

Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Modelling Time in Geodatabases“

vorgelegt von

Lukas Künzel BSc,
U1267, UNIGIS MSc Jahrgang 2006

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)“

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Wien, 29.04.2008

Acknowledgements

At this point I would like to thank everybody who supported me at my work for this thesis. I'm indebted to Marcus Scheiber for giving me the basic conditions for completing this work. Sincere thanks goes to my colleges at ms.gis as they provided their helpful support within regular discussion meetings which resulted in valuable input for my work. Special thanks go to Alexander Zidek for guiding me through this work and for reminding me that there is a light at the end of the tunnel.

Finally I would like to thank my girlfriend Jennifer for her continuous motivation. The birth of our son Tyler was a delightful event and reminded me that new challenges are waiting in the wings.

Abstract

“Space and time are among the most fundamental of notions. They provide a basis for ordering all modes of thought and belief” (Peuquet, 2002). For the dimension of space it is state of the art to store information about real world phenomena in geodatabases. The fundamental units of geography are called features (OpenGis_Consortium, 1999) which are implemented as rows of database tables including one column for the location on the earth’s surface. To provide appropriate standards for querying and analysing features the Open Geospatial Consortium had developed the relevant specifications. The most important concept was to extend the relational model defined by Codd (Codd, 1970) in order to integrate space into a data base. This concept is described in the simple feature specification (OpenGIS_Consortium, 2006a) where geometries representing a location on the earth’s surface are implemented as user defined data types. The simple feature specification is implemented on a widespread basis by major RDBMS vendors and is the state of the art technique for storing information about real world objects.

Besides space the second fundamental dimension of our life is time (Peuquet, 2002). In the past much effort had been done to integrate time into databases. What was common to all developments was the extension of the relational model defined by Codd (Codd, 1970). But what is left until today is an agreement of major database vendors on how to integrate time. This is in sharp contrast to the dimension of space where there already exists an industry for spatial databases.

The main goal of this work is to transform time into geometry in order to integrate it into geodatabases. The idea is mainly based on the concept of geometry of time within the ISO 19108 standard (ISO, 2002). But already Frank (Frank, Egenhofer, & Colledge, 1998) had identified that it may be a natural approach to transform time into geometry which corresponds in some aspects to the perception of time of humans.

Table of Contents

1. INTRODUCTION	1
1.1 Hypothesis.....	3
1.2 Time as Geometry.....	3
1.2.1 Theory	3
1.2.2 Methods for Implementation	4
1.2.3 Data	5
1.3 Expected Results	6
1.4 Issues not covered	6
1.5 Audience	6
1.6 Structure of the Thesis.....	7
BACKGROUND RESEARCH	8
2. THE RELATIONAL MODEL.....	9
2.1 Integrity constraints.....	10
2.2 Normalisation.....	10
2.3 Structured Query Language SQL.....	11
2.4 Contribution.....	11
3. SIMPLE FEATURES.....	12
3.1 Definitions	13
3.2 Features	14
3.3 Geometries.....	14
3.4 Spatial Operations.....	15
3.5 Geodatabases	15
4. ABOUT TIME.....	16
4.1 Types of Time	16
4.2 Abstractions of Time	18
4.3 Dimensions of Time	19
4.3.1 Valid Time.....	19
4.3.2 Transaction Time.....	19
4.3.3 User Defined Time	19

4.4	Discrete vs. Continuous	21
4.5	Temporal Data types.....	23
4.5.1	Instants.....	23
4.5.2	Intervals.....	23
4.5.3	Periods	24
4.6	Subtleties	25
4.6.1	Leap Year.....	25
4.6.2	Leap Seconds.....	25
4.6.3	Daylight Saving Time	25
5.	TEMPORAL RELATIONS	26
5.1	Timestamps.....	26
5.1.1	Attribute Level.....	26
5.1.2	Tuple Level	27
5.1.3	Interval representation	27
5.1.4	Point representation	27
5.2	Modifying Temporal Relations.....	28
5.2.1	Insert	28
5.2.2	Update.....	29
5.2.3	Delete	30
5.3	Normalisation.....	31
5.3.1	Functional Dependencies.....	32
5.3.2	Temporal Primary Key.....	33
5.3.3	Temporal Foreign Key	35
5.4	Current State.....	37
5.4.1	Temporal Partitioning	37
5.5	Temporal Databases	38
5.6	Temporal Data models.....	39
6.	GEOMETRY OF TIME	41
6.1	Definitions	41
6.2	Temporal geometric primitives	42
6.3	Temporal reference system.....	43

6.4	From Temporal Objects to Spatio Temporal Objects	44
MAIN PART -TIME AS GEOMETRY.....		45
7. TIME AS GEOMETRY – A FORMAL DESCRIPTION		46
7.1	Background	46
7.2	A Discrete View of Time	47
7.3	A discrete view of time based on an interval scale	49
7.4	Functions.....	51
7.4.1	TimeStampToNumber(fpOrigin, fpDate)	52
7.4.2	NumberToTimeStamp(fpOrigin, NumberOfSeconds).....	53
7.5	Temporal geographic primitives	54
7.5.1	Instant	55
7.5.2	Period	56
7.6	Summary	57
8. TIME AS GEOMETRY – A PROTOTYPICAL IMPLEMENTATION.....		58
8.1	Scope.....	58
8.2	Functional Specification.....	59
8.2.1	Need to have	59
8.2.2	Nice to have	59
8.2.3	Not to have.....	59
8.3	Base technology	60
8.3.1	Oracle Express Edition with Oracle Locator	60
8.3.2	Oracle SQL Plus Worksheet.....	60
8.3.3	Esri ArcView	60
8.4	Functions.....	61
8.4.1	TimeStampToNumber	61
8.4.2	TimeToGeometry	63
8.4.3	NumberToTimeStamp.....	65
8.4.4	GeometryToTime	66
8.5	Spatial operations in temporal context	67
8.5.1	Disjoint	68
8.5.2	Touch.....	68

8.5.3	Covers.....	69
8.5.4	Contains	69
8.5.5	Equal.....	70
8.5.6	Overlap.....	70
8.5.7	Before or After	71
8.5.8	Intersection	72
8.5.9	Union.....	72
8.5.10	Difference.....	73
8.5.11	Length.....	73
8.6	The bus station scenario	74
8.6.1	Entity Relationship Model.....	75
8.6.2	Initial Data Loading	76
8.6.3	Spatial Questions	77
8.6.4	Temporal Questions.....	78
8.6.5	Spatio-temporal questions.....	79
9.	SUMMARY.....	82
9.1	Conclusion.....	83
9.2	Future Research	84
10.	BIBLIOGRAPHY	85

List of Figures

Figure 1 Transforming time into geometry	4
Figure 2 Bus station scenario	5
Figure 3 Structure of the thesis	7
Figure 4 Geometry basic classes (ISO, 2003)	12
Figure 5 Geometry class hierarchy (OpenGIS_Consortium, 2006a)	14
Figure 6 DE-9IM Matrix (OpenGIS_Consortium, 2006a)	15
Figure 7 Types of Time (Frank, Egenhofer, & Colledge, 1998)	16
Figure 8 Branching Time (Ott & Swiaczny, 2001)	17
Figure 9 Cycling Time (Ott & Swiaczny, 2001)	18
Figure 10 Valid Time and Transaction Time (Ott & Swiaczny, 2001).....	20
Figure 11 Queries on both dimensions of time (Ott & Swiaczny, 2001, adoptep from Langran)	20
Figure 12 Chronos with different granularities (Ott & Swiaczny, 2001).....	22
Figure 13 Temporal Update Cases (Snodgrass, 2000, S. 196)	29
Figure 14 Cases of Temporal Deletions (Snodgrass, 2000, S. 191).....	30
Figure 15 Temporal Databases (Worboys, 1994)	38
Figure 16 Temporal Data Models (Snodgrass, 1992).....	39
Figure 17 Comparison of Temporal Data Models (Kaiser, 1998).....	39
Figure 18 Temporal Objects (ISO, 2002)	42
Figure 19 Temporal geometric primitives (ISO, 2002).....	43
Figure 20 Temporal Objects (Roosmann, Busch, Gorczyk, & Mauersberger, 2003)	44
Figure 21 Spatio temporal object (Roosmann, Busch, Gorczyk, & Mauersberger, 2003).....	44
Figure 22 Discrete view of time (Gregorian Calendar + UTC).....	48
Figure 23 Calendar time vs. Interval scale	49
Figure 24 Functions needed for transformation	51
Figure 25 Date specification to number	52
Figure 26 Number to date specification	53
Figure 27 Temporal geometric primitives (ISO, 2002).....	54
Figure 28 Process of transforming an instant into a point	55
Figure 29 Process of transforming a period into a line	57

Figure 30 TimeStampToNumber source code	61
Figure 31 Transforming a date specification into a number	62
Figure 32 TimeToGeometry source code.....	63
Figure 33 Transforming an instant into a point geometry	64
Figure 34 Transforming a period into a line geometry.....	64
Figure 35 NumberToTimeStamp source code	65
Figure 36 Transforming a number into a date specification	65
Figure 37 GeometryToTime source code	66
Figure 38 Transforming geometry into a date specification	66
Figure 39 Spatial and temporal operators (Ott & Swiaczny, 2001)	67
Figure 40 Disjoint relationship.....	68
Figure 41 Touch relationship	68
Figure 42 Covers relationship	69
Figure 43 Contains relationship	69
Figure 44 Equal relationship	70
Figure 45 Overlap relationship	70
Figure 46 Before or After relationship.....	71
Figure 47 Intersection operation	72
Figure 48 Union operation.....	72
Figure 49 Difference Operation	73
Figure 50 Length calculation	73
Figure 51 Bus station scenario	74
Figure 52 Bus station ERM	75
Figure 53 Inserting values into the bus schedule	76
Figure 54 Definition of the origin.....	76
Figure 55 Definition of my time	76
Figure 56 Bus schedule viewed on the time line	77
Figure 57 Spatial question	77
Figure 58 Temporal question.....	78
Figure 59 Spatio temporal Question I.....	79
Figure 60 Spatio temporal question II	80
Figure 61 Spatio temporal question III	81

List of Tables

Table 1 A relation viewed as a two dimensional table	9
Table 2 Temporal operators vs. Oracle Locator operations	67
Table 3 Temporal set operations vs. Oracle set operations	67

1. Introduction

Based on the definition of Peuquet *“Space and time are among the most fundamental of notions. They provide a basis for ordering all modes of thought and belief”* it is the main motivation of this work to provide concepts of modelling space and time together in relational data base management systems (RDBMS).

The underlying concept of RDBMS is the relational model developed by Codd (Codd, 1970) which is nowadays the most widespread data model used for database applications (Steiner, 1998). To integrate the dimension of space into RDBMS much effort had been made in the past which resulted in the Open Geospatial consortium defining standards in the field of Geographic Information Systems (GIS). The *“Simple Feature Access”* standard is an important contribution to this thesis as it describes how space is conceived into RDBMS. The standard has been widely accepted as the main vendors in the field of GIS rely on it. Databases providing support for space are called geodatabases. Their support consists of providing spatial data structures, spatial operations, spatial indexing methods and much more. It is the result of the work of the Open Geospatial consortium and the acceptance in the GI community that there is a broad consensus on how space is used within RDBMS. Moreover the concepts of integrating space into RDBMS are proved to work as they are implemented on a widespread basis. In geodatabases it is possible to ask for the location of things or the spatial relationships of objects and their neighbours.

