.35.

4.5.1 Operation Create (insertion of TTC)

Like the OTC Create operation, the Create operation for TTC is also recursive, starting from the insertion (creation) of the boundary of TTC, i.e., OTC and boundary of OTC, i.e., ZTC. This operation too can be viewed from three perspectives, i.e., a TTC can intersect with ZTC, OTC and TTC. The cases where TTC intersects with TTC is discussed here; the other two cases, i.e., TTC-OTC and TTC-ZTC, are semi-symmetric (because the geometry is the same, while the spatio-temporal topology is different) to the OTC-TTC and ZTC-TTC intersections.

For example, when OTC at time T2 intersects with TTC at time T1, then TTC may or may not be killed (Figure 4.27). When TTC at time T2 intersects with OTC at time T1, then OTC may or may not be killed. When the boundary of TTC (A1) intersects with the boundary of OTC (a1), then OTC (a1) remains unaltered (Figure 4.36). However, when the boundary of TTC (A1) intersects with the interior of OTC (a1), then OTC (a1) is killed (Figure 4.37).

The TTC-TTC intersection is discussed here. Let TTC (A) and TTC’ (B) be a TTC at time T1 and T2, respectively. Using the point-set approach, a 2x2 intersection matrix can be constructed:

$$TT_{14} = \begin{pmatrix} \partial A \cap \partial B' & \partial A \cap \partial B \\ \partial A \cap \partial B & \partial A \cap \partial B' \end{pmatrix}$$
Chapter 4: Logical Data Modeling

The empty and non-empty combinations are shown in Figure 4.38. Out of 16 possible intersections, only seven (TT 1, 7, 9, 10, 11, 13 and 14) are valid, while the last (16) is a non-intersection case and TT 11 and 12 are symmetric.

Egenhofer (1993) derived binary topological relations between two regions, using this approach. Figure 4.38 is a general illustration of TTC-TTC intersections; each intersection may have various combinations, some of which are associated with valid intersections and are discussed in the following sections. To simplify the illustration of the TCT structure, A and B, the TTCs (Figure 4.38), are represented by 1 and 2, respectively.

The empty and non-empty combinations are shown in Figure 4.38. Out of 16 possible intersections, only seven (TT 1, 7, 9, 10, 11, 13 and 14) are valid, while the last (16) is a non-intersection case and TT 11 and 12 are symmetric.

Egenhofer (1993) derived binary topological relations between two regions, using this approach. Figure 4.38 is a general illustration of TTC-TTC intersections; each intersection may have various combinations, some of which are associated with valid intersections and are discussed in the following sections. To simplify the illustration of the TCT structure, A and B, the TTCs (Figure 4.38), are represented by 1 and 2, respectively.
4.5.1.1 Boundary of TTC Intersects with boundary of TTC’ (TT-1)

Five combinations can be realized in this intersection:

a] Both TTCs intersect at ZTC; Figure 4.39[a]:
Create OTC (a2), TTC (2) and TCTs (c3 and c4).

b] OTC of TTC (1) intersects with ZTC of TTC (2); Figure 4.39[b]:
Kill OTC (a1), TTC (1) and TCTs (c1 and c2). Create ZTC (n2), OTCs (a2, a3 and a4), TTCs (2 and 3) and TCTs (c3, c4, ... c12).
c) ZTC and OTC of TTC (1 and 2) intersect; Figure 4.39[c]:
Kill OTC (a1), TTC (1) and TCTs (c1 and c2). Create ZTC (n2), OTCs (a2, a3 and a4), TTCs (2 and 3) and TCTs (c2, c3, … c14).

d) OTC of TTCs (1 and 2) intersects; Figure 4.39[d]:
Kill OTC (a1), TTC (1) and TCTs (c1 and c2). Create ZTCs (n2 and n3), OTCs (a2, a3, a4 and a5), TTCs (2 and 3) and TCTs (c2, c3, … c18).

e) ZTC and OTC of TTCs (1 and 2) intersect; Figure 4.39[e]:
Kill OTC (a1), TTC (1) and TCTs (c1 and c2). Create ZTC (n2), OTCs (a2, a3 and a4), TTCs (2 and 3) and TCTs (c2, c3, … c14).

The configuration formed in Figure 4.39[a], [b], [c], [d] and [e] is called a 2-temporal complex (2-TC) or 2D spatio-temporal object (TSTO). A 2-TC is a collection such that each TTC in 2-TC is connected through a common face.

4.5.1.2 Boundary of TTC intersects with boundary-interior of TTC' (TT-7)
When the interior of TTC (1) intersects with the boundary of TTC (2), then the following actions are taken (Figure 4.40):

Kill TTC (1) and TCT (c2). Create ZTC (n2), OTC (a2) and TCTs (c3, c4 and c5).

TCT c2 is replaced by c3, because the co-boundary of OTC (a1) is changed while the OTC (a1) remains unchanged. At time T1, the co-boundary of a1 was <0,1> and at time T2 it was changed to <0,2>. This is one of the advantages of the implicit topology storage approach. If the topology is stored in an implicit fashion, then the object also has to be updated in order to update the topology, because topology is associated with the spatial object, e.g., in ARC/INFO the co-boundary (left and right polygon) information is associated with arc.

4.5.1.3 Interior of TTC' intersects with boundary-interior of TTC (TT-9)
When the boundary-interior of TTC (1) intersects with the interior of TTC (2), then the following
actions are taken (Figure 4.41):

![Figure 4.40 Create TTC: interior of TTC intersects with boundary-interior of TTC']

![Figure 4.41 Create TTC: interior of TTC' intersects with boundary-interior of TTC]

Kill TCT (c2). Create ZTC (n2), OTC (a2) and TCTs (c3, c4 and c5).

In this case, TTC (1) remains unchanged but its spatio-temporal topology is adjusted, i.e., TCT c2 is replaced by c3.

4.5.1.4 Boundary-interior of TTC Intersects with boundary-interior of TTC' (TT-10)

The operation rejects the TTC (2) because this cell is identical or the same as TTC (1).

4.5.1.5 Boundary of TTC Intersects with boundary-interior of TTC' and interior of TTC Intersects with boundary of TTC' (TT-11)

Kill TTC (1), OTC (a1) and TCT (c1 and c2). Create ZTCs (n2, n3 and n4), OTCs (a2, a3, a4, a5, a6 and a7), TTCs (2, 3 and 4) and TCTs (c3, c4, .. c26).

The process is shown in Figure 4.42.

4.5.1.6 Interior of TTC Intersects with boundary-interior of TTC' and boundary of TTC Intersects with boundary of TTC' (TT-13)

The boundaries of TTC 1 and 2 are their OTC. This can be considered as OTC-OTC intersections, where OTCs can intersect in many ways. Three cases are illustrated here.

a] Boundary of OTC intersects with interior of OTC (Figure 4.43[a]).

Kill TTC (1) and TCT (c2). Create OTC (a2), TTCs (2 and 3) and TCTs (c3, c4 and c5).

TCT c2 is replaced by c3 because the co-boundary of OTC (a1) is changed to 2 (at time T2) from 1 (at time T1).

b] Interior of OTC intersects with interior of OTC (Figure 4.43[b]).

Kill TTC (1) OTC (a1) and TCTs (c1 and c2). Create ZTC (n2), OTCs (a2, a3 and a4), TTCs (2 and 3) and TCTs (c3, c4, .. c12).
Figure 4.42 Create TTC: boundary of TTC intersects with boundary-interior of TTC' and interior of TTC intersects with boundary of TTC'.

Figure 4.43 Create TTC: interior of TTC intersects with boundary-interior of TTC' and boundary of TTC intersects with boundary of TTC'.
c) Both boundary and interior of OTC intersects each other (Figure 4.43[c]).

Kill TTC (1) OTC (a1) and TCTs (c1 and c2). Create ZTC (n2 and n3), OTCs (a2, a3, a4 and a5), TTCs (2 and 3) and TCTs (c3, c4, .. c18).

These examples show that a different number of TCTs are generated depending upon the geometric configurations of the temporal cells, although topologically they are all the same.

4.5.1.7 Boundary of TTC intersects with boundary-interior of TTC and interiors of TTC intersect interior of TTC’ (TT-14).

This is similar to the previous case (TT-13), except the TTC at time T1 is not killed (Figure 4.44).

4.5.2 TTC Kill operator (↓)

The same convention as in the case of ZTC and OTC is applied to kill a TTC. While applying Kill operators to TTC, two scenarios can be realized.

a) The face of TTC is not shared by other TTCs or isolated TTCs (Figure 4.45[a]): Kill ZTC (n1), OTC (a1), TTC (1) and TCTs (c1 and c2).

All the faces and TTC itself is killed.

b) The face of TTC is shared by another TTC (Figure 4.45[b]): Kill OTC (a1), TTC (1) and TCTs (c5, c6, c7, c8, c10 and c12).

All the faces are killed except common face(s).
4.6 DESTROY (⇓) AND REINCARNATE (↑) OPERATOR

In this research, the Create and Kill operators are discussed in detail. The Destroy / Delete operators are based on Kill operators, i.e., the same algorithm is applied to the Destroy operator as applied to the Kill operator. Once the objects (n-tcells) are killed, they can be purged from the database. The Destroy operator purges the database by permanently deleting n-tcells instead of making them inactive. Therefore, these cells are no longer available for the Reincarnate operator. The Reincarnate operator turns an inactive cell into an active cell by replacing the upper bound (ST_Until) of the time interval by a null value. One example is considered here to demonstrate the function of the Reincarnate operator. This operator is pragmatic in retroactive changes. For example, at time T1, there was one TTC (A); at time T2, two new TTCs (B and C) were created. The TTC (A) has been killed because of the Create operation at time T2. Scenario 1 is shown in Figure 4.46. At time T3, it was realized that the TTCs (B and C) had been wrongly created (wrong configuration). Actually they have to be created in the fashion shown in scenario 2, which is the actual configuration (Figure 4.47). In scenario 1, at time T3, the configuration of B and C (scenario 2) could not be achieved, because the TTC (A) is no longer an active object. Prior to achieving the scenario-2 configuration as shown at time T2 in scenario 1, TTCs B and C have to be killed and TTC A must be reincarnated. These steps are explained as follows and demonstrated in the Figure 4.48.

At time T2:
- a] Wrong configuration, which needs to be corrected.

At time T3:
- b] Destroy (⇓) TTCs (B and C) and corresponding faces and TCT (not shown in the figure for simplification reasons).
- c] Reincarnate (↑) TTC (A), which includes the reincarnations of TCTs (not shown for simplification reasons).
- d] Create TTC (B) to achieve desired temporal cell complex configuration.

The example shown here for TTC can be applied for ZTC and OTC. However, more work is needed to analyze the scenario when applying / designing the Reincarnate operators for ZTC and OTC.

Figure 4.45 Kill TTC: a] isolated TTC and b] face shared by another TTC.
4.7 SPATIO-TEMPORAL RELATIONS

Static operators are query operators. These operators are used to query the spatial, temporal and spatio-temporal relations. Relations in spatio-temporal databases can be categorized into three classes:

- Spatial relations
- Temporal relations
- Spatio-temporal relations.

Some spatial relations were discussed in Chapter-2. Worboys (1992a) groups these relations into four classes: set-oriented, metric, topological and Euclidean relations. Kainz (1989) introduced spatial order relations. These spatial relations can be grouped into five categories (Table 4.2):

- Spatial metric relations
Spatial topological relations
Spatial order relations
Set-oriented spatial relations
Euclidean spatial relations.

Similarly, the temporal relations can be sub-classified into four categories:
- Temporal metric relations
- Temporal topological relations
- Temporal order relations
- Set-oriented temporal relations

Some of the temporal relations (temporal topological) discussed in Chapter-2 can be applied to CTSTDM. Worboys (1992a) proposed nine spatial topological relations, i.e.,
interior, closure, boundary, components, extremes, begin, end, inside and clockwise. These are valid for spatial objects of dimension 0 ≤ n ≤ 2. Egenhofer et al. (1993) and Pullar and Egenhofer (1988) used the point-set approach to derive eight topological relations between two spatial objects of dimension 2 and eight topological relations between two spatial objects of dimension 1, respectively.

The conceptual schema for CTSTDM was discussed in Chapter-3. Various spatial and spatio-temporal objects (n-tcells) were identified. Figure 4.49 shows the hierarchy of spatial, temporal and spatio-temporal classes / objects. One spatial object (point), three spatio-temporal objects or STO (ZTC, OTC and TTC) and linear time class were identified in Chapter-3. Spatio-temporal relations are shown in Table 4.2. Table 4.2 lists the five categories of spatio-temporal relations (operators). For each operator, the operands, results and syntax (in UML) are shown. They are all temporal in nature and therefore called spatio-temporal relations. However, in some cases, they may be time-invariant, depending upon where they are applied, e.g., when applied to a STClass object, an operator may be time-invariant but when applied to STOClass this may be time-dependent.

For example, metric operator distance; when applied to two point objects (ST objects) at time T1 and T2, the distance between them remains the same at time T2; but when applied to two objects (object of STA class) at time T1 and T2, the distance may change at time T2. Consider Figure 4.50, at time T1, the ITC building was located at point p1 and the Dish hotel at p2. The distance between ITC and Dish was 500 meters. At time T2, the ITC building moved to another locations, i.e., point p3, now the distance between ITC and Dish hotel changed to 800 meters. However, the distance between point p1 and p2 remained same.

Some relations are time-invariant, i.e., they are purely spatial relations such as metric and Euclidean, while others are time-dependent, such as co-boundary, inside, meet etc. Metric relations such as distance between two points, length of an OTC and perimeter of a TTC do not change with the
passage of time. Similarly, Euclidean relations such as bearing between two point objects and area of TTC are time-invariant.

Topological relations may or may not change with the passage of time. For example, as time passes, the interior, closure and boundary relations of TTC or OTC remain unchanged. Co-boundary or inside relations may change with time. These relations are either time-invariant or not, they may be considered as spatio-temporal relations because they are measured on a time line. Moreover, the operands of most of them are STO, except point objects. Spatio-temporal relations between ZTC, OTC and TTC are discussed in the following sections.

Spatial relations that are valid for a certain time period are called spatio-temporal relations. Topological relations are considered for further discussion. Most of these relations (spatio-temporal topology) can be derived from TCT the structure.

4.7.1 Spatio-temporal topological relations

These spatio-temporal relations are preserved in TCT structure, as given in Chapter-3. Recalling this, a TCT is a set of C and T, i.e.: 

\[ TCT = \{C, T\} \]

where C is a set of cells
\[ C = \{c_0, c_1, c_2, \ldots, c_n | c_i \in TCC\} \]

and T is a time interval (1-T)
\[ T = \{T_{From}, T_{Until} | (T_{From} < T_{Until}) \land (T_{From}, T_{Until} \in ST)\} \]

Therefore,
\[ TCT = \{c_0, c_1, c_2, \ldots, c_n, T_{From}, T_{Until}\} \]

Boundary (\(\partial\)) and coboundary (\(\Phi\)):

Initially, the boundary and co-boundary relations were introduced in SpatioTemporalClass (Chapter-3). The boundary (\(\partial\)) of a \(n\)-tcell is its (\(n-1\)) faces at time \(t\). The co-boundary (\(\Phi\)) of a \(n\)-tcell produces the (\(n+1\)) cells incident with \(n\)-tcell at time \(t\).

The \(\Phi\) and \(\partial\) history of \(k\)-tcell at time \(T_i\) can be formalized as:

\[ \Phi((k-tcell)_{T_i}) = \{\forall (k+1)tcell \mid T_{From} \leq T_i\} \]

\[ \partial((k-tcell)_{T_i}) = \{\forall (k-1)tcell \mid T_{From} \leq T_i\} \]

This formalization of boundary assumes that singularities in TCC are not incorporated and this definition may provide wrong boundary information. For example, Figure 4.32[a]:

**Boundary of A1 at time T1** is \((a_1, a_2)\), i.e.,

\[ \partial(A1)_{T1} = \{\forall OTC \mid T_{From} = T_1\} \]

= \(a_1, a_2\)

while boundary of A1 at time T2 yields three OTCs \((a_1, a_2, a_3)\)

\[ \partial(A1)_{T2} = \{\forall OTC \mid T_{From} = T_2\} \]

= \(a_1, a_2, a_3\)

However, OTC \((a_3)\) is not a boundary of A1.
Similarly, in Figure 4.29, the boundary of A1 at time T1 and T2 is,
\[ \partial(A1)_{T1} = \{a1, a2\} \]
\[ \partial(A1)_{T2} = \{a1, a2, a3\} \]

To avoid this anomaly, the boundary is discerned in two steps
\[ \partial(k-tcell)_{T1} = \{\forall (k-1)tcell \mid T_{From} = T_1 \land k-tcell_1 \neq k-tcell_2\} \]
where
\[ \Phi(k-1) - \text{tcell} = \{k-\text{tcell}1, k-\text{tcell}2\} \]

This definition can remove the extra boundary information. Hence, in Figure 4.29, the boundary of A1 at time T2 can be calculated as:

\[ \partial(A1)_{T2} = \{a1, a2, a3\} \]
\[ \Phi(a1) = \{\emptyset, A1\} \]
\[ \Phi(a2) = \{\emptyset, A1\} \]
\[ \Phi(a3) = \{A1, A1\} \]

Therefore, a3 is excluded from the boundary of A1 because the co-boundary of a3 is the same.

\[ \partial(A1)_{T2} = \{a1, a2\} \]

The boundary of 1-TC (OSTO) and 2-TC (TSTO) can be formalized as:

\[ \partial(k-\text{TC})_{T1} = \{ \forall (k-1)\text{tcell} \mid T_{From} = T1 \land ( ( (k-1)-\text{tcell}p) \cup \partial(k-\text{tcell}j+2)) – ( ( (k-1)-\text{tcell}p) \cap \partial(k-\text{tcell}j+2)) ) \} \]

where
\[ j = 1, 2, \ldots, n \]
\[ (k-1)-\text{tcell}p = ( \partial(k-\text{tcell}j) \cup \partial(k-\text{tcell}j+1) ) – ( \partial(k-\text{tcell}j) \cap \partial(k-\text{tcell}j+1) ) \]
\[ k-\text{tcell}j \in k-\text{TC}, \]
\[ k = 1, 2 \text{ and } p = j \]

For example, in Figure 4.42 2-TC = (2,3,4). The boundary of this 2-TC at time T2 is

\[ \partial(2-\text{TC})_{T2} = \{(1-\text{tcell}1 \cup \partial(4)) – (1-\text{tcell}1 \cap \partial(4))\} \]

where
\[ 1-\text{tcell}1 = \{ (\partial(2) \cup \partial(B(3)) - (\partial(2) \cap \partial(3)) ) \}
\[ = \{ (a2, a3, a4) \cup (a4, a5) ) - ((a2, a3, a4) \cap (a4, a5) ) \}
\[ = \{ (a2, a3, a4, a5) \} \]
\[ = \{a2, a3, a5\} \]

Therefore,

\[ \partial(2-\text{TC})_{T2} = \{(1-\text{tcell}1 \cup \partial(4)) – (1-\text{tcell}1 \cap \partial(4))\} \]
\[ = \{(a2, a3, a5, a6, a7) – (a5) \} \]
\[ = \{a2, a3, a6, a7\} \]

Co-boundary:
In Figure 4.39[\text{c}], the co-boundary of ZTC (n1) at time T1 and T2 is:

\[ \Phi(n1)_{T1} = \{\forall \text{OTC} \mid T_{From} = T1\} \]
\[ = \{a1\} \]
\[ \Phi(n1)_{T2} = \{\forall \text{OTC} \mid T_{From} = T2\} \]
\[ = \{a2, a3, a4\} \]

Similarly, in Figure 4.40, the co-boundary of OTC (a1) at time T1 and T2 is:

\[ \Phi(a1)_{T1} = \{\forall \text{TTC} \mid T_{From} = T1\} \]
\[ = \{1, 0\} \]
\[ \Phi(a1)_{T2} = \{\forall \text{TTC} \mid T_{From} = T2\} \]
\[ = \{1, 2\} \]

Disjoint (\Omega):
Disjoint(P:STO, P:STO): Boolean

The two \( n \)-tcells (\( n = 1, 2 \)) are disjoint if the intersection of their faces is empty. Disjoint relations of point and ZTC are straightforward. The \( \Omega \) relations of OTC and TTC can be expressed as:

\[
\text{Disjoint(P:TTC}_{T1}, P:TTC_{T2}): \text{Boolean} \\
\{2\text{-tcell}_{T1}, \Omega 2\text{-tcell}_{T2} = true | \partial(2\text{-tcell}_{T1}) \cap \partial(2\text{-tcell}_{T2}) = \emptyset \}
\]

\[
\text{Disjoint(P:OTC}_{T1}, P:OTC_{T2}): \text{Boolean} \\
\{1\text{-tcell}_{T1}, \Omega 1\text{-tcell}_{T2} = true | \partial(1\text{-tcell}_{T1}) \cap \partial(1\text{-tcell}_{T2}) = \emptyset \}
\]

Contains (\( \alpha \)):
The containment relations can be between spatio-temporal objects of the same spatial dimension or different spatial dimensions. For example, a TTC can contain a TTC, an OTC or a ZTC; these relations are depicted in Figure 4.40, Figure 4.29 and Figure 4.10, respectively.

At time \( T_i \), 2-tcell \( j \) contains 2-tcell \( k \);
\[
\text{Contains(P:TTC, P:TTC): Boolean} \\
\{2\text{-tcell}_{j} \alpha \text{-tcell}_{k} = true | T_{From} = T_i \land \partial(2\text{-tcell}_{j}) \cap \partial(2\text{-tcell}_{k}) = \partial(2\text{-tcell}_{j}) \}
\]

At time \( T_i \), 2-tcell contains 1-tcell;
\[
\text{Contains(P:TTC, P:OTC): Boolean} \\
\{2\text{-tcell} \alpha \text{-tcell} = true | T_{From} = T_i \land \partial(2\text{-tcell}) \cap \partial(1\text{-tcell}) = \partial(1\text{-tcell}) \}
\]

At time \( T_i \), 2-tcell contains 0-tcell;
\[
\text{Contains(P:TTC, P:ZTC): Boolean} \\
\{2\text{-tcell} \alpha \text{-tcell} = true | T_{From} = T_i \land \partial(2\text{-tcell}) \cap 0\text{-tcell} = 0\text{-tcell} \}
\]

For example, to check whether TTC contains a TTC or not, consider Figure 4.40, where at time \( T_2 \), \( \text{TTC(3)} \alpha \text{TTC(2)} \).
\[
\{\partial(3) \cap \partial(2) \} = \{\partial(2)\} \\
\{(a_1, a_2) \cap (a_2) \} = \{(a_2)\} \\
\{a_2\} = \{a_2\}
\]

Inside (\( \chi \)):
At time \( T_i \), a ZTC, OTC or TTC can be inside a TTC. The same logic is employed to discern the \( \chi \) relations between two \( n \)-tcells. For example:

At time \( T_i \), 2-tcell \( j \) is inside 2-tcell \( k \);
\[
\text{Inside(P:TTC, P:TTC): Boolean} \\
\{2\text{-tcell}_{j} \chi 2\text{-tcell}_{k} = true | T_{From} = T_i \land \partial(2\text{-tcell}_{j}) \cap \partial(2\text{-tcell}_{k}) = \partial(2\text{-tcell}_{j}) \}
\]

At time \( T_i \), 1-tcell is inside 2-tcell;
\[
\text{Inside(P:OTC, P:TTC): Boolean} \\
\{1\text{-tcell} \chi 2\text{-tcell} = true | T_{From} = T_i \land \partial(1\text{-tcell}) \cap \partial(2\text{-tcell}) = \partial(1\text{-tcell}) \}
\]
At time $T_i$, 0-tcell is inside 2-tcell;
Inside(P:ZTC, P:TTC): Boolean

$\{\text{0-tcell} \chi \text{2-tcell} = \text{true} \mid T_{\text{from}} = T_i \land \partial(\partial(\text{2-tcell})) \cap \text{0-tcell} = \text{0-tcell}\}$

**Equal ($\equiv$):**
Checking Equal relations between two points or ZTCs is straightforward. TTC at time $T_1$ is in equal relations with TTC at time $T_2$ if the boundaries of both are the same.

$\{2\text{-tcell}_{T_1} = 2\text{-tcell}_{T_2} \mid \partial(\partial(\text{2-tcell}))_{T_1} = \partial(\partial(\text{2-tcell}))_{T_2}\}$

Although Equal is a topological relation, the Equal relation between two OTCs may not be checked correctly in the TCT structure (based on boundary / co-boundary relations), because these OTCs can be defined by different intermediate points, regardless of the same boundary. A geometric calculation is needed to check this relation.

**Meet ($\delta$):**
A TTC at time $T_1$ can Meet with TTC, OTC or ZTC at time $T_2$. Similarly, an OTC at time $T_1$ can Meet with OTC or ZTC at time $T_2$.

Meet(P:TTC, P:TTC): Boolean

$\{2\text{-tcell}_{T_1} \delta 2\text{-tcell}_{T_2} \mid \partial(\partial(\text{2-tcell}))_{T_1} \cap \partial(\partial(\text{2-tcell}))_{T_2} \neq \emptyset\}$

Meet(P:TTC, P:OTC): Boolean

$\{2\text{-tcell}_{T_1} \delta 1\text{-tcell}_{T_2} \mid \partial(\partial(\text{2-tcell}))_{T_1} \cap \partial(\partial(\text{1-tcell}))_{T_2} \neq \emptyset\}$

Meet(P:TTC, P:ZTC): Boolean

$\{2\text{-tcell}_{T_1} \delta 0\text{-tcell}_{T_2} \mid \partial(\partial(\text{2-tcell}))_{T_1} \cap (0\text{-tcell})_{T_2} \neq \emptyset\}$

Meet(P:OTC, P:OTC): Boolean

$\{1\text{-tcell}_{T_1} \delta 1\text{-tcell}_{T_2} \mid \partial(\partial(\text{1-tcell}))_{T_1} \cap \partial(\partial(\text{1-tcell}))_{T_2} \neq \emptyset\}$

Meet(P:OTC, P:ZTC): Boolean

$\{1\text{-tcell}_{T_1} \delta 0\text{-tcell}_{T_2} \mid \partial(\partial(\text{1-tcell}))_{T_1} \cap (0\text{-tcell})_{T_2} \neq \emptyset\}$

For example, consider Figure 4.39[d]. At time $T_2$, TTC (2) and TTC (3) have Meet relations.

$\{\partial(\partial(\text{2})) \cap \partial(\partial(\text{3})) \neq \emptyset\}$

$\{\partial(a2, a3, a4) \cap \partial(a3, a5) \neq \emptyset\}$

$\{(n1, n2, n3) \cap (n2, n3) \neq \emptyset\}$

$\{(n2, n3) \neq \emptyset\}$

Similarly, consider Figure 4.37. At time $T_2$, TTC (A1) and OTC (a3) have Meet relations.

$\{\partial(\partial(\text{A})) \cap \partial(\partial(\text{a})) \neq \emptyset\}$

$\{\partial(a4, a5) \cap (n2, n3) \neq \emptyset\}$

$\{(n3, n4) \cap (n2, n3) \neq \emptyset\}$

$\{(n3) \neq \emptyset\}$

**Covers ($\gamma$):**
A TTC at time $T_2$ can cover a TTC or OTC at time $T_2$; these relations are shown in Figure 4.43 and Figure 4.32, respectively. Similarly, an OTC at time $T_1$ can cover OTC at time $T_2$. However, this relation could not be captured in TCT structure because it does not maintain the interior of OTC.

Covers(P:TTC, P:TTC): Boolean
\{2-\text{tcell}_1 \gamma 2-\text{tcell}_2 | (\partial(\partial(2-\text{tcell})_{T1}) \cap \partial(2-\text{tcell})_{T2} \neq \emptyset) \land (\Phi(\partial(2-\text{tcell})_{T1}) \cap (2-\text{tcell})_{T2} \neq \emptyset) \}\)

\text{Covers}(P: \text{TTC}, P: \text{OTC}): \text{Boolean}
\{2-\text{tcell}_1 \gamma 1-\text{tcell}_2 | (\partial(\partial(2-\text{tcell})_{T1}) \cap \partial(1-\text{tcell})_{T2} \neq \emptyset) \land (\partial(2-\text{tcell})_{T1} \cap (1-\text{tcell})_{T2} \neq \emptyset)\}

Consider Figure 4.43[c]. At time T2, TTC (2) covers TTC (3).
\{ (\partial(\partial(2)_{T1}) \cap \partial(3)_{T2} \neq \emptyset) \land (\Phi(\partial(2)_{T1}) \cap (3) \neq \emptyset) \}\)
\{ (\partial(a_4, a_5) \cap (a_2, a_3, a_4) \neq \emptyset) \land (\Phi(a_4, a_5) \cap (3) \neq \emptyset) \}\)
\{ ((n_2, n_3) \cap (n_1, n_2, n_3) \neq \emptyset) \land ((2,3) \cap (3) \neq \emptyset) \}\)
\{ ((n_2, n_3) \neq \emptyset) \land ((3) \neq \emptyset) \}\)

\text{CoveredBy} (\eta):
An OTC or TTC at time T1 can be covered by a TTC at time T2. Figure 4.42[c]; at time T2, TTC (3) is covered by TTC (2). Figure 4.32, at time T2, OTC (a3) is covered by TTC (A1). Similarly, an OTC at time T1 can be covered by OTC at time T2. However, this relation is not captured in TCT structure. The \(\eta\) relation is similar to \(\gamma\) relations and is not discussed further.

\text{Overlap} (\kappa):
A TCC is a partition of spaces, therefore, TTC or OTC cannot overlap each other. However, a TSTO and OSTO at time T1 can overlap with TSTO and OSTO at time T2, respectively, if their intersection of TTC or OTC is non-empty (\(\neq \emptyset\)).

\text{Overlap}(P: \text{TSTO}, P: \text{TSTO}): \text{Boolean}
\{\text{TSTO}_1 \kappa \text{TSTO}_2 | (X_{T1} \cap (Y_{T2} \neq \emptyset) \}\)
where
\(X_{T1} = \{\text{TTC}_1, \text{TTC}_2, \ldots \text{TTC}_n\}\)
\(Y_{T2} = \{\text{TTC}'_1, \text{TTC}'_2, \ldots \text{TTC}'_n\}\)
\(X_{T1} \in \text{TSTO}_1\)
\(Y_{T2} \in \text{TSTO}_2\)

\text{Overlap}(P: \text{OSTO}, P: \text{OSTO}): \text{Boolean}
\{\text{OSTO}_1 \kappa \text{OSTO}_2 | (X_{T1} \cap (Y_{T2} \neq \emptyset) \}\)
where
\(X_{T1} = \{\text{OTC}_1, \text{OTC}_2, \ldots \text{OTC}_n\}\)
\(Y_{T2} = \{\text{OTC}'_1, \text{OTC}'_2, \ldots \text{OTC}'_n\}\)
\(X_{T1} \in \text{OSTO}_1\)
\(Y_{T2} \in \text{OSTO}_2\)

Consider Figure 4.43. Let \(\text{TSTO}_1 = \{2, 3\}\) and \(\text{TSTO}_2 = \{3, 4\}\) at time T2. These two 2-spatio-temporal objects overlap because
\((\{2,3\} \cap \{3,4\} \neq \emptyset)\)
\((\{3\} \neq \emptyset)\)

\text{Start}, \text{Ends}, \text{Left}, \text{Right and Clockwise}:
Direction (start and ends) and orientation (left and right) have been discussed in Chapter-3. If \(c = \{c_1, c_2, \ldots, c_n, 1-T\}\) is a set of TCT. Let \(c_i = \{n, a, A, 1-T\}\) and \(c_j = \{n', a', A', 1-T\}\) be subsets of c, such that \(i < j\). For any OTC a:

\text{Start}(P: \text{OTC}): \text{ZTC}
\text{Start}(\text{OTC}) = \text{ZTC} = \{n | n \in c_i\}\)

\text{Ends}(P: \text{OTC}): \text{ZTC}
\text{End}(\text{OTC}) = \text{ZTC} = \{n' | n' \in c_j\}\)

\text{Left}(P: \text{OTC}): \text{TTC}
Left(OTC) = TTC = \{ A \mid A \in c_i \}

Right(P: OTC); TTC
Right(OTC) = TTC = \{ A' \mid A' \in c_j \}

These relations can be captured through TCT structure. For example, the TCTs for OTC \((a5)\) at time T2 are \(c7, c8, c9\) and \(c10\).

\[
c7 = (3, 5, 3, T2, *) \\
c8 = (3, 5, 4, T2, *) \\
c9 = (2, 5, 3, T2, *) \\
c10 = (2, 5, 4, T2, *)
\]

Start \((a5) = n3\) (ZTC)
Ends\((a5) = n2\) (ZTC)
Left \((a5) = 3\) (TTC)
Right \((a5) = 5\) (TTC)

Clockwise(P: OTC): Boolean
Clockwise relations of OTC are not maintained in TCT structure and are calculated by other geometric techniques.

### 4.8 CONCLUSIONS

One of the components of the spatio-temporal data model, i.e., operations, is discussed in this chapter. Two types of operators are identified: dynamic and static operators. Dynamic operators can change the system’s state, while static operators are query operators and do not affect the system status. Four fundamental types of dynamic operators (Create, Kill, Destroy and Reincarnate) are designed for CTSTDM. Other operators may be designed based on these four fundamental operators. These operators associated with ZeroTCellClass, OneTCellClass and TwoTCellClass have been discussed in detail. The Create operation in spatio-temporal databases is basically an overlay process, where existing cells (objects) are not thrown out, but rather kept with a valid time stamp with consistent topology. Overlay itself is one of the most complex and time-consuming processes in spatial databases. Attaching the time dimension aggravates the process. Deducting the intersection between two spatial objects is a prerequisite for any overlay process. In a unified approach, eight possible intersections can be realized when ZTC, OTC and TTC can intersect with ZTC, OTC and TTC. The problem of intersection is geometric in nature, but using topology it has been demonstrated how these operators can update TCC. A novel point-set topology approach has been employed to discern the intersection between two n-tcells. Two topological invariants, i.e., boundary and interior, have been used to discern these intersections, based on empty or non-empty intersection of boundary and interior. Various scenarios and valid intersections are discussed, when boundary and interior OTC and TTC can intersect each other. Only valid intersections have been considered for further discussion of the Create operator. There are three valid intersections between ZTC-OTC, 10 between OTC-OTC, eight between OTC-TTC and seven between TTC-TTC. The Kill operator for ZeroTCellClass, OneTCellClass and TwoTCellClass has been discussed. The cells can be killed based on the definition of active and inactive cells. The Destroy operator is similar to the Kill operator, except it purges the database or permanently deletes the records from the database. The Reincarnate operator is very useful for retroactive changes. It is demonstrated how these changes can be incorporated.