In contrast to that there still is not a consensus on how time is integrated into RDBMS respectively geodatabases (Chomicki, 2005). In the past many concepts of temporal relational models had been developed as a basis for integrating time into RDBMS. But nowadays state of the art RDBMS lack at supporting time as they usually capture a single state of data, most often the current one (Steiner, 1998). None of the major RDBMS vendors provide a native support for time as they do for space. However database near applications which are working based on previously developed temporal concepts are available. One movement to overcome the lack of time support in geodatabases had been done by the International Standardisation Organisation (ISO). So far it developed a standard called *“ISO 19108 Geographic Information – Temporal Schema”*. The standard provides concepts needed to describe the temporal characteristics of geographic information. As this standard had identified that when conceiving time into geodatabases then there are some analogies

between space and time. Hence within this standard the concept of geometry of time is introduced which is about representing time as geometry. The important thought behind that is that a geodatabase already provides the necessary functionality when time is represented as geometry. Hence a spatial framework consisting of data structures, operations, indexing methods and much more is reused in the temporal context.

This thesis will follow the idea of representing time as geometry in order to integrate it into geodatabases. Therefore this work provides a formal basis to transform time into geometry. Based on this formal description this work provides a prototypical integration of time into a geodatabase.

The scope of this work is to overcome the lack of time support and integrate time represented as geometry into geodatabases. This is based on fundamental findings of important contributions in the field of combining space and time in databases. Worboys identified that *“much information which is referred to space is also referenced to time”* (Worboys, 1994). Moreover Peuquet pointed out that *“things change in space over time ... to exist is to have being within both space and time. ”* (Peuquet, 2002). In her work Peuquet mentioned that in human history space and time were so basic to our understanding that they were regarded to the source of our world.

Now integrating time into geodatabases provides a basis for ordering all modes of thought and belief as Peuquet (Peuquet, 2002) mentioned. It raises new perspectives and probably delivers interesting insights of data. A geodatabase providing support for time is able to answer the question where things are located and when they had happened. It is further possible to find out if things existed together because they occurred within the same period of time at the same place. How a place looked like ten days ago or how this place will look like in one year? These are the common scenarios of so called spatio-temporal issues as space and time are considered together. This work will point out that when time is represented as geometry it is possible to completely solve spatio temporal problems with the spatial framework of a geodatabase.

1.1 Hypothesis

Based on identified analogies of the two fundamental dimensions space and time it is a natural approach to represent time as geometry in order to integrate it into a geodatabase. The spatial framework a geodatabase provides allows meaningful operations applied in temporal context.

1.2 Time as Geometry

This section gives a short introduction of the main part of this thesis.

1.2.1 Theory

The main idea of this work is to model time as geometry in order to integrate time into geo data bases. The formal basis for this approach is now introduced so for a detailed description it is necessary to read chapter 7.

This work uses a discrete view of time that is absolute and not relative as defined by Einstein. Therefore the speed of time is assumed to be independent from an object's own motion. But in this work time is always relative to an arbitrary origin which marks the earliest possible date specification. Further it is assumed that time is totally ordered and that it is possible to measure the distance between two date specifications. The next assumption is that time linearly extends from the origin to the future. Therefore it is not possible that time branches in the future or that time cycles in a repetitive manner. Once an event had happened it will never happen again at that time. It is assumed that any date specification uses the Gregorian calendar to identify years, month and days. The Coordinated World Time UTC is used to identify hours, minutes and seconds. As a result any date specification used for the formal description of the approach is of the form {YYYY.MM.DD HH:MIN:SEC}. The smallest time unit used in this work is one second.

In order to transform a date specification into geometry it is necessary to transform date specifications into numbers. Therefore an interval scale based view of time is defined in chapter 7.3. The natural numbers are the underlying values of any transformed date specification. Chapter 7.4 identifies two functions which are necessary to transform date specifications into geometry and geometry back into date specifications. The functions are described on a formal basis, their implementation is provided in chapter 8.4. Finally in chapter 7.5 the formal description provides a mapping from temporal primitives to geometric primitives. Therefore an instant of time is represented as point geometry and a

Introduction

period of time is represented as simple line geometry. As a result the following figure summarises the process of transforming time into geometry.

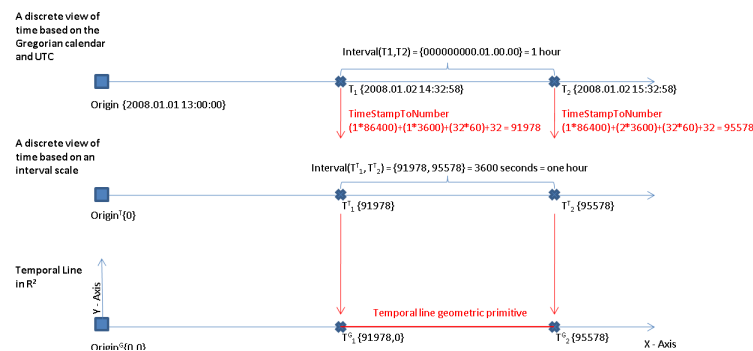


Figure 1 Transforming time into geometry

1.2.2 Methods for Implementation

The implementation of the provided formal description is described in chapter 8. It provides the necessary implementation steps to integrate time into geodatabases. Before time is converted into geometry time is mapped onto the SQL-92 data type `TIMESTAMP` which stores an anchored position in time at a granularity of seconds. Any relative temporal information is mapped onto the SQL-92 data type `INTERVAL DAY TO SECOND`. The goal of the implementation is to integrate the transformation functions as database functions which are described in chapter 8.4. Oracle Express Edition including Oracle Locator is used as a database including a spatial framework. It provides a rich spatial framework consisting of a spatial data type, spatial operations, spatial analytical methods, indexing methods and much more. It is also compliant to the Open Geospatial simple feature specification as it provides functions to transform geometry from its native representation into a well known binary or well known text representation. As the transformation functions are implemented and time therefore is represented as geometry it is possible to reuse the existing spatial framework in temporal context. Chapter 8.5 provides a mapping from temporal operations to spatial operations. For each spatial operation it is evaluated if it provides a meaningful result in temporal context. For example if one period is during another period this question is transformed into if one line geometry is inside another line geometry. Finally the power of the spatial framework applied in temporal context is shown with a bus station scenario where spatio temporal questions are provided.

Introduction

1.2.3 Data

To test the outcome of the implementation a bus scenario is provided in chapter 8.6. The goal of the bus station scenario is to provide a basis for interesting spatial, temporal but also spatio temporal questions. The following figure provides an overview of the bus station scenario.

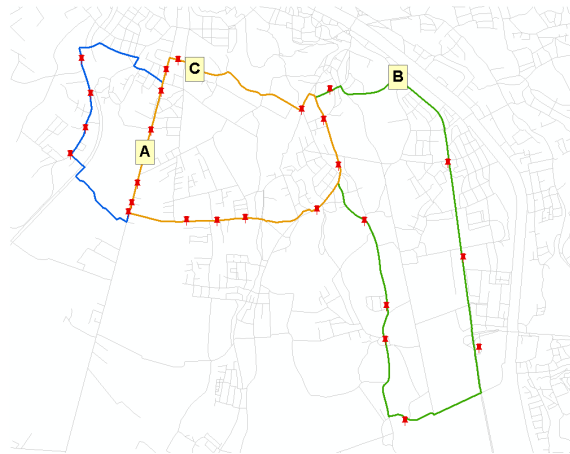


Figure 2 Bus station scenario

There are three bus lines called A,B and C. Each bus line has a number of stations where the bus stops. At each bus station there is a schedule providing the information when a bus arrives. The time a bus stops at a bus station is implemented as a period as it is close to reality that a bus takes time at the station when passengers enter or exit. For this scenario it is assumed that a bus needs 15 seconds from the arrival to the departure at a station.

In order to reuse the functionality of the spatial framework the period is transformed into a line geometry via the Time_To_Geometry function and the location of the bus station is a spatial point. As a result this work should be able to answer the following questions:

- Find those stations which are within a walking distance of 200 meters of a given location
- Find those stations at which a bus arrives or departures within a given period of time
- Find those stations within a walking distance of 500 meters where a bus stands at a given time point
- Based on a given location and a given time find those stations where someone will arrive before a bus and must no wait longer than for ten minutes at that station. A walking speed of 5km/h is assumed.

1.3 Expected Results

The expected result of this work is the integration of time into a geo database where time is represented as geometry. To accomplish this goal a formal basis for the transformation of time into geometry must be provided. A prototypical implementation within a geo database gives answer about the feasibility of the provided formal description. Finally when integration is finished it should be possible to answer the previous mentioned spatial, temporal and spatio temporal questions. Therefore the questions are translated into SQL statements where both spatial and temporal problems are solved via the spatial framework as time is represented as geometry.

1.4 Issues not covered

This work assumes time to be a finite continuum and therefore a discrete view of time is used. For the provided concept of transforming time into geometry time is interpreted to be absolute and not relative. Hence this work is based on a Newtonian view of time rather than Einstein's relativity theory. As a result of this view of time there are a number of topics not covered by this work including the hot topic are of moving objects. Güting and Schneider provide (Güting & Schneider, 2005) an important work in the subject area of moving objects. Aleshkeikh (Alesheikh) and Somayeh (Somayeh & Alesheikh) provided an appropriate entry point in the subject area.

Another issue that is not covered by this work is the visualisation of temporal information. Of course the provided examples and use cases show graphical representations of time but it is not the scope of this work to provide concepts related to the visualisation.

Real time systems are not covered by this work too. It is assumed that there is period of time that goes by from making an observation in the real world to storing that information in the database.

In the past dozens of concepts of temporal relations had been developed. It is beyond the scope of this work to evaluate and compare them in detail but they serve as an important basis for all thoughts about this work.

1.5 Audience

This thesis may be a contribution to everyone interested in modelling space and time in relational database management systems. As this work integrates time into geo databases it may also be of interest in the field of Geographic Information Systems.

1.6 Structure of the Thesis

The following figure provides information of the structure of this thesis.

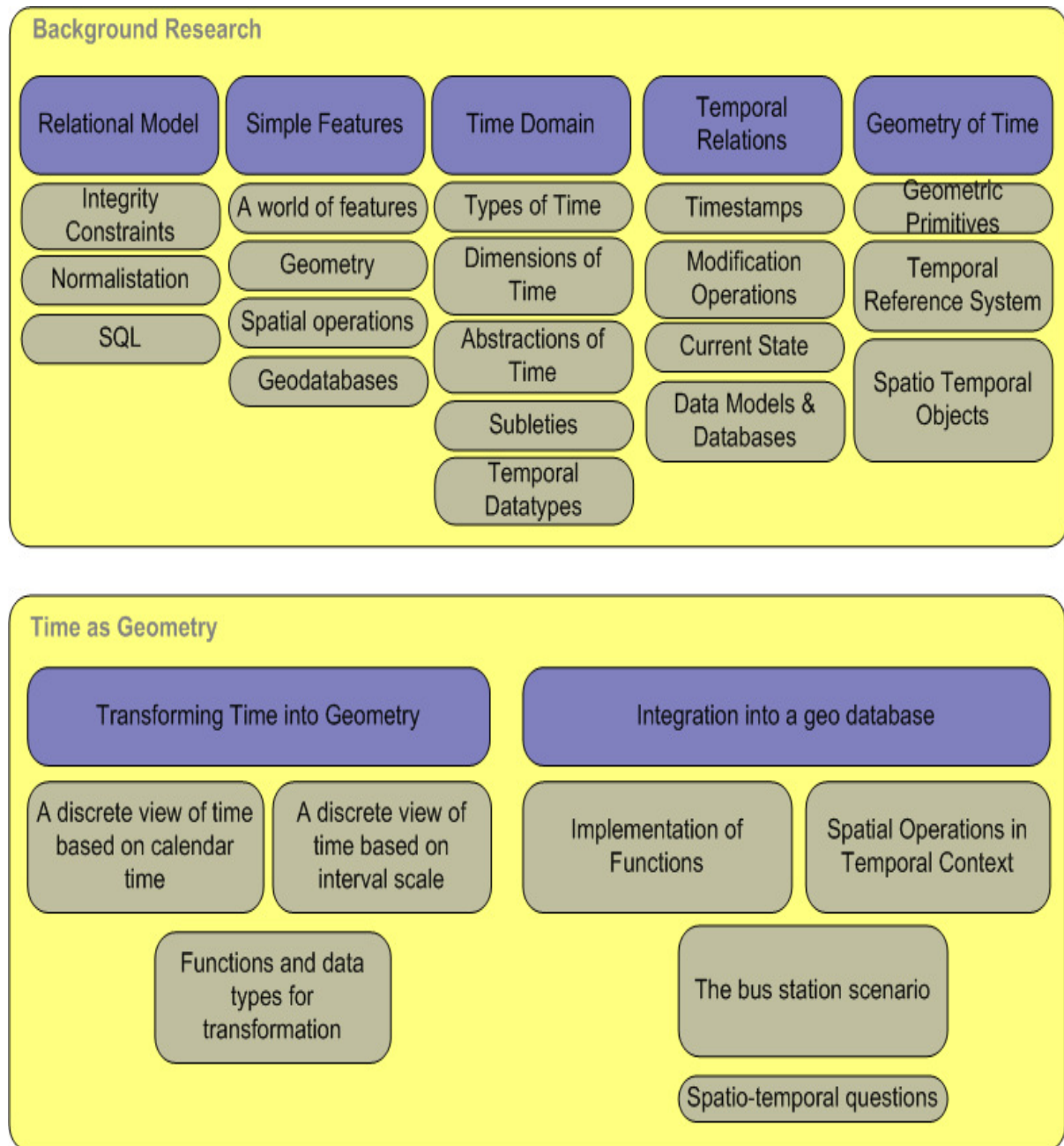


Figure 3 Structure of the thesis

This work is separated into one part providing an evaluation of relevant literature in the subject area which is called background research. The second part is the main part of this work. It is separated into a formal description for the transformation of time into geometry and a prototypical implementation of that description within a geo database. Each previously evaluated paper of the background research is more or less a contribution to the main part.

Background Research

The scope of this section is to describe and identify the main contributions to this work. This work is based on the following fundamental concepts. The first is the relational model defined by Codd (Codd, 1970) as it provides the basic modelling paradigms for nowadays state of the art Relational Database Management System (RDBMS). Based on this the second important concept is how the dimension of space is conceived into RDBMS, so called Geo Databases. The related standards of the International Standardisation Organisation (ISO) and the Open Geospatial Consortium (OGC) are evaluated. The third important concept is how the dimension of time is conceived into RDBMS. Previously developed modelling paradigms and concepts extending the relational model to the temporal domain are described. The background research is completed by the ISO Standard “ISO 19108 Geographic Information –Temporal Schema” which introduces the idea of geometry of time.

2. The Relational Model

The relational model was first introduced by Codd (Codd, 1970) in his paper “*A relational Model of Data for Large Shared Data Banks*”. Codd published follow-up papers where he brought his idea on a mathematical framework (Codd, 1972). The main idea of this model was to protect users from having to know how data is organized in the machine, hence in such a model data are described only by its natural structure (Codd, 1970). At the time of his paper there existed ordering, indexing and access path data dependencies which he identified. The scope of the relational model was to overcome these data dependencies. Nowadays the relational model is the most widespread data model used for database applications and is implemented in state of the art relational database management systems (Steiner, 1998). Jensen (Jensen, Soo, & Snodgrass, 1994) pointed out the success of the relational model is related to its simplicity.

The relational model consists of a set of relations.

“Given sets S_1, S_2, \dots, S_n , (not necessarily distinct), R is a relation on these n sets if it is a set of n -tuples each of which has its first element from S_1 , its second element from S_2 , and so on. More concisely, R is a subset of the Cartesian product $S_1 \times S_2 \times \dots \times S_n$.” (Codd, 1970).

To better understand this mathematical formal definition a more general definition of a relation is needed. Steiner defined a relation in its simplest form as the Cartesian product of its domains (Steiner, 1998). From Codd’s perspective a relation is the provided data structure in the relational model to store information about real world phenomena. A relation consists of domains which describe the properties of such phenomena. As Codd mentioned a relation can be viewed as a two dimensional table, where a row in that table is called a tuple (Codd, 1970).

BUS_STATION	ARRIVAL	DEPARTURE
FIRST_BUS_STATION	01.01.2008 07:40	01.01.2008 07:42
SECOND_BUS_STATION	01.01.2008 08:30	01.01.2008 08:32

Table 1 A relation viewed as a two dimensional table

2.1 Integrity constraints

To apply operations and integrity constraints on relations Codd (Codd, 1970) introduced the concepts of primary key and foreign key. He defined a primary key as the one domain or combination of domains which uniquely identify each tuple of a relation. Hence, a primary key is non redundant. To enable cross references between relations foreign keys are defined. Codd defined a foreign key as the one domain or combination of domains which is not the primary key of a relation but the primary key of another relation. In his definition it is not excluded that the two involved relations of a cross reference are the same.

2.2 Normalisation

Another concept introduced by Codd is called normalisation. To describe this concept it first must be distinguished between simple domains and non simple domains. Simple domains are domains whose elements are atomic they only consist of scalar values. Non simple domains consist of elements whose values include for example relations (Codd, 1970). The process of eliminating such non simple domains from a relation is called normalisation. Codd described a common way of this process by excluding non simple domains from the original relation and redefine them as new relations. They are related to their original relation through the concept of a foreign key. As a result a normalised table only consists of simple domains and can again be viewed as a two dimensional table.

Codd further introduced permutation, projection, join and other operations applied to relations. For example a join operation is understood as setting relations in relation to each other through a domain they have in common (Codd, 1970). Moreover he mentioned as time progresses a relation may be subject to insertion of additional tuples. Existing tuples may be deleted or altered.

All the basic ideas and concepts of Codd are implemented in nowadays state of the art relational database management systems. The implemented counterpart of a relation is called a database table. Domains of relations are implemented as attributes of such tables. The concepts of primary key, foreign key are implemented as integrity constraints. Normalisation of relations as an idea has become accepted although there are reasons why the process of normalisation may be relaxed. The operations applied to relations are implemented too.

2.3 Structured Query Language SQL

What has come along with all this is the definition of a computer language which makes it possible to use the relational model in all its aspects. Such a standard is the Structured Query Language (SQL). Originally SQL had been developed by Donald D. Chamberlin and Raymond F. Boyce under the name “SEQUEL”. The scope of it was to access data of relations based on the relational model of Codd (Chamberlin & Boyce, 1974). Later on this language had been adopted as a standard by the International Standardisation Organisation (ISO). In state of the art RDBMS the standard SQL-92 of the year 1992 is implemented. Following standards like SQL:1999 or SQL:2003 are not implemented on a widespread basis (Snodgrass, 2000). The SQL in its nowadays implemented form provides a means of using the concepts related to Codd on a computational basis.

2.4 Contribution

The relational model and the SQL standard are both main contributions to this work. They provided the basic concepts behind state of the art database management systems. Steiner (Steiner, 1998) summarises that a data model consists of data structures, data operations and integrity constraints. Data structures provide a means to store data, operations applied to them provide a means to query and manipulate this data. Finally integrity constraints provide a means to keep data integer and avoid redundancy. As it is the scope of this work to model time into geo databases the above presented concepts need temporal and spatial counterparts. Space and time had previously considered separately to fit into the relational model. On the one hand geo databases provide data structures and operations dealing with spatial data types. On the other hand temporal data models consist of temporal data structures, temporal operations and temporal integrity constraints. How space and time are conceived into the relational model is discussed in the following chapters. Within the main part of this work concepts are provided how the two dimensions are together considered into the relational model.

3. Simple Features

The goal of this chapter is to provide an overview of state of the art standards and techniques to use spatial information inside data bases, so called geo data bases.

The Open Geospatial Consortium (OGC, available at <http://www.opengeospatial.org/>) is a non profit organisation which was founded in 1994. The major task of the OGC consists in developing open standards and specifications in the field of GI-Systems. Their developed standards are separated into Abstract Specifications and Implementation Specifications. The former type of standard is written at an abstract level and often in relation with an according standard of the International Standardisation Organisation (ISO). The latter type of standard is considered for practical use and is mainly based on Unified Modelling Language (UML) models. Therefore such a standard is independent from specific software architecture.

For this thesis there are two abstract specifications important. The first is called “Topic 2 – Geometry” which is also an ISO standard (ISO, 2003) called “ISO 19107 Geographic Information – Spatial Schema”. In this specification the different types of geometry are described in detail using UML models. Figure 4 shows the basic geometry classes.

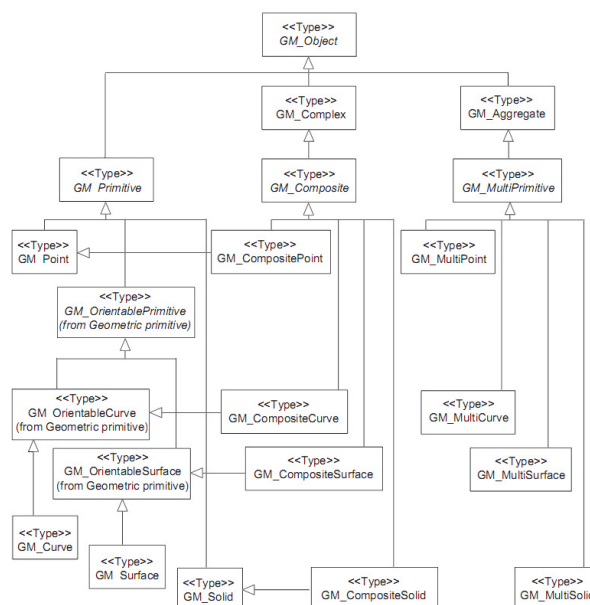


Figure 4 Geometry basic classes (ISO, 2003)

3.1 Definitions

To understand the different types of geometry this standard provides important definitions on the following terms:

Position is *“described by a single set of coordinates within a coordinate reference system”* (ISO, 2003).

Interior is the *“set of all direct positions that are on a geometric object but which are not on its boundary”* (ISO, 2003).

Boundary is the *“set that represents the limit of an entity”* (ISO, 2003).

Exterior is the *“difference between the universe and the closure”* (ISO, 2003).

With these definitions in mind it is possible to define the geometric primitives which are relevant for this thesis.

Point is a *“0-dimensional geometric primitive which represents a position”* (ISO, 2003). Further it states that the boundary of a point is the empty set the interior of a point is its position.

Curve is a *“1-dimensional geometric primitive representing the continuous image of a line”* (ISO, 2003). The boundary of a curve is defined as the set of points at either ends of the curve. As a curve has a direction the first point is called start point and the last point is called end point. The interior of a curve consists of all points between the start and end points based on an interpolation. A common method is linear interpolation, where the start and end points are connected by a straight line.

There exist definitions on surface and many other types of geometry too, but they are not relevant to this work.

3.2 Features

Based on these important definitions on types of geometry the second abstract specification important for this thesis is called “Topic 5 – Features”. Within this specification it is defined how real world phenomena are modelled into a world of features. *“The fundamental unit of geographic information is called a feature”* (OpenGIS_Consortium, 1999).

3.3 Geometries

In addition to the above mentioned abstract specifications one implementation specification is an important contribution to this work as well. It is separated into two parts, the first part is called “Simple Feature Access – Part I: Common Architecture” and it defines the different types of geometries which are considered as a basis for simple features. Figure 5 shows an UML model of the considered types of geometry.