The static operator is a query operator that is associated with TemporalCellTupleClass. These query operators are used to retrieve the spatial, temporal and spatio-temporal relations. Five types of spatio-temporal and four types of temporal relations are identified. One type of spatio-temporal relations, i.e., spatio-temporal topological relations, has been discussed in detail. It has been proved that almost all spatio-temporal relations between OTC-OTC and TTC-TTC, in the spatial domain as suggested by
Egenhofer, and some other relations can be derived from temporal cell tuple structure, which is based on the boundary and co-boundary of cells. In the explicit topology approach, topology is associated with spatio-temporal objects (e.g., ARC/INFO, where topology is stored with spatial object arcs and polygons). Therefore, objects have to be updated to update the topology, irrespective of whether the object is changed or not. In some cases, the spatio-temporal topologies of the objects are changed without any change in spatio-temporal objects. One of the advantages of the implicit topology approach of TCT is that spatio-temporal topology is preserved without any changes in spatio-temporal objects, e.g., Figure 4.27[a] and Figure 4.41, when the co-boundary is adjusted. In the next chapter, the application of CTSTDM in urban applications is shown.
5.1 INTRODUCTION

Urbanization is a dynamic and complex process. Understanding a dynamic process means studying change. Moreover, a change is closely linked with time. According to Parkes and Thrift (1980), changes are time. In this chapter, various urban applications are discussed to gain a better understanding of this dynamic nature. The focus of this research is the role and modeling of time in selected urban applications. There are a number of applications in urban areas, e.g., health, education, utilities (gas, power, water, sewerage etc.), infrastructure (roads, telecommunications, power supply etc.), traffic management etc. All these applications depend on urban planning, which includes town planning, land registration, building permits etc. The core data needed for urban planning are time series land use data. The aim is to discuss these core data, their sources and related information. Karachi has been selected as a study area because of its complexity from the urban planning and management perspective.

The chapter is divided into two parts. The first part gives a brief introduction to the study area and documents the need of urban planning agencies for spatio-temporal data. The second part deals with the application of the cell-tuple-based spatio-temporal data model (CTSTDM) in the urban context, i.e., land use and associated applications are considered. In this case (land use), AttributeClass of CTSTDB plays a vital role. Both detailed and general land uses are discussed. The potential of updating land use data through operational planning processes (identified in Chapter-1) to strengthen the strategic planning processes is also highlighted. Object-oriented concepts are employed to preclude the complexity of the dynamic nature of urban land use change.

5.2 STUDY AREA KARACHI

Karachi has been one of the world's fastest growing cities since the creation of Pakistan in 1947. It is also the transport hub of the nation; it contains the country's two major ports as well as Pakistan's largest international airport (Figure 5.1). It is expected to maintain its dominant role as a major industrial and commercial center in the future. Karachi metropolitan (urban) area comprises some 180,000 hectares (Figure 5.2). A lot of investments in development programs concern housing, infrastructure and so on (Mission Report_UNCHS, 1990). Rapid urbanization has placed enormous stress on the city's ability to provide sufficient urban services for all its residents. "The range of anticipated population growth in Karachi suggested that the capacity of government and private institutions to plan and provide even most basic urban services will be severely strained" (Karachi Development Plan-2000, 1991). Coping with the problems of the city, which has an annual population growth rate of 5%, has been a challenging task for urban planning agencies. The latest population census shows that the population of Karachi rose from 5.4 million in 1981 to 9.2 million in 1998 (Dawn, 1998), while the Karachi Development Plan-2000 (1991) has predicted a population of 11.77 million in 2000. Karachi still accounts for about 10% of the country's population and 25% of the country's urban population. More than 25 urban planning agencies are providing services to the citizens, but they lack coordination. Coordinated efforts may be improved if their databases are of higher quality and designed with data sharing in mind. The need of urban planning agencies for spatio-temporal data is discussed in § 5.2.1.

The Karachi Development Authority (KDA) has developed 45 housing schemes. About 500,000 parcels (plots) have been planned by the KDA (Karachi Development Plan-2000, 1991). For urban planning and design purposes, each scheme is divided into two types of zones, i.e., analysis zones
Chapter 5: Time in Urban Planning

(ANZ) and master planning zones (MPZ). Fifty-eight ANZs evolved from 241 MPZ (Figure 5.3).

Block 4 of scheme-5 of ANZ 16 has been selected for implementation purposes. Details of this block are discussed in Chapter-8.

5.2.1 Need of urban planning agencies for spatio-temporal data

What makes urban data unique is that the same geographic information (parcel) is shared by many agencies. If these data are updated in a consistent manner, duplication can be avoided, the sharing of information can be improved and better coordination can be expected. Information sharing is reviewed in a typical urban environment, i.e., Karachi. As mentioned earlier, urbanization is a complex process because it involves many players performing a variety of tasks, e.g., public utilities...
Chapter 5: Time in Urban Planning

Nine major agencies/departments are included to assess the need for spatio-temporal data:
1. Master Plan and Environmental Control Department (MPECD), KDA
2. Directorate of Planning and Urban Development (DPUD), KDA
3. Traffic Engineering Bureau (TEB), KDA
4. Karachi Water and Sewerage Board (KWSB)
5. Pak Telecommunication Department (PTC)
6. Karachi Electric Supply Corporation (KESC)
7. Sui Southern Gas Company (SSGC)
8. Sindh Katchi Abadi Authority (SKAA)
9. Survey of Pakistan (SOP)

The Karachi Municipal Corporation (KMC) was excluded because of insufficient information. Basically, the SOP is not an urban planning agency but rather a data producing agency for many planning agencies. These nine agencies use a variety of maps in terms of spatial and temporal scales. First, the functions of these agencies/departments are summarized; then the information content in these departments is discussed (Appendix-2). This forms the information flow matrix in terms of data owners and users of this information. Special emphasis is given to the temporal nature of the information.

None of the agencies has in-house ongoing digital mapping projects. However, they are all looking towards the World Bank-funded project “Digital Mapping Project for Karachi” (DMPK), which is still (until 1998) awaiting clearance from the Government of Pakistan. It is quite frustrating for these agencies that the project is not through yet, because the initial work, which started in 1994, has been done. Almost all maps are in non-digital format. None of the agencies has a clear idea about spatial accuracy. The frequency of using these maps varies from department to department, according to the particular map and the scale of the map. No specific spatial accuracy for the maps used by these departments was reported.

Not surprisingly, most of the agencies need land use data, layout plans and base maps. Out of nine agencies/departments, five are using land use data and seven are using layout plans. The base map ranks number one in use (Table 5.1). Table 5.2 shows the frequency of map use and update per agency. In some cases, multiple frequency is reported, e.g., layout plans are updated daily (2) and as and when required (1), as reported by the MPECD and the DPUD. This could be expected because these maps are used for operational and strategic planning. Frequency of use/update therefore varies from one to five years. In the case of base maps, most of the agencies reported use and update as and when required. Similarly, the land use data are used as and when required; however, update frequency is five years because of financial constraints and the time taken to complete the field survey. Layout plans are used daily by most of the agencies, while they are updated daily, weekly or as and when required.
Table 5.1 and Table 5.2 give a general overview of the use and update frequency of spatio-temporal data. Nevertheless, detailed interviews with these urban planning agencies are needed to find out the
exact use of these data and the corresponding frequency of use. Land use and layout plans are considered for further discussion as the detailed land use is registered on layout plans or parcels. Layout plans are scheme maps.

5.3 ATTRIBUTECLASS

AttributeClass and its subclasses were defined in Figure 3.35 (Chapter-3). Like many other features (reality), land use has three dimensions, i.e., space, time and attribute. These three dimensions are associated with SpatialClass, TemporalClass and AttributeClass. The first two classes have already been discussed in previous chapters, while AttributeClass is discussed in this chapter. AttributeClass is a metaclass of LanduseClass; this class is considered to be purely attribute, although a land use definition can also be changed and may have temporal characteristics. AttributeTemporalClass is an aggregation of AttributeClass, WorldTimeClass and DataBaseClass. AttributeTemporalClass has two subclasses: EventTemporalClass and EvidenceTemporalClass. However, for modeling and further discussion only EvidenceClass is considered (Figure 5.4).

The data members (attributes) of LanduseClass are type (residential, industrial, commercial etc.), classification level (level-i, level-ii or level-iii) and description, i.e., the definition of particular land use (e.g., residential detached houses or apartments). The data members of EvidenceTemporalClass are eviname (evidence name), such as field survey, remote sensing, building permit etc. DBT and WT. These classes are discussed in detail, while modeling the land use changes, in § 5.5.2 and 5.5.3.

5.4 URBAN PLANNING, TIME AND MODELING LAND USE CHANGE

Modern urban planning is a recursive process, which consists of temporal information systems (Temporal LIS and GIS), models and methods, and planning processes. Urban planning processes start with problem definition, goals, objectives, alternatives, evaluation, choice, implementation and monitoring (Batty, 1992). Although monitoring is an integrated component of this planning process, it cannot be performed well without time series data. There are a few systems which have been adapted to monitoring urban change (Batty, 1992). Current ISs are not sufficient to deal with time in GISs. The urban planning process proposed by Batty (1992) does not include the time component in IS. Nevertheless, temporal IS (TGIS and TLIS) is an essential part of the overall urban planning process (Figure 5.5). Two types of urban planning processes can be envisaged (both demand time series data), i.e., the operational and strategic planning processes, sometimes called the routine and non-routine planning processes (Batty, 1995, 1992). Almost all urban planning agencies have to deal with at least these two processes (Figure 5.8). The first deals with the day-to-day work or operations of urban planning agencies, such as the issuance of building permits; the allocation, distribution or redistribution of parcels; land registrations; traffic management etc.
The second is a highly formalized planning process based on an explicit process of rational decision, and is conducted over a long period of months or even years. It encompasses long-term planning, i.e., projecting housing demands for the next 10 years, estimating the power load after 15 years, forecasting the shift in population or demands for health and education etc. Initially, in the 1950s computers were used for strategic planning, although data were collected in a routine fashion. Nowadays, computers are used in a more pragmatic manner (Batty, 1995; Brail, 1989; Bruijn, 1990; Butler, 1989; Yeung and Hall, 1989; Laurini, 1994; Mullin and Kittilesen, 1994; van Helden, 1994). Operational planning, such as parcel mapping, permit tracking, vacant land inventories etc., ranks foremost among GIS applications in urban planning agencies (French et al., 1989). The research focuses on the operational planning process to strengthen the strategic planning.

Little has been done with regard to modeling time aspects in urban planning, particularly modeling urban land use change, which is the backbone of many planning (strategic or operational) exercises. A notable exception is the work on time in cadastral systems, carried out by Al-Taha (1992, 1993). Although the importance of the temporal dimension was evident in early 1979, “there are very few GISs which can be adapted to handling temporal or space-time data” (Batty, 1992). Some prototypes do exist, e.g., TempSet for particular forest applications. Here, the importance of land use data for monitoring purposes, together with relevant sources, is discussed with particular regard to the urban setting, followed by modeling of detailed and general land use change.

5.4.1 Urban land use

Land use is the function of land determined by natural conditions and human interventions. Much has been written about the importance and study of urban land use (ULU) data from government or research perspectives. As urbanization is increasing, the demands for urban commodities are also increasing. Now, the study of land use is a matter of concern not only for government agencies but for the private sector as well. It is not only the backbone of many urban planning exercises but is also important for other research activities. Urban land use data help these agencies to keep a balance between users’ demands and supply. Continuous monitoring of ULU data enables these urban players to perform their tasks more efficiently. Its practical and political importance is not a point of argument. “..management of land use change is the fundamental rationale for the theory and practice of land use planning” (Kaiser et al., 1995). The term denotes the human employment of land areas. The usual aim of conventional land use data was merely twofold (Jeffers, 1970; as cited by Rhind and Hudson, 1980):

1) to produce estimates of the proportions of land use for various defined purposes and hence of the actual land area devoted to these purposes

Figure 5.5 Urban planning process (adapted from Batty, 1992).
to produce maps of the area, showing the spatial distribution of the various land use classes.

Coppock, (1970, as cited by Rhind and Hudson, 1980) however, saw the role of land use data in a more policy-oriented sense to guide the formation of new policies and to assist the implementation of these policies when chosen. It is both possible and desirable, however, to examine the technical uses of land use data in more detail. Amplifying Duek er and Talcott's (1973; as cited by Rhind and Hudson, 1980) list, we can state these as follows:

1. to provide area and/or volume descriptions of the land uses within a defined region, usually on statistical basis, or to define the land use at a particular point in space
2. to ’overlay’ data for an area at different moments in time (temporal data) and thus produce rate-of-change in land use
3. to overlay different data sets for the same area (land use with soil, geology, geochemical distribution and so on) to determine the coincidence of physical features and gain hints as to the factors responsible for the observed land use
4. to overlay different data sets for the same area to observe relationships between physical features and socio-economic or medical attributes, or transport and other network configurations for the same reason as (3)
5. to carry out statistical analysis to explore relationships within an area - either aspatially (such as: is there an inverse correlation between the extent of land use A and that of land use B?) or spatially (is land use A next to land use D more frequently than one would expect from random scatter of uses in space? Is there clear evidence of sectoral structures in the city's land use pattern?)
6. to compare areas at the same moment in time (comparison can be carried out at different levels, e.g., the comparison of towns or countries)
7. to produce a graphic representation of the data, either as maps or graphs.

5.4.2 Why urban land use data?

Land use data are necessary because they are used by many players in the urban planning process. They are used for urban planning and management tasks, inventory, monitoring, prediction and education, or non-policy-oriented research (Douglas, 1994). The importance of land use can be viewed from the practical and political perspectives. On the one hand, we all require land on which to live; on the other, the use of any given parcel of land affects not only those who reside there or have use of the land for whatever purpose, but also those who live on or use adjacent and surrounding areas. Due to the finite supply of land and the competition for land, especially in urban areas, for certain locations perceived as particularly desirable, the purchase price of land can be extremely high. Since the financial ability to purchase land and the consequent right to influence or control land use may bring considerable power to those who posses it, control of land use has been and remains a politically contentious issue. It is patently obvious that land has also frequently been at the center of struggles over its use and control, some of these involving physical aggression and war, or some more subtle means. However, the central position of land in economic and social life is different from political aspects, which generally lead to considerable state intervention (direct or indirect) in land use control and the land market. At a more mundane level, interaction occurs between everyday behavior and future land use patterns. Existing land use arrangements in part determine where people live, where they work and how and when they travel there, where they shop, where they play etc., while such behavior in turn helps to shape future land use patterns (Rhind and Hudson, 1980).

The study of urban land use data helps us to assess the use and misuse of urban areas and their functional zones; it also gives an idea of subsequent development processes and the origin of the land use pattern of the city. The process of urban land use study assumes an added importance since it helps a planned growth in order to conserve land and regulate its utilization (Gautam, 1978). Some uses of urban land use data, according to Gautam (1978), may include the following:

1. It provides first hand familiarity with the general layout of the city.
2. It gives an idea of congested areas and slum areas separately.
3] It indicates areas of inappropriate use, e.g., areas that should be commercial or industrial but are devoted to residential or vice-versa.
4] It indicates the potential land for future town-extension / expansion.
5] It gives an overall picture of land use patterns in a city as they exist in their inter-relationships.
6] Land use data help in implementing the zoning programs of town and city planning authorities.

### 5.4.3 Sources and users of land use

Land use data can be collected from various sources. Detailed discussion of these sources is beyond the scope of this study. We shall consider the major sources of land use data and focus on the limitations in terms of spatial resolution (map scale) and temporal resolution (frequency of collection).

Land use data can be obtained from three major sources i] remote sensing (RS), ii] field surveys and iii] administrative records (Figure 5.6). Remote sensing is a technique of collecting land use data from a distance (relatively high, may be hundreds of feet, miles or more) without touching the object. We consider the RS technique, which includes: i] satellite-based (imagery from satellites) and ii] aircraft-based (photo interpretation and photogrammetry) remote sensing data.

1. **Satellite Imagery:** Satellite imageries are more useful in rural studies than urban studies (Rhind and Hudson, 1980). The advantages of satellite imageries are frequency, repetitiveness, consistency of recording and interpretation (in the case of automatic procedures), low unit cost per scene and lack of any air traffic problems etc. However, launching a satellite is an expensive task. The aircraft-based RS platforms provide more flexibility as regards temporal resolution, spatial resolution and photo scale, but incur problems with climate, geographic location, hiring aircraft, air traffic and security.

2. **Field Surveys:** Almost all field surveys require topographic maps (accurate or less accurate) or layout/development plans of the area under study. Field surveys are flexible in terms of frequency and, scale (depending upon the scale of the topo map or layout plans), and are more accurate than RS data; however, they are time-consuming, labor-intensive and often difficult to standardize.

3. **Administrative Records:** Administrative records such as building permits and historical data (previous records) can be used for updating land use data. The merits of building permits are the high frequency of update (normally daily), almost no extra costs (operational planning data used), and the recording of detail-level land use and spatial resolution (depending upon the scale of the topo map or layout plans). These data are collected at parcel level. However, the demerits of the technique includes the limited coverage of all land uses (only legal land uses are recorded). Illegal constructions or misuses are not captured. If the authorities are not notified of completion plans, land use might not be correctly recorded, and the interval between issuance of building permits and actual construction / use is a critical factor. Previous records / maps are very useful historical information, subject of course to availability.

![Figure 5.6 Sources to update land use data.](image-url)
Different types of data collection techniques provide data of different spatial resolution, and thus suitable for different applications. Timeliness and temporal consistency (should be carried out at the same time) are two important elements in the usefulness of land use data.

Table 5.3 shows the classification of potential sources of land use data, with corresponding spatial resolution. Table 5.4 illustrates the temporal characteristics of land use data. Table 5.5 outlines the uses and users of land use data. Data collected at level-A are both up-to-date and temporally consistent.

<table>
<thead>
<tr>
<th>Classification Level</th>
<th>Data Source</th>
<th>Map Output Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Landsat or similar</td>
<td>1:5,000 to 1:10,000</td>
</tr>
<tr>
<td></td>
<td>a IKONOS (680,000 m)</td>
<td>1 to 4 m resolution</td>
</tr>
<tr>
<td></td>
<td>b LANDSAT7 (705,000 m)</td>
<td>15 to 30 m resolution</td>
</tr>
<tr>
<td></td>
<td>c IRS-1C (905,000 m)</td>
<td>5.8 m resolution</td>
</tr>
<tr>
<td></td>
<td>d RADARSAT-1 (793,000 m)</td>
<td>8 m resolution</td>
</tr>
<tr>
<td>II</td>
<td>High altitude (12,500 m+)</td>
<td>1:80,000 and smaller</td>
</tr>
<tr>
<td></td>
<td>Photography, reconnaissance survey</td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>Medium altitude (3,000 to 12,500 m)</td>
<td>1:20,000 to 1:100,000</td>
</tr>
<tr>
<td></td>
<td>Photography, field survey</td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>Low altitude (below 3,000 m)</td>
<td>1:2,500 to 1:20,000</td>
</tr>
<tr>
<td></td>
<td>Field survey, individual building</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Or sub-building/plot, building permits.</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 Data sources and resolution (adapted from Rhind and Hudson, 1980).

Table 5.4 Temporal characteristics of data (adapted from Rhind and Hudson, 1980).

<table>
<thead>
<tr>
<th>Classification Level</th>
<th>Temporal characteristics of the data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Updated when change occurred or within a few months or years</td>
</tr>
<tr>
<td>B</td>
<td>Collected within last two years or up to two years between surveys</td>
</tr>
<tr>
<td>C</td>
<td>Collected between two and 15 years ago or between two and 5 years between surveys</td>
</tr>
<tr>
<td>D</td>
<td>Collected more than 15 years ago</td>
</tr>
</tbody>
</table>

Notice that the data sources, map output scale, temporal characteristics and areal extent have some correlation. For example: i) it is impossible to produce land use maps of scale 1:100 from Landsat or a similar data source and ii) it may be possible to produce land use data of large areas (particular areal extent) at temporal level A through data-source level I, but it is not possible to use data-source level V for the same area and temporal level. The choice of data source, temporal characteristics and scale of map depend upon the particular application, user and area under study (areal extent). In the urban environment, classification levels III to V are commonly used for updating general and comprehensive (detailed) land use data.

What is significant in a temporal GIS but not in an atemporal GIS is the concept of change. The general concept of change has already been discussed (Chapter-2). There are many facets of land use change, ranging from classification level, spatial resolution (scale) to temporal resolution (frequency). The change in land use is associated with these facets. Moreover, while updating the land use it is desirable to know the source or evidence of land use change, which provides the metadata regarding the land use change. These sources have some restrictions in terms of spatial and temporal resolution and the application of data.
A land use may change because of government policy (re-zoning, new housing or industrial scheme), flood, natural disaster (earthquake), migration etc. The change in land use is associated with the level of classification, and spatial (scale) and temporal resolution. These three factors are interdependent. If the level of classification is higher, then change may not be detected (Figure 5.7). If the spatial scale is smaller, the change cannot be detected. On the other hand if the spatial scale is larger, then data volume increases and generalization may be desired for strategic planning. Similarly, if the temporal resolution is not finer, then change may not be detected and some interpolation techniques may be required to express the change between two time periods. On the other hand, if the temporal resolution is finer, more storage is required. Data volume may expand exponentially. What is significant is that the change occurs in either the spatial and / or attribute part of STAO (land use).

If the spatial scale is smaller, the change cannot be detected. On the other hand if the spatial scale is larger, then data volume increases and generalization may be desired for strategic planning. Similarly, if the temporal resolution is not finer, then change may not be detected and some interpolation techniques may be required to express the change between two time periods. On the other hand, if the temporal resolution is finer, more storage is required. Data volume may expand exponentially. What is significant is that the change occurs in either the spatial and / or attribute part of STAO (land use).

### Table 5.5 Uses and users of urban land use data (adapted from Rhind and Hudson, 1980).

<table>
<thead>
<tr>
<th>Type of use</th>
<th>Example</th>
<th>Spatial resolution ¹</th>
<th>Temporal resolution ²</th>
<th>Area extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning, administration and land management</td>
<td>- District planner checking planning applications.</td>
<td>V, I</td>
<td>A</td>
<td>District</td>
</tr>
<tr>
<td></td>
<td>- Central planner involved in planning appeals.</td>
<td>V, I</td>
<td>A</td>
<td>Sub-district</td>
</tr>
<tr>
<td></td>
<td>- Equalisation of x assessment.</td>
<td>V, I</td>
<td>A</td>
<td>Country</td>
</tr>
<tr>
<td></td>
<td>- District planner returns on land central governor.</td>
<td>IV, I</td>
<td>A</td>
<td>District</td>
</tr>
<tr>
<td>Planning initiatives</td>
<td>- Inventory</td>
<td>IV-I</td>
<td>B-C</td>
<td>Area of Administrative responsibility</td>
</tr>
<tr>
<td></td>
<td>- Monitoring</td>
<td>V-I</td>
<td>A-B</td>
<td>All levels</td>
</tr>
<tr>
<td></td>
<td>- Prediction</td>
<td>V-I</td>
<td>A-C</td>
<td>All levels</td>
</tr>
<tr>
<td>Academic or non-policy-oriented research</td>
<td>- Comparing different areas and rural areas economists, sociologists.</td>
<td>V-IV</td>
<td>A-C</td>
<td>Usually local</td>
</tr>
<tr>
<td></td>
<td>- Collection of data and analysis of classification effects.</td>
<td>V-IV</td>
<td>A-C</td>
<td>Selected urban areas</td>
</tr>
<tr>
<td>Business or commercial activities</td>
<td>- Private companies looking for new fuel stations, products and introduce new items.</td>
<td>IV-I</td>
<td>B-A</td>
<td>Selected urban area</td>
</tr>
</tbody>
</table>

¹ see Table 5.3 ² see Table 5.4.

If the spatial scale is smaller, the change cannot be detected. On the other hand if the spatial scale is larger, then data volume increases and generalization may be desired for strategic planning. Similarly, if the temporal resolution is not finer, then change may not be detected and some interpolation techniques may be required to express the change between two time periods. On the other hand, if the temporal resolution is finer, more storage is required. Data volume may expand exponentially. What is significant is that the change occurs in either the spatial and / or attribute part of STAO (land use).

### 5.4.4 Monitoring urban land use changes

Timely detection of changes in land use is an important component of many urban and agriculture studies, land use planning activities and environmental assessments (Shepard, 1964; as cited by...
Jeffrey, 1991). Land use monitoring helps us to improve our understanding of the links among human activities, land use change and environmental change (Douglas, 1994). Land conversion in urban areas is spatially and temporally discontinuous, it is therefore essential for urban planners to monitor the changes in order to determine short-, medium- and long-term rates of growth and change (Martin et al., 1989). Monitoring land use change forms an important part of the process by which plans are continuously reviewed and updated. The initial plan may have been over-generous in designating land for residential uses and rather niggardly where industry was concerned. In particular, higher residential densities are complemented by a proliferation of extensive light industrial and commercial uses, employing only small numbers of people. Such changes within land use types may form a major aspect of the evolution of urban areas and it is crucial to monitor trends in order to establish the emerging pattern. A new town in a largely rural area might attract a disproportionate amount of new development into its immediate vicinity, adversely affecting growth elsewhere. Planning restrictions, intended to rectify this, might force expansion into other areas where it is undesirable or discourage growth altogether. Careful monitoring of land use change could reveal these effects, enabling planners to modify strategies accordingly. Moreover, monitoring may identify areas where particular types of change should be encouraged (or discouraged) by future plans.

Conventional change detection techniques have historically relied on the visual comparison of aerial photographs or satellite images with one another or with a corresponding topographic map (Masry and McLaren, 1979; and Usery and Welch, 1989; as cited by Jeffrey, 1991). The majority of land use monitoring projects today use a combination of ground work, aerial photograph interpretation and satellite image analysis. Recently, the digital comparison of two satellite images of the same area, obtained on separate dates, has demonstrated the potential to yield information regarding urban change (Jensen, 1983; and Martin, 1989; as cited by Jeffrey, 1991). However, traditional ground survey techniques are still important and it is impossible to assess the accuracy of other methods without them. Incorporating a structural component in automatic urban change detection using satellite image data is currently an area of research. Automatic spatial pattern recognition is still a theoretical subject; advances in expert systems and artificial intelligence may one day lead to more accurate classification of land uses, but truly operational systems are, as yet, not available.

Land use data can be updated, or change can be detected by various techniques. Figure 5.6 illustrates major sources of updating land use data. Each technique has its own advantages and disadvantages. For example, the field survey technique is labor-intensive, costly and time-consuming, but very accurate. As far as aerial photographs are concerned, these are difficult to obtain (especially in developing countries due to cost and security reasons), depend on climatic conditions, require good interpreters and are expensive. Building permits are also a cheaper and quicker source of updating land use, but they provide only limited coverage of the area. Moreover, the applicability of this technique in less developed countries is limited to planned areas, as this technique cannot be applied to informal settlements or slum areas.

### 5.4.5 Types of land use changes

Two types of land use change can be contemplated, i.e., detailed and general land use change. The former needs parcel information, while the latter does not require any parcel information. General land use change has been discussed in § 5.5.3 and can also be found in Raza (1996) and Raza et al., (1998). We first focus on detailed land use change, where parcel information is mandatory. Conventionally, land use data are collected through field surveys and administered at parcel or plot level. Advances in technology have also penetrated this technique and attempts have been made to update these land use data through remote sensing data. However, there has been little success in urban areas to update land use data at parcel level (Martin et al., 1989 and Kaiser et al., 1995). Normally, this technique is useful for general land use data / change. Another approach to update urban land use data at parcel level is through the operational planning process, i.e., building permits (Brais, 1989; Bruijn, 1990). No matter how and from where the time series land use data are collected for urban planning and management, the issue of data management remains unresolved. Without
going into details about the sources, this research focuses on updating land use data through field survey and building permits at parcel level, i.e., detailed land use change. Parcel-level land use data are indispensable for many operational and strategic planning exercises, e.g., utilities, health, education, environmental impact assessment etc. The majority of urban functions are dependent on these data (van Helden, 1994). This is also evident from user-needs interviews.

During the user-needs interviews, it was found (as shown in Table 5.1) that most of the agencies were using the KDA’s layout plans for planning and operation purposes. Moreover, the focus was the KDA, as this organization is responsible for the maintenance of Karachi land use data. The KDA has various departments, some are performing operational planning tasks, i.e., town planning (preparing layout plans), land registration or allotment, building control etc., and others are carrying out strategic planning, i.e., long-term development plans, the monitoring of land use change etc. Two separate sources of evidence, i.e., building permits and field survey, are considered for updating the land use data. To update the land use data regularly from operational planning, the spatio-temporal data from the town planning, land management (not discussed in detail, as it does not directly support land use change) and building control departments have been collected. The information, which is a prerequisite for detailed land use change applications, is discussed here. This information is associated with parcel, building permit and field survey applications (processes). The object in each application is identified. The concept of change is discussed in each application.

The DPUD and MPECD are responsible for allocating land and preparing the layout plans for the housing schemes. This is considered a parcel application. The Building Control Authority (BCA) is responsible for issuing the building permits for construction. This is considered a building permit application. The MPECD is responsible for updating the land use survey (normally through field surveys). This is considered a field survey application.

Figure 5.8 shows the information flow in a typical urban environment, particularly in DPUD, LMD and BCA applications. Based on these parcels, the LMD issues land registrations to the owners. The spatio-temporal aspects of ownership issues are not covered in this research. Land registration provides the basis for a building permit document. As mentioned earlier, urban planning is a continuous and recursive process; operational and strategic planning support each other. The core of strategic planning is spatio-temporal land use data. Spatio-temporal land use data are utilized for a variety of strategic

![Figure 5.8 Information flow for updating land use data.](image-url)
urban planning exercises, e.g., for the preparation of master plans or resettlement plans (Figure 5.8).

Layout plans or parcels are used for recording the detailed land change, where three types of evidences are possible (as mentioned earlier), i.e., building permits, field surveys and remote sensing (RS) data, including aerial photographs and digital imageries. The first two types are considered here for the further modeling and updating of land use data.

5.5 MODELING LAND USE CHANGE

Land use is a multi-dimensional concept, which includes a complex mix of various characteristics of ownership, physical environment, structures and space uses (Figure 5.9). Planners often break it down into ownership units (parcels), because these are the market units on which development projects and land use changes are based. Using OO concepts to define land use means we have to define the characteristics or responsibilities of this object in terms of data members (attribute) and operations. As the emphasis is on modeling, data members are discussed here. The object land use may be born, die, change shape, or be reincarnated. Land use can be defined by its area, perimeter, spatial and temporal neighbors, event (which causes land use to change) etc. There are two fundamental parts to land use, i.e., the attribute and spatial parts. The ‘land use’ (attribute) is a vital concept in land use change, while spatially (spatial) it may change shape or size, may move to new location etc. Therefore, change in land use creates a new object, while change in other properties merely creates a new version of the same object. This dynamic nature of land use in the case of detailed and general land use is illustrated by examples in the following sections. Parcel, building permit and field survey data needed for modeling detailed land use change are discussed first, followed by general land use change.

5.5.1 Parcel application

Parcel itself is a spatio-temporal-attribute object (STAO), which has spatial, temporal and attribute components. Spatial-temporal extent is defined by a two-dimensional spatio-temporal object called a 2D spatial-temporal object, which is the set of TTC objects. Attribute-temporal is described by its category, area, perimeter, address, owner etc. In the OO approach, each object has a unique ID, certain properties (attributes in UML) and operations. Fundamental operators associated with SpatioTemporalClass such as create, delete, kill and destroy are discussed in Chapter-4 and Raza and Kainz (2000a). The properties (spatio-temporal and attribute-temporal) can be categorized into two types, i.e., essential and non-essential (Raza et al., 1998). The essential properties of the parcel are...
defined by its address, which when changed creates a new parcel object, whereas, the non-essential properties are described by area, shape, perimeter, planning zones, TTC objects etc. As discussed earlier, the city of Karachi consists of various housing schemes and each scheme has a number of blocks. For planning purposes, the scheme is divided into various planning zones (Figure 5.3).

![Figure 5.10 CityClass and ZoneClass, the metaclass of ParcelClass.](image)

The city is divided into many planning zones: building control zones, five administrative zones, and each utility agency (gas, power, telephone etc.) has its own zone. Only two zones (ANZ and MPZ) are considered here, and these are used for master planning purposes. These are used to demonstrate OO concepts in an urban setting. Figure 5.10 shows the city, zone, scheme, block and parcel hierarchy. Each of them is considered as an object. The three classes TwoTCell-, Attribute- and LinearTime-Class identified earlier are aggregated to form SpatioTemporalAttributeClass (STAClass). Semantically, STAClass is the metaclass of CityClass, PlanningZoneClass, SchemeClass, BlockClass and ParcelClass. Spatially, CityClass is the aggregation of either PlanningZoneClass or SchemeClass. PlanningZoneClass has two subclasses, i.e., AnalysisZoneClass and MasterPlanningZoneClass. AnalysisZoneClass is the aggregation of one or more MasterPlanningZoneClasses. SchemeClass is the aggregation of one or more BlockClasses and BlockClass is the aggregation of one or more ParcelClasses. Each class, i.e., City-, Scheme-, Block-, Parcel-, AnalysisZone- and MasterPlanningZoneClass, has one or more versions. The data members (attributes) of CityClass are name, population, area, density, set of TTCs, WT_From, WT_Until, DBT_From etc. Analysis- and MasterPlanningZone-Class are defined by zonename, area, population, WT_From, WT_Until etc. The data members of SchemeClass, BlockClass and ParcelClass are shown in Figure 5.11. The instances or objects of these classes are depicted in Figure 5.12

### 5.5.1.1 ParcelClass

A parcel consists of two parts: the first part indicates the land use category and the second is the
parcel number. For example, R-1 and R-2 mean it is a residential (R) parcel with IDs 1 and 2, respectively. Similarly, categories A, P, C and I represent agricultural, park, commercial and industrial use, respectively. Although parcels inherently contain land use information (category), land use surveys or other evidence are necessary to determine the actual use, because these parcel categories merely indicate the planned use of parcels. The mismatch between the parcel’s planned category and actual use provides information regarding the illegal use or misuse of land. The objects scheme and block are the same as long as the number is unchanged. Therefore, a new object is spawned whenever the scheme or block number changes. Any other change in the attributes (data members) of scheme or block causes the birth of a new version.