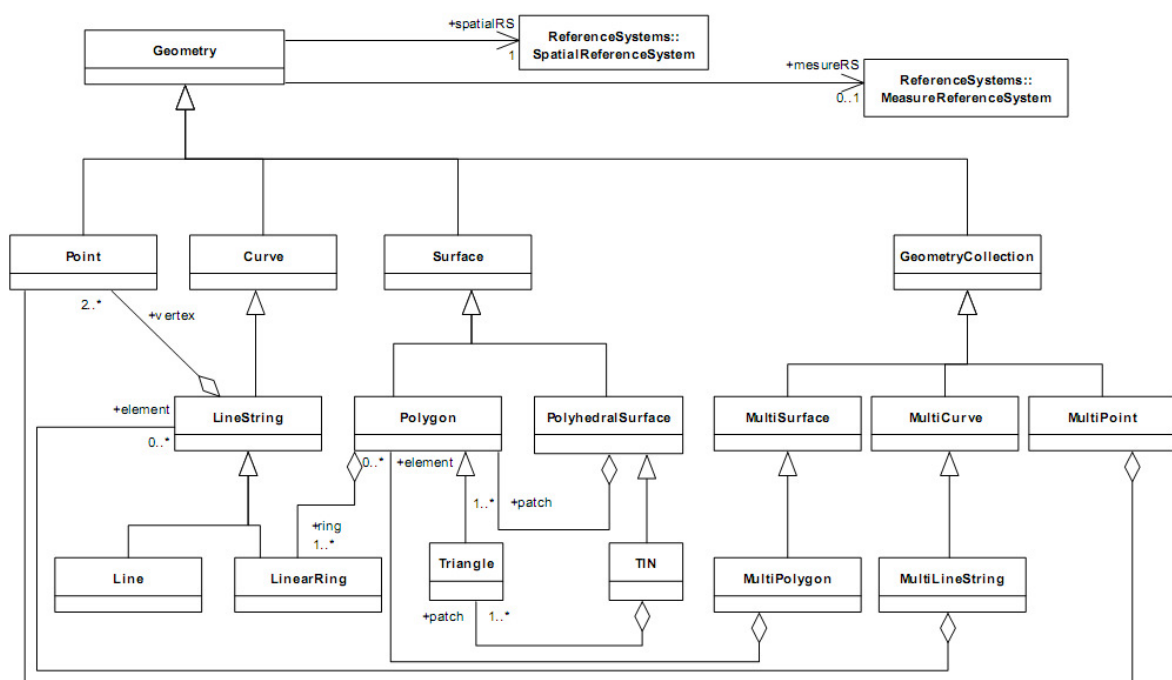


Figure 5 Geometry class hierarchy (OpenGIS_Consortium, 2006a)

3.4 Spatial Operations

Further this standard defines the properties for each type of geometry as well as relational operations applied to them. For example the well known Dimensionality Extended Nine-Intersection Matrix (DE-9IM) describing the relations between two geometries is provided (OpenGIS_Consortium, 2006a, S. 35). This matrix is based on the assumption that geometric primitives have an interior, a boundary and an exterior.

	Interior	Boundary	Exterior
Interior	$\dim(I(a) \cap I(b))$	$\dim(I(a) \cap B(b))$	$\dim(I(a) \cap E(b))$
Boundary	$\dim(B(a) \cap I(b))$	$\dim(B(a) \cap B(b))$	$\dim(B(a) \cap E(b))$
Exterior	$\dim(E(a) \cap I(b))$	$\dim(E(a) \cap B(b))$	$\dim(E(a) \cap E(b))$

Figure 6 DE-9IM Matrix (OpenGIS_Consortium, 2006a)

The second part of the simple feature specification is called “Simple Feature Access – Part II: SQL Option” and it defines how features (remember the above mentioned standard “Topic 5 – Features”) are conceived into the relational model defined by Codd (Codd, 1970). Hence the scope of this part is to describe a SQL schema that supports storage, retrieval, query and update of collections of features (OpenGIS_Consortium, 2006b). Besides the SQL implementation option there are other options like the implementation option based on the Component Object Model (COM). For this work only the SQL option is considered.

3.5 Geodatabases

In the SQL specification a table is called feature table within a geodatabase. Columns of such a table are whether spatial or non spatial and describe the attributes of a feature which itself is represented as a row of a feature table. Non spatial attributes are derived from the SQL implementation and spatial attributes are derived from the OGC Implementations Specification. The spatial attributes may be implemented as user defined data types (UDT) (OpenGIS_Consortium, 2006b). Further this standard identifies different SQL operations applied to feature tables as well as how the different types of geometry are stored in a feature table.

4. About Time

The scope of this chapter is to give an introduction to the time domain. General concepts, definitions as well as methodologies from the main contributors of the subject area are identified. The chapter starts with abstract concepts related to the time domain like the taxonomic model developed by Frank (Frank, Egenhofer, & Colledge, 1998) which identified different types of time. Further different views of time and the topic how time can be conceived into computer systems are provided. The chapter ends up with the definitions of time related data types within relational database managements systems.

4.1 Types of Time

What is time and how do we understand time? Such questions are almost very philosophical and cannot be answered by this work. To give an appropriate introduction Frank (Frank, Egenhofer, & Colledge, 1998) had developed a taxonomic model which identified different types of time. Of course in reality there is only one time, the different types of time refer to our conception of time and how time can be conceived into computer systems. He had also identified that on different types of time different operations may be applied. Figure 7 shows the different types of time as identified by Frank (Frank, Egenhofer, & Colledge, 1998).

		total order	partial order	branching	multiple perspective
Linear	Ordinal	<i>Single Experience</i>	<i>Multiple Experiences</i>	<i>Branching Time</i>	<i>Time With Multiple Perspectives</i>
	Continuous	<i>Continuous Time</i>			
Cyclic	Ordinal	<i>Cyclic Time</i>			
	Continuous				

Figure 7 Types of Time (Frank, Egenhofer, & Colledge, 1998)

The first consideration is whether an event occurs at an instant or it has duration over a period of time. Instants of time are absolute and represent a point in time. Once this time point had occurred it is forever in the past. An example for a time instant is January 1st. In contrast, a time interval represents duration between two time instants. Intervals are called

About Time

periods if they are considered to be absolute. For example the period between 1st January and 7th January is an absolute period defined by its two delimiting time instants. If intervals only represent duration then they are considered to be relative. For example the expression “two weeks” is considered as a relative interval. The according data types to instants, intervals and periods are discussed in chapter 4.5.

The second consideration is whether time is interpreted linear or cyclic. To use ordinal or interval scale values is the third consideration. Moreover an important role plays the ordering of events. As Frank mentioned linear time may have total order, a partial order or a branching order. In a total order system one event occurs after the other. Linear time extends from the past to the present throughout the future. From Frank's perspective a partial order exists if the observations of two observers are considered together but each observation do not know its temporal relation to the other one.

Branching time is defined as linear time with different future states. One observation may have multiple states for the same period of time. This type of time is useful for planning scenarios or predictive tasks. Figure 8 shows how time branches from the past to the future.

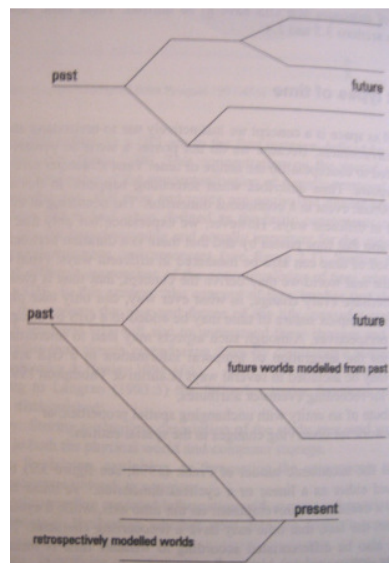


Figure 8 Branching Time (Ott & Swiaczny, 2001)

About Time

When time is interpreted as cyclic then it has a repetitive pattern. For example many cultural or biological processes, like the four seasons or breeding seasons, occur on a cycling basis (Frank, Egenhofer, & Colledge, 1998). One character of cycling time is that an ordering of events is meaningless as for example morning is before but also after evening.

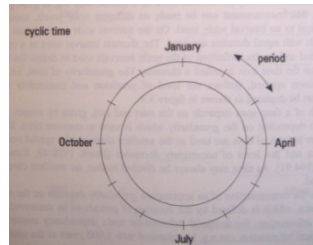


Figure 9 Cycling Time (Ott & Swiaczny, 2001)

Lastly Frank identified multiple perspectives of time which is often referred to dimensions of time. This topic is discussed in chapter 4.3.

As a summary it should be mentioned that many of the presented considerations can be combined. A total order, a partial order or a branching order are the possibilities if linear time is used. Further the scale of measurement can be considered too. Frank also introduced the concept of tolerance. He stated that the temporal information of observations may include errors of measurement. To better compare temporal information a tolerance may be used to define at what precision to values are considered equal. This concept is related to the concept of using a discrete view of time which is discussed in chapter 4.4.

4.2 Abstractions of Time

From the beginning of research the view of time either as absolute or relative produced much discussion. Peuquet (Peuquet, 2002) mentioned that from ancient history until Newton time always had been considered to be absolute. But with the first relative theory of Einstein the understanding changed to a relative view of time. It states that the relativity of time is related to the movement of objects. In the context of this work the movement of objects is the topic of moving objects which is not covered within this work as mentioned in chapter 1.4. Therefore this work interprets time to be absolute rather than to be relative. Chapter 7.2 provides the definition of an absolute view of time as a basis for the implementation described later in this work.

4.3 Dimensions of Time

In literature there is much discussion about dimensions of time which Frank (Frank, Egenhofer, & Colledge, 1998) had identified as multiple perspectives of time. In general there are two dimensions, valid time and transaction time. Any other dimension is defined as user defined time (Snodgrass, 1992).

4.3.1 Valid Time

Valid time is defined as the *“time when a fact is true in the abstracted reality.”* (ISO, 2002). It defines the time of an observation that had existed in the real world. The valid time dimension is also called the valid time period which is discussed in later chapters. Valid time often describes the lifetime of real world phenomena. As Jensen stated all facts have a valid time by definition (Jensen, 2000). One important fact is that in the context of database systems valid time must always be provided by the user it can never be provided by the database itself (Steiner, 1998).

4.3.2 Transaction Time

Transaction time is defined as the *“time when a fact is current in a database and may be retrieved.”* (ISO, 2002). It is the time the recorded fact of the reality is available in the database. Hence this dimension of time does not say anything about when the recorded fact was valid in the real world. Jensen stated that the period from insertion to deletion is the duration of transaction time (Jensen, 2000). Steiner (Steiner, 1998) mentioned that there is often a delay between when a fact is observed in reality and when a fact is true in the database. Hence transaction time is an important dimension of time as it provides the possibility to query database changes. Transaction time must always be provided by the database system, it cannot be provided by the user.

4.3.3 User Defined Time

Any other dimension of time except from valid time and transaction time is referred to user defined time. A real world object may have zero, one or more user defined time dimensions. For example the periods of legal aspects can be modelled as user defined time.

About Time

Valid time and transaction time can be viewed as two orthogonal axes (Ott & Swiaczny, 2001). The following figure shows the relations between valid time and transaction time.

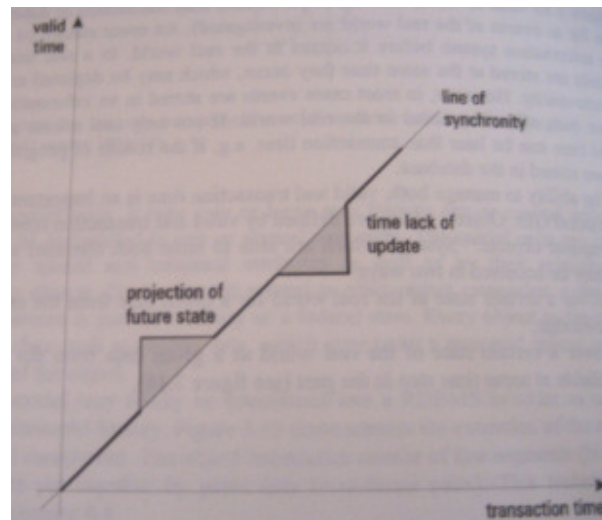


Figure 10 Valid Time and Transaction Time (Ott & Swiaczny, 2001)

Valid time and transaction time are synchronic as long as the period of valid time overlaps the period of transaction time. In such a case the observation of the real world is valid while it is also stored within the database. In Figure 10 this relation between valid time and transaction time is called “line of synchrony”. The time delay between the occurrence in the real world and the storage in the database is called “time lack of update”. When an event is stored in the database before it happens in the real world it is called a “projection of future state”. Further Ott and Swiaczny (Ott & Swiaczny, 2001) provided an overview of possible queries if both dimension are supported. These queries are shown in Figure 11.

World = Database	What do we know of the world on this date, as of the same date?
World < Database	To the best of our knowledge as of a given date, how did the world appear on a given past date?
World > Database	Based on our knowledge as of a given date, how would we expect the world to appear at a later date?

Figure 11 Queries on both dimensions of time (Ott & Swiaczny, 2001, adopted from Langran)

4.4 Discrete vs. Continuous

There are two possible abstractions of time called continuous view and discrete view. Frank (Frank, Egenhofer, & Colledge, 1998) mentioned that in a continuous view time is dense and therefore between any two time instants another one can be inserted. Thus such an approach is isomorphic to the real numbers (Steiner, 1998). In contrast in a discrete view temporal information is isomorphic to the natural numbers. As a result between any two time instants no other time instant can be inserted. In her work Peuquet (Peuquet, 2002) pointed out the important difference between both views. She stated that the continuous view focuses on space and time as the subject matter. In such a view time and space are the two important subjects of interest and everything that exists within these subjects is just derived information. In contrast in a discrete view objects are the subject matter (Peuquet, 2002) and the derived information is about space and time.

Roosman (Roosmann, Busch, Gorczyk, & Mauersberger, 2003) and others said in their work that discrete time is used if time is measured at certain time points or time intervals and the variation is discontinuous between them. In contrast a continuous time may be more appropriate to describe processes as it probably will exist a theory to interpolate a value for every time point on the continuous time axis. As a continuous view of time is more appropriate to describe processes and therefore is more appropriate to describe the movement of objects this view tends to apply to the topic of moving objects which is not covered by this work. Moreover Steiner stated that as using time in relational database management systems is build on a discrete computing device it also may be used in a discrete manner (Steiner, 1998). Clifford and Tansel mentioned that *"... from a practical standpoint the natural numbers seem a more useful candidate for modelling properties of database time."* (Clifford & Tansel, 1985). Further they mentioned in their work that the discrete view is more appropriate as a recording instrument has at a best a finite quantum. Snodgrass (Snodgrass, Temporal Databases, 1992) provides several reasons why a discrete model is more appropriate than a continuous model in the context of relational database management systems. For example he stated that observations are already measured on a discrete basis. Moreover we all tend to use time on a more discrete view as we are using clocks and calendars.

In a discrete view time is abstracted as a finite dimension with equal sized time instants (Steiner, 1998). Clifford and Tansel (Clifford & Tansel, 1985) defined the discrete view as a

About Time

set of equal distanced points, these discrete time points are called chronos. Between two consecutive time points there is duration of one time unit. The points are totally ordered and relative to an origin.

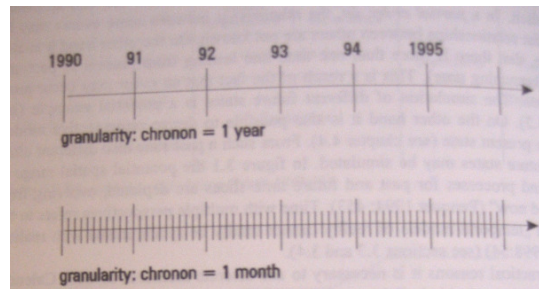


Figure 12 Chronos with different granularities (Ott & Swiaczny, 2001)

Chronos are the main concept when modelling time discrete “... *the sequence of chronons may be thought of as representing a portioning of the real time line into equal-sized, indivisible segments ...*” (Roosmann, Busch, Gorczyk, & Mauersberger, 2003). One chronon itself has no duration, but the temporal distance between two consecutive chronos has duration. The ISO Standard 19108 introduced the Gregorian calendar as a prominent example of a discrete view of time. The smallest time unit is one day. Within one day humans use clock time to discrete time. Therefore clock time in combination with the Gregorian calendar is a very popular discrete view of time and is the basis for this work.

Frank introduced that a discrete view can be built with varying degrees of tolerance but also with different levels of granularity. The granularity is the smallest unit used in a discrete view, whereas the tolerance is equal to or greater than the granularity. A discrete model can have a granularity of one day, but for specific applications a tolerance of one week may be preferred. Therefore two time instants are considered equal if they fall into the same level of tolerance. For example consider a granularity of one day. Now compare December 31st 2007 with January 1st 2008 at level of granularity they are not equal. If a tolerance of one year is used they are still not equal but for example if a tolerance of one week is used they may be considered equal.

4.5 Temporal Data types

Temporal values are the core of temporal applications, they are the stuff of which temporal applications are made as Snodgrass (Snodgrass, 2000) mentioned in his work. Data types are needed to model time in relational database management systems. In literature there are three main types defined: instants, intervals and periods. These types had been defined by the International Standardisation Organisation (ISO) within the standard “ISO 19108 Geographic Information – Temporal Schema” which is an important contribution to this work.

4.5.1 Instants

“An instant is an anchored location on the time line” (Snodgrass, 2000). In general an instant represents a point in time. Once it has occurred it is forever in the past and will not occur again (Snodgrass, 2000). Within the SQL-92 standard there are two different data type definitions for instants. The first data type is called “DATE”. According to the definition a value of data type date includes calendar date as well as clock time. The default calendar is the Gregorian calendar with UTC as clock time. The clock time is modelled with a precision of a second. The newer data type for instants is called “TIMESTAMP”. Timestamps are able to represent clock time up to a precision of nanoseconds. Further this data type provides three possibilities of time zone support. First is no support only UTC time is stored. The second possibility is to store the time zone offset from UTC in the temporal values. This approach is called timestamp with time zone. The third possibility is called timestamp with local time zone. For this approach the time zone offset is not stored in the values but values are always represented in the session time zone of the user.

The ISO standard (ISO, ISO 19108 Geographic information - Temporal schema (draft version), 2002) defines an instant as a *“0-dimensional geometric primitive representing position in time.”* (ISO, 2002). This geometric view of a time instant is considered as one of the main ideas of this work and described in chapter 7.5.1.

4.5.2 Intervals

“An interval is an unanchored contiguous portion of the time line” (Snodgrass, 2000). Intervals are relative they represent the temporal distance respectively the duration between two instants (Snodgrass, 2000). For example the temporal information “3 weeks” is an interval. As instants also intervals are defined within the SQL-92 standard. From a general

perspective they are often mixed up with periods which are described in the next chapter. There are two different types of intervals defined. The first type of interval is called “year to month” interval. The second interval type is called “day to second” interval (Snodgrass, 2000). This distinction is a result of the unsteady character of time. For example a year consists of 12 month this is always the same. But a year either consists of 365 days or in the case of a leap year of 366 days. Hence it is not possible to define an interval type of “year to day” as the number of days varies. The same applies to months. Therefore the separation into two types of intervals had been done. The “year to month” interval provides the possibility to store intervals from years to months. The interval type “day to second” represents intervals from days to seconds.

4.5.3 Periods

Periods are not defined in the SQL-92 standard they are part of the SQL 3 standard. At the moment they are not implemented within a state of the art relational database management system. Snodgrass (Snodgrass, 2000) defines a period as “an anchored duration of the time line”. Further he stated that they may had not been included in state of the art RDBMS as they can be implemented as a pair of two time instants. To express the time a fact is valid periods are used. Periods are most often interpreted as closed-open. This means that the last time instant of the period is not included. In contrast if a closed-closed approach is used then the last time instant is included. There are less prominent options called open-open and open-closed (Snodgrass, 2000).

The lack of support of a period data type in current databases is one of the main motivations of this work. Further the ISO standard (ISO, ISO 19108 Geographic information - Temporal schema (draft version), 2002) defines a period as a *“one-dimensional geometric primitive representing extent in time.”* (ISO, 2002). This geometric view of a period is considered as one of the main ideas of this work and described in chapter 7.5.2.

4.6 Subtleties

The following sub chapters provide a short description of well known subtleties. Snodgrass (Snodgrass, 2000) provided an extensive description of time related problems in his work.

4.6.1 Leap Year

The first subtlety is related to leap years. As it is strictly defined when a year is a leap year they are not a general problem. But as you cannot assume a year to be 365 days long it was not possible in the past to define only one interval data type. As discussed in chapter 4.5.2 there are “year to month” and “day to second” types of intervals. Further interval calculations which include February 29th must always carefully considered. Computer systems in general do not have a problem to handle the leap year topic but the ones who are interpreting the output from computer systems must always be aware of the impact of interval calculations in leap years.

4.6.2 Leap Seconds

Leap seconds are introduced by a committee (Snodgrass, 2000) with the scope to adjust the atom clock to the astronomic clock. The main problem with leap seconds is that their occurrence is agreed by a committee and is not strictly defined as leap years. This committee agrees on a date when the leap second should be implemented. Most often they are implementing it in the last minute of year which then consists of 61 seconds. Leap seconds cannot be considered in relational database management systems as their implementation date is not predictable.

4.6.3 Daylight Saving Time

The existence of daylight saving time brings along some heavy subtleties. The first impact to consider is that a day cannot be assumed to last for 24 hours. In October the day where daylight saving time comes into play lasts for 25 hours as the interval between 2 pm and 3 pm is repeated. The according day in March lasts only for 23 hours as the interval between 2 pm and 3 pm does not exist. The next problem is related to the periods where time changes. As the period in October repeats what this does mean to things happened in that period. The period between 2pm and 3pm in March does not exist therefore a data base system must not allow to store temporal information within this period.

5. Temporal Relations

This chapter provides an overview of important concepts of integration time into RDBMS.

5.1 Timestamps

Timestamps are the main method to implement the valid time dimension on database level. Any other dimension of time like transaction time also relies on this methodology. Time stamping data is common to all previously developed temporal data models but there are important differences in considering what is time stamped within a database and how data is time stamped. On the one hand there are concepts time stamping on tuple level, on the other hand there are concepts time stamping on attribute level. This is either implemented through a pair of time instants delimiting a period or as a single time instant.

5.1.1 Attribute Level

Concepts allowing attributes being time stamped have the advantage that not only for the tuple but also for a particular attribute the valid time period is provided. This is implemented by adding two timestamp attributes for each attribute of interest of a temporal relation. According to Kaiser (Kaiser, 1998) one major drawback is that an attribute now consists of a triplet of attributes. Such a triplet consists of the attribute itself and two time stamp values defining its valid time period. In Codd's original perspective (Codd, 1970) such a relation consisting of triplets has non atomic attribute values. Hence, first normal form is already violated through this fact. Concepts like classification of attributes (Kaiser, 1998), or temporal partitioning (Snodgrass, 2000, S. 206) try to overcome this problem by grouping attribute triplets together into new relations and cross reference them via a foreign key. Kaiser (Kaiser, 1998) classified attributes into time stamp attributes, time independent attributes and time dependent attributes. On the basis of this classification he divided a relation into one relation for each class of attributes. As result the original relation is separated into one including the primary key and the valid time period and other relations containing the same classes of attributes. They are all cross referenced to the original table via foreign key concepts. In the strict case each attribute with its related time stamps must be implemented as a relation and cross referenced via foreign key concepts. In practice it would be an enormous effort to implement a model where for each original attribute a relation is maintained.