The unique combination of category (land use), number, block number and scheme number determine the address of the parcel. However, for modeling and implementation purposes, a unique combination of category and number is considered as a parcel ID. Therefore, a parcel is defined by the unique combination (ID) of category and number, e.g., A-1, ST-10, FL-20 etc. Categories indicate the type of land use e.g., A is residential, ST is amenities and FL is apartment. Considering a parcel as an object, a single parcel can appear, disappear, transform or change size, position and shape (Figure 5.13).
In the case of two or more parcels, parcels can be subdivided or amalgamated (Figure 5.13). Parcels A-1, A-2 and A-3 are amalgamated to form parcel A-1. Parcel A-1 is subdivided into two parcels, A-1 and A-2. Therefore, parcels too have mutations. We consider a parcel to be the same as long as its category and parcel number is the same. Therefore, the cases listed (except transformation) in Figure 5.13 illustrate the mutation of parcel A-1, where change in size, shape and position triggers a new version of the same parcel. Change in a category and / or parcel number creates a new parcel, e.g., transformation (A-1 to A-2). In this research, the focus is on modeling detailed land use change, where the parcel object is one of the fundamental components.

While observing land use changes at parcel level, Huxhold (1991) indicated the following potential sources of changes:
1] Buildings are built and demolished.
2] Existing buildings are renovated (conversion of house into commercial enterprise without demolishing it and rebuilding).
3] Boundaries of parcel change.
4] New subdivision is created.
5] Resize or relocation of public right-of-way.
6] A street is vacated.
7] Property outside the jurisdiction is annexed.
8] Errors are detected and corrected.

Some of the changes are attribute changes, e.g., 1, 2 and 6, others are spatial changes, e.g., 3, 4, 5 and 7, while 8 could be either one or both. These changes can be explained by following the example shown in Figure 5.14 in a chronological order. Figure 5.14 shows the parcel, land use, land use objects and temporal cell complex. From T1 to T10, various changes trigger the birth and / or death of new objects or versions (depending upon the type of change). In this example, three objects are dealt with, i.e., parcel, land use and TTC objects. Spatio-temporal characteristics and the modeling of parcels are discussed below, followed by the modeling of land use objects.

5.5.1.2 Spatio-temporal characteristics of parcel objects

To understand the spatio-temporal characteristics of parcel objects, the following example is considered:
- At time T1, there were two parcels, R-1 and A-1.
- At time T2, parcel A-1 changed to P-1 (park or recreational).
- At time T4, parcel P-1 subdivided into two parts C-1 and I-1.
Chapter 5: Time in Urban Planning

**Figure 5.14 Land use change from T1 to T10: object, versions and corresponding temporal cell complex.**

**Figure 5.15 Parcel change from T1 to T10: object and versions.**
At time T6, parcel C-1 expanded in size, parcel R-1 ceased to exist and a new parcel I-2 came into existence.

At time T8, three parcels I-2, C-1’ and I-1 merged (amalgamated) to form parcel I-1’.

At time T10, parcel I-1’ changed (transformed) to R-2.

The parcel categories R, A, P, C and I represent the nature of the parcel, i.e., residential, agricultural, recreational, commercial and industrial, respectively. These categories indicate the planned land use for each parcel. C-1’ and I-1’ represent the new state, i.e., versions of the parcel C-1 and I-1, respectively.

5.5.1.3 Modeling parcel change

The parcel object and its various versions are shown in Figure 5.15. The life of parcel object R-1 is from T1 to T4. The life of parcel object I-1 is from T4 to T9. This object has two versions, i.e., I-1,1 and I-1,2. The life span of these versions is from T4 to T7 and T8 to T9, respectively. Similarly, the life span of each parcel-version is indicated in Figure 5.15. Each parcel object version is defined by a set of TTCs (Figure 5.14 and Figure 5.15), i.e.:

\[
R-1,1 = \{p1\}, \quad A-1,1 = \{p2\}, \quad P-1,1 = \{p2\}, \quad C-1,1 = \{p3\}, \quad I-1,1 = \{p4\}, \quad I-2,1 = \{p5\}, \\
C-1,2 = \{p6, p3\}, \quad I-1,2 = \{p3, p4, p5, p6\} \quad \text{and} \quad R-2,1 = \{p3, p4, p5, p6\}.
\]

The parcel-parent relationship is indicated by parcel / parent, e.g., P-1/A-1 means that the parent of parcel P-1 is A-1. The details of parcel-parent, parcel-version and version-parent relationships are shown in Figure 5.16. Each parcel object has a single parent, e.g., the parent of parcel I-2 is R-1 and the parent of C-1 and I-1 is P-1 (Figure 5.16[a]). When a parcel object has multiple parents, the youngest parent is assigned to the resulting parcel. For example, if two parcels X1T1,* and X2T2,* are amalgamated to form a new parcel X3T3,* then X3 gains X2 as a parent, such that time T1 < T2 and T2 < T3. However, for modeling and implementation purposes only a single parent is contemplated. Each parcel object may have one or many versions, e.g., parcel R-1 has one version R-1,1 and parcels C-1 and I-1 have two versions <C-1,1 and C-1,2> and <I-1,1 and I-1,2>, respectively (Figure 5.16[b]).

A finer decomposition can be achieved by establishing the parcel version-parent relationship (Figure 5.16[c]). Each parcel version may have one or many parents, e.g., the parent of parcel version I-2,1 is R-1,1 (single-parent), but the parents of parcel version C-1,2 are C-1,1 and R-1,1 (multiple parents).
Chapter-5: Time in Urban Planning

To simplify the model, the multiple relationship is not incorporated; only the single parent relationship is considered. The object parent and object version forms a tree graph structure while the version parent forms a poset structure.

5.5.2 Modeling detailed land use changes

In this description (Figure 5.14), the land use residential, agricultural, recreational, commercial and industrial are represented by Res., Agr., Rec., Com., and Ind., respectively. At time T1, there were two land uses, Res. (O1) and Agr. (O2). These two land uses were associated with parcels R-1 and A-1. At time T3, Agr. was changed to Rec. (O3) and at time T5 Rec. split into Com. (O4) and Ind. (O5). At T7, Ind. remained unchanged, Com. expanded in size and became (O’), and Res. converted into Ind. (O6). At T9, all Ind. (O6), Com. (O’5) and Ind. (O4) transformed into Ind. (O’4). Finally, at T10, all Ind. (O’4) converted to Res. (O7).

During the period T1 to T10, seven land use objects (O1, O2, ..,O7) were spawned. The object land use is the same as long as its land use is unchanged; any other change, such as parcel change, merely creates a new version (Raza et al., 1998). For example, at T3 object O3 was generated from O1 because the land use was changed, while at T7 version O5 came into existence due to the change in parcel C-1,1 (from C-1,1 to C-1,2). The land use object (O) is a collection of versions (V), where the spatial extent of V is described by a single parcel version. As mentioned earlier, a version is a set of TTCs. In Figure 5.17, the land use object O4 has two versions O4,1 and O4,2. Version O4,1 is defined by parcel version I-1,1 and O4,2 by parcel versions I-1,1,2. Similarly, other land use objects are defined by parcel objects and TTCs. Each cube (object) in Figure 5.17 represents the time, land use, land use object and parcel object, while each cube (version) depicts the time, land use, land use version, parcel version and TTC.

The conceptual schema presented in Figure 3.35, Chapter-3, is a generic model for cell tuple-based spatio-temporal data modeling. This can be extended to parcel-level (detailed) land use change by incorporating parcel and land use class (Figure 5.18). AttributeClass is a metaclass of LanduseClass and EvidenceClass. EvidenceTemporalClass is the aggregation of EvidenceClass and TemporalClass (WorldTime and DataBaseTime). EvidenceTemporalClass has two subclasses: BuildingPermitClass and FieldSurveyClass. SpatioTemporalAttributeClass is the metaclass of ParcelClass, ParcelLanduseFSClass and ParcelLanduseBPClass. ParcelLanduseFSClass and ParcelLanduseBPClass indicate that the land use is updated through field survey and building permits, respectively. ParcelLanduseFSClass is the aggregation of LanduseClass, FieldSurveyClass and ParcelClass. The ParcelLanduseBPClass is the aggregation of LanduseClass, BuildingPermitClass and ParcelClass. From the modeling perspective, both are the same, the only difference is the evidence.

Each class, i.e., ParcelLanduseFSClass and ParcelLanduseBPClass, has one subclass, i.e., ParcelLanduseVersionFSClass and ParcelLanduseVersionBPClass, respectively. The objects of both classes and subclasses can have many children. The objects of ParcelLanduseFSClass and ParcelLanduseBPClass can have one or more versions.
Figure 5.17 Decomposition of land use objects into versions and parcels.
Figure 5.18 Conceptual schema for parcel-level land use change.
The modeling and spatio-temporal aspects of field survey and building permits are not discussed in detail. They are merely used as evidence of land use change. Each parcel is defined by a set of TTCs, while a TTC may have zero or many parcels. Each parcel has one or many parcel versions. A parcel parent may have one or many children (Figure 5.18). The conceptual schema for detailed land use is demonstrated in Figure 5.18. The data members of FieldSurveyClass and BuildingPermitClass are shown in Figure 5.19 and Figure 5.20, respectively. Let T_From and T_Until be the lower and upper bounds of 1-T, respectively. The temporal consistency constraints pertaining to ParcelClass and ParcelLanduseClass can be defined as:

- T_From of any object (ParcelLanduse or parcel) must be less or equal to T_Until.
- T_From of ParcelLanduse object must be greater or equal to T_From of parcel object.

### 5.5.3 Modeling general land use change

In the case of general land use change, the spatial component of the object (land use) is represented by a polygon or TTC. For example:

At time T1, there were two land use objects, residential (Res: O₁) and agricultural (Agr: O₂). O₂ changed to recreational (Rec: O₃) at time T2. At time T3, O₃ subdivided (disappeared) into two commercial parts (Com: O₄) and industrial (Ind: O₅). O₅ is divided into two commercial parts (Com: O₆) and industrial (Ind: O₇). At time T6, O₇ turned into residential (Res: O₈). The objects, versions and corresponding temporal cell complex are shown in Figure 5.22.

The same Figure 5.22 is shown in three dimensions (space, time and attribute axes) to illustrate the changes in attribute and spatial components, which trigger the change in land use or STAO (Figure 5.23).
Worboys (1992a and 1992b) considered STA-object as a finite collection of disjoint atoms. It is not clear how the two industrial objects O₄ and O₆ will be considered in his approach. Will both industrials be assigned the same ID or different ones? We extend this definition and consider an STAO as a finite collection of disjoint versions/atoms, which are spatially or geometrically (topologically touch relation) connected to one another. Therefore, objects O₄ and O₆ are regarded as separate objects; O₄ and O₆ are disjoint but O₄ is spatially (geometrically) disconnected from O₆. At time T₅, two industrial objects O₄ and O₆ merged to form a new state of the industrial object. The new industrial object is spatially connected to O₄ and O₆. What will be the object ID in this case, O₄ or O₆? Priority is given to the oldest object, i.e., O₄; therefore the ID of the new industrial object O'₄ is another version (state) of object O₄.

Object parent ¹TOid described the object’s parent ID (parentid), lifetime (I-T) and ID (objectid). Version ²TTCOobject, versionid described the version’s lifetime (I-T), TwoTCell (TTC), ID (objectid) and version ID (versionid). The characteristics of these objects and their versions (atoms) and are summarized below and shown in Figure 5.24 and Figure 5.25:

- ¹TOid = \{ T₁-T₄ p₁O₁, T₁-T₄ p₂O₂, T₂-T₃ p₁O₃, T₃-T₅ p₁O₄, T₃-T₅ p₁O₅, T₄-T₅ p₁O₆, T₆-* p₁O₇, T₆-* p₂O₇\}
- ²TTCOobject, versionid = \{ T₁-T₄ p₁O₁, T₁-T₄ p₂O₂, T₂-T₃ p₁O₃, T₃-T₅ p₁O₄, T₃-T₅ p₂O₅, T₄-T₅ p₁O₆, T₆-* p₁O₇, T₆-* p₂O₇\}

As each land use object (STAO) is composed of various versions or atoms, each STAO can be decomposed to identify the various versions or atoms. In general land use, change in the spatial component triggers a new version(s), while change in the attribute triggers a new object. Decomposition of the object into versions is shown in Figure 5.24. Each version is associated with TTC.
Figure 5.23 Spatial and attribute change in land use along time line.
Figure 3.35 (Chapter-3) is a generic model. This can be applied for general land use change. The complete conceptual schema for general land use change is presented in Figure 5.26. AttributeClass is the metaclass of EvidenceClass and LanduseClass. EvidenceTemporalClass is the aggregation of

Figure 5.24 Decomposition of objects into versions.

Figure 5.25 Object-parent and version-parent relationship.

Figure 3.35 (Chapter-3) is a generic model. This can be applied for general land use change. The complete conceptual schema for general land use change is presented in Figure 5.26. AttributeClass is the metaclass of EvidenceClass and LanduseClass. EvidenceTemporalClass is the aggregation of
WorldTimeClass, DataBaseTimeClass and EvidenceClass.

SpatioTemporalAttributeClass (as defined earlier) has one subclass, i.e., GenLanduseClass (general land use class). GenLanduseClass is the aggregation of EvidenceTemporalClass, LanduseClass and TwoTCellClass. The object of this class is called GLUO, and each GLUO (parent) has many children. GenLanduseClass has one subclass, i.e., GenLanduseVersionClass. The object of GenLanduseVersionClass is called GLUV. Various associations are defined between GenLanduseVersionClass and EvidenceTemporalClass and TwoTCellClass. Each GLUO has one or more versions (GLUVs). A GLUV can have one evidenceid and a set of TTCs. Each GLUV (parent) has one or more children. Details of the relations or associations are captured in a logical schema (Chapter-6).

Figure 5.26 Conceptual schema for general land use change.
Chapter 5: Time in Urban Planning

The concept of change in object parcel and land use (general and detailed) has been discussed. It appears from the discussion that both have similar concepts regarding the triggering of new objects and versions. Therefore, in the case of detailed land use, the parcel object is considered the basis for recording or modeling land use changes. In this case, a ParcelLanduse object is the same as long as the parcel is the same. Any version of parcel triggers a new version of the ParcelLanduse object. This concept is different from general land use change, where a new land use object is created whenever there is a change in land use, while a new version is created whenever there is a spatial change in the land use. Therefore, the concept of change in detailed land use depends directly on the concept of change in parcel object.

5.6 Conclusions

Modern urban planning is a recursive process, where monitoring plays a vital role. Monitoring demands time series data. Current ISs such as GIS do not have the capabilities to deal with time explicitly. A TGIS is desirable to make the urban process more effective. Time in urban planning has been discussed in this chapter. Not much has been done in this area. The need of urban planning agencies for spatio-temporal data has been explored.

One issue in the urban planning process is the sharing of spatio-temporal data. Data sharing has been reviewed in a typical urban environment, i.e., Karachi. Nine major urban planning agencies were interviewed to document their need for spatio-temporal data. Some of the agencies are data users or producers and others are both. Spatio-temporal data are being used for a variety of applications, such as planning, operations, maintenance, management etc. The use and update frequency varies from agency to agency depending upon their specific application. Nonetheless, most (60%) of the agencies are utilizing land use data for planning and management purposes. Spatio-temporal land use data are the backbone of many planning exercises. The modeling of land use change and associated applications (parcel) has been discussed.

Three sources of evidence (field surveys, RS and the operational planning process / official record) for updating these land use data have been identified and their limitations are discussed. One of the sources is the operational planning process, i.e., building permit data that could be used for updating the land use data. Operational planning could be used to strengthen strategic planning. Field survey and building permits are both used for modeling land use change.

Land use change is a dynamic and complex process. Many facets (such as spatial and temporal resolution or classification level) of land use have been discussed while updating the land use change. From the modeling perspective, two types of land use can be contemplated, i.e., detailed and general land use. To preclude the complexity of land use change, an OO approach is employed. Each land use is considered as an object. Essential and non-essential properties of the land use object are defined. Change in essential properties (detailed and general land use) triggers a new land use object. The process of generating versions in detailed and general land use change is slightly different. In the case of general land use, non-essential properties are defined by area, shape etc., while in the case of detailed land use another property is added, i.e., parcel. In detailed land use, spatial extent is defined by parcel(s), while in general land use it is defined by TTC(s). In general land use change, a new version is spawned when non-essential properties (size, area, shape etc) are changed, while in detailed land use change a new version can be created when there is a change in parcel too. Detailed land use change relies on parcel information; therefore, the spatio-temporal characteristics of parcel objects are discussed. Based on the OO approach, parcel hierarchy is constructed and various parcel attributes are identified. For modeling purposes, two attributes (properties) are considered as essential properties, i.e., parcel category and number. These two form a unique parcel ID. Change in either property gives birth to a new parcel object. Change in a non-essential property (area, shape, size etc.) triggers a new parcel version. Each land use object has a single parent and may have many versions. The object parent and object version form a tree graph structure, while the version-parent forms a poset structure. Establishing these types of relations, together with the time stamp, provides a relative and absolute
temporal scale, respectively. This land use change model is applied to CTSTD1M and a conceptual schema is presented for both general and detailed land use changes. The logical schema is discussed in Chapter-6.
6.1 INTRODUCTION

This chapter presents the logical schema for CTSTDM. The conceptual schema proposed in Chapter-3 is a generic schema, while that in Chapter-5 is application-dependent, i.e., for parcel land use applications. Two types of conceptual schema have been presented for land use applications: for general and for detailed land use change. In this chapter, generic and detailed land use change schemas are dealt with further.

6.2 LOGICAL MODELING DESIGN ISSUES

Traditionally, a conceptual schema has been mapped to a logical schema by using network, hierarchical or relational data models. The first two types are now obsolete, while the relational model is still a very widely used model. Recently, attempts have been made to map a conceptual schema to a logical one by using OO or deductive data models. These emerging technologies are particularly important for spatio-temporal databases. Each model has its own inherent advantages and disadvantages. The hierarchical data model provides a tree data structure. The network data model is not much used in spatial data modeling (Laurini and Thompson, 1996). The OO approach containing the concepts of network and hierarchy, is best suited to spatio-temporal data. Various hierarchies for CTSTDM are shown in Chapter-3 and Chapter-5. The problem with this approach is that it requires greater implementation effort (Pilouk, 1996). The trend recently has been to adopt the object-relational approach, which combines the power of the relational and OO approaches. First, the relational and OO data models are briefly discussed, followed by the mapping of conceptual schema for a CTSTDM to a logical schema, where the object-relational approach will be employed.

6.2.1 Relational data model

A relational database is a database that is perceived as a collection of tables (Date, 1990). All the entities and relations are transformed into tables. Relational databases are based on the concept of the relational model, initially proposed by E. F. Codd. A relational schema does not include data, rather it provides a structure, while relations include the data. A relational schema can be defined as follows (Worboys, 1995):

A relational schema is a set of attribute names and a mapping from each attribute name to a domain (such as character, integer, date etc.). A relation is a finite set of tuples associated with a relational schema in a relational database such that:

- each tuple is a labeled list containing as many data items as there are attribute names in the relational schema
- each data item is drawn from the domain with which its attribute type is associated.

A database schema is a set of relational schemes and a relational database is a set of relations. Some properties of the relations are:

- the ordering of tuples in the relation is not significant
- tuples in the relation are all distinct from one another
- attributes (columns) are ordered so that data items correspond to the attributes in the relational schema with which they are labeled.

The relational model does not allow non-atomic values and repeating columns (attributes). All values
are explicitly stored at the intersections of tuples and columns. However, there are no links or pointers between tables. The only way the tables are linked is through foreign keys. One of the great advantages of the relational model is the query power, which is inherited by the relational algebra. This algebra consists of a set of operations (such as union, intersection difference and product) performed on relations (tables) to produce other relations (O’Neil, 1994). These query capabilities and / or operations form the basis of SQL. Pure relational data models are not suitable for many applications (CAD, CAM, CASE, GIS, LIS, TGIS etc) because of the complex data types and relationships among objects.

**Structured Query Language (SQL)**
The first prototype of SQL, known as ‘sequel’, was developed by IBM in the 1970s. Since then various versions of SQL have been floating around, e.g., SQL-86 and SQL-89. Currently, SQL2 (SQL-92) is implemented in most commercial RDBMSs, with various levels of compliance. SQL2 defines four levels of compliance, i.e., entry, transitional, intermediate and full. A conforming SQL implementation must support at least entry SQL. For example, Oracle8, release 8.0, fully supports entry SQL and has many features that conform to transitional, intermediate or full SQL (Oracle8 SQL Reference Release 8.0). The SQL3 version released by the American National Standard Institute (ANSI) / International Organization for Standardization (ISO) includes a number of OO concepts but has not yet been implemented by vendors.

### 6.2.2 OO data model

OO databases integrate object orientation (discussed in Chapter-3) and database functionality (Khoshafian, 1993). Many OO concepts have been successfully implemented in OO programming languages (e.g., C++, Smalltalk, Java++ etc). One of the motivations behind OO databases was to incorporate these concepts in databases. Object orientation allows more direct representation of reality and is based on three notions: abstract data type (ADT) or class, inheritance and object identity. As explained earlier (Chapter-3), in the OO approach, objects encapsulate the data structure (data members) and operations (data members). Database functionality is needed to ensure the persistent and concurrent sharing of information in applications. Therefore, an OO database can be defined as follows:

$$\text{OO database} = (\text{object orientation}) + (\text{database functionality})$$

$$= (\text{ADT} + \text{inheritance} + \text{object identity}) + (\text{persistence} + \text{concurrency} + \text{transaction} + \text{recovery} + \text{querying} + \text{versioning} + \text{integrity} + \text{security} + \text{performance})$$

The database functionalities are not dealt with here. These functionalities are addressed in commercial object-oriented database management systems (OODBMSs) such as ObjectStore or O2 OODBMS. Different vendors offer different capabilities in OODBMS, e.g., ObjectStore, Versant, GemStone, O2 etc provide combinations of these functionalities. Blaha and Premerlani (1998), O’Neil (1994) and Khoshafian (1993) have discussed some of these OODBMS functionalities or capabilities. As the focus here is on OO data modeling for spatio-temporal databases, these functionalities are not dealt with.

What makes the OO database more suitable for spatio-temporal databases is the fact that objects in spatio-temporal databases are never deleted (see Chapter-4, Kill and Destroyed operators for CTSTDM); rather they are kept with a valid time stamp. Objects are either created, die, reincarnate or transform into another object’s state, called version of the object. The object versioning mechanism (as mentioned in functionalities) is one of the fundamental requirements of an OO database. This object versioning takes the form of ‘pointers’ to navigate the object’s versions of varying time. These pointers are known as object IDs (OIDs). These OIDs are comparable to row IDs (RIDs) in relational database systems. However, in OO databases, the object carries data item OIDs pointing to another object, whereas this is forbidden in the relational model (O’Neil, 1994). Access to previous or following object states is an inherent part of many (complex) applications (Khoshafian, 1993). These applications are not necessarily temporal in nature, e.g., engineering design applications CAD, CAM
and CASE or office automations. But, efficient version management is indispensable to spatio-
temporal databases. Some commercial GIS provide version management techniques, e.g.,
SmallWorld GIS. Various land use objects and their versions are discussed in Chapter-5. Without
versions, a database will be populated with a large number of objects instead of versions. Objects are
the concepts that are semantically different from the other objects in the database. However versions
are semantically equal to objects; they are merely another state of an object. Two types of versioning
schemes can be contemplated:

- Linear scheme
- Branching scheme.

In the linear versioning scheme all versions are generated sequentially (e.g.,
Figure 5.16[b]), while in the branching
versioning scheme alternate versions may
be created (Figure 6.1). In Figure 6.1,
object O1 consists of two alternate
versions, i.e., V2 and V3. V1 and V7 are
the initial and final versions, respectively.
The path V1 to V3 to V6 to V7 is the
linear versioning path from V1 to V7. The
linear versioning scheme for land use and
parcel change is followed in this research.

CAD, CAM, CASE etc. are examples
where the OODBMSs have proven their power. In these systems the queries are routine or less
dynamic compared with those of spatio-temporal databases. Most of the OODBMSs provide ad hoc
query facilities (Khoshafian, 1993). These queries cannot be compared with relational calculus such
as SQL.

### 6.2.3 OO versus relational database management systems

OO DBMSs and RDBMSs have many strengths and shortcomings. These are summarized in Table
6.1. Other strengths of RDBMS are data dictionaries, associative queries (computation on a collection
of records) and security. Relational database systems have a fully integrated data dictionary available
to all users. If the data is properly indexed then data access performance can be improved. Many
commercial systems provide the facility of user-defined indexes. However, defining these indexes is
difficult for average or amateur users. By means of grant and view commands RDBMSs enforce
security.

From the temporal perspective, an added advantage of RDBMS is that a large amount of research has
been conducted into temporal query language. Most of the work is based on relational algebra to
completely match with the relational model. Examples include temporal operators for the historical
relational data model (HRDM) by Clifford and Croker (1993); an extension of SQL, i.e., temporal
query language (TempSQL), by Gadia and Nair (1993); temporal query operators for the interval-
extended relational model (IXRM) by Lorentzos (1993); the temporal relational model (TRM) and
associated temporal query language (TSQL) by Navathe and Ahmed (1993); HSQL (Historical SQL)
for historical databases by Sarda (1993); temporal query language (Tquel), an extension to Quel
(query language for INGRES), by Snodgrass (1993); temporal relational algebra (TRA) by Tansel
(1993) etc.

Both the OO and relational approaches have a number of strengths and shortcomings. Most of the OO
features can be nicely fitted into current RDBMSs. Many vendors have started to realize the strengths
of the relational and OO models and are striving to fuse these two powerful capabilities by adopting
the object-relational approach.
<table>
<thead>
<tr>
<th>Relational database systems</th>
<th>Object-oriented database systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Navigation</strong></td>
<td></td>
</tr>
<tr>
<td>Slow:</td>
<td>Fast: Navigational queries are as fast as accessing data in memory because of smart caching strategies.</td>
</tr>
<tr>
<td>Joins are inefficient for extensive navigation.</td>
<td></td>
</tr>
<tr>
<td><strong>Advanced features</strong></td>
<td></td>
</tr>
<tr>
<td>Lack advanced features:</td>
<td>Support advanced features.</td>
</tr>
<tr>
<td>Inheritance: No inheritance mechanism.</td>
<td></td>
</tr>
<tr>
<td>Schema evolution: Quality of database to change the structure of database populated with data</td>
<td></td>
</tr>
<tr>
<td>Versioning: Various states of an object.</td>
<td></td>
</tr>
<tr>
<td>Configuration: Set of mutually consistent objects.</td>
<td></td>
</tr>
<tr>
<td>Long transaction: A series of database commands that extend over a long period of time (days, weeks, months etc). RDBMS provides short transactions normally executed within few seconds.</td>
<td></td>
</tr>
<tr>
<td>Change notification: Ability of database to notify the user whenever an object is changed.</td>
<td></td>
</tr>
<tr>
<td><strong>Locking protocols</strong></td>
<td></td>
</tr>
<tr>
<td>Inflexible: Automatically lock data for commands. Locking behavior is inflexible because it does not make low-level primitive available for other kinds of behavior.</td>
<td>Flexible: The programmers can access low-level data management functionality, i.e., they have control over concurrent locking. A user may use clean read locks (transaction-consistent data), dirty read (reading without lock), write locks, long transaction locks and versions of data.</td>
</tr>
<tr>
<td><strong>Data types</strong></td>
<td></td>
</tr>
<tr>
<td>Few: New data types cannot be defined.</td>
<td>No limit: Many (complex) data types can be constructed from primitive data types, such as point from x and y and ZTC from point data types.</td>
</tr>
<tr>
<td><strong>Paradigm</strong></td>
<td></td>
</tr>
<tr>
<td>Table: Sometimes data does not naturally fit within the confines of a table and decomposition is required.</td>
<td>An object naturally fits in a file / table. No decomposition is required.</td>
</tr>
<tr>
<td><strong>Theory and standard</strong></td>
<td></td>
</tr>
<tr>
<td>Based on formal theory of relation and SQL standards. Therefore, products are standardized. Vendor independent.</td>
<td>Not based on formal theory, therefore products are not standardized and vary from vendor to vendor.</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td></td>
</tr>
<tr>
<td>Due to standard and well-known data structure, relational model is widely deployed.</td>
<td>Not widely available.</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td></td>
</tr>
<tr>
<td>Much extensibility: The ability to change database schema without changing the applications programs.</td>
<td>Lack of logical extensibility: Existing products lack logical data independence; this is not a flaw of OO databases but rather of commercial products.</td>
</tr>
<tr>
<td><strong>Database corruption</strong></td>
<td></td>
</tr>
<tr>
<td>Database and applications run in separate processes; therefore chances of data corruption are less, although some performance is lost.</td>
<td>Database is run in the application process space. Therefore, database runs the risk of security violation or corruption by wild pointers.</td>
</tr>
</tbody>
</table>

Table 6.1 Relational versus OO databases: some advantages and disadvantages (adapted from Blaha and Premerlani, 1998; and Khoshafian, 1993).
6.2.4 Object-relational approach

The object-relational approach is a compromise between the features of the OO and relational models. This approach is sometimes called the extended relational data model. Many OO features have been fitted in many commercial RDBMSs. For example, Illustra (now called Informix) and UniSQL were the first commercial software companies who incorporated many OO features in the relational data model. Recently, Oracle introduced this concept (object-relational) to overcome the shortcomings of the relational data model. The RDBMS provides a simple table structure, and operators to manipulate (through SQL) these tables and constraints (by enforcing integrity constraints, e.g., referential integrity). As discussed earlier in Chapter-3, these three components form a data model. The OO approach provides the powerful semantics needed for many complex applications, such as the spatio-temporal data model (TGIS) or 3D-GIS. This object-relational approach is pursued to derive the logical schema for unified CTSTDM.

The conceptual schema can be mapped to a logical schema through the dependency diagram technique and by checking the functional dependencies. The dependency diagram technique was first proposed by Smith (1985) for non-spatial data and later extended by Van Roessel (1986) for spatial data structure. Raza and Kainz (1999) further extended this technique to the spatio-temporal data model.

6.3 LOGICAL SCHEMA

Two conceptual schemas are presented in previous chapters: the generic in Chapter-3 and for detailed land use change in Chapter-5. These schemas can be translated into a logical schema (relational) by employing the object-relational approach. First, the generic schema is considered, then the schema for detailed land use change. As mentioned earlier, one of the reasons for adopting this approach is to fully utilize the functionality of the relational data model and SQL. To map the OO model to a relational model, initially each concrete class (Figure 3.35) is considered as a table, and data members as attributes or data fields of the table. Subclasses of LinearTimeClass are not considered as separate tables because these classes are special classes, having only data types (0-T and 1-T). To normalize (remove the update anomalies), the dependency check technique by Van Roessel (1986) for a spatial data model has been used. Van Roessel’s approach is the extension of the Smith (1985) approach. This approach has been extended to spatio-temporal data and presented by Raza and Kainz (1999). The dependency check starts by defining dependency statements, creating the dependency diagram, mapping diagrams into relations and introducing surrogate keys (if necessary). Here another approach is adopted, which is based on a procedure proposed by Blaha and Premerlani (1998). This procedure is explained in § 6.5. Both approaches resulted in similar relational structures, but the second approach provides more direct mapping from the OO to relational model. First the normalization and associated terms (functional dependencies) are defined, followed by the relational structure for the CTSTDM.

6.4 NORMALIZATION

Normalization is the process of removing the update anomalies and redundancies. These anomalies could be associated with the insert, delete or update operations. A relation is said to be in a particular normal form if it satisfies certain conditions. The process can be characterized as the successive reduction of a given collection of relations to some more desirable form (Date, 1990). A relation is said to be in a first normal form (1NF) if and only if it contains atomic values. Starting from 1NF, these normal forms can go up to $n$NF (1NF, 2NF, ..., $n$NF). Normally, 3NF is desirable, but the guiding factor is converting a set of relations into a more desirable form (say from 1NF to 2NF). A relation is said to be in 3NF if and only if the non-key attributes, if there are any, are mutually independent and fully dependent on the primary key. Two or more attributes are mutually independent if none of them is functionally dependent on any combination of the others (Date, 1990). This implies that these attributes can be updated without affecting the others. The original 3NF is
known as Codd 3NF and the modified one as Boyce/Codd normal form (BCNF). A relation is said to be in 2NF if it is in 1NF and all attributes entirely depend on the primary key. A relation is in 3NF if it is in 2NF and no attribute transitively depends on the primary key.

Dependencies play a central role in normalization and provide a basis for 3NF or BCNF. Two types of dependencies can be anticipated in a relational schema, i.e., functional and multi-valued dependencies. These dependencies are the constraints on the relational schema.

**Functional dependencies (FDs)**

Let $R = \{A_1, A_2, \ldots, A_n\}$ be a relation schema and $X \subseteq R$ and $Y \subseteq R$.

$X \rightarrow Y$ (X functionally dependent on Y)

For all valid relations $r$ of $R$

if $\forall t_1, t_2 \in r | t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

This means for every attribute $X$ there is a unique corresponding value in attribute $Y$.

**Multi-valued dependencies (MVDs)**

Multi-valued dependencies are the generalization of FDs, i.e., every FD is an MVD, but the converse is not true (Date, 1990). Let $R = \{A_1, A_2, \ldots, A_n\}$ be a relation schema and $X \subseteq R$ and $Y \subseteq R$.

There is a multi-valued dependency of $Y$ on $X$ or $X$ multi-determines $Y$ ($X \rightarrow\rightarrow Y$) if given the values for the attributes of $X$, there is a set of zero or more associated values for the attributes of $Y$, and this set of $Y$-values is not connected in any way to the values of the attributes in $R-X-Y$ (Ullman, 1995).

$X \rightarrow\rightarrow Y$ (X multi-determines Y)

For all valid relations $r$ of $R$

if $t_1, t_2 \in r | t_1[X] = t_2[X]$ then $t_3, t_4 \in r$, where

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[R-X-Y] = t_1[R-X-Y]$
- $t_4[R-X-Y] = t_1[R-X-Y]$

FDs and MVDs are applied to the CTSTDM to achieve the normalized relations.

### 6.5 RELATIONAL STRUCTURE FOR CTSTDM

The conceptual schema (Figure 3.35 and Figure 5.18) is mapped to the logical schema (relational form). SpatioTemporalClass and SpatioTemporalAttributeClass, and their subclasses, are considered for transition from classes to a relational schema (Figure 6.2). AttributeClass and its subclasses are subjective or applications-dependent and will be discussed in land use application in § 6.6. Mapping the OO conceptual schema to relational schema requires the transformation of classes, associations and generalization to tables (relation). The procedure proposed by Blaha and Premerlani (1998) is employed for achieving the normalized relational form. The processes start by implementing the classes, associations and generalization.