5.1.2 Tuple Level

Another approach is to time stamp the valid time period on tuple level. This is implemented by adding two time stamp values only to the relation and not to an attribute. Hence, each row of database table now has the valid time period defined. As a result all the attributes of a row are only valid within this period. Hence it is not possible that a particular attribute has a different valid time period. The advantage of this approach is that attributes remain atomic, so first normal form is not violated. The disadvantage is that such a concept generates vertical anomalies (Steiner, 1998) as the primary key is not unique anymore. The challenge is to find new uniqueness concepts which take the valid time period on row level into account. The concepts are described in chapter 5.3.2. Other concepts like coalescing (Snodgrass, 2000, S. 159) which aim at removing unnecessary duplicates (duplicates with the same values, except the timestamp values) are not without controversy in literature. This work will further use time stamping on tuple level as the preferred concept.

5.1.3 Interval representation

According to Clifford and Tansel (Clifford & Tansel, 1985) there are two different concepts to define the valid time period. The first approach is an interval representation, where the interval is defined either as two timestamps or as a period data type. The disadvantage of the interval representation is that without further constraint approaches it is possible to define overlapping periods. Moreover workarounds are needed to define that the valid time period is valid until now. Such workarounds are described in chapter 5.4. When interval encoding is used it must be considered whether to use closed-closed or closed-open intervals.

5.1.4 Point representation

The second approach mentioned by Clifford and Tansel (Clifford & Tansel, 1985) is called point representation. The start of the valid time period is implemented by one timestamp on the tuple. The end of the period is defined by the timestamp of the next (chronological) tuple. The disadvantages of the point representation are that the valid time period is only implicit defined on a tuple as the end of the period is only available through the chronological next tuple. Hence very complex database queries with sub queries are required for answering simple temporal questions.

5.2 Modifying Temporal Relations

Inserts, updates and deletes are the three important manipulation operations which can be applied to database tables. Insert operations are adding new rows to tables, update operations are modifying attributes of existing rows and delete operations are deleting existing rows. In the case of temporal relations each operation has a temporal counterpart. In his work Chomicki (Chomicki, 2005) pointed out that at least from an abstract point of view there are no differences between standard operations and their temporal counterparts. In practice the SQL statements for insert, update and delete operations on temporal relations are getting quite complex. In temporal relations there are some issues to consider which Snodgrass (Snodgrass, 2000) described in his work. He developed a classification of modifying operations where he distinguishes between current modifications, sequenced modifications and non sequenced modifications. For this work only sequenced modifications are of interest as they apply to the past, present and future (Snodgrass, 2000). It is assumed that the valid time dimension is implemented via two time stamps on tuple level. Sequenced modification operations are further assumed to comply with the SQL-92 standard. Each operation is described in the following chapters.

5.2.1 Insert

In a sequenced insertion the user or the application must provide the valid time period. The challenge of the temporal insertion statement is to maintain integrity. Hence the primary key as well as foreign keys must not be violated. Snodgrass (Snodgrass, 2000, S. 189) had defined the following steps in the case of a sequenced insert. The first step of Snodgrass recommendation is to maintain entity integrity. To accomplish this check the SQL statement is expanded to check whether there are rows with the same primary key overlapping with the provided valid time period. The second step is to maintain referential integrity. Therefore the statement is expanded to check whether there exist rows in the referenced table which cover the provided valid time period with their own valid time period. The third step further checks if the periods of validity of the rows of the referenced table have no gaps. Snodgrass stated that it is up to the application whether the integrity is directly checked within the insert statement or through database triggers or assertions.

5.2.2 Update

A sequenced update is the temporal counterpart of the standard update operation. It is assumed that the application provides the valid time period which applies to the row after the update operation. In the following discussion this period is called period of applicability (PA). In contrast the valid time period (PV) is the period before the sequenced update. Snodgrass (Snodgrass, 2000) identified four cases of how the PA and the PV have to be considered in the case of an update operation.

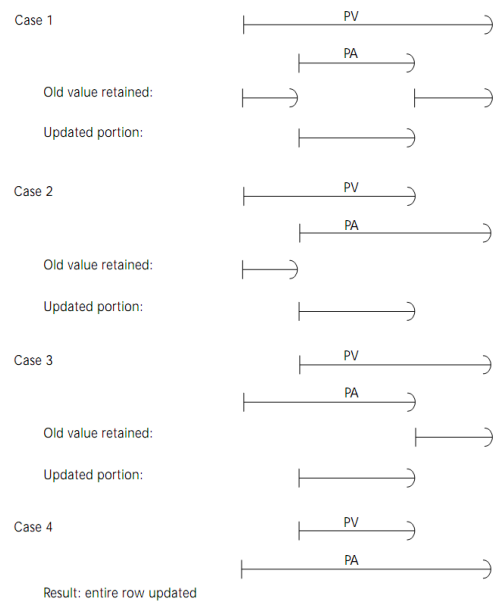


Figure 13 Temporal Update Cases (Snodgrass, 2000, S. 196)

In the first case the PA is within the PV. The original PV is separated into three parts. The first part extends from the original start of the PV to the start of the PA. The attribute values of the original row remain for this part. The second part is the PA itself. The attribute values of this part are those which are affected by the update statement and are therefore updated. The third part extends from the end of PA to the original end of the PV. The attribute values of the original row remain for this part. A result of case 1 is that a sequenced update statement affecting one row causes three new rows to be inserted. The old row is deleted but their attribute values are remained. Case 2 and case 3 are considered in a similar manner. In case 4 the PA extends the whole PV and therefore the entire row is updated.

5.2.3 Delete

A sequenced deletion is the temporal counterpart of the standard delete operation. It is assumed that the application provides the valid time period which defines the period that the affected rows should not be valid after the statement. In the following discussion this period is called period of applicability (PA). In contrast the valid time period (PV) is the period before the sequenced deletion. Snodgrass (Snodgrass, 2000) identified four cases of how the PA and the PV have to be considered in the case of a delete operation.

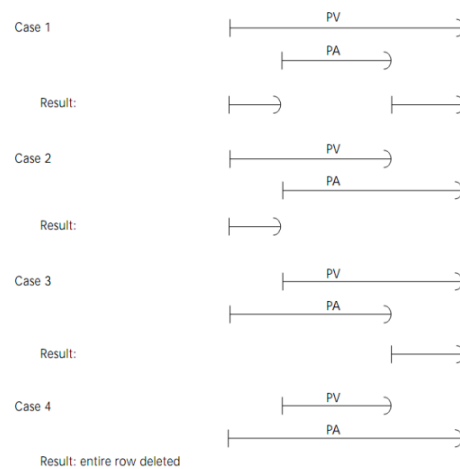


Figure 14 Cases of Temporal Deletions (Snodgrass, 2000, S. 191)

In the first case the PA is within the PV. The original PV is separated into three parts. The first part extends from the original start of the PV to the start of the PA. The attribute values of the original row remain for this part. The second part is the PA itself. The attribute values of this part are those which are affected by the delete statement and are therefore not valid anymore. The third part extends from the end of PA to the original end of the PV. The attribute values of the original row remain for this part. A result of case 1 is that a sequenced delete statement affecting one row causes two new rows to be inserted but they are not valid for the PA of the delete statement. Case 2 and case 3 are considered in a similar manner. In case 4 the PA extends the whole PV and therefore the entire row is not valid anymore.

5.3 Normalisation

Normalisation is a major concept in database theory. Codd (Codd, 1970) introduced normalisation with its relational model and described it as the process of eliminating non simple domains from a relation. As with other concepts in the context of relations normalisation has a temporal counterpart as well.

“There is also a need for temporal normal forms and underlying concepts that may serve as important guidelines during temporal database design.” (Jensen, Snodgrass, & Soo, 1996, S. 1). In their work Jensen, Snodgrass and Soo (Jensen, Snodgrass, & Soo, 1996) provided a description of previously developed temporal normal forms and related concepts like functional dependency. What they pointed out in their work is that a major disadvantage of the reviewed normal forms is that they are all related to different underlying temporal data models. Further they pointed out that the models do not satisfy all requirements of the conventional normal forms. Moreover they are not following the same goals as their conventional counterparts.

In a later work Jensen mentioned that *“the conventional normalization concepts are not applicable to temporal relational data models because these models employ relational structures different from conventional relations.”* (Jensen, 2000). But Jensen and Snodgrass (Jensen, Snodgrass, & Soo, 1996) identified two issues that at least should be satisfied by temporal normalisation. First, as the process of normalisation consists of decomposing relations with non simple domains into relations with simple domains this decomposition should be lossless. Hence the contents of the original relation should be available from the decomposed relations via natural join operations. Second, the process of decomposition should preserve existing dependencies.

However, normalisation has become a common methodology during database design and therefore it is worth considering it in the case of temporal relations. Normalised temporal relations will benefit from the framework defined for conventional relations. Therefore storage structures, query evaluation techniques and further concepts will remain the same as in the relational model (Jensen, Soo, & Snodgrass, 1994).

5.3.1 Functional Dependencies

Functional dependencies are a concept related to normalisation. They describe dependencies between domains respectively attributes of a relation. Further they are a basis for normalisation and for the definition of the primary key of a relation. As a primary key is the set of domains which uniquely identify a tuple in a relation all other domains of that relation are fully functional dependent on that primary key. Armstrong (Armstrong, 1974) had defined axioms respectively rules which make it possible to find all functional dependencies of a relation. In literature these rules are known as Armstrong's axioms. Functional dependencies are used as a concept during database design they don't have an implemented counterpart in a database.

As with other concepts functional dependencies have a temporal counterpart too, called temporal functional dependencies. At an abstract point of view temporal functional dependencies are divided into intrastate and interstate dependencies (Jensen, Snodgrass, & Soo, 1996). Intrastate dependencies are those which apply to individual snapshots of a temporal relation. Hence they are understood as conventional functional dependencies applying to attributes of a temporal relation ignoring time. In contrast interstate functional dependencies are those dependencies across snapshots. They describe functional dependencies of attributes over time. For this work only intrastate dependencies are considered and serve as a basis for concepts related to temporal primary keys.

5.3.2 Temporal Primary Key

Snodgrass (Snodgrass, 2000) pointed out that when adding time support to relations it is one of the first steps to consider the valid time period as a part of the primary key. A primary key uniquely identifies each row in a table. Originally Codd (Codd, 1970) defined a primary key as the set of domains of a relation which uniquely identify tuples of that relation. Therefore the goal of a temporal primary key is to uniquely identify tuples within temporal relations at any given time. At first sight it seems that adding the valid time period to the primary key is the solution. The following examples of a relation storing information about an employee's department attendance show the opposite.

The employment relation consists of the employee column and the department column. The employee column is the primary key. An employee can only attend to one department. For convenience it is assumed that the name of the employee is unique. For such a relation it is not possible to store information about the history of an employee's department attendance.

<u>EMPLOYEE</u>	<u>DEPARTMENT</u>
Lukas	Development
Alexander	Business

After adding a valid time period to the relation and adding the department column to the primary key it is possible to store the history of an employment.

<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	FROM	TO
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	25.2.2008
Alexander	Business	1.1.2008	31.1.2008

The primary key consisting of EMPLOYEE and DEPARTMENT does not cover the case that an employee may attend twice to a department over time. The red marked cells in the table below show this case which violates the given primary key.

<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	FROM	TO
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	29.2.2008
Alexander	Business	1.1.2008	31.1.2008
Lukas	Development	1.3.2008	31.3.2008

Temporal Relations

Now time has come to include the valid time period to the primary key. The table below shows the relation with the valid time period added to the primary key.

<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	<u>FROM</u>	<u>TO</u>
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	29.2.2008
Alexander	Business	1.1.2008	31.1.2008
Lukas	Development	25.2.2008	31.3.2008

Within the period between 25.2.2008 and 29.2.2008 employee Lukas attends to more than one department but the primary key is not violated.

<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	<u>FROM</u>	<u>TO</u>
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	29.2.2008
Alexander	Business	1.1.2008	31.1.2008
Lukas	Development	25.2.2008	31.3.2008

As a result Snodgrass (Snodgrass, 2000) stated adding the valid time period to the primary key is not enough to get a temporal primary key. Because it is the period between the two valid time values which must be unique and not two valid time value themselves. Hence it is not possible to define a primary key the usual way. In his work Snodgrass (Snodgrass, 2000) provided SQL statements which assure that rows with the same primary key are not overlapping each other in time. There are two possibilities to implement such SQL statements. First they can be included in manipulating operations such as insert, update and delete. Second they can be implemented within database functionality like triggers. Snodgrass (Snodgrass, 2000) summarises that a temporal primary key is the analogue of a conventional primary key if it assures that at any given time there are no two rows with overlapping valid time periods with the same primary key.

5.3.3 Temporal Foreign Key

Codd (Codd, 1970) defined a foreign key as the one domain or combination of domains which is not the primary key of a relation but the primary key of another relation. This concept enables to define cross references between relations and is called referential integrity. The temporal counterpart concept is called temporal referential integrity. Snodgrass (Snodgrass, 2000) distinguished between four cases considering which side of the cross reference is a temporal relation. For this work the case that both relations are temporal relations is important and now described in detail. In his work Snodgrass (Snodgrass, 2000) pointed out three steps which are needed to ensure referential integrity if both relations are temporal:

- 1) For all rows r in the referenced table there must be that key value in the referencing table when r starts,
- 2) for all rows r in the referenced table there must be that key value in the referencing table when r ends
- 3) and there are no gaps in the referencing table within the valid time period of all rows r with that key value

With other words, the valid time period of a row with a foreign key in the referenced table must be completely covered by the valid time period of referenced primary key of the referencing table. The following tables show an example of temporal referential integrity including again an employment relation in this case as the referenced table. An employee relation storing all employees of a company is considered as the referencing table. In the first case below temporal referential integrity is not violated as the valid time period of the employee completely covers the referenced valid time periods of the employment relation.

EMPLOYEE				
<u>EMPLOYEE</u>	<u>FROM</u>	<u>TO</u>	AGE	SEX
Lukas	1.1.2008	01.06.2008	24	Male

EMPLOYMENT			
<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	<u>FROM</u>	<u>TO</u>
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	29.2.2008

Temporal Relations

The second case below shows a violation as the employment in the business department goes beyond the valid time period of the employee.

EMPLOYEE				
<u>EMPLOYEE</u>	<u>FROM</u>	<u>TO</u>	AGE	SEX
Lukas	1.1.2008	01.06.2008	24	Male

EMPLOYMENT			
<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	<u>FROM</u>	<u>TO</u>
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	31.6.2008

The third case below shows a violation as the employee attended to the company twice. Hence there is a gap between the valid time periods of the employee relation. The employment within the business department falls within the gap therefore temporal referential integrity is violated.

EMPLOYEE				
<u>EMPLOYEE</u>	<u>FROM</u>	<u>TO</u>	AGE	SEX
Lukas	1.1.2008	01.06.2008	24	Male
Lukas	1.10.2008	31.12.2008	24	Male

EMPLOYMENT			
<u>EMPLOYEE</u>	<u>DEPARTMENT</u>	<u>FROM</u>	<u>TO</u>
Lukas	Development	1.1.2008	31.1.2008
Lukas	Business	1.2.2008	31.6.2008

Snodgrass (Snodgrass, 2000) pointed out that it is not possible to define temporal referential integrity the usual way. Therefore he again provided SQL statements which can be implement either within manipulation operations or within database functionality like triggers. Temporal referential integrity is the temporal counterpart of referential integrity if it assures that valid time periods of the referenced table are completely covered by valid time periods of the referencing table. Further during valid time periods of the referenced table there are no gaps in valid time periods of the referencing table allowed.

5.4 Current State

This section provides an overview of concepts for querying current state of temporal relations. In conventional relations there is no other state than the current state so this topic is unique to temporal relations and has no conventional counterpart.

According to Snodgrass (Snodgrass, 2000) there are two possibilities to define the current state into a valid time period. It is assumed that the valid time period is encoded via a pair of time instants. One time instant is called start date and one time instant is called end date. The end of the valid time period is set to the maximum date a system provides. It is possible to query the current state of such a relation through comparing the end date with the maximum date. A major disadvantage of this approach is that it is not possible to distinguish between rows that are valid until the maximum time and a row which is valid now. Rows whose valid time period ends before the maximum date fall out of the query condition. As a result this approach is not practicable. Another approach is to set the end date to null. A value of null defines that the end of the valid time period of a row is unknown and therefore always considered as valid now. The disadvantage of this approach is that null values may cause errors in database functions comparing dates (Snodgrass, 2000).

In his work Snodgrass (Snodgrass, 2000) pointed out that the current state is not more important than any other state of a temporal relation. Therefore he provided SQL statements that are querying any given state out of a temporal relation.

5.4.1 Temporal Partitioning

Another approach to query current state out of the temporal relational model is called Temporal Partitioning. Snodgrass (Snodgrass, 2000) introduced that approach, which is implemented by separating current state from past state. Therefore a temporal relation is split up into one relation storing the current state and one relation storing rows whose valid time period had expired. An extension of temporal partitioning is to separate future states into a third relation too. The advantage of temporal partitioning is that querying the current state is possible with conventional querying methods. On the other hand when querying history states and current states together, queries get much more complex as relations have to be joined within such a query statement.

5.5 Temporal Databases

In this chapter previously developed temporal databases are shortly introduced.

	<i>No support for transaction time</i>	<i>Support for transaction time</i>
<i>No support for event time</i>	static	rollback
<i>Support for event time</i>	historic	bitemporal

Figure 15 Temporal Databases (Worboys, 1994)

Figure 15 shows the classification of temporal databases from Worboys (Worboys, 1994). Static databases also called snapshot databases store information about real world phenomena for a certain state, most often the current state. Manipulating operations such as update statements overwrite this state. Therefore past states are not accessible in snapshot databases. They neither support valid time nor transaction time. Historic databases record the information about real world phenomena for multiple states. They only support valid time, hence the period must be provided by the user and not by the system (Steiner, 1998). Rollback databases only support transaction time, hence the valid time period is maintained by the system. The database keeps track of every change in data such a concept can be viewed as append-only (Steiner, 1998). Finally bitemporal databases support both dimensions of time. According to Steiner (Steiner, 1998) such databases have the properties of historical and rollback databases. Therefore within bitemporal databases it is possible to query when an observation of the real world had happened and when this fact was available in database (Worboys, 1994).

Nowadays state of the art RDBMS are snapshot databases, they neither support valid time nor transaction time. Most often the support of time does not go beyond the support of temporal data types. But there are many solutions built on top of RDBMS which temporally enable them. For example TimeDB¹ is a java based interpreter which translates conventional SQL statements into temporal statements. Relations of a database schema are converted to bitemporal relations. But also major RDBMS vendors like for example Oracle may provide time support through applications on top of RDBMS like the Oracle Workspace Manager². What is missing is a native support of temporal relations.

¹ <http://www.timeconsult.com/>

² http://www.oracle.com/technology/products/database/workspace_manager/index.html

5.6 Temporal Data models

In the past dozens of temporal data models had been developed. It is beyond the scope of this work to provide a complete list or to describe each model in detail. The important concepts of temporal relations had been already described in the previous chapters. Many of the provided concepts originate to a previously developed temporal data model. Anyway this section gives a short overview of temporal data models. Snodgrass (Snodgrass, 1992) provided a list of temporal data models which is shown in Figure 16.

Data Model	Citation	Temporal Dimension(s)	Homogeneous	Identifier
—	@cite[Snodgrass86A]	both	yes	Ahn
Temporally Oriented Data Model	@cite[Ariav86A]	both	yes	Ariav
Time Relational Model	@cite[Ben-Zvi82]	both	yes	Ben-Zvi
Historical Data Model	@cite[Clifford83]	valid	yes	Clifford-1
Historical Relational Data Model	@cite[Clifford87A]	valid	no	Clifford-2
Homogeneous Relational Model	@cite[Gadia88B]	valid	yes	Gadia-1
Heterogeneous Relational Model	@cite[Gadia88A]	valid	no	Gadia-2
TempSQL	@cite[Gadia92]	both	yes	Gadia-3
DM/T	@cite[Jensen91D]	transaction	N/A	Jensen
LEGOL 2.0	@cite[Jones79]	valid	yes	Jones
DATA	@cite[Kimball78]	transaction	N/A	Kimball
—	@cite[Lomet89A]	transaction	N/A	Lomet
Temporal Relational Model	@cite[Lorentzos88B]	valid	no	Lorentzos
—	@cite[Lum84]	transaction	yes	Lum
—	@cite[McKenzie91C]	both	no	McKenzie
Temporal Relational Model	@cite[Navathe89]	valid	yes	Navathe
HQL	@cite[Sadeghi87B]	valid	yes	Sadeghi
HSQL	@cite[Sarda90]	valid	yes	Sarda
Temporal Data Model	@cite[Segev87]	valid	yes	Shoshani
TQuel	@cite[Snodgrass87A]	both	yes	Snodgrass
Postgres	@cite[Stonebraker87D]	transaction	no	Stonebraker
HQuel	@cite[Tansel86B]	valid	no	Tansel
Accounting Data Model	@cite[Thompson91A]	both	yes	Thompson
Time Oriented Databank Model	@cite[Wiederhold75]	valid	yes	Wiederhold

Figure 16 Temporal Data Models (Snodgrass, 1992)

For each temporal data model Snodgrass mentioned which temporal dimension is supported. An often used distinction between the previously developed models is whether they are satisfying first normal form or not (Chomicki, 2005). Often this fact is related to whether a model uses attribute or tuple level time stamping as time stamping on attribute level violates first normal form. Kaiser (Kaiser, 1998) provided a comparison of the most important temporal data models which is shown in Figure 17.

- V1: Welche Zeitdimension wird unterstützt?
V2: Wird die Entitätsintegrität angemessen unterstützt?
V3: Wird die referentielle Integrität angemessen unterstützt?
V4: Wird ein coalescing automatisch durchgeführt?
V5: Ist das Datenmodell zum relationalen Modell aufwärts kompatibel?
V6: Ist die Datenbanksprache zu SQL92 (syntaktisch) aufwärts kompatibel?
V7: Gibt es einen (kommerziell) verfügbaren Prototyp der Sprache?

	<u>TSQL2</u>	<u>TSQL</u>	<u>SQL^I</u>	<u>IXSQL</u>	<u>ATSQL2</u>	<u>SQL/Temporal</u>
V1	bi	Gültigk.	Gültigk.	Gültigk.	bi	Gültigk.
V2	nein	ja ¹	k.A. ²	ja	ja? ³	k.A. ²
V3	nein	nein	k.A. ²	nein	ja? ³	k.A. ²
V4	ja	ja	nein	nein	nein	nein
V5	nein	ja	nein	ja	ja	ja
V6	ja	ja	ja	ja	ja	ja
V7	nein	nein	nein	nein	ja	nein

Figure 17 Comparison of Temporal Data Models (Kaiser, 1998)

Kaiser pointed out which temporal dimension a temporal data model supports and to which level normalisation concepts like primary key and foreign key are supported. One important consideration is if a temporal data model is available in commercial RDBMS. According to Kaiser there is only the ATSQL2 temporal data model commercially available. In their work Ott and Swiaczny (Ott & Swiaczny, 2001) provided an overview of temporal models and approaches. They identified a snapshot approach as the simplest approach to provide temporal data in a GIS. Further they described the concepts of a space time cube, topology of time and the space time composite which are all developed by Langran. One of the most important developments had been done by Snodgrass, Jensen and others by developing a unifying model which is called the Bitemporal Conceptual Data Model further abbreviated as BCDM. The scope of this model was to unify previously developed concepts into one temporal data model (Jensen, Soo, & Snodgrass, 1994). *"The idea behind the BCDM is to retain the simplicity of the relational model while also capturing the temporal aspects of the facts stored in a database."* (Jensen, 2000). The core elements of the BCDM are bitemporal relations also called bitemporal tables. They include four time stamp columns, two for each dimension of time. Hence a bitemporal table is bitemporal as it supports valid time as well as transaction time. As a result of the development of the BCDM Snodgrass and others developed the TSQL2 Temporal language. Within this language a period data type was defined to overcome the problem of defining periods via pairs of time stamps. In later papers Snodgrass extended the BCDM by defining temporal integrity constraints.