**Tables**

Each concrete class of the conceptual schema presented in Figure 3.35 is considered as a relation (table). The objectID of each class becomes the primary key in a table.

**Association**

1] Distinct table for many-to-many (m:n) associations:
Each many-to-many association is mapped to a distinct table. The primary key of the association is the combination of the primary key from each table.

2] Buried one-to-many (1:n) association:
   In the one-to-many association, the foreign key is buried in the class with many associations.

3] Buried zero-or-one-to-many association:
   Bury the foreign key in the class with the zero-or-one association.

4] Buried the other one-to-one association:
   Bury the foreign key in either class.

5] Sequence associations:
   Sequence or ordered association can be implemented by including a sequence-number attribute.

6] Symmetric associations:
   Symmetric association is an association between objects of the same class that have interchangeable roles / associations, e.g., parent-child association in TwoTCellClass. Blaha and Premerlani (1998) did not provide any solution to this symmetric association and proposed that it should be avoided if possible. However, this type of association is quite common in spatio-temporal databases. In this research this association is dealt as # 2 (1:n).

7] Aggregation:
   This is a kind of association, as shown in Figure 3.35. It is dealt with in a similar fashion to other associations.

Generalization
In the case of the metaclasses in Figure 3.35, these are abstract classes and do not need any further consideration.

Based on the classes and association type, relational schemas (tables) are derived as shown in Figure 6.2. Primary and foreign keys are underlined and double underlined, respectively. Association with dot (•) is an obligatory relation, e.g., a child must have a parent. Dependency statements corresponding to the conceptual model (Figure 3.35) are given here:

1] The object of PointClass is denoted by a PointID and represented by (unique) x- and y-coordinates.

2] The object of ZeroTCellClass is denoted by a ZTCID and each object has one PointID and ST_{0-T}.

3] The object of OneTCellClass is represented by a OTCID and has a set of ordered PointIDs represented by PointSequence. Each OneTCell object has a ParentID and is defined by ST_{1-T}.

4] The object of TwoTCellClass is represented by a TTCID and has a set of OTCIDs. Each TwoTCell object has a ParentID and is defined by ST_{1-T}.

5] The object of TemporalCellTupleClass is represented by a TCTID. Each TCellTuple consists of a ZeroTCellID, OneTCellID, TwoTCellID and ST_{1-T}.

6] The object of SpatioTemporalAttributePointClass is represented by a STAOPID. Each STAOPID has ZTC, AttributePoint and versions (VersionID), and is defined by WT_{1-T} and DT_{1-T}. Each version is defined by WT_{1-T} and DT_{1-T}.

7] The object of SpatioTemporalAttributeLineClass is represented by a STAOLID. Each STAOLID has OTC, AttributeLine and versions (VersionID), and is defined by WT_{1-T} and DT_{1-T}. Each version is defined by WT_{1-T} and DT_{1-T}.

8] The object of SpatioTemporalAttributeAreaClass is represented by a STAOAID. Each STAOAID have TTC, AttributeArea and versions (VersionID); and defined by WT_{1-T} and DT_{1-T}. Each version is defined by WT_{1-T} and DT_{1-T}.

Figure 6.2[1] shows the mapping of the Point and ZeroTCell classes, which resulted in two relations (tables), i.e., Point and ZeroTcell tables. Mapping the Point and OneTCell class generates three tables, i.e., Point, Onetcell and Onepointcell. The Point table is not shown in Figure 6.2[2] because it has already appeared in previous mapping (Figure 6.2[1]). Similarly, other tables are omitted when they have appeared in previous mapping processes. The Onepointcell table states that OTCID and PointID functionally determine PointSequence, but there are some other dependencies; for example, OTCID
and PointSequence determine PointID. Mapping between OneTCell and TwoTCell classes spawned three relations (tables), i.e., Onetcell, Twotcell and Twoonetcell tables (Figure 6.2[3]). The Onetcell table is not shown because of the reason stated earlier. Other relations are derived in a similar fashion. Initial tables (corresponding to the class), possible functional dependencies and normalized tables (relations) based on functional dependencies can be summarized as:

1] Point (PointID, X, Y)
   PointID → X, Y
   R1: Point (PointID, X, Y)

2] Zerotcell (ZTCID, PointID, ST_From, ST_Until,....)
   ZTCID → PointID
   R2: Zerotcell (ZTCID, PointID, ST_From, ST_Until,....)

3] Onetcell(OTCID, PointID, PointSequence, ParentID, ST_From, ST_Until,....)
   OTCID, PointSequence → PointID
   OTCID → ParentID

Figure 6.2 Transition from conceptual schema to relational schema for generic CTSTDM.

1] Point (PointID, X, Y)
   PointID → X, Y

diagram
R3: Onetcell(OTCID, ParentID, ST_From, ST_Until,…..)
R4: Onepointcell(OTCID, PointSequence, PointID)

4] Twotcell(TTCID, OTCID, ParentID, ST_From, ST_Until,…..)
   TTCID → OTCID
   TTCID → ParentID

R5: Twotcell(TTCID, ParentID, ST_From, ST_Until,…..)
R6: Twoonetcell(TTCID, OTCID)

5] Temporalcelltuple(TTTID, ZTCID, OTCID, TTCID, ST_From, ST_Until,…..)
   TTTID → ZTCID, OTCID, TTCID

R7: Temporalcelltuple(TTTID, ZTCID, OTCID, TTCID, ST_From, ST_Until)

6] STAOpoint(STAOPID, ZTCID, AttributePointID, WT_From, WT_Until, DBT_From, DBT_Until …..)
   STAOPID → AttributePointID
   STAOPID, VersionID → ZTCID

R8a: STAOpoint(STAOPID, AttributePointID, WT_From, WT_Until, DBT_From, DBT_Until …..)
R8b: STAOpointversion(STAOPID, VersionID, ZTCID, WT_From, WT_Until, DBT_From, DBT_Until ….)

7] STAOline(STAOLID, OTCID, AttributeLineID, WT_From, WT_Until, DBT_From, DBT_Until …..)
   STAOLID → AttributeLineID
   STAOLID, VersionID → OTCID

R8c: STAOline(STAOLID, AttributeLineID, WT_From, WT_Until, DBT_From, DBT_Until ….)
R8d: STAOlineversion(STAOLID, VersionID, OTCID, WT_From, WT_Until, DBT_From, DBT_Until ….)

8] STAOarea(STAOAID, TTCID, AttributeAreaID, WT_From, WT_Until, DBT_From, DBT_Until …..)
   STAOAID → AttributeAreaID
   STAOAID, VersionID → TTCID

R8e: STAOarea(STAOAID, AttributeAreaID, WT_From, WT_Until, DBT_From, DBT_Until ….)
R8f: STAOareaversion(STAOAID, VersionID, OTCID, WT_From, WT_Until, DBT_From, DBT_Until ….)

These functional dependencies validate the relations derived in Figure 6.2. The TCT functionally
determines the unique combination of ZTCID, OTCID and TTCID. The STAOPID functionally
depends on AttributePointID, while the unique combination of STAOPID and VersionID determines
the ZTCID. Therefore, the relation STAOpoint is broken down into two pieces: relation STAOpoint
and STAOpointversion. Similarly, the relations STAOline and STAOarea are split into two parts:
STAOline and STAOlineversion and STAOarea and STAOareaversion. Details of these relations
when applied to urban applications are provided in § 6.6. The primary key (single or composite) is
underlined, while the foreign key is double underlined. The above relations are mapped into the
relational diagram (Figure 6.3).
6.6 LOGICAL SCHEMA FOR PARCEL-LEVEL (DETAILED) LAND USE CHANGE

A relational schema for the generic unified cell tuple-based spatio-temporal data model (CTSTDM), which emphasizes the modeling and mapping of SpatialClass and SpatioTemporalClass, is illustrated in Figure 6.3. This can be applied to any application. Various urban applications are discussed in Chapter-5, e.g., land use and parcel change applications. The conceptual schema pertaining to the detailed and general land use is discussed in Chapter-5. The conceptual schema for detailed (parcel-level) land use change (Figure 5.18) is transformed into a logical schema (Figure 6.4). In this case, emphasis is given to the modeling and mapping of AttributeClass and SpatioTemporalAttributeClass. Two subclasses (LanduseClass and EvidenceTemporalClass) of AttributeClass were identified in Figure 5.18. Three subclasses (ParcelLanduseFS, ParcelLanduseBP and ParcelVersion) of SpatioTemporalAttributeClass are shown in Figure 5.18. The same procedure is followed to derive the logical (relational) schema as in the case of deriving the relational schema for generic CTSTDM (Figure 6.3). The classes and their relationships are more complex in this case. Dependency statements pertaining to each class are given below, followed by tables (relations) corresponding to the classes.

1] The object of ParcelVersionClass is denoted by a ParcelID and its version by a VersionID. Each parcel has an area and a set of TTCs; and is defined by WT_From, WT_Until, DBT_From, DBT_Until. A parcel may have many versions and a version (parcel) must have an object (parcel). Each version (parcels) is defined by WT_From, WT_Until, DBT_From and DBT_Until. A parcel (object) may have many children and a child (parcel) must have a parent (parcel).

2] The object of LanduseClass is defined by a LanduseID. A land use object has a description and classification level.

3] The object of FieldSurveyClass is denoted by a FieldsurveyID. A field survey is bounded by Surveydatefrom, Surveydateuntil, DBT_From and DBT_Until.

4] The object of ParcelLanduseFS is represented by a ParcelUID. Each ParcelUID has a
LanduseID, and is defined by WT_From, WT_Until, DBT_From and DBT_Until. A ParcelLUID may have many children and a child must have a parent (ParcelLUID).

5] A ParcelLanduseVerFS is represented by a ParcelLUVerID. Each ParcelLUVerID has a ParcelID, a VersionID and a FieldsurveyID, and is defined by WT_From, WT_Until, DBT_From and DBT_Until. A ParcelLUVerID may have many children and a child must have a parent (ParcelLUVerID).

Relations corresponding to the aforementioned statements are:

1] ParcelVersion( ParcelID, VersionID, TTCID, Area, BlockNo, SchemeNo, WT_From, WT_Until, DBT_From, DBT_Until, MPD, ANZ, DigitizedBy, CheckedBy, ParcelParent, VersionParent)

2] LandUse( LanduseID, Description, Classlevel)

3] FieldSurvey( FieldSurveyID, SheetNo, ConductedBy, CheckedBy, SurveyDateFrom, SurveyDateUntil, DBT_From, DBT_Until, Remarks)

4] ParcelLanduseFS( ParcelLUID, LanduseID, WT_From, WT_Until, DBT_From, DBT_Until, Parent)

5] ParcelLanduseVerFS( ParcelLUID, ParcelLUVerID, FieldSurveyID, ParcelID, VersionID, WT_From, WT_Until, DBT_From, DBT_Until, PLUParent, PLUVersion)

6] BuildingPermit( BuildingPermitID, ParcelID, LanduseID, OwnerName, OwnerAddress, NoOfStories, ApprovalDate, DBT_From,..)

7] ParcelLanduseBP( ParcelBPLUID, LanduseID, WT_From, WT_Until, DBT_From, DBT_Until, Parent)

8] ParcelLanduseVerBP( ParcelBPLUID, ParcelBPLUVerID, BuildingPermitID, ParcelID, VersionID, WT_From, WT_Until, DBT_From, DBT_Until, PBPLUParent, PBPLUVersion)

At this stage, all relations are not in normalized form; moreover these relations have associations with other relations, e.g., relation ParcelVersion is associated with TwoTCell and relation ParcelLanduseVerFS is associated with ParcelVersion and ParcelLanduseFS etc. These relations are shown in Figure 6.4 and can be dissolved into further relations based on the criteria explained in § 6.5. During this mapping process, some redundant relations are generated, e.g., ParcelVersion and ParcelVerTTC in Figure 6.4[8] appear in Figure 6.4[9] too. Figure 6.5 shows the relations after removing these redundant relations and introducing the ParcelLanduseBP and ParcelLanduseVerBP classes (Figure 6.5[10]). The same procedure is used as in the case of deriving the relations from the ParcelLanduseFS and ParcelLanduseVerFS classes.

Based on Figure 6.5, the following statements summarize initial tables (corresponding to the class), functional dependencies and normalized tables (relations) based on functional dependencies:

1] ParcelVersion( ParcelID, VersionID, TTCID, Area, BlockNo, SchemeNo, WT_From, WT_Until, DBT_From, DBT_Until, MPD, ANZ, DigitizedBy, CheckedBy, ParcelParent, VersionParent)

ParcelID, VersionID → Area, BlockNo, SchemeNo, MPD, ANZ, ...
ParcelID, VersionID → TTCID

R8: ParcelVersion (ParcelID, VersionID, Area, BlockNo, SchemeNo, WT_From, WT_Until, DBT_From, DBT_Until, MPD, ANZ, DigitizedBy, CheckedBy)

R9: ParcelVerTTC( ParcelID, VersionID, TTCID)

2] LandUse( LanduseID, Description, Classlevel)

LanduseID → Description, Classlevel

R10: LandUse( LanduseID, Description, Classlevel)
Figure 6.4 Transition from conceptual to relational schema for parcel-level land use change.

Figure 6.5 Relational schema (partial) for parcel-level land use change based on Figure 6.4.

3) FieldSurvey( FieldsurveyID, SheetNo, ConductedBy, CheckedBy, SurveyDateFrom. SurveyDateUntil, DBT_From, DBT_Until, Remarks)
   FieldsurveyID → SheetNo, ConductedBy, …
   R11: FieldSurvey( FieldsurveyID, SheetNo, ConductedBy, CheckedBy, SurveyDateFrom. SurveyDateUntil, DBT_From, DBT_Until, Remarks)

4) ParcelLanduseFS( ParcelLUID, LanduseID, WT_From, WT_Until, DBT_From, DBT_Until, Parent)
ParcelLUID \rightarrow \text{LanduseID}, \ldots

R12: \text{ParcelLanduseFS( ParcelLUID, LanduseID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, Parent)}

5] \text{ParcelLanduseVerFS( ParcelLUID, ParcelLUVerID, FieldSurveyID, ParcelID, VersionID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, PLUParent, PLUVersion)}

ParcelLUID, ParcelLUVerID \rightarrow \text{ParcelID, VersionID, \ldots}

R13: \text{ParcelLanduseVerFS( ParcelLUID, ParcelLUVerID, FieldSurveyID, ParcelID, VersionID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, PLUParent, PLUVersion)}

6] \text{BuildingPermit( BuildingPermitID, ParcelID, LanduseID, OwnerName, OwnerAddress, NoOfStories, ApprovalDate, DBT\_From, \ldots)}

BuildingPermitID \rightarrow \text{ParcelID, OwnerName, \ldots}

R14: \text{BuildingPermit( BuildingPermitID, ParcelID, LanduseID, OwnerName, OwnerAddress, NoOfStories, ApprovalDate, DBT\_From, \ldots)}

7] \text{ParcelLanduseBP( ParcelBPLUID, LanduseID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, Parent)}

ParcelBPLUID \rightarrow \text{LanduseID}, \ldots

R15: \text{ParcelLanduseBP( ParcelBPLUID, LanduseID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, Parent)}

8] \text{ParcelLanduseVerBP( ParcelBPLUID, ParcelBPLUVerID, BuildingPermitID, ParcelID, VersionID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, PBPLUParent, PBPLUVersion)}

ParcelBPLUID, ParcelBPLUVerID \rightarrow \text{ParcelID, VersionID, \ldots}

R16: \text{ParcelLanduseVerBP( ParcelBPLUID, ParcelBPLUVerID, BuildingPermitID, ParcelID, VersionID, WT\_From, WT\_Until, DBT\_From, DBT\_Until, PBPLUParent, PBPLUVersion)}

Figure 6.5 shows the relational schema for detailed (parcel-level) land use change. This schema is integrated with the relational schema for generic CTSTDM (Figure 6.3) to design the relational schema for detailed (parcel-level) land use change and is presented in Figure 6.6.

This schema is implemented in relational database management systems (RDBMSs), i.e., Oracle, and is discussed in Chapter 7 (Architecture, implementation and prototype).
6.7 CONCLUSIONS

Two major issues are studied in this chapter: the transition from a conceptual schema (for CTSTDM) to a logical one, and normalization process for a logical schema. A conceptual schema can be mapped to logical schema in four ways, i.e., network, hierarchical, relational and OO. Relational and OO models are good candidates for a logical data model. Both have many strengths, but also shortcomings. These strengths and shortcomings are discussed in general as well as particularly from the spatio-temporal data modeling perspectives.

Based on these strengths, an object-relational approach is employed to achieve the logical model. This approach has many advantages, i.e., the relational data model is based on solid theory of relational algebra and supported by SQL. SQL too is based on relational algebra and any operation is closed under any operator, i.e., any operation results in a table (relation). SQL is a simple yet powerful query language and has proven its strength in many successful commercial / business applications. Many commercial GIS software vendors are incorporating SQL in their products, e.g., ARC/INFO, SDE (Spatial Data Engine). Most of the research on temporal query languages is based on relational algebra. By employing this technique, the research in temporal query languages can be exploited.

Many OO features are discussed in this chapter and Chapter-3. The core of OO is the ADT, inheritance and object identity. ADT and inheritance are embossed in the conceptual schema (Chapter-3 and Chapter-5), while OIDs are comparable to row IDs (RIDs) in RDBMSs. It is shown that these three features can be nicely fitted into a relational data model. The ADT or classes are mapped to table(s), inheritance is a semantics process and object identity is an implementation issue. Transitions from the conceptual schema (OO) to the logical schema (relational) are performed by a procedure proposed by Blaha and Premerlani (1998). The relational schema achieved by this rigorous process is normalized through the functional dependency check technique. Two relational schemas are proposed, i.e., relational schema for a generic CTSTDM and parcel-level (detailed) land use change. Implementation of the latter is discussed in the next chapter.
Chapter-7
Architecture, Implementation and Prototype

7.1 INTRODUCTION

Previous chapters focused on various aspects of spatio-temporal data modeling. As explained earlier, a data model consists of data structure, operators and consistency rules. The structure of the model is dealt with in Chapter-3 and Chapter-6. The conceptual and logical schemas (object-relational) for the CTSTDM are presented in Chapter-3 and Chapter-6, respectively. Chapter-4 proposed the operators needed for this data model (CTSTDM). This chapter moves toward the implementation of the CTSTDM, i.e., construction and implementation of these operators. Methods are the implementation of the operation and specification of the algorithm associated with the operation.

The data are stored in relational (table) format and implementation is performed in OO programming language MS Visual C++ (VC++). Various aspects of linking these two paradigms are discussed. This provides the architecture of the CTSTDM. The interface to interact with these two components is provided through an OO graphical user interface (OOGUI). The CTSTDM is implemented for urban applications (discussed in Chapter-5). Some of the results (prototype) are presented in Raza and Kainz (2000b). This chapter also explains the steps needed for the construction of the CTSTDM and the maintenance of parcel and urban land use change data. First, the architecture of the OO TGIS is presented, followed by details of the construction, implementation and prototype.

7.2 ARCHITECTURE

The relational schema for parcel-level land use change based on CTSTDM is presented in Figure 6.6. As explained earlier, one of the advantages of the relational data model is that it provides a uniform platform for data storage and data manipulation. Data are stored in relations (tables) and communicated through SQL. SQL provides a data definition language (DDL) and a data manipulation language (DML). In spite of all the strengths of SQL92 (SQL2 or SQL), SQL is a computationally incomplete language (O'Neil, 1994). Two options can compensate for this communication deficiency:

- Embedded SQL
- Open database connectivity driver (ODBC).

Both techniques enable programmers to:

- Combine the SQL statements with other procedural programming languages such as C, C++, Java, COBOL, FORTRAN, ADA etc.
- Access the logical and physical levels. SQL works at the logical level while general-purpose programming languages (C++, BASIC, COBOL etc.) can work at the physical (system) level.

The first option is product- and platform-dependent and the second is product-dependent but platform-independent. Embedded SQL is a platform- and language-dependent option. The codes (embedded SQL) written in COBOL or C++ (product) for one platform (e.g., UNIX) may not be executable on another platform (e.g., Windows NT, Windows 2000, DOS etc.). On the other hand, the ODBC option is product-dependent but platform-independent, i.e., through the same ODBC instructions / strings data can be accessed from any platform (UNIX or Windows NT). One of the chief advantages of ODBC is the standardization of SQL. ODBC is based on the standardized version of SQL (Krunlinski et al., 1998). This standardization facilitates the writing of product- or platform-independent applications. Via ODBC, users can communicate with standard databases (e.g., relational
format) and non-standard databases (e.g., ASCII format).

The DLL (dynamic linked library)-based architecture of ODBC makes the system modular. A DLL is a library file that contains functions. A user can integrate a DLL file in a program and use the DLL’s functions. Figure 7.1 shows the architecture of ODBC. An application program has a top-level DLL (ODBC.DLL), which defines the API (applications programming interface). ODBC.DLL loads database-specific DLLs (drivers) during program execution. With the aid of the operating system (e.g., Windows Registry in Windows NT), ODBC.DLL keeps track of several databases and associated DLL and enables a single application program to communicate with different remote and local databases (Oracle, Sybase, Informix, SQL Server, DB2 etc) simultaneously.

This architecture enables the designers to construct a client / server and / or distributed system where data located locally and / or remotely can be shared. This client / server architecture is particularly significant for designing an integrated urban information system. Parcel and land use data can be located on one server and can be accessed by many clients (planning departments / agencies), each with certain rights, i.e., create / update or view rights only. Depending upon the rights, a clear distinction can be made between data owners and data users (as discussed in Chapter-5 and shown in Table 5 .1). In this fashion, consistency in the database can be guaranteed, because data duplication (often very common in the urban environment) can be avoided. Figure 7.2 is an overview of the system architecture for cell tuple-based OO TGIS. This is a three-tiered architecture (i.e., it has three layers):

- User interface (OOGUI)
- Graphical user interface controller (GUIC) and
- Databases (RDBMS).

The first two layers form an application program (AP). In OO terminology, it is an object where an OOGUI may be considered as an attribute and a GUIC as an operator, which carry the messages and talk to the database. Two types of links can be contemplated in this architecture, direct and indirect links. First, the three layers are explained, then the links (§ 7.2.4).

7.2.1 OO graphical user interface (OOGUI)

At the front, there is an OO graphical user interface (OOGUI) to communicate with spatio-temporal data, i.e., it provides forms to create, update, delete and query the spatio-temporal data.

An OOGUI can be characterized as overlapping windows, pulldown menus, buttons, icons and other graphical techniques to represent the objects (real or artificial). Each object identified earlier (Point, ZTC, OTC, TTC, Parcel, ParcellLU etc.) is linked with one form / window. Figure 7.3 illustrates the main menu for OO temporal GIS. The three classes, i.e., SpatiotemporalClass (STClass), AttributeClass and SpatioTemporalAttributeClass (FeatureClass), appear in the main menu bar (Figure 7.3). One spatial class (Point) and three spatio-temporal classes (ZeroTCell, OneTCell and TwoTCell) are shown as menu items of STClass (Figure 7.4). Each menu item, when selected, provides a form for a particular object, e.g., a form for a Point, ZTC, OTC or TTC object.
Figure 7.5 and Figure 7.6 show the forms for OTC and TTC objects, respectively. In the OTC form the user is asked to enter the first point, the vertex and the last point. In the case of a loop, the first and the last point are the same. Conceptually, there is no limit to the number of points for an OTC; however, for implementation, the limit is 500 points. The TTC form provides two options for creating a TTC.

Figure 7.2 System architecture for cell tuple-based OO Temporal GIS.
A user is asked to enter the first point and the next point; in this case, the last point is always the first point. In the second option, a user can load the coordinates (points) from an ASCII (American standard code for information interchange) file. Like OTC, conceptually, there is no limit to the number of points; however, for implementation the limit is 500 points.

The menu bar AttributeClass has two menu items, i.e., Landuse and EvidenceTemporal (Figure 7.7). EvidenceTemporal has two menu items, i.e., FieldSurvey and BuildingPermit. All menu items are associated with a form. These forms enable a user to add, delete and / or update the information related to land use, field surveys and building permits.

The menu bar FeatureClass provides access to the feature (spatio-temporal-attribute) objects. It has three menu items, ParcelLandUse, Draw and Parcel (Figure 7.8). ParcelLandUse has one sub-menu item, i.e., Evidence, which itself is associated with FieldSurvey and BuildingPermit. Each has a form to create, update and / or draw land use data. The Draw menu is not discussed here. The Parcel menu item has three sub-menu items, i.e., Create / Update, Create and Draw. Each has a form to perform these tasks.
Figure 7.5 The OneTCell Form.

Figure 7.6 The TwoTCell Form.
7.2.2 Graphical user interface controller (GUIC)

The GUIC acts as a mediator or bridge between user and database (Figure 7.2). A GUIC is set of VC++ codes (routines) and an application programming interface (API). The routines are written to communicate and perform operations on the database (Figure 7.9). An API is a set of routines that an application uses to request and carry out lower-level services performed by a computer’s operating system (MSDN, 1999). An API defines how programmers utilize a particular computer feature. APIs exist for windowing systems, file systems, database systems and, of course, networking systems. For computers running a graphical user interface, an API manages an application's windows, icons, menus and dialog boxes. These APIs are platform-independent (Windows NT, OS/2, UNIX etc.). For example, Figure 7.1, represents the API-based ODBC architecture. Another example could be obtaining the position of the mouse cursor. In an object-relational approach, the data and operations are separated. The data are stored in a database (RDBMS) and the operations in a GUIC as operators (although, procedures can be stored on a server in RDBMS). For example, operators such as create OTC or kill OTC are located in a GUIC. These GUICs are transparent to the users and some of them are discussed in the implementation (§ 7.4). In Microsoft (MS) VC++, the two main ODBC classes are CDatabase and CRecordSet (Krunilinski, 1998). Objects of class CDatabase establish connection with the data source (Oracle in this case) and objects of CRecordSet control the scrollable rowsets / recordset (Figure 7.10).
To make the records scrollable, each table of Oracle is mapped to an object in VC++, called CObjectSet, e.g., CPointSet, CZeroTCellSet, COneTCellSet etc. This CObjectSet class is based on the CRecordSet class. A recordset has a number of functions for manipulating the data, e.g., Open(), Update(), Delete(), Edit(), MoveFirst(), MoveNext(), GetRecordCount(), IsBOF(), IsEOF() etc (see MSDN (1999) for more functions and explanation of these functions).

A CRecordSet object represents a set of records selected from a data source (Oracle server). It is typically used in two forms: dynasets and snapshots (MSDN, 1999). A dynaset remains synchronized with data updates made by other users. A snapshot is a static view of the data. A dynaset form is used in the implementation. Each form represents a set of records fixed at the time the recordset is opened, but when a user scrolls to a record in a dynaset, it reflects changes subsequently made to the record, either by other users or by other recordsets in the application. CObject is the metaclass of all classes. CRecordSet is the subclass of the CObject class and CObjectSet (PointSet, ZeroTCellSet, OneTCellSet etc) is the subclass of the CRecordSet.
In an object-relational approach, each table (relations) of Figure 7.12 is de-normalized and assumed as a class. For example,

- Point table is mapped as CPointClass
- ZeroTCell table is mapped CZeroTCellClass
- OneTCell and OnePointTCell tables are mapped as COneTCellClass
- TwoTCell and TwoOneTCell tables are mapped as CTwoTCellClass
- TemporalCellTuple table is mapped as CTemporalCellTupleClass etc.

These classes are shown in Figure 7.11. Two types of classes are shown, i.e., Microsoft foundation class (MFC) and user-defined class (UDC). The latter is shown in shade. The MFC CObject class has a number of subclasses, e.g., CRecordSet, CTime, CDialog etc. CPointSet, CZeroTCellSet, COneTCellSet, COnePointTCellSet, CTwoTCellSet, CTwoOneTCellSet and CCellTupleSet are the subclasses of CRecordSet; therefore, these classes inherit all operations (data members) of the CRecordSet class. CPointClass is the aggregation of the CPointSet, CPoint and CDialog classes. The CPoint is a UDC and CDialog is an MFC. As the user interface is provided by the way of forms, the CDialog class provides all functions for designing a form. COneTCellClass is the aggregation of COneTCellSet, COnePointTCellSet, CDialog, CZeroTCellClass and CTimeClass (CTimeClass is a subclass of CTime class). Similarly, other inheritance and aggregation are shown in Figure 7.11. Figure 7.11 shows some of the classes implemented in VC++. It focuses on the generic part of the model. The purpose is to grasp the idea of integrating UDCs with MFCs. Further details are provided in Implementation (§ 7.4).

Figure 7.11 Implementation of classes: relationship between CObject class (MFC) and user-defined classes (UDCs).
7.2.3 Database

The data can be located locally or distributed over the net (Intranet). No matter how the data are distributed, a GUIC can talk to these data through the API (ODBC). In this architecture, it is assumed that data are located either on a local machine (client) or on a remote machine (server). In this implementation, data are stored on an Oracle server (Figure 7.12). The data are stored in relational form and are normalized utilizing the process explained in Chapter-6 and shown in Figure 7.12.

Some of the features of the Oracle server can be summarized as (Oracle8, 1998):

- Client/server (distributed processing) environments
- Large databases and space management
- Many concurrent database users
- High transaction processing performance (Oracle maintains the preceding features with a high degree of overall system performance. Database users do not suffer from slow processing performance)
- High availability (works 24 hours)
- Controlled availability (e.g., an administrator can disallow use of a specific application so that the application's data can be reloaded, without affecting other applications)
- Openness, industry standards.

7.2.4 User-database link

Two types of links are provided, i.e., direct and indirect links.

7.2.4.1 Direct link

A user can interact with databases via structured language and structured and / or programming language. The first option provides a direct link between user and database while the second is an indirect link. Using structured language such as SQL via Net8 (an API), a user can directly communicate with spatio-temporal data (database). Net8 enables the machines in the network to talk with one another (Oracle8, 1998). It facilitates and manages communication sessions between a client application and a remote database. Specifically, Net8 performs three basic operations:

- Connection: opening and closing connections between a client (or a server acting as a client) and a database server over a network protocol.
- Data transport: packaging and sending data such as SQL statements and data responses so that it can be transmitted and understood between a client and a server.
- Exception handling: initiating interrupt requests from the client or server.

However, information through this direct link is non-graphical (tabular form). The two-directional arrows indicate that users can read and write information / data to the database. However, writing spatio-temporal data to a database is not a simple task, e.g., writing / storing TTC in a database requires the writing of Point, ZTC and OTC objects. Therefore, this flow is indicated by a non-filled arrow. The filled arrow indicates that the information retrieved is complete, i.e., using SQL statements, complete information can be retrieved. This is a weak link. Better communication between users and the database is provided through an OOGUI.

7.2.4.2 Indirect link

The indirect link is provided through the OOGUI. Each layer is communicated with through links, i.e., links between

- User and OOGUI
- OOGUI and GUIC and
- GUIC and database (RDBMS).
Chapter 7: Architecture, Implementation and Prototype

Figure 7.12 Oracle database server (RDBMS).

User and OOGUI
A user can create, update and / or retrieve graphic and non-graphic information - graphic information in the form of maps and non-graphic in the form of tables or other ASCII files. Here, a user can use structured language, i.e., SQL (with limited functionalities) and non-structured language (with more functionalities). SQL statements can be used to retrieve spatio-temporal information; however, this option does not provide flexibility to the users as the connection to the database is maintained via ODBC. In this case too, the underlying connection is established through Net8 (in the case of Oracle). Current ODBC drivers (e.g., Microsoft ODBC for Oracle or Oracle ODBC drivers) do not allow a user to post a dynamic query, e.g., query where the tables (relations) can be joined dynamically. If the tables have to be joint, then they have to be joint at the time of establishing a connection (i.e., in GetDefaultSQL() statement in VC++) between ODBC and the database server. Updating spatio-temporal data is a recursive operation. SQL alone cannot support this recursive operation. As stated earlier, SQL is not a complete language; it lacks the recursive operators. Moreover, ODBC does not provide dynamic query facilities. These two factors motivate the use of other programming languages (in this case VC++).

OOGUI and GUIC
This link is transparent to the user. It uses the message / instruction given by the user at the first link (front end) through the OOGUI. All these messages are translated into SQL and VC++ codes and passed to the GUIC. Based on the message, appropriate actions are taken, i.e., operators (fundamental or general) are activated along with APIs.

GUIC and RDBMS
This link maintains the connection between the GUIC and the database through ODBC. All the operations are carried out via this link. The database can be located locally or remotely (Intranet). Like all other links, this link too, is transparent to the user. It seamlessly transfers the data between the GUIC and the RDBMS.

7.3 CONSTRUCTION OF CTSTDM
Construction of the model can be seen from two perspectives, i.e., feature object to spatio-temporal object and spatio-temporal object to feature object. In both cases, reality is perceived as a point, line or area object. The first case starts by defining the feature object and then the geometry (spatio-temporal) object is defined. In the second case, the geometry (spatio-temporal) object is defined, then the feature object. The steps needed to construct the unified OO CTSTDM are illustrated in Figure 7.13.

These steps focus on the generic part of the model. Steps needed for application-dependent objects are not dealt with here. The procedure needed to update application-specific objects, i.e., parcel and land use objects is discussed in Prototype (§ 7.5). The steps in respect to constructing the CTSTDM updating of ST-objects are as follows:

- At the top, reality is captured, i.e., data are collected by any technique (field surveys, RS, building permits etc.) in the form of STAO. The updating of ST-objects is considered here.
- Each ST-object, i.e., ZTC, OTC and TTC objects is checked to see whether it is duplicate or overlaps. If a ZTC object is a duplicate object, then the process is stopped; else a new ZTC object is created and the cell tuple-based spatio-temporal database (CTSTDB) is updated with a valid time stamp.
- If an OTC is a duplicate object, then the process of creating OTCs is stopped; else if it overlaps with another OTC object, then an overlay process is initiated. This overlay process is called ‘CTSTDM overlay’. This process consists of an algorithm to kill old OTCs and create new OTCs with a valid time stamp.
If a TTC is a duplicate object, then the process of creating TTCs is stopped; else an overlay process is started. In this case, the ARC/INFO overlay processor is utilized to discern the various intersections between two or more TTC objects (Appendix-7). The old TTC object(s) is (are) killed and a new object(s) is (are) created with a valid time stamp and the CTSTDB is updated.

Figure 7.13 Steps in constructing unified OO CTSTDM.

- If a TTC is a duplicate object, then the process of creating TTCs is stopped; else an overlay process is started. In this case, the ARC/INFO overlay processor is utilized to discern the various intersections between two or more TTC objects (Appendix-7). The old TTC object(s) is (are) killed and a new object(s) is (are) created with a valid time stamp and the CTSTDB is updated.
• If no ST-object is found, the process of creating a new object starts again.

There are at least three approaches for topology building: on-line, off-line and on-fly. Most of the commercial GISs provide off-line topology building (e.g., ARC/INFO). In the new version of ARC/INFO, an on-fly topology approach will be adopted. Without going into the argument ‘What is the best approach for GIS?’, an on-line topology approach has been used. This choice was pragmatic, as the topology of the objects in the CTSTDM has to be deducted and preserved before new objects are born and topology is set. The updating process requires step-by-step topology building, which is not possible in the off-line topology approach. At every update, the spatio-temporal topology of the objects is discerned; the question is why not store this topology at the time of each update rather than use the on-fly approach, which consumes processing time.

On-the-fly topology options may be a trade-off in the case of spatial databases, but not in spatio-temporal databases. On-the-fly topology is achieved by geometric computation. In the CTSTDM, an ST-object may have different associations at different times. For example, a ZTC or OTC may have different co-boundaries at different times. Spatio-temporal topology cannot be retrieved by geometry. Storing topology is also important in an OO approach, where the topology of the objects is checked for possible object versions when feature objects (parcel or land use) are updated (see § 7.5.3 and 7.5.4).

As explained earlier, this is a unified approach where all the ST-objects are treated uniformly. For example, when a ZTC object is created (Create operation), it can intersect with ZTC, OTC or TTC object(s). The details of how these operations are performed have already been explained in Chapter-4. In Figure 7.13, the emphasis is on the Create operator for ZTC, OTC and TTC objects. A user can make spatial, temporal or spatio-temporal queries (topological or non-topological). The output can be in tabular or graphical format.

7.4 IMPLEMENTATION

Various levels of spatio-temporal modeling for the CTSTDM (conceptual to logical) have been discussed in Chapter-2 to Chapter-6. Chapter-3 provides a conceptual schema for the CTSTDM; Chapter-4 investigates the fundamental operations and spatio-temporal query capabilities of the CTSTDM. Chapter-5 focused on the CTSTDM from the application perspective, and the conceptual schema for parcel and land use change applications is presented. Chapter-6 moves toward the logical schema. The logical schema for generic, parcel and land use change applications is acquired in Chapter-6. The question is: What is implemented? The conceptual and logical schemas provide a skeleton of the model; the operators discussed in Chapter-4 are the core of the model. In OO terms, the skeleton paves the way for objects, while operators carry the messages to communicate with these objects. The operators are the messengers. Without these messengers, a user cannot properly talk to the database.

7.4.1 Structure and consistency rules

The logical schema captures the structure of the model, which is physically implemented in Oracle8 (Release 8.0.5 for Windows NT) object-relational database management system (ORDBMS). This structure is shown in Figure 7.12. Various consistency rules can be defined in the database. They are defined at different levels. Three levels of consistency are identified in Chapter-2:

- Database (integrity) consistency
- Spatial and temporal topological consistency and
- Scene consistency.

Database consistency rules (integrity constraints) are the laws that govern which operations are allowed on the data and structures of a database. They protect the data and the structures of a database. These rules are implemented while defining the logical schema, using the data definition
language (DDL). Various database consistency constraints are imposed in spatio-temporal data, e.g.,
- Unique constraints (single and composite)
- Not null constraints
- Primary key constraints (single and composite)
- Referential integrity constraints (single and composite)
- Check constraints etc.

In database management systems, these database consistency constraints are well understood. Some of these (e.g., unique X,Y or ST_Until >= ST_From) constraints pertaining to Figure 7.12 are given in Appendix-3. pk_, fk_ and unq_ represent the primary, foreign and unique key constraints, respectively.

Spatial and temporal topological consistency rules are defined in Chapter-3. These rules cannot be defined in tables or by using SQL. An example is spatio-temporal consistency rule-3, i.e., every TTC is bounded by a closed cycle of ZTCs and OTCs. These consistency rules are embedded in ST-objects and implemented as an operation (see Kainz, 1995 for scene consistency). Consistency in database is not limited to the three classes of consistency. Data may be consistent with respect to the aforementioned consistencies, but might be inconsistent semantically. Semantics is a branch of linguistics that deals with the study of meaning. Although semantic consistency plays a vital role in the spatio-temporal database, it is not discussed here, as the focus of this research is the database and spatio-temporal consistency.

7.4.2 Operators

Operators are applied to the spatio-temporal database. While applying these operators or updating the spatio-temporal database, a consistent (initial or empty) database should exist. This is the initial state of the database indicates the world-TTC. As discussed in Chapter-4, the world-TTC is represented by a void (W). At implementation (physical in database) level, a primary key cannot have a null value. This world-TTC represents the large space and time where all ST-objects are embedded. To create an empty database, an abstract ID for the world-TTC is considered a 0 (zero). A 0 is an abstract ID for all objects in the database, which represents the initial state of the object or null value of the object (empty database). This is also necessary to enforce the primary and referential integrity constraints in the database. Therefore, in an empty database:
- PointID = {0}
- ZTCID = {0}
- OTCID = {0}
- TTCID = {0}
- TCTID = {0}

Four fundamental operators (Create, Kill, Destroy and Reincarnate) for the CTSTDM are designed in Chapter-4. These operators are designed for three fundamental classes of the CTSTDM, i.e., ZTC, OTC and TTC. Two operators, i.e., Create and Kill operators, are implemented, Create as a public and Kill as a private operator. Besides these, there are a number of supporting operators to perform various tasks. Appendix-4 provides the list of attributes (data members) and operations (member functions) of the Point, ZeroTCell, OneTCell, TwoTCell, land use and Parcel classes. Appendix-5 is the list of some algorithms for Point and ZeroTCell objects. This research is focused on creating the spatio-temporal database; therefore, implementation of the Create operator is discussed in the following sections. The Create operator basically performs an overlay operation, particularly in TGIS and more importantly in temporal cell complex structure. This operator has been discussed in Chapter-4. Overlay is the generic name for a set of closely related spatial operations. Overlay is a computationally intensive operation as it is combinatorial in nature, requiring, as part of the overall process, the comparison of many geometries for the purpose of detecting intersections (Harding et al., 1998). See Frank (1987); Harvey (1994); Waugh and Hopkins (1992); Chrisman et al. (1992); Sedgewick (1988); Preparata and Shamos (1990); Hopkins et al. (1992) etc. on various issues in
overlay processing. Wang (1993) presented various algorithms and compared the efficiency in terms of time.

### 7.4.2.1 Create OTC

Two cases can be contemplated while creating (inserting) an OTC, i.e., isolated OTC (OTC intersects with world-TTC) and OTC intersecting with other ST-object. In the context of a unified approach, various scenarios are discussed in Chapter-4 for the latter case, i.e., an OTC can intersect with a ZTC, an OTC or a TTC. Here, the case when an OTC intersects with an OTC is discussed and an algorithm is provided. OTC-OTC intersection is a fundamental requirement of many processes in any spatial information system, e.g., TTC (polygon) overlay, topology building, buffering etc.

**Create non-intersecting OTCs**

Creating a non-intersecting (new) OTC means that it will fall completely within an existing world-TTC (0), say ‘W’. This means it completely falls in the interior of a world-TTC (W). The input parameters are a world-TTC W and a set of non-existent points (p1, p2, ..., pn). The system automatically captures the start and end ZTC, i.e., the first (p1) and last (pn) points are the start and end ZTC, respectively.

Method CreateIsolatedOTC(W: world-TTC, set p = {p1, p2, ..., pn}, ST_FROM = t1 )

```java
{ // Assume that W exist and p1, p2, ... pn does not
  // update point table
  do for  i = 1 to n
    INSERT INTO POINT TABLE
    VALUES (pi, xi, yi);
  end do
  z1 ← new (ZTC (p1) );
  z2 ← new (ZTC (pn) );
  // update zerotcell table
  INSERT INTO ZEROTCELL TABLE
  VALUES (z1, p1, t1,*);
  VALUES (z2, pn, t1,*);
  otc1 ← new (OTC);
  len ← length (otc1);
  // update onetcell table
  INSERT INTO ONETCELL TABLE
  VALUES (otc1, len, t1,*, 0); // len = length, t1 = ST_From, * = ST_Until and 0 = parent of o1
  // update onepointtcell tables
  do for  i = 1 to n
    INSERT INTO ONEPOINTTCELL TABLE
    VALUES (otc1, pi, i);
  end do
  // update tcelltuple table
  do for  i = 1 to 2
    INSERT INTO TCELLTUPLE TABLE
    VALUES (tcti, zi, o1, W, t1,  *); // tcti = ith tcelltuple ID
  end do }
```

**Create intersecting OTCs**

The algorithms that discern the intersection between two or more OTCs are not discussed here, although they are implemented. As mentioned earlier in Chapter-4, the existing algorithms partially provide the solution to the OTC-OTC intersection numbers 1, 2, 3 and 4 (Figure 4.14), but fail in the rest of the cases, where segments / lines are parallel or where the slope of the segments are the same.
There are various ways to compute the intersection between the lines (OTCs). One method is to represent the line in parametric form (parametric expression), where the parameter \( p \) varies from 0 to 1 ( \( 0 \leq p \leq 1 \) ) for the relevant portion of the line (Laurini and Thompson, 1996). A similar NCGIA (National Center for Geographic Information and Analysis) line intersection algorithm presented by (Wise, 1996) is based on the similar concept of parametric line form. As mentioned earlier, the major problem with these algorithms is that they fail when the lines are parallel.

The parametric line approach works for straight lines (segments). Figure 7.14[1] depicts a line \( L \) in parametric form. Between the end points the parameter is \( > 0 \) and \( < 1 \), while at the start and end it is 0 and 1, respectively. Let \( L1 \) and \( L2 \) be two lines, then these lines can be expressed in a parametric form as (Figure 7.14[2]):

\[
\text{Line } L1:
\begin{align*}
x &= x_1 + p_1 \times (x_2 - x_1) \\
y &= y_1 + p_1 \times (y_2 - y_1)
\end{align*}
\]

where \( 0 \leq p_1 \leq 1 \), \( p_1 = (x_1, y_1) \) and \( p_2 = (x_2, y_2) \)

\[
\text{Line } L2:
\begin{align*}
x &= x_3 + p_2 \times (x_4 - x_3) \\
y &= y_3 + p_2 \times (y_4 - y_3)
\end{align*}
\]

where \( P_1 = (x_1, y_1) \), \( P_2 = (x_2, y_2) \), \( P_3 = (x_3, y_3) \), \( P_4 = (x_4, y_4) \) and \( P = (x, y) \)

\[
A = (x_1 - x_3) \times (y_4 - y_3) - (x_4 - x_3) \times (y_1 - y_3)
\]

\[
B = (x_2 - x_1) \times (y_3 - y_1) - (x_3 - x_1) \times (y_2 - y_1)
\]

\[
C = (x_2 - x_1) \times (y_3 - y_4) - (x_3 - x_4) \times (y_2 - y_1)
\]

\[
p_1 = A / C \text{ and } p_2 = B / C
\]

\( P = \) intersection point if, \( 0 \leq p_1 \leq 1 \), \( 0 \leq p_2 \leq 1 \) and \( C \neq 0 \)

If \( C = 0 \), then the lines are parallel and the intersection cannot be computed. To discern the intersection between OTCs, each OTC segment is checked based on this technique. The algorithm is modified to accommodate the inactive cells (OTCs). The cases where the algorithm based on parametric form fails (i.e., where \( C = 0 \)) are extended. This algorithm is not explained here. A sample (C++ codes) of this algorithm is presented in Appendix-6.

### 7.4.2.2 Create TTC

Two algorithms are presented here, i.e., for non-intersecting and for intersecting TTC(s). In the latter case, a polygon (TTC) overlay test has to be performed. The polygon overlay operation is computationally one of the most intensive and expensive operations in spatial databases and frequently proves to be a bottleneck in production-line processing (Harding et al., 1998; Laurini and Thompson, 1996; Wang, 1993 etc.). In particular, it is more complex in spatio-temporal databases because old objects are not thrown out, plus the database itself is larger than a spatial database. Fundamentally, these overlay operations are based on line-line intersections. Typically, the algorithm is divided into several stages (Wagner, 1991 as cited by Harding et al., 1998 and Wang, 1993):

- Transformation: The transformation of the TTC sets to the same coordinate system.
- Geometric intersection: Locating intersections between the geometries in the input data (maps) and subdividing the geometries to create new geometries and OTC (edge) information for the output map. Geometric intersection is the most time-consuming task and might take about 80% of the total time (Wang, 1993).
- TTC (polygon) linking: Using the new TTC (edge) from the intersection stage, TTC (polygon)
boundaries are traced to generate new area records for the output data (map).

- Output of new OTCs (edges) and new area records: TTCs (polygons).
- Database population: Assignment of attributes of parent TTCs (polygons) to child TTCs (polygons).

For the first stage (transformation), it is assumed that both sets of TTC have the same coordinate system. Researchers in computational geometry, computer graphics and GIS have developed a number of algorithms for geometric intersection, e.g., the plane-sweep, the band, the uniform grid and the Delaunay triangulation. Harding et al. (1998) and Wang (1993) discussed the algorithms in parallel processing, to exploit the multiprocessor environment. Still, these algorithms are incomplete for spatio-temporal data, particularly for the temporal cell complex data structure. Steps 2 to 5, need further modification to accommodate or to deal with the inactive objects and also to assign parents to the OTC objects. After modification, broadly, there were two alternatives:

- Implement from scratch
- Exploit the existing software functionalities for overlay operation or geometric intersection (stages 2 and 3).

The first alternative is time-consuming and requires more programming effort. Due to limited time, it was not possible to follow the first alternative; moreover, it was not the object of this research to investigate and fully implement this algorithm. The second alternative uses existing software, e.g., ARC/INFO. As stated earlier, one of the problems with existing systems is that they are closed systems. Converting the data from the CTSTDM to the format needed for ARC/INFO and reformatting the data from ARC/INFO for the CTSTDM demands extensive programming effort (although less than the first alternative). The first alternative is more flexible, faster (in terms of processing) than the second software-independent alternative, while the second is less flexible, slower (in terms of processing) than the first software-dependent alternative (ARC/INFO in this case). The ARC/INFO overlay (Union) command is used as an overlay processor to generate the overlaid coverage (temporal cell complex). This processor is indicated in Figure 7.13. The arc macro language (AML) program used for this process is listed in Appendix-7.

Prior to starting the ARC/INFO overlay process, for a given TTC (ttc') at time T, its centroid is computed based on the direction of its constituent OTC(s). These algorithms are briefly explained in § 7.4.3. The ttc' has to be compared with all existing TTCs in spatial-temporal data with time T' (such that T ≥ T') for possible geometric intersection. If N and M are the number of geometries (points, ZTCs and OTCs) in ttc' and existing TTCs, respectively, the time complexity is O(N x M). This can be improved by employing efficient techniques or so-called localization strategies (Harding et al., 1998 and Wise, 1996), such as:

- Bounding boxes
- Spatial sorting
- Plane sweep
- Domain decomposition.

The bounding box technique is employed to check the overlap between TTCs. The various advantages and disadvantage of these techniques can be found in (Harding et al., 1998). The overlapping TTCs (existing and given TTCs) are then passed to ARC/INFO to generate the geometric intersection. This method is adopted in many commercial GIS (Harding et al., 1998). The TTCs have bounding boxes called minimum bounding rectangles (MBR). The bounding box (MBR) is defined
by two values, i.e., \((X_{\text{min}}, Y_{\text{min}})\) and \((X_{\text{max}}, Y_{\text{max}})\). The MBR for TTC is shown in Figure 7.15. A simple test is performed to determine whether MERs overlap; if they do not, then there is no intersection. For example, if \((X_{\text{min}}, Y_{\text{min}})\), \((X_{\text{max}}, Y_{\text{min}})\), \((X_{\text{max}}, Y_{\text{max}})\) and \((X_{\text{min}}, Y_{\text{max}})\) and \((X_{\text{min}}, Y_{\text{min}})\), \((X_{\text{max}}, Y_{\text{min}})\), \((X_{\text{max}}, Y_{\text{max}})\) and \((X_{\text{min}}, Y_{\text{max}})\) are the values for MBR 1 and 2, respectively, the overlap test can be performed simply by checking:

\[
\begin{align*}
\text{If} & \quad (X_{\text{min}} > X_{\text{max}}) \text{ or } (X_{\text{min}} > X_{\text{max}}) \text{ or } (Y_{\text{min}} > Y_{\text{max}}) \text{ or } (Y_{\text{min}} > Y_{\text{max}}) \\
\text{then} & \quad \text{MBR do not overlap} \\
\text{else} & \quad \text{MBR do overlap.}
\end{align*}
\]

The following are the two algorithms, i.e., for isolated TTCs and for overlapping TTCs.

**Algorithm for non-overlapping TTCs:**

Method CreateIsolatedTTC(W: world-TTC, set p = \{p1, p2, ..., pn\}, ST_FROM = t1)

\[
\begin{align*}
& \text{// Assume that W exists and p1, p2, ..., pn does not} \\
& \text{// update point table} \\
& \text{for } i = 1 \text{ to } n \\
& \quad \text{INSERT INTO POINT TABLE} \\
& \quad \text{VALUES (pi, xi, yi);} \\
& \text{end do} \\
& z1 ← \text{new (ZTC (p1));} \\
& \text{// update zero_tcell table} \\
& \quad \text{INSERT INTO ZERO_TCELL TABLE} \\
& \quad \text{VALUES (z1, p1, t1,*);} \\
& otc1 ← \text{new (OTC);} \\
& \text{len ← length (otc1);} \\
& \text{ori ← orientation (otc1);} \quad \text{// ori = orientation of o1 (clockwise or anti-clockwise)} \\
& \text{// update one_tcell table} \\
& \quad \text{INSERT INTO ONE_TCELL TABLE} \\
& \quad \text{VALUES (otc1, len, t1, *, 0);} \quad \text{// len = length, t1 = ST_From, * = ST_Until and 0 = parent of o1} \\
& \text{// update one_point_tcell tables} \\
& \text{for } i = 1 \text{ to } n \\
& \quad \text{INSERT INTO ONE_POINT_TCELL TABLE} \\
& \quad \text{VALUES (otc, pi, i);} \\
& \text{end do} \\
& ttc1 ← \text{new (TTC);} \\
& cen ← \text{centroid (TTC);} \quad \text{// cen = centroid} \\
& \text{area ← area (TTC);} \\
& \text{peri ← perimeter (TTC);} \\
& \text{// update two_tcell table} \\
& \quad \text{INSERT INTO TWO_TCELL TABLE} \\
& \quad \text{VALUES (ttc1, area, peri, cen, t1, *, 0);} \quad \text{// 0 = parent of ttc1} \\
& \text{// update two_one_tcell tables} \\
& \quad \text{INSERT INTO TWO_ONE_TCELL TABLE} \\
& \quad \text{VALUES (ttc1, otc1);} \\
& \text{// update tcell_tuple table} \\
& \text{if ori = clockwise} \\
& \quad \text{INSERT INTO TCELL_TUPLE TABLE} \\
& \quad \text{VALUES (tct1, z1, otc1, W, t1, *);} \quad \text{// tct = tcell_tuple ID} \\
& \text{else} \\
& \quad \text{INSERT INTO TCELL_TUPLE TABLE} \\
& \quad \text{VALUES (tct1, z1, otc1, ttc1, t1, *);} \\
\end{align*}
\]
VALUES (tct2, z1, otc1, W, t1, *)

end if

}  

Algorithm for overlapping TTCs:
Method CreateTTC(W: world-TTC, set p = {p1, p2, ..., pn}, ST_FROM = t1 )
{
   // Assume that W exists
   // This method detect the intersection of two or more TTCs, perform the overlay operations and update the
temporal cell complex
   // The intersection means the intersection of boundary and / or interior of two TTCs
   'ttc ← new (TTC);
   m = count(TTCs);    // m = number of existing TTCs
   // check intersection of 'ttc and existing TTCs
   do for  j = 1 to m
      i=0;
      if 'ttc ∩ TTCj // TTCj = j th TTC
         i ++;    // number of intersected TTC
      end do
   // create 2-temporal cell complex
   do for k = 1 to i
      tcc = overlay('ttc,TTCk); // tcc = 2-temporal cell complex
   end do
   n = count (tcc);    // n = number of TTCS | TTCS ∈ tcc
   in tcc
   do for j = 1 to m
      ↓ TTCj   // ↓ = kill j th TTC
   end do
   do for k = 1 to n
      ttck ← new (TTC);   // ttck = k th TTC | TTC ∈ tcc
      partttck = parent(ttck);  // partttck = parent of ttck
      l = count (OTC); // l = number of OTCs | OTC ∈ ttck
      do for x = 1 to l
         otckx ← new (OTC);  // otckx = x th OTC | OTC ∈ kth TTC
         lxttc = left(otckx);  // lxttc = left TTC of x th OTC | OTC ∈ kth TTC
         rxttc = right(otckx);  // rxttc = right TTC of x th OTC | OTC ∈ kth TTC
         parotckx = parent(otckx);   // parotckx = parent of otckx
         y = count(Point); // y = number of Points | Points ∈ otcx
         do for m = 1 to y
            pm ← new (Point); // pm = m th Point of x th OTC;
            // update point table
            if pm exist
               INSERT INTO POINT TABLE
               VALUES (pm, xm, ym);
            end if
            if (m == 1 or m == y)
               z1 ← new (ZTC (pm) );
               // update zerotcell table
               INSERT INTO ZEROTCELL TABLE
               VALUES (z1, pm, t1,*);
            end do
            if otckx ¬ exist
               break
            end if
         end do
      end do
   end do
}
{ len ← length (otckx);
ori ← orientation (otckx); // ori = orientation of otckx (clockwise or anti-clockwise)
// update onetcell table
  INSERT INTO ONETCELL TABLE
  VALUES (otckx, len, t1, *, parotckx); // len = length, t1 = ST_From and * = ST_Until
// update onepointntcell tables
  do for m = 1 to y
    INSERT INTO ONEPOINTCELL TABLE
    VALUES (otc, pm, m);
  end do
}

if ttck ¬ exist
{
  cen ← centroid (ttck); // cen = centroid
  area ← area (ttck);
  peri ← perimeter (ttck);
  // update twotcell table
  INSERT INTO TWOTCELL TABLE
  VALUES (ttck, area, peri, cen, t1, *, parttck);
// update twoonettcell tables
  x = count (OTC);  // x = number of OTCs | OTC
  do for l = 1 to x
    INSERT INTO TWOONETCELL TABLE
    VALUES (ttck, otcl);
  end do
}
end do

// update tcelltuple table
  do for k = 1 to n
  {
    ttck ← new (TTC);   // ttck = kth TTC | TTC
    l = count (OTC); // l = number of OTCs | OTC
    do for x = 1 to l
      otckx ← new (OTC);  // otckx = xth OTC of kth TTC;
      lxttc = left(otckx);  // lxttc = left TTC of xth OTC | OTC
      rxttc = right(otckx);  // rxttc = right TTC of xth OTC | OTC
      ori = orientation(otckx);
      if ori = clockwise
        tct = max(ttcid)
        if ( x == 1 )
          z1 ← ZTC (otckx);
          INSERT INTO TCELLTUPLE TABLE
          VALUES (tct, z1, otckx, lxttc, t1, *);  // tct = tcelltuple ID
          VALUES (tct+1, z1, otckx, rxttc, t1, *);
        else if (x == l)
          z2 ← ZTC (otckx);
          INSERT INTO TCELLTUPLE TABLE
          VALUES (tct, z2, otckx, lxttc, t1, *);
      else
        ... // insert remaining code
      end if
    end do
  } end do
} end do
VALUES (tct+1, z2, otckx, rxttc, t1, *);
end if
}

7.4.3 Algorithm: direction of OTC and centroid of TTC

The orientation of each constituent OTC of TTC is computed by a side operation (Worboys, 1995). Consider Figure 7.16, where an OTC is represented by a1, P_s and P_e are the start and end ZTCs and P_c is a given point (centroid in the case of TTC). The side of P_c of a1 is given as:

\[
\text{Side}(P_c) = \left( \frac{(P_c.x \cdot P_s.y - P_c.y \cdot P_s.x) + (P_s.x \cdot P_e.y - P_s.y \cdot P_e.x) + (P_e.x \cdot P_c.y - P_e.y \cdot P_c.x)}{2} \right)
\]

If Side \((P_c) < 1\); \(P_c\) is on the right-hand side of a1.
If Side \((P_c) > 1\); \(P_c\) is on the left-hand side of a1.
If Side \((P_c) = 0\); \(P_s\), \(P_e\) and \(P_c\) are collinear.

A centroid or a center of gravity of an areal object (TTC in this case) is the point at which it would balance if it were cut out of a sheet of material of uniform density (Worboys, 1995). The centroid \(P_c(X_c, Y_c)\) coordinates are calculated as an average of all points (ZTC and intermediate).

\[
X_c = \frac{1}{N} \sum_{i=1}^{N} X_i
\]
\[
Y_c = \frac{1}{N} \sum_{i=1}^{N} Y_i
\]

A point in polygon test is performed to validate this centroid. Several algorithms exist, one of the most well-known is based on the half-line (Jordan) theorem (Laurini and Thompson, 1996). In order to know whether a point \(P\) (in this case \(P_c\) centroid) is inside or outside a TTC (\(A\)), a half-line is drawn from \(P\) and extended to the extreme x-coordinate. A count is made for the number of intersections of this half-line with TTC boundaries. The count could be either odd or even. If it is odd then the point \(P\) is inside, otherwise it is outside the TTC (Figure 7.17).

The TTC in Figure 7.17 is not a convex hull. The above algorithm for point in polygon (TTC) holds for a convex hull or convex TTC (temporal polygon). In the case of concavities, the centroid may lie outside the TTC; in that case, the same algorithm is employed but the point \(P\) is moved along the line, which has a longer segment forming the intersection with the TTC.
7.5 PROTOTYPE

In the prototype, first the data transfer mechanism is explained, followed by the description of parcel and land use data and updating procedures. The prototype also demonstrates the formulation of various types of queries (using SQL) to access this parcel and land use data.

7.5.1 Data transfer

The parcel and land use data of Karachi are used to demonstrate the applicability of the model. The two types of information come from different KDA departments. The former come from DPUD and the latter from the MPECD. Various aspects of the spatio-temporality of these two types of data have been discussed in Chapter-5. Various attributes associated with these data have been identified earlier. Land use and parcel records from 1970 to 1998 have been collected, digitized and stored in the relational form. The land use data of 1987 and 1993 were available in AutoCAD format. The land use data of 1998 were collected through field survey during the fieldwork. These too, were digitized in AutoCAD. Although a land use survey was conducted in 1973 because these data were stored in paper format, the data were lost and could not be collected. Therefore, land use of 1973 was assumed as vacant. All land use records were imported to ARC/INFO to be finally stored in Oracle. A translator to read the data from ARC/INFO and to store them in the CTSTDM has been designed and implemented in VC++ (Figure 7.18).

7.5.2 Spatio-temporal data: land use and parcel records

Layout plans and associated attribute data of parcels, indicating the various parcel mutations, have been collected from the parcel registers of the DPUD. Oracle (8.01) has been used for storing the parcel and land use data and VC++ has been used for the implementation of operators. Like any other RDBMS, Oracle too lacks recursive functions and certain consistency checks could not be performed. Therefore, these consistency checks are implemented in VC++. The status of land use (Block 4 of scheme 5 of ANZ 16) in 1987, 1993 and 1998 is shown in Appendix-8. In 1970 Block 4 contains about 400 parcels.

7.5.3 Updating land use objects

A menu for FeatureClass is shown in Figure 7.7. Two types of feature objects are considered here, i.e., parcel and land use. A form is provided to the user for both features, to create / update and draw these features (Figure 7.19 and Figure 7.21). To create and / or update parcel-level land use data, a user is given an option to enter land use ID, field survey ID (evidence ID), parcel and its versions ID. The interval time is provided by WT_From and WT_Until. In most of the cases, only WT_From is required; WT_Until is assigned by the system. It may be required in cases where the object life-span is known, e.g., entering past objects, which are no longer alive. When new objects are born and existing objects cease to exist, this WT_From becomes the start time for the former and end time for the latter. DBT_From and DBT_Until are assigned by the system.
Initially, the concept of change in land use (both general and parcel-level) has been discussed in Chapter-5. This concept is elaborated in Figure 7.20. Figure 7.20 illustrates the process of creating and/or updating the parcel land use object, using the UML activity diagram. In this activity diagram, states are actions or sub-activity states, and transitions are triggered by the completion of actions or sub-activities, with the focus on flow-driven by internal processing rather than external events (UML 1.3, 1999). An action state indicates an entry action and at least one outgoing transition involving the implicit event of completing the entry action. The action may be described by natural language, pseudocode or programming language. A decision icon is used when conditions are employed to indicate different possible transitions that depend on Boolean conditions of the owning object. The transition fork indicates that activities occur in parallel.

Figure 7.19 The parcel land use form.
When a land use (LU) and a parcel object (ParO) are entered at time $t'$, the parcel land use object (PLUO) of given ParO is retrieved. The $t''$ is the lower bound of WT of PLUO, where $t' \geq t''$. If the land use (LU1) of PLUO is equal to the given land use (LU), then the process is discarded. A request for new land use made. If LU1 is not equal to LU, then the topology of the parcel object (ParO) is detected, i.e., all the neighboring parcel objects (nParO) are loaded. The land use (nLU) associated with these neighboring parcel objects is acquired. If the given LU is not equal to nLU, then PLUO is killed and a new object PLUO is created. If the given LU is equal to nLU, then the parcel land use object (nParO) of nLU is retrieved, PLUO is killed and a new version (PLUO'') of PLUO' is spawned. In this activity diagram, it is assumed that the parcel object exists. The updating of parcel objects is discussed below.

In the draw option, a user can enter either parcel land use ID (with version ID) or land use, thus defining the point time (Time From) or interval time (Time From and Time Until). These provide a crude way of retrieving or displaying the land use data. The more flexible or powerful way is provided through SQL. This option is discussed in § 7.5.5.

### 7.5.4 Updating parcel objects

As mentioned earlier, a parcel object has spatial, temporal and attribute components. A parcel form is shown in Figure 7.21, where the user can enter parcel number and attributes, e.g., block number, scheme number, ANZ etc., as identified in Chapter-5. Temporal information is provided through World-Time interval, spatial information by defining the TTC(s). This can be achieved in two ways:
• By defining a new TTC(s)
• By defining the existing TTC(s).

Selecting the first option will lead to the TTC form (Figure 7.6). The form has already been explained earlier. The second option assumes that the TTC(s) already exist(s), if not, the system will automatically jump to the TTC form. Various spatio-temporal characteristics or dynamic aspects of parcel objects are discussed in Chapter-5 and shown in Figure 5.13. A single parcel can appear, disappear, transform or change size, position and shape. In the case of two or more parcels, a parcel can be subdivided or amalgamated. Subdivision and change in a single parcel can be captured by these two options. Parcels can be amalgamated by defining the parcel and version list in the Amalgamate option (Figure 7.21). The processes of creating and / or updating parcel objects are depicted in the activity diagram (Figure 7.22).

Figure 7.21 The parcel form.
For example, at time $t'$, a parcel object ($\text{ParO}$) is added to the database (Figure 7.22). If the user selects the option (Existing TTC), say $\text{ttc2}$, the parcel / version ID ($\text{ParO}_1$) of $\text{ttc2}$ is compared with $\text{ParO}$. If both are equal, the operation returns to the initial state for a new parcel object; if not, the neighboring (assumed single parcel object) parcel / version ($\text{ParO}_3$) of $\text{ParO}_1$ is compared with $\text{ParO}$. If both are equal, $\text{ParO}_1$ is killed and a new version of $\text{ParO}_3$ ($\text{ParO}_3'$) is generated, else a new object ($\text{ParO}'$) is spawned. If a user defines a new TTC ($\text{ttc1}$), a check is performed to see whether $\text{ttc1}$ overlaps with existing TTCs ($\text{ttc3}$). To simplify the illustration / concept, it is assumed $\text{ttc3}$ overlaps with a single TTC. If $\text{ttc1}$ does not overlap with any TTCs, a new parcel object / version is created. If $\text{ttc1}$ does overlap with existing TTCs ($\text{ttc3}$), a parcel / version ($\text{ParO}_2$) associated with $\text{ttc3}$ is retrieved, $\text{ttc3}$ is killed and new TTCs, i.e., $\text{ttc1}'$ and $\text{ttc1}''$ are generated. A comparison is made to check whether $\text{ParO}_2$ is equal to $\text{ParO}$; if yes, a version of $\text{ParO}_2$ is killed and a new version of $\text{ParO}_2$ ($\text{ParO}_2'$) is created. This new version is associated with $\text{ttc1}'$ and $\text{ttc1}''$. If $\text{ParO}_2$ and $\text{ParO}$ are not equal, the parcel / version of $\text{ParO}_2$ is killed and a new parcel / version ($\text{ParO}$) associated with $\text{ttc1}'$ and $\text{ttc1}''$ comes into existence. The process of amalgamation is rather simpler and is not shown in Figure 7.22.

*Figure 7.22 UML activity diagram: process of updating parcel objects.*
In summary, apart from AML and DLL, a program of 21898 source code lines has been written in VC++ to implement the model. Although the number of source code lines is not a criterion to judge the efficiency of the program, an efficient program should have a minimum number of source codes. As the focus of the research is on the spatio-temporal database, emphasis has been given to the pragmatic approach and the Create and Kill operators for classes in Figure 5.18 have been implemented.

7.5.5 Accessing spatio-temporal objects

Accessing spatio-temporal objects / data demands temporal (when, now, before, after, meet etc.) and spatial (left, right, inside, meet etc.) operators. Theoretical work on temporal relational databases by Allen (1983, 1984), Gadia and Nair (1993) etc. for these operators is not fully implemented in commercial databases such as Oracle. None of the commercial GIS stores all the spatial data (topological) in a relational form. Spatial operators in Oracle are dedicated to the spatial data model called Spatial Cartridge, which is a non-topological model. These functions may not be utilized for a cell tuple-based spatio-temporal data model. Without going into detail about these functions, some examples are shown to demonstrate the retrieval of spatio-temporal data through ‘Entry-level SQL92’, which is implemented by Oracle. A conforming SQL implementation must support at least Entry SQL (Oracle8, 1998).

Langran (1992a) indicated four types of queries in spatio-temporal databases:

- Simple temporal query: What was the state of the object(s) at time $t_i$? In this type of query, point time is given.
- Temporal range query: What was the state of the object(s) during the time interval $t_i$ to $t_j$? In this type of query interval $1-T'$ $[t_i, t_j]$ time is given.
- Simple spatio-temporal query: What was the state of the objects at time $t_i$?
- Spatio-temporal range query: What was the state of the objects during the time interval $t_i$ to $t_j$?

For the last two types a query window has to be developed on the screen. These queries are not discussed here. Apart from these four types of queries, a TGIS should be able to scan the history of any object. For example how has the object evolved? Knowledge of this type of evolution can be captured through the object-parent relationship. How has an object evolved? or what was the parent of the object? The following are examples of queries of this type, followed by simple temporal and temporal range queries.

7.5.5.1 Scanning history of objects

Query 1  How has parcel D-139 evolved? Or show the history of parcel D-139?

```
Select parcelversion.parcelid, parcelversion.versionid, parcelversion.WT_From, parcelversion.WT_Until, parcelversion.parcelparent, parcelversion.versionparent
From parcelversion
Where parcelversion.parcelid='D-139' Order by parcelversion.parcelid
```

The above query shows that parcel D-139 has three versions (Table 7.1). Versions 1, 2 and 3 were valid from 01-Jan-72 to 02-Apr-89, 02-Apr-89 to 12-Nov-94 and 12-Nov-94 till now, respectively. Further, it indicates the ParcelParent and ParcelVersionParent. To display this parcel and its version, the spatial information (Point, ZTC, OTC and TTC) is also needed. This can be achieved by modifying the SQL statement

```
Select parcelversion.parcelid, parcelversion.versionid, parcelversion.WT_From, parcelversion.WT_Until, parcelversion.parcelparent, parcelversion.versionparent, parcelversion.parcelid, parcelversion.versionid, parcelversion.parcelparent, parcelversion.versionparent
From parcelversion
```

<table>
<thead>
<tr>
<th>PARCELID</th>
<th>VERSIONID</th>
<th>WT_FROM</th>
<th>WT_UNTIL</th>
<th>PARCELPARENT</th>
<th>VERSIONPARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D-139</td>
<td>2</td>
<td>02-APR-89</td>
<td>12-NOV-94</td>
<td>D-139</td>
<td>1</td>
</tr>
<tr>
<td>D-139</td>
<td>3</td>
<td>12-NOV-94</td>
<td>D-139</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1 History of parcel D-139.
Table 7.2 is the tabular form of parcel D-139 and its versions. To simplify the SQL statements, the SQL statement to retrieve spatial information is omitted from the rest of the query examples because it is the same in all cases.

<table>
<thead>
<tr>
<th>PARCELID</th>
<th>VERSION</th>
<th>WT_FROM</th>
<th>WT_UNTIL</th>
<th>TWOTCELLID</th>
<th>ONETCELLID</th>
<th>POINTID</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>2</td>
<td>02-APR-89</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>2</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>3</td>
<td>12-NOV-94</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>3</td>
<td>12-NOV-94</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>2</td>
<td>02-APR-89</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>2</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>3</td>
<td>12-NOV-94</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>3</td>
<td>12-NOV-94</td>
<td>12-NOV-94</td>
<td>436</td>
<td>1182</td>
<td>703195.31</td>
<td>178252.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
<tr>
<td>D-139</td>
<td>1</td>
<td>01-JAN-72</td>
<td>02-APR-89</td>
<td>136</td>
<td>446</td>
<td>703196.5</td>
<td>178229.5</td>
<td></td>
</tr>
</tbody>
</table>

Query 2 What was the land use of C-38 before 1993?

Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid, Parcellufs.wt_from, Parcellufs.wt_until, Parcellufs.parent
From Parcellufs, parcelluverfs
Where parcelluverfs.parcelid = 'C-38' and Parcellufs.parcelluid = parcelluverfs.parcelluid and Parcellufs.wt_from <= to_date('1993', 'yyyy') and Parcellufs.wt_until >= to_date('1993', 'yyyy') or Parcellufs.wt_until is null
Order by Parcellufs.parcelluid, Parcellufs.wt_from, Parcellufs.wt_until

Table 7.3 Previous land use of parcel C-38.

<table>
<thead>
<tr>
<th>PARCEL</th>
<th>LANDUSEID</th>
<th>WT_FROM</th>
<th>WT_UNTIL</th>
<th>POINTID</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>17</td>
<td>C-38</td>
<td>01-JUN-87</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query 3 What is the land use history of parcel ST-0 to date?

Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid,
Table 7.4 shows that in 1998 the land use of parcel ST-0 was 13 (government office), in 1993 it was 17 (utilities) and in 1972 it was 26 (vacant). This list of land use codes and description is provided in Appendix-9.

It can be seen from Queries 1 to 3 that no explicit time is required, i.e., the SQL statement does not contain any WT or DBT condition, although the output of the queries is spatio-temporal and / or attribute-temporal.

### 7.5.5.2 Simple temporal query

The parcel and land use objects are embedded by interval time; the point time $t_i$ (0-T) could relate to 1-T in various ways. These relations are discussed in Chapter-2 (Figure 2.28). Broadly, $t_i$ can be related to 1-T in two ways, i.e., $t_i$ is covered by 1-T or $t_i$ is not covered by 1-T. To demonstrate simple query language, i.e., in the first case when $t_i$ is covered by 1-T, there could be three possibilities (Figure 7.23):

1. $t_i$ is in between time interval (1-T) or $t_i$ is during 1-T
2. $t_i$ is a lower bound of 1-T or $t_i$ meets 1-T
3. $t_i$ is an upper bound of 1-T or $t_i$ finishes 1-T.

**Query 4** What was the state of residential land use in 1990?

This query will retrieve all the objects valid at time $t_i$. This $t_i$ could be any point between time interval 1-T [$t_0$, $t_n$].

```sql
Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid,
Parcellufs.wt_from, Parcellufs.wt_until
From Parcellufs, parcelluverfs
Where Parcellufs.landuseid = '1' and Parcellufs.wt_from <= to_date('1990', 'yyyy') and
(Parcellufs.wt_until >= to_date('1990', 'yyyy') or Parcellufs.wt_until is null ) and
parcellufs.parcelluid = parcelluverfs.parcelluid
Order by Parcellufs.parcelluid, Parcellufs.wt_from
```

**Query 5** What was the state of residential land use in 1995?

```sql
Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid,
Parcellufs.wt_from, Parcellufs.wt_until
From Parcellufs, parcelluverfs
Where Parcellufs.landuseid = '1' and Parcellufs.wt_from <= to_date('1995', 'yyyy') and
(Parcellufs.wt_until >= to_date('1995', 'yyyy') or Parcellufs.wt_until is null ) and
parcellufs.parcelluid = parcelluverfs.parcelluid
Order by Parcellufs.parcelluid, Parcellufs.wt_from
```

The output of Queries 4 and 5 is shown in Table 7.5, Table 7.6, Figure 7.24 and Figure 7.25, respectively. Similarly, the land use in 1999 is shown in Figure 7.26.
Chapter-7: Architecture, Implementation and Prototype

Query 6  Show land uses which turned to residential in 1998.

This type of query will retrieve all the objects where \( t_i \) is equal to the lower bound of \( 1-T \).

Select  
Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluid, parcelluverfs.parcelluid, Parcellufs.wt_from, Parcellufs.wt_until, Parcellufs.parent

From  
Parcellufs, parcelluverfs

Where  
parcellufs.landuseid = '1' and round(Parcelluverfs.wt_from, 'Year') = to_date('01-01-1999', 'dd-mm-yyyy') and parcellufs.parcelluid = parcelluverfs.parcelluid

Order by Parcellufs.parcelluid, Parcellufs.wt_from

<table>
<thead>
<tr>
<th>PARCEL</th>
<th>LAND-</th>
<th>PARCEL-</th>
<th>PARCELID</th>
<th>WT_FROM</th>
<th>WT_UNTIL</th>
<th>PARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>1</td>
<td>2</td>
<td>C-11/1</td>
<td>01-JUN-87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>1</td>
<td>3</td>
<td>C-12/1</td>
<td>01-JUN-87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>1</td>
<td>C-31</td>
<td>01-JUN-87 01-JUN-93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>3</td>
<td>C-33</td>
<td>01-JUN-87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>1</td>
<td>D-127</td>
<td>01-JUN-87 01-JUN-93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>1</td>
<td>1</td>
<td>F-74</td>
<td>01-JUN-87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>1</td>
<td>1</td>
<td>F-75</td>
<td>01-JUN-87</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

72 rows selected.

Queries 4 and 5 provide the state of residential in 1990, 1995 and 1999, while Query 6 and Figure 7.27 render the changes that occurred in 1998.

7.5.5.3 Temporal range query

In this type, the given time interval \( 1-T' \) may relate to \( 1-T \) in several ways. Allen (1983, 1984) compares these two intervals. Thirteen temporal relations are discussed in Chapter-2 and shown in Figure 2.30. Various spatio-temporal queries can be made based on these temporal relations. Here we consider four types of relations, i.e., when \( 1-T' \) is within \( 1-T \) (Figure 7.28):

- \( 1-T \) left cover \( 1-T' \) or \( 1-T \) started \( 1-T' \)
- \( 1-T \) covers \( 1-T' \)
- \( 1-T \) right cover \( 1-T' \) or \( 1-T \) finishes \( 1-T' \)
- \( 1-T \) equals \( 1-T' \)
Figure 7.24 Residential in 1990

Figure 7.25 Residential in 1995.
Figure 7.26 Residential in 1999.

Figure 7.27 Change to residential in 1998.
Various queries could be formulated based on these relations. For example, get all land uses from 1972 to 1980, get all land uses from 1975 to 1985, get all land uses from 1985 to 1987 or get all land uses during period 1972 to 1987.

**Query 7** Show residential land uses during 1990 and 1992.

```sql
Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid, parcelluverfs.wt_from, parcelluverfs.wt_until
From Parcellufs, parcelluverfs
Where Parcellufs.landuseid = '1' and to_date('1990', 'yyyy') >= parcelluverfs.wt_from and parcelluverfs.wt_from >= to_date('1987', 'yyyy') and (to_date('1992', 'yyyy') <= parcelluverfs.wt_until or parcelluverfs.wt_until is null ) and (trunc (parcelluverfs.wt_until, 'year') <= to_date('1994', 'yyyy') or parcelluverfs.wt_until is null ) and parcellufs.parcelluid = parcelluverfs.parcelluid
Order by Parcellufs.parcelluid, parcelluverfs.wt_from
```

Table 7.8 and Figure 7.29 show the outputs of this query.

**Query 8** Show residential land uses which were valid from 1994 till now.

```sql
Select Parcellufs.parcelluid, Parcellufs.landuseid, parcelluverfs.parcelluverid, parcelluverfs.parcelid, parcelluverfs.wt_from, parcelluverfs.wt_until
From Parcellufs, parcelluverfs
Where Parcellufs.landuseid = '1' and to_date('1993', 'yyyy') <= parcelluverfs.wt_from and to_date('1998', 'yyyy') >= parcelluverfs.wt_from and parcelluverfs.wt_until is null and parcellufs.parcelluid = parcelluverfs.parcelluid
Order by Parcellufs.parcelluid, Parcellufs.wt_from
```

The output is shown in Table 7.9 and Figure 7.30.
Table 7.8 Residential land use from 1990 to 1992.

Table 7.9 Residential land use from 1994 till now.

Figure 7.29 Residential land use from 1990 to 1992.
Similarly, parcels and their versions can be retrieved based on these temporal relations. For example, get all the parcels during the period 1970 to 1975. For both simple temporal and temporal range queries, a time stamp (WT or DBT) is mandatory.

7.5.5.4 Spatio-temporal topology

Langran (1992a) mentioned four types (as discussed earlier) of queries in TGIS. But queries in TGIS are not restricted to these types. Another example is retrieving the spatio-temporal topology of objects. These types of queries can be a combination of any of these four queries. Langran’s queries search space, time or space-time. Spatio-temporal queries search space topology, time topology and space-time topology. Space topology is common in any conventional GIS. Time topology is shown in scanning the history of objects (Queries 1 to 3). The spatio-temporal relations to capture the spatio-temporal topology of the objects for the CTSTDMin have been discussed in Chapter-4. Some of the relations are demonstrated here. It is shown how the spatio-temporal topology of the objects can be achieved based on this CTSTDMin. Two of the fundamental spatio-temporal relations are the boundary and co-boundary relations. These relations are discussed in Chapter-3 and Chapter-4. Often, planning agencies are interested in knowing the topology of the objects (parcels or land use). Some planning organizations / municipalities have to know the status of neighboring parcels prior to issuing building permits. For example, when issuing a building / environmental permit for parcel A (residential), apart from knowing the history of parcel A, a planner also needs to know the history of surrounding parcels, say parcels B and C. A planner needs to know whether B or C has ever been contaminated. Parcel B or C may be contaminated because of industry (perhaps the textile industry or a leather factory). Because of B and C, it would not be advisable to issue a residential permit to parcel A. For example, to check the neighboring parcels (spatio-temporal topology) of parcel F-29, a TCT structure is used in the following query. First, the boundary (OTCs) of parcel F-29 is retrieved; based on this boundary, the co-boundaries (TTCs) of OTCs are determined. The parcels associated with these TTCs are then loaded. The corresponding SQL statements (sub-query) are given below.

![Figure 7.30 Residential land use from 1994 till now.](image)
Query 9 What is the present land use of parcel F-29 and the land use history of neighboring parcels of F-29.

```
Select Distinct parcelluverfs.parcelid, parcelluverfs.landuseid, parcelluverfs.wt_from, parcelluverfs.wt_until
From  parcelluverfs, parcellufs
Where  parcellufs.parceluid = parcelluverfs.parceluid and parcellufs.parcelid In
(Select parcelversion.parcelid from parcelversion, parcelverttc
Where  parcelversion.parcelid = parcelverttc.parcelid and parcelverttc.twotcellid !=0 and
parcelverttc.twotcellid in
(Select tcelltuple.twotcellid from tcelltuple Where onetcellid In
(Select onetcellid From tcelltuple
Where  tcelltuple.twotcellid In
(Select parcelverttc.twotcellid from parcelverttc, parcelversion
Where  parcelversion.parcelid = parcelverttc.parcelid and parcelversion.parcelid In
(Select parcelluverfs.parcelid from parcelluverfs
Where parcelluverfs.parcelid = 'F-29')))))))
```

The output of the query is shown in Table 7.10. Parcel F-29 has two neighbors, i.e., parcels F-28 and F-30. Parcel F-28 was vacant (26) from 01-JAN-72 to 01-JUN-93 and residential (3) from 01-JUN-93. From 01-JAN-72 to 01-JUN-93, the land use of parcel F-30 was vacant (26), from 01-JAN-93 to 01-OCT-98 it was residential (3) and from 01-OCT-98 till now it has been used for health facilities (10). Parcel F-29 itself has undergone land use change. During the periods 01-JAN-72 to 01-JUN-93 and 01-JUN-93 to 01-OCT-98, it was vacant (26) and residential (3), respectively, while on 01-OCT-98 it changed to educational (10). Queries like this, based on spatio-temporal topology, are quite common in operational and strategic urban planning. This query demonstrates the use and applicability of the TCT structure for retrieving spatio-temporal information based on spatio-temporal topology.

### Bounded / unbounded intervals

The question of bounded or unbounded time-intervals is an important issue. Four types of interval can be defined in mathematics and they are discussed and shown in Figure 2.29. The issue of bounded and unbounded intervals is vital as it is a deciding factor in drawing the line between life and death of objects. For example, in Table 7.4, the life of ParcelLU objects 67 and 143 is from 01-JAN-93 to 01-JUN-93 and 01-JUN-93 to 01-OCT-98, respectively. The question is: do the objects 67 and 143 exist on 01-JAN-93?, i.e., is it a bounded interval? The answer is yes. For implementation, the time-intervals are considered as bounded; therefore the death-time of one object shares the birth-time of its children. According to Table 7.4, the time-interval is bounded; this can be verified with finer time granularity, i.e., seconds. The SQL statement of Query 3 can be reformulated as:

```
Select  Parcellufs.parceluid, Parcellufs.landuseid, parcelluverfs.parcelid,
        to_char(Parcellufs.wt_from, 'DD-MM-YYYY HH24:MI:SS') "WT_FROM",
        to_char(Parcellufs.wt_until, 'DD-MM-YYYY HH24:MI:SS') "WT_UNTIL", Parcellufs.parent
From  Parcellufs, parcelluverfs
Where  parcelluverfs.parcelid = "ST-0" and parcelluverfs.parceluid = Parcellufs.parceluid
Order by Parcellufs.parceluid, Parcellufs.wt_from
```

The result of this query is shown in Table 7.11. The WT_Until of object 76 is 01-06-1993 14:35:59, which is the WT_From of its child object 143. Similarly, this holds for objects 143 and 208.
Another example

Many more queries can be structured using SQL92. For example, to know how many parcels have changed or land use change per year may provide the evidence of the rapid growth of the city.

Query 10  How many parcels changed per year in Block 4?

```
Select  To_char(wt_from, 'YYYY') "WT_From", Count(parcelid) "No. of Parcels" From parcelversion
        Group by to_char(wt_from, 'YYYY')
```

The output of this query is shown in Table 7.12 and Figure 7.31. Initially there were 414 parcels in 1972. Note that the changes shown here are based on the sample data collected during the fieldwork. This does not reflect the actual changes during this period because the data were incomplete. The actual changes could be more than shown in this table. This table indicates both spatial and attribute changes in a parcel.

Similarly, the change or rate of change in land use for any given time can be calculated. The DBT can be used to know the status of any object at that time.

### 7.6 Performance Issues of CTSTDM

The performance of the model has not been checked, nor it was the objective of the study. The prototype implemented for about 500 parcels yields the following CTSTDM characteristics. Interesting results are obtained. Table 7.13 shows the number of ST-objects, cumulative numbers (in brackets) and the percentage of change. At base state (Jan-2000), the number of ZTC, OTC, TTC and TCT objects was the same, i.e., 1 (empty database). There were 363, 466 and 22 ZTCs at the times Feb-2000, Mar-2000 and Apr-2000, respectively. Similarily, the number of OTC, TTC and TCT objects are shown for Feb-2000, Mar-2000 and Apr-2000. The change in the number of ZTCs, OTCs and TTCs from Feb to Mar is almost the same, i.e., about 130%, but the increase in TCTs is 144%. From Mar to Apr, the change in ZTCs, OTCs, TTCs and TCTs is 2.6%, 5.6%, 14% and 12%, respectively (Figure 7.33 and Figure 7.34). The change in ZTCs, OTCs and TCTs is relatively smaller than TTCs, because the same ZTCs and OTCs participate in defining the new TTCs. The cumulative number of ZTCs, OTCs, TTCs and TCTs are shown in Figure 7.32.
Data structures have characteristic capabilities that make them acceptable or unacceptable for different applications (Langran, 1992a). The three characteristics for spatio-temporal data that dictate the acceptance or non-acceptance of the data structure are whether the data structure:

- supports range query
- defines a search space along any of its dimensions (space, time or both)
- treats dynamic data.

The first two characteristics (performance issues) are not tested and need more research efforts. However, the temporal cell complex (CTSTDM) is a dynamic data structure, i.e., it supports not only frequent create, kill, destroy and reincarnate operations, but also supports the changes of spatio-temporal topology of the objects. It can restructure itself based on these operations. According to Langan (1992a), its fluidity comes at the cost of deteriorating performance over time. Whether a TGIS needs a dynamic or static data structure depends upon application. The choice of a dynamic structure for urban applications is pragmatic. As mentioned earlier (Chapter-5), urban planning is a dynamic process. To describe this dynamic process, a dynamic data model (dynamic data structure) is needed. Particularly, it is important in the case of operational planning processes, where data are updated on a daily basis, e.g., parcel, building permit, land registration, cadastral, land use, traffic management data etc. Two urban applications (parcel and land use) are considered in this research, and their dynamic characteristics are discussed. It has been shown how these dynamic characteristics can be implemented in the CTSTDM.

<table>
<thead>
<tr>
<th>ST-objects</th>
<th>System time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZTC</td>
<td>1</td>
</tr>
<tr>
<td>OTC</td>
<td>1</td>
</tr>
<tr>
<td>TTC</td>
<td>1</td>
</tr>
<tr>
<td>TCT</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7.13 Number of ST-objects by system time.

![ST-objects by system time](image)

Figure 7.32 Number (cumulative) of ST-objects by system time.

Data structures have characteristic capabilities that make them acceptable or unacceptable for different applications (Langran, 1992a). The three characteristics for spatio-temporal data that dictate the acceptance or non-acceptance of the data structure are whether the data structure:

- supports range query
- defines a search space along any of its dimensions (space, time or both)
- treats dynamic data.

The first two characteristics (performance issues) are not tested and need more research efforts. However, the temporal cell complex (CTSTDM) is a dynamic data structure, i.e., it supports not only frequent create, kill, destroy and reincarnate operations, but also supports the changes of spatio-temporal topology of the objects. It can restructure itself based on these operations. According to Langan (1992a), its fluidity comes at the cost of deteriorating performance over time. Whether a TGIS needs a dynamic or static data structure depends upon application. The choice of a dynamic structure for urban applications is pragmatic. As mentioned earlier (Chapter-5), urban planning is a dynamic process. To describe this dynamic process, a dynamic data model (dynamic data structure) is needed. Particularly, it is important in the case of operational planning processes, where data are updated on a daily basis, e.g., parcel, building permit, land registration, cadastral, land use, traffic management data etc. Two urban applications (parcel and land use) are considered in this research, and their dynamic characteristics are discussed. It has been shown how these dynamic characteristics can be implemented in the CTSTDM.
7.7 DISCUSSION AND CONCLUSIONS

This chapter concludes with the implementation of cell tuple-based OO TGIS. Some of the stages of system design and implementation are covered in this chapter, i.e., system architecture, construction and implementation of the model. The OO approach has been adopted for the design and implementation of cell tuple-based OO TGIS. This system proposed for cell tuple-based OO TGIS is a three-tiered architecture that consists of three layers, i.e., OOGUI, GUIC and database. The first layer, i.e., OOGUI, is designed to create, update, delete and/or query spatio-temporal data. All communication between users and database is controlled by the second layer, i.e., GUIC. As discussed in Chapter-6, in spite of the powerful functionalities of SQL, computationally SQL is an incomplete language, which creates a communicate gap. At least two options emerged to fill this gap, i.e., embedded SQL or the ODBC approach can be used to talk to a relational database. Both provide a programmer to combine SQL statements with procedural programming language (C, C++, Java, COBOL, FORTRAN, ADA etc.) The first option is product- (programming language) and platform- (operating system) dependent while the latter is product-dependent but platform-independent. One of the major advantages of the latter option is the standardization, i.e., ODBC is based on a standardized version of SQL. This controller comprises two components, i.e., VC++ sources codes and API (ODBC). Fundamentally, the OOGUI and the GUIC constitute the application program (AP).

Data can be stored on a local or a remote machine. The third layer physically stored the data in relational format on the Oracle server (remote machine). This provides the client / server system architecture. An AP (layer OOGUI and GUIC) stays on the client site, while the database stays on the server site (Oracle server). Many users can access the spatio-temporal data from any terminal (client). The client / server architecture approach is used to implement the system, however current GISs do not fully exploit this architecture. The client / server architecture is desirable in an integrated urban information system, where parcel and land use objects are shared by many departments and agencies for varying purposes (Raza and Kainz, 2000b). This architecture may boost the performances of urban planning agencies by designing an integrated information system. This architecture can be extended to design a client / server and distributed database architecture. Two links are provided to communicate with spatio-temporal data stored on the server, i.e., direct and indirect links. The first link (weak and non-graphic) is via Net8 and the latter (strong and graphic) via AP (ODBC + VC++).

Basically, this is a hybrid (object-relational) architecture, where an OO skin is built on top of the RDMBS. The AP is an OO environment and the database is a relational one. The advantage of this approach is twofold, i.e., it fully utilizes the semantics and modular power of the OO paradigm and the power of the RDBMS. The advantage of the RDBMS is threefold. First, any commercial RDBMS can be utilized. Second, the powerful functionality of the RDBMS, such as unified data management, data recovery (roll back), data back up, consistency constraints, multi-user access, widespread use, extensive infrastructure of products and standard support etc. can be advantageous. Third, the power
of SQL can be exploited to retrieve the spatio-temporal data.

In the construction phase, it is shown how the CTSTDM can be constructed. Implementation of the CTSTDM is explained from three perspectives: structure, consistency rules and operators. The structure of the model at a logical level is realized in relational format (Oracle database management system). Not all consistency rules governing the CTSTDM can be implemented in Oracle. They are implemented in different layers. The database consistency rules have been implemented in Oracle (database layer) using DDL, while spatio-temporal consistency rules have been implemented in the GUIC layer, using VC++. The operators are implemented in the GUIC. Two fundamental Create and Kill operators and other associated operators have been implemented for consistent updating of spatio-temporal database. The Create operator associated with OTC and TTC objects is discussed in detail. Algorithms for Create OTC and TTC have been defined. For each object, two cases are discussed, i.e., for overlapping and non-overlapping objects. Discerning the geometric intersection of overlapping ST-objects (OTC and TTC) is an overlay process. Ten valid intersections between OTC-OTC and seven between TTC-TTC have been identified in Chapter-4. The existing algorithm for OTC-OTC (line-line) intersections fails where the slope of two OTC objects is the same. A new algorithm has been designed for overlapping OTCs. Researchers in computational geometry, computer graphics and GIS have developed a number of algorithms for polygon intersection, such as the plane-sweep, uniform grid or Delaunay triangulation etc. None of them treats the temporal element and old objects are thrown out. It was not possible to improve and implement any of these algorithms. Therefore, for TTC ARC/INFO overlay functions have been employed to discern the geometric intersection. A translator has been developed and implemented (in AML and VC++), which reads and writes spatio-temporal data to and from ARC/INFO and CTSTDM. However, this marriage slows down the system performance because at every update the existing objects in the CTSTDM are checked for possible overlap, using the MBR technique. For better performance, it is suggested these algorithm be modified (to accommodate temporal element) and implemented under one system (VC++), using a parallel processing technique (in the case of multi-processors machines).

One block of Karachi, Pakistan, is selected to demonstrate the applicability of the model. The time series parcel and land use of 1972 (assumed vacant), 1987, 1993 and 1998 has been used for prototyping. A parcel-level land use is indispensable for many operational and strategic planning exercises. A procedure for creating and / or updating OTC, TTC, parcel and detailed (parcel-level) land use objects has been discussed and implemented. The on-line topology building approach has been adapted. The approach for creating / updating is based on OO concepts. This approach can be extended to applications similar to these applications. The object-relational approach adopted here may help to alleviate the accessibility problem of land use data in an urban environment. Through SQL, it is demonstrated how the spatio-temporal queries can be made. Langran (1992a) mentioned four types of queries, which search space, time and / or space-time. In TGIS, we need time topology, space topology and space-time topology queries as well. Based on Langran’s queries, it has been shown how the simple temporal query and temporal range query can be formulated in SQL. In these types of queries an explicit time stamp is mandatory. Space topology is common in conventional GIS. Using time topology queries, it has been demonstrated how the object history can be scanned, where the explicit time may not be imperative. By employing the temporal cell tuple structure, space-time topology of the objects is retrieved. All these queries are based on SQL. It is shown that at any time not only can the state of an object be determined, but also the change can also be discerned. However, in the absence of temporal SQL and spatial operators, careful formulation of SQL statements is suggested. Further, OO extension of SQL (SQL3) may be useful for keeping the SQL shorter and simpler, yet powerful. As discussed earlier, spatio-temporal data grow very fast and require large storage space. A good index mechanism is indispensable for retrieving these bulky data. Most of the commercial RDBMS (e.g., Oracle or Sybase) employ the B-tree indexing technique. Oracle implicitly creates indexes on the columns of all unique and primary keys that are defined with integrity constraints. These indexes are the most selective and the most effective in optimizing performance (Oracle8, 1998). However, these indexes are not suitable for certain queries (sub-queries in SQL), such as spatio-temporal topology (Query 9). Langran (1992a) implemented some indexing techniques.
(R-tree) for spatio-temporal data, which need more investigation for the CTSTDM. Langran’s indexing is based on a space-time composite model. Oracle provide a user-defined index function this function may be useful for spatio-temporal indexing for the CTSTDM.

The format of spatio-temporal data stored in relational form does not conform to the format needed for graphic display. It is difficult in SQL to achieve the desired format. Therefore, some interface is needed to make the data directly available for graphic display. Although an interface is made in VC++, more work is needed to make this interface more interactive and user-friendly.

The performance of the model is not checked in scalability terms (space, time and different data set). However, a simple comparison is made to correlate the ST with the number of ZTCs, OTCs, TTCs and TCTs. Two time periods (Feb-Mar and Mar-Apr) are compared. In the first period, the percentage of change for ZTC, OTC and TTC (130%) was almost same but was higher for TCT (144%) than for the rest of the objects. In the second period, the percentage of change in ZTC, OTC, TTC and TCT was 2.6%, 5.6%, 14% and 12%. It shows that after a certain time the increase rate in ZTC, OTC and TCT is stable because the data are compact. To define new TTC(s), existing ZTC(s), OTC(s) and TCT(s) are used, but fewer numbers of ZTC, OTC and TCT are needed. In this way, the size of the database (temporal cell complex structure) is controlled. However, different data sets and time series data are needed to validate this claim.
Chapter-8

Conclusions and Recommendations

8.1 INTRODUCTION
This chapter summarizes the research carried out during the study. The major outcome of the research is briefly outlined in the summary. The chapter concludes the discussion on the research and provides recommendations for future work.

8.2 SUMMARY
According to many forecasts, rapid urban growth is inevitable. The world urban population currently totaling some 2.9 billion is expected reach 5.1 billion by the year 2025. About 70 to 80% will be living in the less developed countries. Urban planners have suggested two approaches to cope with this ever-increasing population pressure on urban areas: first, by population control measures, and, second by efficient urban management. The first approach has been found extremely difficult, especially in the less developed countries. Urban management is based on two planning processes, i.e., operational and strategic planning. The core data needed for urban planning is parcel and land use data. Parcel data is used for a variety of purposes by various planning agencies for operational planning, e.g., building permits, gas, power, water connections, legal purposes, land registration, taxation etc. Land use data is used for operational and strategic planning. This data is used for analyzing trends, monitoring changes and for prediction purposes in scenario building exercises. Where to get this data for urban planning and management is one issue, but how to manage this time series data is a particular challenge. This core data, if updated frequently, raises the question of structuring, storing, querying and display. Modeling, querying and displaying changes in parcel and land use data has been a challenging task for many designers and planners.

This thesis addresses the technical issues of data management in urban planning and computer science. Technologically, there is no commercial or operational TGIS to manage this data. Therefore, two issues have been taken up in the research, i.e., the management of parcel and land use data and the design and implementation of a TGIS. In the past, many issues and barriers have been identified in the design and implementation of a TGIS, e.g., the concept of time and change, spatio-temporal data modeling, system design, architecture, algorithms, the clustering of data, technological development, process- and event-oriented data spatio-temporal modeling, spatio-temporal query language, the role of event and evidence etc. Some of the issues are application-dependent, such as, the concept of change, while others are more fundamental and relevant to any generic TGIS.

One of the fundamental enigmas and impediments in designing a generic TGIS is the spatio-temporal data model. To design such a model has been a challenging task for many researchers because of the complex data structure. Few integrated approaches exist that treat spatio-temporal data in a unified fashion. The glaring limitations of the existing models have been discussed by many researchers. One of the problems in adopting these models is that they are based on existing systems that are closed and cannot be extended or modified. Spatio-temporal data modeling, and parcel and land use change are dynamic and complex processes. The OO concepts and mathematical theory are used to design the model. OO concepts are utilized at various stages of system development, i.e., OO system development (conceptualization, analysis, design and implementation), OO languages (modeling language UML and programming VC++) and the OOGUI. UML has been employed as a visual programming language mainly for the conceptual schema and VC++ as an implementation language. The following section provides chapter-wise, an overall summary of the research.
Various issues associated with the development of a TGIS are encompassed in this research. Starting from Chapter-1, which explained the motivation for carrying out the research, two problem areas have been identified, i.e., the technological issue and applications. The main focus of this thesis is on the former. On the technological front, there is a bewildering array of alternatives for the development of a TGIS.

The fundamental concepts of space and time are studied in Chapter-2. The definitions of space and time are given. Space is considered as a system of relationships. Various spaces (metric or topological) and associated spatial relations are discussed. Time is considered as a phenomenon that can be perceived by its effects. The relations (meet, before, after etc.) associated with time are defined in this chapter. The similarities and differences between space and time are highlighted. A general model of change is proposed, which facilitates the definition of the object and version of any object as a result of any change. This concept is used throughout the research for every object.

The data model consists of data structure, operations and consistency rules. A class is a set of objects with common properties. A class consists of data members and member functions (operations). The former are called static and the latter dynamic aspects of the class. The structure (schema) of the data model is defined by data members (data), while operations and consistency rules are defined by operations. SpatialClass, TemporalClass and AttributeClass represent the space, time and attribute components of a real world feature object, respectively. SpatioTemporalClass and AttributeTemporalClass are the aggregations of SpatialClass, TemporalClass and AttributeClass. The three classes, which are aggregated to generate SpatioTemporalAttributeClass provide a basis for modeling STAO, which is the backbone of any TGIS. SpatialClass has one PointClass (subclass). SpatioTemporalClass is a superclass of three classes, i.e., ZeroTCellClass, OneTCellClass and TwoTCellClass. The objects of ZeroTCellClass, OneTCellClass and TwoTCellClass are ZTC, OTC and TTC, respectively. The objects ZTC, OTC and TTC define their own spatial and temporal configurations. ZTC, OTC and TTC are members of the temporal cell complex (TCC). The definition of the SpatioTemporalClass object and TCC are given in Chapter-3. TemporalCellTupleClass is formed as an aggregation of ZeroTCellClass, OneTCellClass and TwoTCellClass, which preserves the temporal cell tuple (TCT) structure. A TCT structure encapsulates the spatio-temporal topology of each spatio-temporal object. Three dimensions of time are incorporated as three specialized classes of LinearTimeClass, i.e., DBT, WT and ST. ST reflects the time at which spatial changes occur in the system. ST is explicitly associated with the spatial object and is independent of DBT and WT. It is different from DBT in the sense that the latter represents the updating of STAO in the database, while the former indicates the updating of the spatial object. In LinearTimeClass, two data types are defined, 0-T and 1-T. The integration of these classes form the unified CTSTDM. The generic conceptual schema, and the temporal and spatio-temporal consistency rules governing TemporalClass and SpatioTemporalClass, respectively, are defined in Chapter-3.

The fundamental operators associated with ZTC, OTC, TTC and TCT are defined in Chapter-4. Two types of operators are identified: dynamic and static operators. Four dynamic operators, i.e., Create, Kill, Destroy and Reincarnate, are designed for the CTSTDM. These operators are associated with ZTC, OTC and TTC. Static operators are query operators, which are used to query spatial, temporal and spatio-temporal relations. As the spatio-temporal topology is encapsulated in TCT, these operators are associated with the TCT class. The dynamic operators change the status of the system, while the static operators do not affect the system’s state. The objects ZTC, OTC and TTC can be born, die or reincarnate. These objects are the same as long as they occupy the same space and retain the same geometry. The dead objects are called inactive objects and the alive objects are called active. The inactive do not participate in any dynamic operation except Reincarnate.

The need of major urban planning agencies / departments (nine) of Karachi for spatio-temporal data are documented in Chapter-5. Based on interviews during the fieldwork, a data flow matrix is constructed to assess their needs. Various users and owners of this data have been identified. The frequency of use and update is determined. This enables people to know who is sharing what type of
spatio-temporal data; sharing information is one of the chief problems in urban planning. The majority of agencies are utilizing land use data for planning and management purposes. Two types of land use change have been modeled, i.e., general and parcel-level land use. Various spatio-temporal characteristics of parcel and land use are elaborated. Each is treated as an object. The CTSTDM is extended to parcel and land use applications. These objects are an instance of SpatioTemporalAttributeClass. The change in parcel and land use objects causes change in the objects of either SpatioTemporalClass, AttributeTemporalClass or both classes, simultaneously. The object land use is the same as long as its land use is same; similarly, the object parcel is the same as long as its address is unchanged. The conceptual schema for general and parcel-level land use change is devised in Chapter-5.

The conceptual schema devised for generic and land use change is mapped to a logical schema. After reviewing the strengths and weaknesses of various approaches, an object-relational approach has been utilized. The conceptual schemas (generic and land use change) have been translated into relational form (logical schema). This translation was performed by an approach suggested by Blaha and Premerlani (1998). These relations were normalized by checking their functional dependencies. This mapping process is detailed in Chapter-6.

The next stage is the implementation and prototype system. Further, not only do the logical schemas elaborated earlier need to be implemented at the physical level, but also the system architecture and operators, so that the user can communicate with the database. A three-tier (three-layer) architecture is devised. The first layer reflects the OOGUI, the second the GUIC and the third the database. The link between each layer is established. Various forms have been designed for the OOGUI to facilitate updating, querying and visualizing ZTC, OTC, TTC, parcel and land use objects. The methods are the implementations of the operations and describe the algorithms associated with the operations. Some of the methods for creating OTC and TTC have been described in Chapter-7. Time series parcel and land use data from 1970 to 1998 was collected. This data was initially available in AutoCAD, and then translated to ARC/INFO and finally to CTSTDM format (in Oracle). The conceptual framework for updating the parcel and land use objects has been explained. It is shown when objects or versions are spawned and when they cease to exist. Many spatio-temporal queries are formulated using SQL, and the results are shown in terms of tables and graphics (maps).

8.3 CONCLUSIONS

Two issues emerging from the technological and urban planning side are dealt with in the research. The urban problem of managing time series parcel and land use data has been dealt with from the technological / computer science perspective. An attempt has been made to develop a unified spatio-temporal data model based on sound mathematics and OO concepts. The OO paradigm has been employed due to the fact that it deals with complexity in a more systematic manner. These concepts are useful, in terms of extensibility and flexibility, for the modeling of space, time and attributes. Extensibility means additional classes can be added, e.g., this model can be extended to include branching time or cyclic time. Flexibility means that object attribute and operations can be added or removed at any time. It has been demonstrated that a TGIS can be designed and implemented by utilizing these concepts from conceptual level to the physical level.

The main achievements of the research have been summarized in the previous section (8.2). Two major contributions are made: 1] the development of a unified CTSTDM and 2] the modeling of parcel and land use change. Both are core issues in geo-information / computer science and in urban planning (operational and strategic). The work paves the way for further development of a generic TGIS and enables urban planning agencies to monitor parcel and land use changes and to implement an integrated urban information system.

On the basis of various issues addressed in this research, the following conclusions can be drawn:

- A novel approach is presented to model space, time and attributes in a modular and systematic
manner. The OO design provides a cleaner data model, where each component is defined on its own and integrated in a more structured and flexible mode. The mathematical concepts used provide a sound basis for defining the space; this provides an unambiguous definition of the data model for implementation and a basis for further development of a spatio-temporal query language.

- A simple data structure is provided to address the complexity of spatio-temporal data. However, the processes are still complex. The concept of ST-simplicial and cell tuple structure has been extended to form a cell tuple-based spatio-temporal data model (CTSTM). The ST-simplicial complex is extended to temporal cell complexes and the cell tuple structure to temporal cell tuples for storing spatio-temporal topology. The model is dimension-independent. Therefore, the approach can be extended to higher spatial dimensional objects. All the singularities (cyclic, acyclic and interior) are incorporated in the CTSTDM. However, they have not been implemented.

- In the CTSTDM, objects and topology are stored separately. The spatio-temporal topology of objects is always stored in an implicit fashion. If the topology of an object alters over time, then this is recorded in TCT rather than altering or creating new objects every time. This has a great advantage over the explicit topology approach, where topology is stored with spatial objects (e.g., in ARC/INFO an arc carries a left / right polygon or a start /end node).

- Temporal, spatial and spatio-temporal consistencies are defined and implemented at various levels of implementation.

- Each class has two types of operators: dynamic and static. Four fundamental dynamic operators for the ZTC, OTC and TTC classes are designed, i.e., Create, Kill, Destroy and Reincarnate. The Create and Kill operators for the OTC and TTC classes have been implemented. Since the spatio-temporal data model devised in this research is a unified CTSTDM, the associated operations are complex. Any Create operation has to take care of various possibilities, as an object can intersect with an object of similar, higher or lower dimensional spatial objects. There are eight possibilities where an $n$-cell can intersect with an $n$-cell. In each case, there are various intersections / scenarios. Based on the point-set topology approach, these scenarios have been analyzed. Three valid intersections between ZTC-OTC, 10 between OTC-OTC, eight between OTC-TTC and seven between TTC-TTC have been realized. Static or query operators associated with the TCT class have been identified. The result of any fundamental operation produces an $n$-TCT and guarantees spatio-temporal consistency in the database.

- The CTSTDM is a dynamic data structure; it self-adjusts automatically whenever a dynamic operator is commanded. In this structure, retroactive changes can be committed.

- Five types of spatio-temporal relations are identified. One type, i.e., spatio-temporal topological relations, has been discussed in detail. It has been formulated and demonstrated that, in the spatial domain almost all spatio-temporal relations between OTC-OTC and TTC-TTC (interior, boundary, co-boundary, closure, disjoint, contains, inside, equal, meet, covers, coveredBy, overlap, start and end) as proposed by Egenhofer et al. (1993) and Pullar and Egenhofer (1988), and some other (left and right) relations, can be derived from the TCT structure, which is based on the boundaries and co-boundaries of cells / objects. The clockwise relations have to be determined by a geometric technique. The fundamental relations, i.e., direction (start / end) and orientation (left / right), of OTC have been incorporated in TCT, which were missing in the Pigot (1995) and Brisson (1990) models. It is demonstrated that all singularities can be addressed in the TCT structure. However, the procedure for adding these singularities to the CTSTDM is neither discussed nor implemented.

- At the logical level, the relational schema has been fully normalized to remove the update anomalies. However, normalization is not the only solution in overall system functionalities and performance. In many queries, a join operation has to be performed to de-normalize the relations. The join operation is one of the time-consuming operations. Every effort is needed to improve system performance. Eight concrete classes of the generic conceptual schema resulted in 13 relations (tables). Similarly, 10 concrete classes of parcel-level land use change mapped to 16 relations.

- All spatial (including topological) data is stored in relational form. None of the commercial
systems (GIS) stores all spatial data in a relational form.

- The format of spatio-temporal data stored in relational form does not conform to the format required for graphical display. It is difficult to achieve the desired format in SQL. Therefore, an engine / interface has been built to make this data available for graphical display. However, it needs further improvement.

- The three-tier system architecture is implemented for OO TGIS. The system espouses a client / server architecture and has several advantages, i.e., data can be located on local or remote machines, many users can access data at one time, and different users (planning departments) can view or update data, based on rights assigned by a database administrator. This architecture is suitable for designing integrated urban information systems, because parcel and land use data are the fundamental requirement for many urban applications. This approach may help to alleviate the accessibility problem of parcel and land use data in the urban environment.

- The object-relational approach adopted is a hybrid architecture where an OO skin is built on top of an RDBMS. It has two advantages, i.e., it fully utilizes the power and semantics of the OO paradigm and the functionalities of the RDBMS. The advantage of the RDBMS is threefold. First, any commercial RDBMS can be utilized. Second, the power and functionalities (unified data management, data recovery or roll back, data backup, consistency constraints, multi-user access, widespread use, extensive infrastructure of product and standard support etc.) of the RDBMS are advantageous. Third, the power of SQL can be exploited (it can be extended to SQL3) to retrieve spatio-temporal data. However, different commercial RDBMSs provide varying levels of functionalities, e.g., MS Access does not support the roll back option. Theoretical work on temporal operators is not fully implemented in commercial databases such as Oracle. Moreover, SQL is a computationally incomplete language, e.g., there is a dearth of recursive operators. Certain consistency checks cannot be performed in the RDBMS (Oracle); they are implemented in C++. SQL3, which supports some OO concepts and temporal operators, is not commercially available yet. SQL implemented in commercial RDBMSs does not include spatial and temporal operators. Spatial operators available in commercial databases, e.g., Oracle, are dedicated to their spatial data model (Oracle Spatial Cartridge or Informix 2D / 3D Spatial DataBlade), which is basically a non-topological model. These operators cannot be utilized for the CTSTDM. ODBC has certain limitations, e.g., it does not allow dynamic SQL and is limited to simple select, from and where statements. Other mathematical functions that are part of SQL, such as maximum or minimum, are not supported via ODBC. These functions are fundamental for assigning a new object ID whenever an update operation is performed. In the current implementation of the CTSTDM, these operations are performed in an application program (AP).

- The Create and Kill operators for OTC and TTC objects have been implemented. Create is basically an overlay operation, with geometric intersection as the backbone. Existing algorithms for OTC-OTC and TTC-TTC intersections are not sufficient for the CTSTDM. For OTC-OTC geometric intersection, a new algorithm has been designed and implemented. However, its performance has not been evaluated. For TTC-TTC geometric intersection, the ARC/INFO overlay command has been employed. However, this marriage slows down system performance because at every update the existing objects (TTC) in the CTSTDM are checked for possible overlap, using the MBR technique. For better performance, it is suggested that these algorithms be modified and implemented under one system (C++).

- It has been speculated that the cell tuple model will increase the database size very rapidly. Moreover, extending the cell tuple model to the spatio-temporal domain will adversely affect the system performance because of database size, as spatio-temporal data is already bulky. The prototype implemented for about 500 parcels shows that after a certain time the increase in the number of ZTC, OTC and TCT becomes stable because the data is compact. To define new TTC(s), existing ZTC(s) and OTC(s) are used, but fewer numbers of ZTC and OTC are used. Therefore, only a few TCT(s) are spawned. The results achieved here need further validation.

- OO programming language (e.g., C++) is a powerful language in terms of modularity, aggregation, inheritance and encapsulation. If these concepts are utilized and combined properly, then a system can provide excellent performance. A careful implementation strategy is required.
to optimize these concepts, otherwise the system may react very slowly or even collapse as a result of insufficient memory. When two or more classes / objects are aggregated, they are aggregated along with their attributes and operations. For example, take two objects, A and B, with the usual attributes and operations. Suppose B is a larger object than A. If B is encapsulated in A to hire an operation (single) from B, then it may occupy a large chunk of memory.

- Spatio-temporal characteristics of parcel and land use change (general and parcel-based) have been analyzed and modeled using OO concepts. It has been shown that changes modeled in this way can provide better management of spatio-temporal data. Data can be shared as it is available in standard relational format. The data structure supports querying on the history of the objects, the state of an object at any given time and the change in an object at any given time. Thus, it provides a monitoring mechanism. Although all these are possible using SQL, a careful formulation of SQL statements is suggested. The TGIS is not yet for amateurs.

- The model provides a sufficient basis to further incorporate temporality in various applications, particularly in the applications that involve moderate change processes such as cadastral systems or strategic and operational urban planning.

**8.4 RECOMMENDATIONS**

This research does not address all aspects relating to TGIS and the issues of urban planning and management. This thesis just touches the tip of the iceberg. Yet the research does provide a basis for further TGIS development. The main areas for future work directly relevant to the research are as follows:

- Further investigation on data indexing techniques to improve the data access time, such as R-tree.
- Most of the work on temporal query languages is based on relational algebra; this work directly matches the relational model. This work on temporal SQL should be incorporated. The temporal operators (Oracle8 Time Series Cartridge) provided by Oracle have not been tested. Further research efforts are needed in this respect. Temporal operators and other OO concepts supported in SQL3 need to be tested.
- The research does not address the spatio-temporal visualization issues. How to visualize space-time is another active area of research in cartography. This is still an open question. However, the data structure provides a sufficient basis for visualizing space-time, such as the space-time cube.
- Secondary operators such as copy, move, join etc., based on fundamental operators, need further development.
- The performance of the model should be checked with different data sets and more time series data. It would provide better evaluation if this model could be compared with other spatio-temporal models in terms of database size and data accessing time.
- A relational approach has been utilized to store the data on the RDBMS. In future, the OODBMS may provide better performance, since the conceptual model devised is OO. This could facilitate automatic object and version ID generation. Normally, the RDBMS does not provide this option. The OODBMS may provide better performance and the normalization process could be avoided.
- Further implementation of Destroy and Reincarnate operators should be undertaken.
- Applications where singularities are vital, such as roads between crop fields or dead end streets, need further implementation and testing of the model.
- The CTSTDM has been tested for parcel and land use applications where ST-objects are TTC. More applications based on linear and point features need further investigation and testing.
- The research is restricted to \( n \)-dimensional spatial objects \( (0 \leq n \leq 2) \). The next step is to extend this approach to spatial dimension \( (0 \leq n \leq 3) \).
- Modeling branching time needs further research efforts. A branching time model is indispensable in many scenario building exercises.
- One of the advantages of time series data is the forecasting capability. The CTSTDM can be exploited for forecasting purposes. Forecasting techniques need more investigation.
• The development of enhanced engines for graphical display to visualize the changes in 3D. The VRML (virtual reality modeling language) could be one option. Although it supports 3D modeling, it does not support the time component. Further improvement of the GUI and development of animated maps is recommended.
• Further investigation into the development of interactive visual editing of objects. This might not be a simple task as a single cell (n-tcell) is associated with many feature objects at varying times.
• Formulation of fundamental operators.
• Another interesting area to explore is the temporal buffering concept, similar to spatial buffering.
Bibliography

Batty, M., 1992, Geographic information systems: GIS in urban planning and policy analysis, GIS in development planning, In the proceedings of the study group meeting on regional and global integration, Development Planning and Information Technology, Nagoya, ISBN 4-906236-08-1, pp. 27-60.
Bird, C., 1991, Land use monitoring, Year Book of Association for Geo-Information.
Cheng, T. and Molenaar, M., 1998, A process-oriented spatio-temporal data model to support physical environmental modeling, In the Proceedings of the 8th International Symposium on Spatial Data Handling (SDH), Edited by T.K. Poiker and N. Chrisman, Simon Fraser University, Canada,

Chrisman, N., Dougenik, J. and White, D, 1992, Lessons for the design of polygon overlay processing from the ODYSSEY whirlpool algorithm, In the proceedings of the 5th International Symposium on Spatial Data Handling (SDH), pp. 401-410, Charleston, South Carolina, USA.

Claramunt, C. and Thériault, M., 1995, Managing time in GIS an event-oriented approach, In the proceedings of the International Workshop on Temporal Databases, Recent advances in temporal databases, Zurich, pp. 23-42.


Dangermond, J., 1984, A classification of software components commonly used in geographic information systems, In the proceedings of the Computer-Based Geographic Information Systems, Amherst, IGU, Commission on Geographical Data Sharing and Processing, pp. 70-91.


Dueker, K.J. and Talcott, R., 1973, Statewide land use analysis and information requirements, Working group paper 13, Institute of Urban and Regional Research, University of Iowa.


Egenhofer, J.M., Frank, A. and Jackson, J., 1989, A topological model for spatial databases, In design and implementation of large spatial databases, Edited by A. Buchman, O. Gunther, T.R. Smith, Y.F. Wang, In the proceedings of First Symposium SSD'89, Santa Barbara, California, Springer-Verlag.
Frank, A., 1987, Overlay processing in spatial information systems, In the proceedings of the AutoCarto 8, American Society for Photogrammetry and Remote Sensing and American Congress on Surveying and Mapping, Baltimore, Maryland, USA, pp. 16-31.
Franklin, W.M., 1984, Cartographic errors symptomatic of underlying algebra problems, In the proceedings of the International Symposium on Spatial Data Handling (SDH), Edited by D. Marble, et al., Zurich, Switzerland.
Freeman, E. and Sallons, W., 1971, Basic issues in the philosophy of time, LaSalle, Illinois, Open Count.
Hanrahan, P.M., 1985, Topological shape models, PhD thesis, University of Wisconsin-Madison, USA.
Harvey, F., 1994, Defining unmoveable nodes/segments as part of vector overlay: the alignment overlay, In the proceedings of the 6th International Symposium on Spatial Data Handling, SDH' 94, Talyor & Francis, vol. 1, pp. 159-176.
Hazelton, N., 1992, Beyond 2-D map: a new metaphor for multi-temporal 4-D GIS, In the proceedings of the LIS/GIS' 92, pp. 303-311.
Hazelton, N, Leahy, F. and Williamson, I., 1990a, Incorporating dynamic modeling into geographical information systems, In the proceedings of the AURISA, pp. 77-84.
Hazelton, N. J., Leahy, F. and Williamson, I., 1990b, On the design of temporally-referenced, 3-D
geographical information system: development of four-dimensional GIS, In the proceedings of GIS/LIS, pp. 356-373.
Hermosilla, L., 1994, Unified approach for developing a temporal GIS with database and reasoning capabilities, In the proceedings of the Fifth European Conference and Exhibition on Geographical Information Systems, vol. 1, pp. 123-131
Herring, R.J., 1987, TIGRIS: Topologically integrated geographic information system, In the proceedings of the Auto Carto 8 Conference, Baltimore, Maryland, USA, pp. 282-291.
Hopkins, S., Healey, R. and Waugh, T., 1992, Algorithm scalability for line intersection detection in parallel polygon overlay, In the proceedings of the 5th international symposium on spatial data handling (SDH), pp. 211-218, Charleston, South Carolina, USA.
Kainz, W., 1990, Spatial relationships - topology versus order, In the proceedings of the 4th International Symposium on Spatial Data Handling (SDH), vol. 2, pp. 814-819.
Kirschenhofer, P., 1995, The mathematical foundation of graphs and topology for GIS, Geographic information systems - materials for a post-graduate course, Edited by Frank U. Andrew, Department of Geoinformation, Technical University Vienna.
Langran, G., 1993a, Manipulation and analysis of temporal geographic information, In the proceedings of the Canadian conference on GIS'93, Canada, pp. 869-879.
Langran, G., 1992a, Time in geographic information systems, Taylor and Francis Ltd.
Laurini, R., 1994, Sharing geographic information in distributed databases, In the proceedings of the Urban and Regional Information Systems Associations (URISA), pp. 441-454.
Miller, G.T., Jr., 1988, Living in the environment, Sixth edition, Wadsworth, Belmont, California.
Molenaar, M., 1995, Spatial concepts as implemented in GIS, Geographic Information Systems -
Bibliography

MSDN, 1999, C:\Program%20Files\Microsoft%20Visual%20Studio\MSDN98\98VSa\1033\glossary.chm::/_gloss_glossary_a.htm.
Nunes, J., 1995, General concepts of space and time, Geographic information systems - Materials for a post-graduate course, Edited by Frank U. Andrew, Department of Geoinformation, Technical University Vienna.
Oracle8, 1998, Oracle8 documentation, Oracle Corporation, USA.
Peuquet, D. and Qian, L., 1996, An integrated database design for temporal GIS, In the proceedings of the 7th International symposium on spatial data handling (SDH’7), Advances in GIS research II, Edited by M.J. Kraak and M. Molenaar, vol. 1, pp. 2.1-2.11.
Pigot, S., 1995, A topological model for a 3-dimensional spatial information system, PhD thesis, University of Tasmania.
Ramachandran, B., MacLeod, F. and Dowers, S., 1994, Modeling temporal changes in a GIS using an
object-oriented approach, in the proceedings of 6th International Symposium on Spatial Data Handling (SDH), pp. 518-537.
Raza, A. and Kainz, W., 2000c, Spatio-temporal data modeling for urban land use changes, IJGIS, Taylor and Francis (under review).
Rhind, D. and Hudson, R., 1980, Land use, Methuen & Co., USA.
Ross, J., 1985, Detecting land use change on Omaha’s urban fringe using a geographic information system, in the proceedings of the Auto-Carto 7, Falls Church, VA, ACSM, pp. 463-471.
Tsang, E.P.K, 1983, Time structure for AI, In the proceedings of the 10th IJCAI.
Urban, S. and Delcambre, 1986, An analysis of the structural, dynamic and temporal aspects of semantic data models, in the proceedings of the International Conference on Data Engineering,
Bibliography

IEEE, pp. 382-389.
v an Heldon, P., 1994, An integrated information system for urban land use management, In the proceedings of the Urban and Regional Information Systems Associations (URISA).
van Roessel, J., 1986, Design of a spatial data structure using the relational normal forms, In the proceedings of the SDH, Seattle, USA, pp. 251-272.
v an Oosterom, P., 1993, Reactive data structure for geographic information systems, Oxford University, Bookcraft Ltd.
Worboys, M., 1992a, Object-oriented models of spatio-temporal information, In the proceedings of the GIS/LIS, ASPRS/ACSM, pp. 825-834.
Worboys, M., 1992a, Object-oriented models of spatiotemporal information, In the proceedings of the GIS/LIS, ASPRS/ACSM, pp. 825-834.
Worboys, M., 1992c, A generic model for planar geographical objects, IJGIS, vol. 6, no. 5, pp. 353-372.
Worboys, M., 1994a, Unifying the space and temporal components of geographical information, In the proceedings of the 6th International Symposium on Spatial Data Handling (SDH’ 6), vol. 1, pp. 505-517.
Worboys, M., 1994b, Object-oriented approaches to geo-referenced information, IJGIS, vol. 8, no. 4, pp. 385-399.
Yuan, M., 1994, Wildlife conceptual modeling for building GIS space-time models, In the proceedings of the GIS/LIS, Phoenix, USA, pp. 860-869.
Urban planning and service agencies in Karachi

1. Municipal bodies
   1.1. Karachi Metropolitan Authority (KMC)
   1.2. Karachi Zonal Municipal Committees
   1.3. Karachi District Council and Union Councils

2. Special purpose bodies (Provincial Statutes)
   2.1. Karachi Development Authority (KDA)
   2.2. Sindh Industrial Trading Estate (SITE)
   2.3. Karachi Water and Sewerage Board (KWSB)
   2.4. Sindh Katchi Abadi Authority (SKAA)
   2.5. Sindh Highway Department
   2.6. Cooperative Housing Societies.

3. Special purpose bodies (Federal Statutes)
   3.1. Karachi Port Trust
   3.2. Port Qasim Authority
   3.3. Pakistan Steel
   3.4. Pakistan Public Works Department
   3.5. Military Engineering Services
   3.6. Civil Aviation Authority
   3.7. Defence Housing Authority

4. Cantonment Boards (Federal Statutes)
   4.1. Karachi Cantonment Board
   4.2. Drigh Cantonment Board
   4.3. Malir Cantonment Board
   4.4. Manora Cantonment Board
   4.5. Clifton Cantonment Board
   4.6. Defence Cantonment Board

In addition, the following agencies are responsible for respective services to the residents of Karachi.

- Pak Telecommunication Department (PTC)
- Karachi Electric Supply Corporation (KESC)
- Sui Southern Gas Company (SSGC)
Spatio-temporal data in planning agencies

Three major departments of Karachi Development Authority (KDA) have been selected to document the need for spatio-temporal data in these departments.

1. **Master Plan and Environmental Control Department (MPECD), KDA**
   The MPECD uses maps from various sources for a variety of urban planning and management functions. Some are internally produced, while the rest are acquired from other sources. Basically, four types of map, namely topographic maps, infrastructure maps, land use maps and layout plans, are used by MPECD. Topographic and infrastructure maps are procured from the SOP and other utility agencies; land use maps are generated internally; and layout plans are acquired from the DPUD, land developers, cooperative-housing societies, or generated internally (for a few schemes).

   Topographic maps of scales 1:50,000 to 1:250,000 are used for regional planning and maps of scales 1:1,000 to 1:50,000 are utilized for the urban area. The frequency of use is occasional. Infrastructure maps are occasionally used for planning purposes, mainly for the preparation of development plans. Layout plans of 1:1,000 to 1:4,000 are frequently used for routine town planning tasks and periodically (five years) for registering land use survey information to produce land use maps. These land use maps/information are used for the preparation of development plans. Almost all maps are in non-digital format and are updated manually. A few 1:25,000 scale topographic maps and layout plans are in digital (AutoCad) format.

2. **Directorate of Planning and Urban Development (DPUD), KDA**
   The DPUD is one of the major owners (apart from the cantonment board) of layout plans of various schemes in Karachi. The DPUD acquire maps from the SOP, the board of revenue (Government of Sindh) and private societies to generate the layout plan at the time of designing and planning a new housing scheme. The scale of these maps varies from 1:1,000 to 1:4,000, and they are used as and when required or when a new scheme is designed. All layout plans are internally produced when new housing schemes are planned and designed. The scale of layout plans varies from 1:1,000 to 1:4,000. The layout plans are used for routine urban and town planning exercises and are therefore used daily. Although transactions are daily, requiring daily updates, these maps are not updated daily, but only occasionally. At every transaction (parcel subdivision, amalgamation or extra land), a part plan is created without updating the original layout plan, which leads to the duplication of part plans (parcel) in some cases. The original layout plan is updated at non-specific intervals/frequency.

   The DPUD requires maps from different sources to plan its schemes, for example, geomorphological, soil or topographic maps. The lack of these maps sometimes creates problems in terms of land allocation for specific purposes.

3. **Traffic Engineering Bureau (TEB), KDA**
   The TEB has the task of managing the problems related to rapidly increasing traffic in Karachi. Various base maps are used, mostly from the SOP having a scale of 1:4,000 to 1:50,000. Other maps (drawing) of scale 1:500 are generated within the department. Almost all these maps are used daily for a variety of tasks, i.e., planning, management, maintenance or operation of the traffic system in Karachi. The TEB desired the base maps to be updated once in five years and traffic maps (drawings) to be updated once a year. The TEB does not require maps from different sources to be pieced together.

4. **Karachi Water and Sewerage Board (KWSB)**
   The KWSB is responsible for the provision and maintenance of water supply, sewerage and drainage in the Karachi urban area. This urban area is divided into 14 divisions, and these divisions are
supposed to keep the maps or drawings of water and sewerage system. These drawings/maps show the network and valve locations. None of these drawings/maps is in digital format. They are used as and when required for maintenance, operation and planning purposes. However, most of the work is done manually and there is no organized information. Staff know by heart what services are laid down where.

All maps are internally produced and there is no scientific or systematic approach to update these maps. Needs for spatial accuracy has been shown for laying the pipe network, but no accuracy was mentioned. The spatial scale varies from 1:1,000 to 1:25,000, while the frequency of using these maps is non-specific.

5 Sui Southern Gas Company (SSGC)
The SSGC deals with gas purification, transmission and distribution. For distribution, planning, management and operation, the company acquires maps/layout plans from the KDA, the KMC, the SKAA and cantonment boards. These maps/layout plans are used to indicate plots, valve locations, pipeline details (size and dia), subsurface depth, distance from property etc. No map is internally produced. All maps are in non-digital format and are updated manually on paper maps/layout plans. The spatial scale of these maps varies from 1:300 ft to 1:50 ft. They are used and updated daily for planning, management, operation or maintenance purposes.

6 Pak Telecommunication Department (PTC)
The PTC is responsible for providing the telephone and telecommunication facilities in Karachi. It uses maps for planning, executing and maintaining their services. Topographic maps and guide maps from the SOP are used for laying the primary network. No maps are internally produced. Layout plans from the KDA, the KMC and cantonment boards and base and guide maps from the SOP are being acquired. These layout plans and maps give information regarding cable/duct routes and cable/duct size of primary and secondary cables. The scale of these maps and layout plans varies from 1:500 to 1:25,000. Almost all maps and layout plans are in non-digital format; however, a few drawings of fiber glass cable exchanges are in digital format but not geo-referenced. Maps are updated as and when required. No particular frequency for updating these maps was reported.

7 Karachi Electric Supply Corporation (KESC)
The KESC generates and distributes electric power to industrial and domestic users of Karachi. The base maps, layout plans and drawings (routes of high/low tension lines) are used for planning, operation, maintenance and system planning. The base maps are acquired from the SOP at scales 1:50,000 and 1:100,000; and layout plans from the KDA and the KMC at scales 1:1,000 to 1:5,000. The drawings showing the power lines are internally produced and do not have any specific scale. For planning purposes, the maps are used once in five years, while use for operational purposes is daily. The power maps (drawings) are updated once in five years or as and when required.

8 Survey of Pakistan (SOP)
The SOP is a national mapping agency with a number of regional offices distributed at urban centers. One of the regional offices is in Karachi. One of the tasks of this regional office is to maintain the 1:25,000 topographic map and large-scale (e.g., 1:4,000) maps. These maps are not updated regularly. The latest available 1:25,000 and 1:4,000 series maps date back to the 1970s. Some of the 1:4,000 scale maps were updated in the early 1990s, covering only the coastal areas of the Karachi region.

9 Sindh Katchi Abadi Authority (SKAA)
The SKAA basically uses two types of maps, i.e., layout plans and topographic maps. Both maps are generated internally and are in non-digital format. The spatial scale of the map varies from 1:30 ft to 1:40 ft. These maps are used daily, while layout plans are updated once in two/three months. These maps are used for planning, management and execution purposes.
Appendix

Definition of relations (tables) in Oracle using DDL

Create Table Point (PointID Number(9) Constraint ck_point Check(PointID >=0), X Float Constraint nn_x Not Null, Y Float Constraint nn_y Not Null, Constraint pk_point Primary Key (PointID), Unique (X,Y))

Create Table ZeroTCell (ZeroTCellID Number(9) Constraints ck_ztc Check (ZeroTCellID >= 0), PointID number references Point, ST_From Date Default SysDate, ST_Until Date, Constraints pk_ztc Primary Key (ZeroTCellID), Constraints unq_pt UNIQUE (PointID), Check (ST_Until >= ST_From))

Create Table OneTCell (OneTCellID Number(9) Constraints ck_otc Check (OneTCellID >= 0), Length Float, Parent Number(9), ST_From Date Default SysDate, ST_Until Date, Constraints pk_otc Primary Key (OneTCellID), Constraints fk_otc Foreign Key (Parent) references OneTCell(OneTCellID), Check (ST_Until >= ST_From))

Create Table OnePointTCell (OneTCellID Number(9), PointID Number(9), PointSeq Number(9), Constraints fk_point Foreign Key (PointID) references Point(PointID), Constraints fk_otc_pt Foreign Key (OneTCellID) references OneTCell(OneTCellID), Constraints pk_otc_pt_seq Primary Key (OneTCellID, PointID, PointSeq), Constraints unq_otc_pt_seq Unique (OneTCellID, PointID, PointSeq))

Create Table TwoTCell (TwoTCellID Number(9) Constraints ck_ttc Check (TwoTCellID >= 0), Area Float, Perimeter Float Not Null, Parent Number(9) Not Null, ST_From Date Default SysDate, ST_Until Date, X_Centroid Float, Y_Centroid Float, Constraints pk_ttc Primary Key (TwoTCellID), Constraints fk_ttc Foreign Key (Parent) references TwoTCell(TwoTCellID), Check (ST_Until >= ST_From))

Create Table TwoOneTCell (TwoTCellID Number(9) references TwoTCell, OneTcellID number(9) references OneTCell, Constraints pk_ttc_otc Primary key (TwoTCellID, OneTCellID), Constraints fk_ttc_ttc Foreign Key (TwoTCellID) References TwoTCell(TwoTCellID), Constraints fk_ttc_otc Foreign Key (OneTCellID) References OneTCell(OneTCellID))

Create Table TCellTuple (TCellTupleID Number(9) Constraints ck_tct Check (TCellTupleID >= 0), TwoTCellID Number(9), OneTcellID Number(9), ZeroTCellID Number(9), ST_From Date Default Sysdate, ST_Until Date, Constraints pk_ztc_tct Primary Key (ZeroTCellID), Constraints fk_ztc_tct Foreign Key (ZeroTCellID) references ZeroTCell(ZeroTCellID), Constraints fk_otc_tct Foreign Key (OneTCellID) References OneTCell(OneTCellID), Constraints fk_ttc_tct Foreign Key (TwoTCellID) References TwoTCell(TwoTCellID), Constraints unq_ztc_otc_ttc_tct Unique (ZeroTCellID, OneTCellID, TwoTCellID), Check (ST_Until >= ST_From))
Create Table ParcelVersion ( 
    ParcelID Char(15) Constraints ck_parcel Check (ParcelID = Upper(ParcelID)), 
    VersionID Number(9) Constraints ck_version Check (VersionID > 0), 
    BlockNo Number(9) Constraints ck_block Check (BlockNo > 0), 
    SchemeNo Number(9) Constraints ck_scheme Check (SchemeNo > 0), 
    ParcelParent Char(15), 
    VersionParent Number(9), 
    MPD Number(9) Constraints ck_mpd Check (MPD > 0), 
    ANZ Number(9) Constraints ck_anz Check (ANZ > 0), 
    DigitizedBy Char(30), 
    CheckedBy Char(30), 
    WT_From Date, 
    WT_Until Date, 
    DBT_From Date Default Sysdate, 
    DBT_Until Date, 
    Area float, 
    Constraints pk_parver Primary Key (ParcelID, VersionID), 
    Constraints fk_parcelversion Foreign Key (ParcelParent, VersionParent) references ParcelVersion(ParcelID, VersionID), 
    Constraints Check (WT_Until >= WT_From), 
    Constraints Check (DBT_Until >= DBT_From) 
)

Create Table ParcelVerTTC ( 
    ParcelID Char(15), 
    VersionID Number(9), 
    TwoTCellID Number(9), 
    Constraints fk_parverttc_parver Foreign Key (ParcelID, VersionID) references ParcelVersion(ParcelID, VersionID), 
    Constraints fk_parverttc_ttc Foreign Key (TwoTCellID) references TwoTCell(TwoTCellID), 
    Constraints pk_par_ver_ttc Primary Key (ParcelID, VersionID, TwoTCellID), 
    Constraints unq_par_ver_ttc Unique (ParcelID, VersionID, TwoTCellID) 
)

Create Table ParcelLU ( 
    ParcelLUID Number(9) Constraints ck_parlu Check (ParcelLUID > 0), 
    LanduseID char(10), 
    WT_From Date constraints nn_WT_From Not Null, 
    WT_Until Date, 
    DBT_From Date Default Sysdate, 
    DBT_Until Date, 
    Parent Number(9), 
    Constraints pk_par_lu Primary Key (ParcelLUID), 
    Constraints fk_parlu_parluver Foreign Key (Parent) references ParcelLU(ParcelLUID), 
    Constraints fk_parlu_lu Foreign Key (LanduseID) references Landuse(LanduseID), 
    Constraints Check (DBT_Until >= DBT_From), 
    Constraints Check (WT_Until >= WT_From) 
)

Create Table ParcelLUVer ( 
    ParcelLUID Number(9), 
    ParcelLUVerID Number(9) Constraints ck_parluver Check (ParcelLUVerID > 0), 
    BuildPermitNo Char(30), 
    ParcelID Char(15), 
    VersionID Number(9), 
    WT_From Date constraints nn_WTFrom Not Null, 
    WT_Until Date, 
    DBT_From Date Default SysDate, 
    DBT_Until Date, 
    PLUParent Number(9), 
    PLUVersion Number(9), 
    Constraints pk_par_lu_ver Primary Key (ParcelLUID, ParcelLUVerID), 
    Constraints fk_parluver_parlu Foreign Key (ParcelLUID) references ParcelLU(ParcelLUID), 
    Constraints fk_parluver_parluver Foreign Key (ParcelLUID, ParcelLUVerID), 
    Constraints fk_parlu_parluver Foreign Key (ParcelID, VersionID) references ParcelVersion(ParcelID, VersionID), 
    Constraints fk_parluver_parluver Foreign Key (PLUParent, PLUVersion) references ParcelLUVer(ParcelLUID, ParcelLUVerID), 
    Constraints fk_parlu_bp Foreign Key (BuildPermitNo) references BuildingPermit(BuildPermitNo), 
    Constraints Check (DBT_Until >= DBT_From), 
    Constraints Check (WT_Until >= WT_From) 
)

Create Table FieldSurvey ( 
)
FieldSurveyID Char(30) Constraints ck_fsid Check (FieldSurveyID = Upper(FieldSurveyID)),
SheetNo Char(30),
ConductedBy Char(30),
CheckedBy Char(30),
SurveyDateFrom Date,
SurveyDateUntil Date,
DBT_From Date Default SysDate,
DBT_Until Date,
Remarks Char(80),
Constraints pk_fsid Primary Key (FieldSurveyID),
Check (DBT_Until >= DBT_From)
)

Create Table FieldSurveyParcel (ParcelID Char(15),
VersionID Number(9),
FieldSurveyID Char(30),
LanduseID Char(10),
SurveyDateFrom Date,
SurveyDateUntil Date,
DBT_From Date Default SysDate,
DBT_Until Date,
Remarks Char(80),
Constraints pk_par_ver_fsp Primary Key (ParcelID, VersionID),
Constraints fk_parver_fsp Foreign Key (ParcelID, VersionID) references ParcelVersion(ParcelID, VersionID),
Constraints fk_fs_fsp Foreign Key (FieldSurveyID) references FieldSurvey(FieldSurveyID),
Constraints fk_lu_fsp Foreign Key (LanduseID) references Landuse(LanduseID),
Check (DBT_Until >= DBT_From)
)
Some illustration of implementation in VC++

Figure 1 List of class in CTSTD.

223
Figure 2 Data members (attributes) and members functions (operations) of PointClass.
Figure 3 ZeroTCell class: attributes and operations.

Figure 4 OneTCell class: attributes and operations.
Appendix

Figure 5 TwoTCell class: attributes and operations.

Figure 6 Parcel class: attributes and operations.
Algorithms for Point and ZeroTCell objects

**InsertPoint(Point) / InsertPoint(x,y)**

InsertPoint() is an overloaded function. This operation inserts the Point in the space (Point Table).

**Input:**
A Point data type can be passed as either input or x and y coordinates.

**Output:**
The output is new PointID. If the operation is successful (given Point or (x,y) are inserted) then output is PointID (>0), else PointID is 0.

**Method:**
I1: The Point is first searched for in the space with a certain tolerance level. The IsDuplicate() operation is used. IsDuplicate() returns Boolean value true or false.
I2: If IsDuplicate() is false, the new PointID is calculated and point is inserted in the PointTable.
If IsDuplicate() is true, the existing PointID is returned. If Point cannot be inserted, 0 PointID is returned.

**DestroyPoint(PID)**

DestroyPoint() deletes (permanently) the Point from the Point Table. This is a protected function, as the users of the spatio-temporal data are not allowed to delete the data. This is a special function, which is used to as purge operations.

**Input:**
Given a PointID (PID > 0).

**Output:**
The output is a PID. If the operation is successful then output is PID (=0), else input PID is (>0).

**Method:**
D1: The Point is first searched for in the space with a certain tolerance level. The IsDuplicate() operation is used. IsDuplicate() returns Boolean value true or false.
D2: If IsDuplicate() is true, the Point and associated PointID are deleted from the Point Table. If IsDuplicate() is false, this means Point does not exist.

**IsDuplicatePoint(Point)**

IsDuplicatePoint() checks the space (Point Table) for duplicate Point.

**Input:**
The input is a Point.

**Output:**
The output is a Boolean value. If the operation is successful (the given point exists), then output is true (1), else false (0).

**Method:**
D1: The IsDuplicate() function searches the space with a certain tolerance level. Each record of Point Table is searched (the search depends upon the indexing technique used in the RDBMS). The IsEqual() operation is used to check the duplicate Point. IsEqual() returns Boolean value true or false.
D2: If IsEqual() is true, the Point exists in the space (Point Table); therefore IsDuplicate() returns true. If IsEqual() is false, then given Point does not exist and IsDuplicate() returns false.
IsEqual(P1, P2)
IsEqual() checks whether two points P1 and P2 are equal at a certain tolerance level.

**Input:**
Two Points, i.e., Point P1 and Point P2.

**Output:**
The output is a Boolean value. If the operation is successful (the two points are equal), then the output is true (1), else false (0).

**Method:**
The IsEqual() function checks the x-member of P1 with x-member of P2 and y-member of P1 with y-member of P2. If both are equal with a certain tolerance, then IsEqual() is true, else false.

GetPoint(PID)
Retrieve the Point of given PointID (PID).

**Input:**
A PointID (PID).

**Output:**
A Point (x,y co-ordinates).

**Method:**
GP1:
Search the space (Point Table) for given PID. If found return the corresponding Point, else return the minimum value of float data type, i.e., (-858993460, -858993460).

GetPointID(P)
Get the PointID (PID) of a given Point (P).

**Input:**
A Point (P).

**Output:**
A PointID (PID).

**Method:**
GI1:
Search the space (Point Table) for given P. If found, return the corresponding PID, else return 0.

GetMaximumPointID()
Get the maximum PointID (PID) in the space (Point Table).

**Input:**
Null.

**Output:**
A PointID (PID). Return 0 PID if Point Table is empty.

**Method:**
GI1:
Search the space (Point Table) for a maximum PID.

GetMaximumPoint()
Retrieve maximum value of Point in the space (Point Table).

**Input:**
Null.

**Output:**
A Point (x, y). Return (-858993460, -858993460) when Point Table is empty.

**Method:**
GM1:
Search for the maximum value of x in the space (Point Table). For maximum value of x, search for maximum value of y.
**GetMinimumPoint()**

Get minimum value of Point in the space (Point Table).

**Input:**
Null.

**Output:**
A Point \((x, y)\). Return \((858993460, 858993460)\) when Point Table is empty.

**Method:**
GMi:
Search for the minimum value of \(x\) in the space (Point Table). For minimum value of \(x\), search for minimum value of \(y\).

**InsertZeroTCell(P)**

ZeroTCell is a spatio-temporal object. The operation InsertZeroTCell(), insert a Point \((P1)\) in ZeroTCell Table defined by PointID.

**Input:**
The PointID \((P)\).

**Output:**
ZeroTCellID, in case of successful operation (>0) else (0).

**Method:**
IZ1:
If \(P\) is (>0), IsExistZeroTCell() check the duplicate zerotcellid. If it is false, then unique ZeroTCellID is calculated by MaximumZeroTCellID(). \(P\) is then inserted with unique ZeroTCellID in ZeroTCellTable. Unique ZeroTCellID is returned. If IsExistZeroTCell() is true, then operation is suspended and 0 is returned. If \(P\) is (=0), operation is suspended and 0 is returned.

**DestroyZeroTCell(P)**

Delete ZeroTCell defined by PointID.

**Input:**
ZeroTCellID \((P)\).

**Output:**
ZeroTCellID. In case of successful operation (>0) else (0).

**Method:**
This operation assumes that the Point already exists in space (Point Table). The PointID is returned by InsertPoint() operation. The new (unique) ZeroTCellID is computed by MaximumZeroTCellID() operation and assigned to the \(P\). If \(P\) and ZeroTCellID is inserted successfully, then ZeroTCellID (>0) are returned, otherwise (0), indicating failure of operation.

**GetZeroTCell(P: PointID) / GetZeroTCell(P: Point)**

It is an overloaded operation. Get ZeroTCell return ZeroTCell ID for given PointID \((P)\).

**Input:**
PointID \((P)\).

**Output:**
ZeroTCellID. In case of successful operation (>0) else (0).

**Method:**
Select ZTCID from ZTC_Table
where PointID = \(P\);
if \((ZTCID > 0)\) return ZTCID else 0;

**GetZeroTCell(P: Point)**

Get ZeroTCell return ZeroTCell ID for given Point \((P)\).
Input: The Point (P).
Output: ZeroTCellID. In case of successful operation (>0) else (0).
Method:
Select PID from Point_table where Point = P;
If (PID > 0)
   { Select ZTCID from ZTC_Table
      where PointID = PID;
      if (ZTCID > 0)
         return ZTCID
      else return 0;
   }
else
   return 0;

IsExistZeroTCell(P: PointID)
Check that the given PointID (P) is ZeroTCell.
Input: The PointID (P).
Output: Boolean true if the given PointID is a ZeroTCell, else false.
Method:
Select ZTCID from ZTC_table where PointID = P;
If (ZTCID > 0)
   return true
else return false;

MaximumZeroTCellID()
Get the maximum ZCTID.
Input: Null.
Output: ZeroTCellID.
Method:
Select Max(ZTCID) from ZTC_table;
If (ZTCID > 0)
   return ZTCID
else return 0;
Implementation of algorithm for OTC-OTC Intersection in VC++

```cpp
int COneTCellClass::GetIntersect(CTPoint P1, CTPoint P2, CTPoint P3, CTPoint P4)
{
    NoOfCommonPoints = 0;
    CTPoint Eq1, Eq2, InterP1, InterP2;
    float parameter1, parameter2, NewSegParaAtP1, NewSegParaAtP2, ExistSegParaAtP1, ExistSegParaAtP2, A, B, C, e1x, e1y, e2x, e2y;
    A = ( (P1.x-P3.x)*(P4.y-P3.y) - (P4.x-P3.x)*(P1.y-P3.y) );
    B = ( (P2.x-P1.x)*(P3.y-P1.y) - (P3.x-P1.x)*(P2.y-P1.y) );
    C = ( (P2.x-P1.x)*(P3.y-P4.y) - (P3.x-P4.x)*(P2.y-P1.y) );
    if (C != 0)
    {     // Line-Line intersection rules: 2, 3, 4
        parameter1 = A/C;
        parameter2 = B/C;
        // Check condition that parameter >= 0 and <=1 for straight line
        if ( (parameter1 >= 0) && (parameter1 <= 1) && (parameter2 >= 0) && (parameter2 <= 1) )
        {
            e1x = P1.x + parameter1*(P2.x-P1.x);
            e1y = P1.y + parameter1*(P2.y-P1.y);
            e2x = P3.x + parameter2*(P4.x-P3.x);
            e2y = P3.y + parameter2*(P4.y-P3.y);
            if ( (e1x >= e2x - .0005) && (e1x <= e2x + .0005) && (e1y >= e2y - .0005) && (e1y <= e2y + .0005) )
            {
                P.x = e1x;
                P.y = e1y;
                FirstCommonPoint = P;
                NoOfCommonPoints = 1;
                return NoOfCommonPoints;
            }
        }
    }
    else
    { if (C == 0) ||
        { // Seg-Seg intersection rules: 5, 6, 7, 8, 9, 11, 12, 13, 14, 15
            NewSegParaAtP1 = ParameterAtPoint(P1, P2, P1);
            NewSegParaAtP2 = ParameterAtPoint(P1, P2, P2);
            ExistSegParaAtP1 = ParameterAtPoint(P3, P4, P1);
            ExistSegParaAtP2 = ParameterAtPoint(P3, P4, P2);
            if ( (NewSegParaAtP1 == 0) && (NewSegParaAtP2 == 0) )
            { // Seg-Seg: R:5+14
                if ( (ExistSegParaAtP1 == 0) && (ExistSegParaAtP2 > 0) &&
```
\begin{align*}
&\text{if } (\text{ExistSegParaAtP2 < 1}) \quad \text{||} \quad (\text{ExistSegParaAtP1 == 1}) \& \& (\text{ExistSegParaAtP2 > 0}) \& \& \\
&\text{(ExistSegParaAtP2 < 1})
\end{align*}

\begin{align*}
\text{InterP1} &= \text{PointAtParameter}(P1, P2, \text{NewSegParaAtP2}); \\
\text{InterP2} &= \text{PointAtParameter}(P3, P4, \text{ExistSegParaAtP2}); \\
\text{if } (\text{isEqual(InterP1, InterP2) \&\& isEqual(InterP1, P2)})
\end{align*}

\begin{align*}
\text{NoOfCommonPoints} &= 1; \\
\text{FirstCommonPoint} &= P2; \\
\text{MarkRecForRemove}(P1, P2); \\
\text{return NoOfCommonPoints;}
\end{align*}

\begin{align*}
\text{else}
\text{if } (\text{ExistSegParaAtP1 == 1}) \& \& (\text{ExistSegParaAtP2 == 0}) \quad \text{||} \quad \text{(ExistSegParaAtP2 == 1})
\end{align*}

\begin{align*}
\text{InterP1} &= \text{PointAtParameter}(P1, P2, \text{NewSegParaAtP1}); \\
\text{InterP2} &= \text{PointAtParameter}(P3, P4, \text{ExistSegParaAtP1}); \\
\text{if } (\text{isEqual(InterP1, InterP2) \&\& isEqual(InterP1, P1)})
\end{align*}

\begin{align*}
\text{NoOfCommonPoints} &= 1; \\
\text{FirstCommonPoint} &= P1; \\
\text{MarkRecForRemove}(P1, P2); \\
\text{return NoOfCommonPoints;}
\end{align*}

/// End of R:5+14
//else
// Start R: 9
\begin{align*}
\text{if } (\text{ExistSegParaAtP1 > 0}) \& \& (\text{ExistSegParaAtP1 < 1}) \& \& \\
(\text{ExistSegParaAtP2 > 0}) \& \& (\text{ExistSegParaAtP2 < 1}) \& \& \\
(\text{isEqual(NotExistParaAtPoint(P3, P4, ParameterAtPoint(P3, P4, P1)), P1)}) \& \& \\
(\text{isEqual(NotExistParaAtPoint(P3, P4, ParameterAtPoint(P3, P4, P2)), P2)})
\end{align*}

\begin{align*}
\text{//if } (\text{ParameterAtPoint(P3, P4, P1) > 0}) \& \& (\text{ParameterAtPoint(P3, P4, P2) < 1})
\end{align*}

\begin{align*}
\text{if } \\
\text{((P1.x > P3.x) \& \& (P2.x < P4.x) \& \& (P2.x<P1.x)) ||} \\
\text{(P1.y > P3.y) \& \& (P2.y < P4.y) \& \& (P2.y>P1.y)) ||} \\
\text{((P1.x < P3.x) \& \& (P2.x > P4.x) \& \& (P2.x>P1.x)) ||} \\
\text{(P1.y < P3.y) \& \& (P2.y > P4.y) \& \& (P2.y<P1.y))}
\end{align*}

\begin{align*}
\text{UpdateIntersectionData}(P1, P3, P4); \\
\text{UpdateIntersectionData}(P2, P3, P4); \\
\text{MarkRecForRemove}(P1, P2); \\
\text{NoOfCommonPoints} &= 0; \\
\text{return NoOfCommonPoints;}
\end{align*}

\begin{align*}
\text{else}
\text{if } \\
\text{(((P1.x < P4.x) \& \& (P2.x > P3.x) \& \& (P2.x<P1.x)) || ((P1.y < P4.y) \& \& (P2.y > P3.y) \& \&}
\end{align*}
\{(P2.y<P1.y)) \&\& ((P1.x > P4.x) \&\& (P2.x < P3.x) \&\& (P2.x>P1.x) ) \&\& ((P1.y > P4.y) \&\& (P2.y < P3.y) \&\& (P2.y>P1.y) )\} \\

UpdateIntersectionData(P2, P3, P4);
UpdateIntersectionData(P1, P3, P4);
MarkRecForRemove(P1,P2);
NoOfCommonPoints = 0;
return NoOfCommonPoints;
} // End of R:9
else

// Start R: 7
if ( ( ( ( ExistSegParaAtP1 < 0) &&  (ExistSegParaAtP2 > 1)) ||

((ExistSegParaAtP1 > 1) &&  (ExistSegParaAtP2 < 0)) ) &&

( IsEqual(PointAtParameter(P1, P2, ParameterAtPoint(P1, P2, P3)), P3) ) &&

(IsEqual(PointAtParameter(P1, P2, ParameterAtPoint(P1, P2, P4)), P4) ) )

{ if ( (ParameterAtPoint(P1, P2, P3) > 0) && (ParameterAtPoint(P1, P2, P4) <1) )

{ if ( (P2.x > P1.x) \&\& (P4.x > P3.x) ) \&\& ((P2.y > P1.y) \&\& (P4.y > P3.y) )

{ NoOfCommonPoints = 0;
PushCurrentSeg(P1, P2, P3, P4);
return NoOfCommonPoints;
}

else if ( ( (P2.x < P1.x) && (P4.x > P3.x)) || ( (P2.y > P1.y) && (P4.y > P3.y) ) )

{ NoOfCommonPoints = 0;
PushCurrentSeg(P1, P2, P4, P3);
return NoOfCommonPoints;
}

else if ( ( (P2.x > P1.x) && (P4.x < P3.x)) || ( (P2.y > P1.y) && (P4.y < P3.y) )

{ NoOfCommonPoints = 0;
PushCurrentSeg(P1, P2, P4, P3);
return NoOfCommonPoints;
}

else if ( ( (P2.x < P1.x) && (P4.x < P3.x)) || ( (P2.y < P1.y) && (P4.y < P3.y) )

{ NoOfCommonPoints = 0;
PushCurrentSeg(P1, P2, P3, P4);
return NoOfCommonPoints;
}

} // End of R:7
else

// Start R: 8
if ( ( (ExistSegParaAtP1 == 0) &&  (ExistSegParaAtP2 > 1) ) ||

((ExistSegParaAtP1 > 1) &&  (ExistSegParaAtP2 == 0)) ) ||

( ExistSegParaAtP1 == 1) &&  (ExistSegParaAtP2 < 0) ) ||

( ExistSegParaAtP1 < 0) &&  (ExistSegParaAtP2 == 1) )

{ if ( (ParameterAtPoint(P1, P2, P3) == 0) && (ParameterAtPoint(P1, P2, P4) >0) &&

( ParameterAtPoint(P1, P2, P4) <1) &&

(IsEqual(PointAtParameter(P1, P2, ParameterAtPoint(P1, P2, P4)), P4) ) )

{ if (ReplacePointOfCurrentSeg(P1, P2, P3, P4, P1,P4) == TRUE)
PushAt = Position1;
NoOfCommonPoints = 3;
return NoOfCommonPoints;
}
else
if ( (ParameterAtPoint(P1, P2, P3) == 1) && (ParameterAtPoint(P1, P2, P4) > 0) &&
( ParameterAtPoint(P1, P2, P4) < 1) &&
( IsEqual(PointAtParameter1(P1, P2, ParameterAtPoint(P1, P2, P4), P4) ) )
( ReplacePointOfCurrentSeg(P1, P2, P3, P4, P2, P4) == TRUE )
PushAt = Position1;
NoOfCommonPoints = 3;
return NoOfCommonPoints;
}
else
if ( (ParameterAtPoint(P1, P2, P3) > 0) && (ParameterAtPoint(P1, P2, P3) < 1) &&
( ParameterAtPoint(P1, P2, P4) == 0) &&
( IsEqual(PointAtParameter1(P1, P2, ParameterAtPoint(P1, P2, P3), P3) ) )
if ( ReplacePointOfCurrentSeg(P1, P2, P3, P4, P1, P3) == TRUE )
PushAt = Position1;
NoOfCommonPoints = 3;
return NoOfCommonPoints;
}
else
if ( (ParameterAtPoint(P1, P2, P3) > 0) && (ParameterAtPoint(P1, P2, P3) < 1) &&
( ParameterAtPoint(P1, P2, P4) == 1) &&
( IsEqual(PointAtParameter1(P1, P2, ParameterAtPoint(P1, P2, P4), P3) ) )
if ( ReplacePointOfCurrentSeg(P1, P2, P3, P4, P2, P3) == TRUE )
PushAt = Position1;
NoOfCommonPoints = 3;
return NoOfCommonPoints;
}
if ( (ExistSegParaAtP1 > 0) && (ExistSegParaAtP1 < 1) && (ExistSegParaAtP2 > 1) ||
( ExistSegParaAtP1 > 1) && (ExistSegParaAtP2 > 0) && (ExistSegParaAtP2 < 1) ||
( ExistSegParaAtP1 > 0) && (ExistSegParaAtP1 < 1) && (ExistSegParaAtP2 < 0) ||
( ExistSegParaAtP1 < 0) && (ExistSegParaAtP2 > 0) && (ExistSegParaAtP2 < 1) )
{ if ( Exists paraAtAIP1 > 0) && (ExistSegParaAtP1 < 1) && (ExistSegParaAtP2 > 1) &&
IsEqual(PointAtParameter1(P1, P2, ParameterAtPoint(P1, P2, P4), P4) )
{ NoOfCommonPoints = 1;
FirstCommonPoint = P1;
UpdateCurrentExistingSeg(P1, P2, P1, P4);
//ReplacePointOfCurrentSeg(P1, P2, P1, P4);
SweepPoint1 = P4;
return NoOfCommonPoints;
}
else
if ( (ExistSegParaAtP1 > 1) && (ExistSegParaAtP2 > 0) && (ExistSegParaAtP2 < 1) &&
IsEqual(PointAtParameter1(P1, P2, ParameterAtPoint(P1, P2, P4), P4) )
{ NoOfCommonPoints = 1;
FirstCommonPoint = P2;
//ReplacePointOfCurrentSeg(P1, P2, P2, P4);
UpdateCurrentExistingSeg(P1, P2, P2, P4);
SweepPoint2 = P4;
return NoOfCommonPoints;
}
else
if ( ExistSegParaAtP1 > 0 ) && ( ExistSegParaAtP1 < 1 ) && ( ExistSegParaAtP2 < 0 ) &&
IsEqual(PointAtParameter(P1, P2, ParameterAtPoint(P1, P2, P3) ),P3 ) )
{
NoOfCommonPoints = 1;
FirstCommonPoint = P1;
//ReplacePointOfCurrentSeg(P1, P2, P1, P3);
UpdateCurrentExistingSeg(P1, P2, P1, P3);
SweepPoint1 = P3;
return NoOfCommonPoints;
}
else
if ( (ExistSegParaAtP1 < 0) && (ExistSegParaAtP2 > 0) && (ExistSegParaAtP2 < 1) &&
IsEqual(PointAtParameter(P1, P2, ParameterAtPoint(P1, P2, P3) ),P3 ) )
{
NoOfCommonPoints = 1;
FirstCommonPoint = P2;
//ReplacePointOfCurrentSeg(P1, P2, P2, P3);
UpdateCurrentExistingSeg(P1, P2, P2, P3);
SweepPoint2 = P3;
return NoOfCommonPoints;
}
endl End of R:12
else

// Start R: 16
#endif
{( ((P2.x-P1.x) == (P4.x-P3.x) ) && ((P2.y-P1.y) == (P4.y-P3.y)) ) ||
#endif
((P2.x-P1.x) == (P2.y-P1.y) ) && ((P4.x-P3.x) == (P4.y-P3.y)) )
{
NoOfCommonPoints = 0;
return NoOfCommonPoints;
}
#endif End of cases where Seg1 <= Seg2
#endif End of C==0
NoOfCommonPoints = 0; return NoOfCommonPoints;
AML routines for overlay process

/* Goverlay.aml
 &run GenTTC1.aml
 /* &sys "pause"
 &run GenUnion.aml/*
 /* &sys "pause"
 &run UnGenTTC.aml

 Infodbase ResTTC.pat ResPAT
 Infodbase ResTTC.aat ResAAT

/* GenTTC1.aml
/* generate First TTC (Polygon).
 kill FirstTTC all
 generate FirstTTC
 input TTC1.aml
 lines
 quit
 clean FirstTTC
 build FirstTTC line
 build FirstTTC poly

 arcedit AddID1.aml
 build FirstTTC line
 build FirstTTC poly

 Tables
 &run AdParent.aml

/* GenUnion.aml
/* to generate Union of TTC1 & TTT2 (Polygon).
 &run ithTTC.aml

 &sv .5 = 2

 &label Repeat
 &if %.5% le %.4% &then
 &goto DoOverlay
 &else
 &goto END

 &label DoOverlay
 /* &sys "pause"
 &lv &file TTCNo.aml &format '&sv %1% = %2%'
 &lv

 &run genTTC2.aml
 list lindttc.pat
 /* &sys "pause"
 kill ResTTC all

 union lIndTTC FirstTTC ResTTC
 clean ResTTC
 Build ResTTC line
 Build ResTTC poly

 Tables
 &run UpDatPar.aml
 /* &sys "pause"
 idedit ResTTC poly
 list Resttc.pat
 build resttc line
 build resttc poly
 /* &rem Tables DropCol.aml
 List ResTTC.pat

 kill FirstTTC all
 copy ResTTC FirstTTC

 &sv .5 = %.5% + 1
 &goto Repeat
 /* &goto Repeat &if &range %.4% %.5% 100

 &label END
 &return
Appendix

Land use maps

Figure 7 Block 4: land use 1987.
Land use classes
- Residential: pucca
- Health
- Assembly
- Religious places
- Government offices
- Other government building
- Transportation terminals
- Transportation: right of way
- Utilities
- Parks
- Burial ground
- Residential: semi-pucca
- Quarries
- Agriculture
- Irrigation
- Water bodies
- Annual flood
- Salt pans
- Vacant: developed
- Vacant: approved
- All other vacant
- Restricted area
- Residential: kutcha / huts
- Commercial
- Industrial
- Warehousing
- Mixed uses
- Goth / villages
- Educational
- No data

Figure 8 Block 4: land use 1993.
Appendix

Land use classes
- Residential: pucca
- Health
- Assembly
- Religious places
- Government offices
- Other government buildings
- Transportation terminals
- Transportation: right of way
- Utilities
- Parks
- Burial grounds
- Residential: semi-pucca
- Quarries
- Agriculture
- Irrigation
- Water bodies
- Annual flood
- Salt pans
- Vacant: developed
- Vacant: approved
- All other vacant
- Restricted area
- Residential: kutcha/huts
- Commercial
- Industrial
- Warehousing
- Mixed uses
- Educational
- No data

Figure 9 Block 4: land use 1998.
## Land use codes

<table>
<thead>
<tr>
<th>LANDUSEID</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Residential pucca</td>
</tr>
<tr>
<td>10</td>
<td>Health</td>
</tr>
<tr>
<td>11</td>
<td>Assembly: social and cultural institutions</td>
</tr>
<tr>
<td>12</td>
<td>Religious places: all religious institutions</td>
</tr>
<tr>
<td>13</td>
<td>Government offices: all government offices</td>
</tr>
<tr>
<td>14</td>
<td>Other government buildings</td>
</tr>
<tr>
<td>15</td>
<td>Transportation terminals</td>
</tr>
<tr>
<td>16</td>
<td>Transportation right of ways</td>
</tr>
<tr>
<td>17</td>
<td>Utilities</td>
</tr>
<tr>
<td>18</td>
<td>Parks</td>
</tr>
<tr>
<td>19</td>
<td>Burial grounds</td>
</tr>
<tr>
<td>2</td>
<td>Residential semi-pucca</td>
</tr>
<tr>
<td>20</td>
<td>Quarries</td>
</tr>
<tr>
<td>21</td>
<td>Agriculture</td>
</tr>
<tr>
<td>22</td>
<td>Irrigation</td>
</tr>
<tr>
<td>23</td>
<td>Water bodies</td>
</tr>
<tr>
<td>24</td>
<td>Annual flood</td>
</tr>
<tr>
<td>25</td>
<td>Salt pans</td>
</tr>
<tr>
<td>26</td>
<td>Vacant land</td>
</tr>
<tr>
<td>27</td>
<td>Vacant approved</td>
</tr>
<tr>
<td>28</td>
<td>All other vacant</td>
</tr>
<tr>
<td>29</td>
<td>Restricted area</td>
</tr>
<tr>
<td>3</td>
<td>Residential kutcha / jhuggi /huts</td>
</tr>
<tr>
<td>4.1</td>
<td>Commercial shops &amp; offices</td>
</tr>
<tr>
<td>4.2</td>
<td>Commercial temporary shops</td>
</tr>
<tr>
<td>5</td>
<td>Industrial</td>
</tr>
<tr>
<td>6</td>
<td>Warehousing</td>
</tr>
<tr>
<td>7.1</td>
<td>Residential/commercial</td>
</tr>
<tr>
<td>7.2</td>
<td>Industrial / residential</td>
</tr>
<tr>
<td>7.3</td>
<td>Industrial / commercial</td>
</tr>
<tr>
<td>8</td>
<td>Goths / villages</td>
</tr>
<tr>
<td>9</td>
<td>Educational</td>
</tr>
</tbody>
</table>
Completed PhD studies at ITC

2. Pan He Ping, 1990, 90-9003757-8, Spatial structure theory in machine vision and applications to structural and textural analysis of remotely sensed images.
7. Sharifi, A., 1991, 90-6164-074-1, Development of an appropriate resource information system to support agricultural management at farm enterprise level.
22. Woldai, T., 1995, The application of remote sensing to the study of the geology and structure of the Carboniferous in the Calafias area, pyrite belt, SW Spain.
27. Hoanh Chu Thai, 1996, 90-6164-120-9, Development of a Computerized Aid to
Integrated Land Use Planning (CAILUP) at regional level in irrigated areas: a case study for the Quan Lo Phung Hiep region in the Mekong Delta, Vietnam.


31. Al-Amir, S., 1996, 90-6164-116-0, Modern spatial planning practice as supported by the multi-applicable tools of remote sensing and GIS: the Syrian case.


41. Ceccarelli, T., 1997, 90-6164-135-7, Towards a planning support system for communal areas in the Zambezi valley, Zimbabwe; a multi-criteria evaluation linking farm household analysis, land evaluation and geographic information systems.

42. Peng Wanning, 1997, 90-6164-134-9, Automated generalization in GIS.


44. Bijker, W., 1997, 90-6164-139-X, Radar for rain forest: A monitoring system for land cover Change in the Colombian Amazon.


47. Bishr, Y., 1997, 90-6164-141-1, Semantic Aspects of Interoperable GIS.


50. Turkstra, J., 1998, 90-6164-147-0, Urban development and geographical information: spatial and temporal patterns of urban development and land values using integrated
geo-data, Villaviciencia, Colombia.


57. Moameni, A., 1999, Soil quality changes under long-term wheat cultivation in the Marvdasht plain, South-Central Iran.


60. Wolski, Piotr, 1999, 90-6164-165-9, Application of reservoir modelling to hydrotopes identified by remote sensing.


65. Eleveld, M., 1999, 90-6461-166-7, Exploring coastal morphodynamics of Ameland (The Netherlands) with remote sensing monitoring techniques and dynamic modelling in GIS.


73. Schetselaar, Ernst, 2000, 90-6164-180-2, Integrated analyses of granite-gneiss terrain from field and multisource remotely sensed data. A case study from the Canadian Shield.


Curriculum Vitae

Ale Raza, born on 5th January 1965, received his MSc degree in Statistics (with distinction) from the University of Karachi, Pakistan, in 1988. From 1988 to 1994, he worked as an Assistant Director in the Computer Unit of Karachi Development Authority (KDA), Pakistan. At KDA, he was actively involved with the UNDP/UNCHS team in the development of urban databases, urban growth models and digital maps. In 1991, he received training in Computer Hardware Maintenance, at the Asian Institute of Technology (AIT), Bangkok, Thailand. From 1991 to 1994, he was a visiting faculty member of one of the largest computer institutes of Pakistan, where he was engaged in designing and conducting computer training programs for professionals and multinational companies. In October 1994, he became a student at the International Institute for Aerospace Survey and Earth Sciences (ITC), Enschede, The Netherlands. At ITC in September 1996, he received a MSc degree (with distinction) in GIS for urban applications. Prior to joining ITC in 1994, he completed his fourth semester (out of six) in MSc Computer Science at NED University of Karachi. In January 1997, he started his PhD research that is reported in this dissertation. He is the author and co-author of several international publications and has received an award for the best paper.