One of the latest developments in the field of temporal data models was the integration of temporal concepts into the SQL3 standard. According to Snodgrass (Snodgrass, 2000) the integration process had suffered from many disagreements of the involved researches. But one of the main goals had been accomplished as a period data type had been defined within SQL3. Moreover temporal predicates had been defined in order to query for relationships of periods (Snodgrass, 2000). Anyway today there is no state of the art relational database compliant to the SQL3 standard in the temporal sense (Steiner, 1998). Most often the temporal support does not go beyond the support of simple date and time data types excluding a period data type.

6. Geometry of Time

The International Standardisation Organisation (ISO) developed a standard called “ISO 19108 Geographic information – Temporal Schema”. The standard provides concepts needed to describe the temporal characteristics of geographic information. *“Temporal characteristics of geographic information include feature attributes, feature operations, feature associations, and metadata elements that take a value in the temporal domain.”* (ISO, 2002). The standard emphasises valid time rather than transaction time. Temporal structures provided in the ISO standard are not only intended for use in the area of Geographic Information Systems. To understand the most important characteristics of temporal information the ISO Standard provides definitions which are important for this work

6.1 Definitions

Calendar is a *“discrete temporal reference system that provides a basis for defining temporal position to a resolution of one day”* (ISO, 2002)..

Coordinated Universal Time (UTC) is a *“time scale maintained by the Bureau International des Poids et Mesures (International Bureau of Weights and Measures) and the International Earth Rotation Service (IERS) that forms the basis of a coordinated dissemination of standard frequencies and time signals ...”* (ISO, 2002)..

Day is a *“period having a duration nominally equivalent to the periodic time of the Earth's rotation around its axis.”* (ISO, 2002).

Event is an *“action which occurs at an instant.”* (ISO, 2002).

Instant is a *“0-dimensional geometric primitive representing position in time.”* (ISO, 2002).

Month is a *“period approximately equal in duration to the periodic time of a lunar cycle.”* (ISO, 2002).

Period is an *“one-dimensional geometric primitive representing extent in time.”* (ISO, 2002).

Temporal coordinate system is a *“temporal reference system based on an interval scale on which distance is measured as a multiple of a single unit of time.”* (ISO, 2002).

Temporal position is a *“location relative to a temporal reference system.”* (ISO, 2002).

Temporal reference system is a *“reference system against which time is measured.”* (ISO, 2002).

6.2 Temporal geometric primitives

Within the ISO Standard one chapter is titled “Geometry of Time” (ISO, 2002, S. 7) which corresponds to the main idea of this work. It says that time is a dimension analogous to any of the spatial dimensions. As a spatial point occupies a position in space an instant in time occupies a position in relation to a temporal reference system. According to the standard one important difference between space and time is that time has single dimension and cannot be reused (ISO, 2002). The following figure shows an abstract UML Model defining temporal objects.

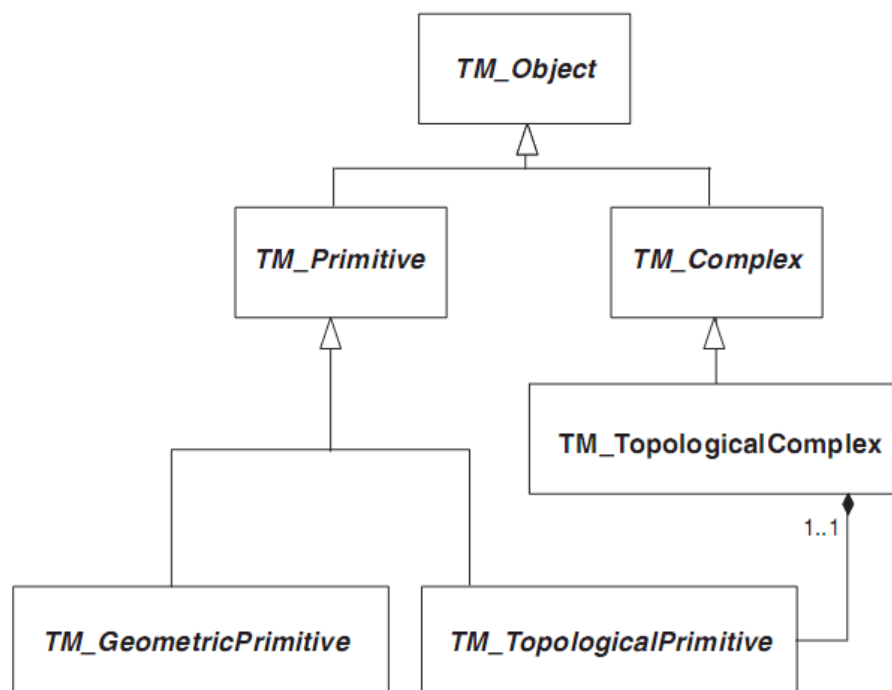


Figure 18 Temporal Objects (ISO, 2002)

Figure 18 shows that temporal primitives which are namely instants and periods are conceived to geometric primitives. Hence an instant in time is equivalent to a point in space. In practice, an instant is an interval whose duration is less than the resolution of the time scale (ISO, 2002). The same applies to a period. A period is equivalent to a curve in space and it is bounded by one instant at the beginning and one instant at the end. As a result of this a period has a length, a so called duration, which itself is equal to the temporal distance of the two bounding instants (ISO, 2002).

Figure 19 shows the relation between periods and instants. Both are derived from the abstract class `TM_GeometricPrimitive`.

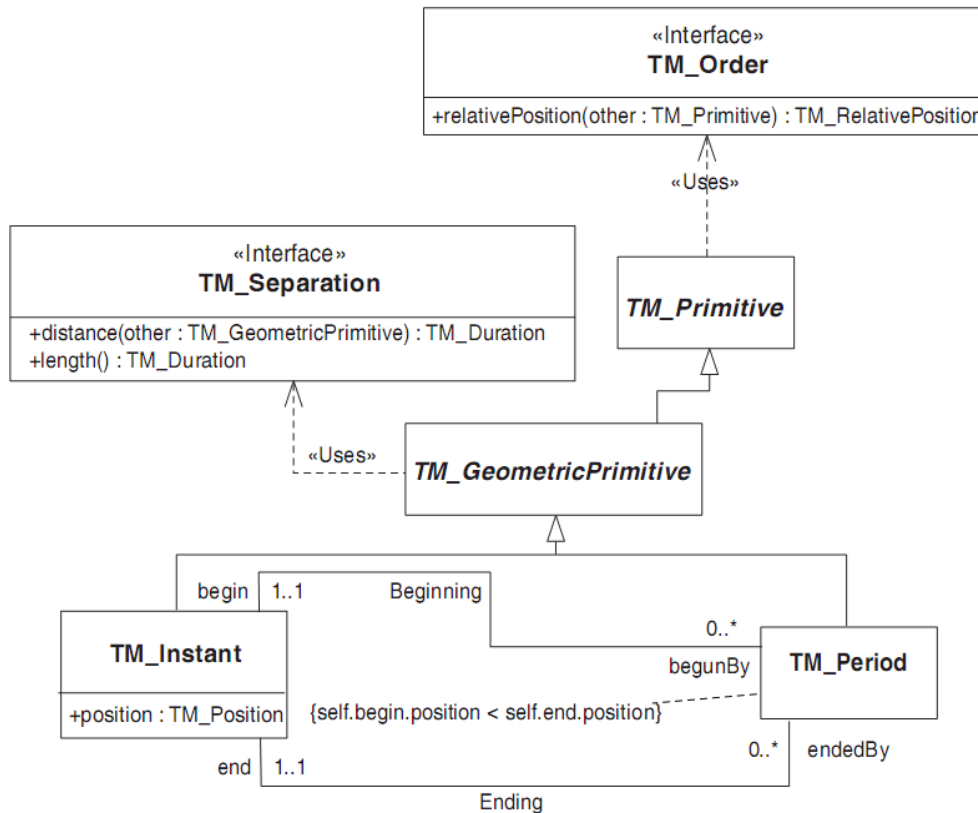


Figure 19 Temporal geometric primitives (ISO, 2002)

The classes `TM_Instant` and `TM_Period` are relevant to this work and discussed in chapter 7.5.

6.3 Temporal reference system

As it was defined previously in this work a value in the time domain is a temporal position measured relative to a temporal reference system. The ISO Standard recommends using the Gregorian calendar in combination with the UTC. The Gregorian calendar identifies temporal information at a granularity of one day and the UTC identifies temporal position within a day operating on a precision beyond seconds.

It says in the ISO standard that for other application other temporal reference systems may be appropriate. But as recommended this work will use Gregorian calendar in combination with UTC. This combination is the common one provided by most database systems.

6.4 From Temporal Objects to Spatio Temporal Objects

Roosman and others (Roosmann, Busch, Gorczyk, & Mauersberger, 2003) followed the concepts of the ISO standard and provided concepts of modelling time as geometry. In their work they provided an UML model which defines a temporal object.

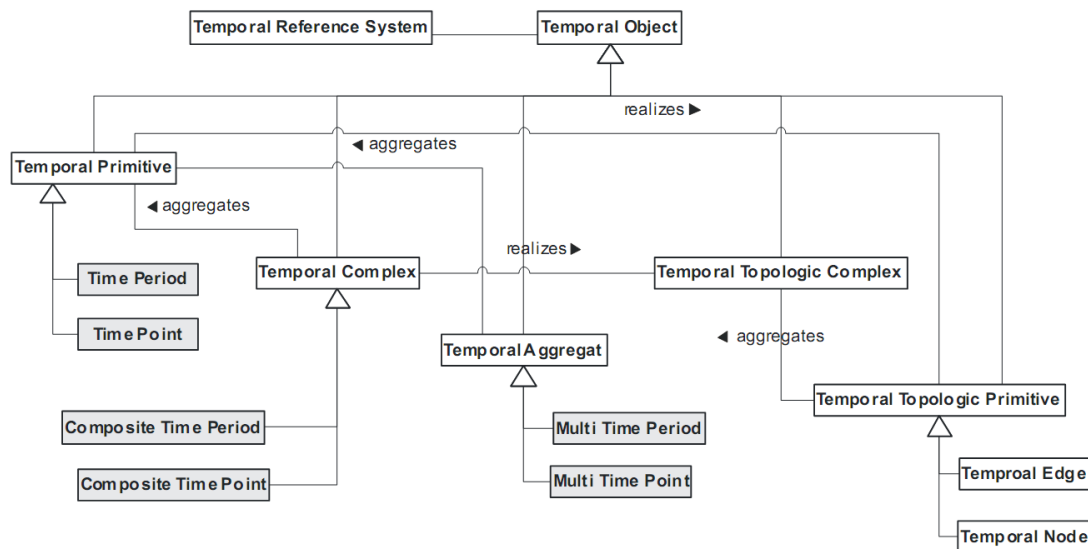


Figure 20 Temporal Objects (Roosmann, Busch, Gorczyk, & Mauersberger, 2003)

Figure 20 shows the definition of a temporal object which is always based on a temporal reference system. A temporal object consists of temporal primitives or temporal complexes. Moreover a temporal object may consist of temporal aggregates or temporal topological primitives. For this work only temporal primitives such as a time point or time period are of interest and described in chapter 7.5. Roosman and others (Roosmann, Busch, Gorczyk, & Mauersberger, 2003) identified the relations between temporal, spatial and thematic objects which resulted in an UML model of a spatio temporal object as shown in the following figure.

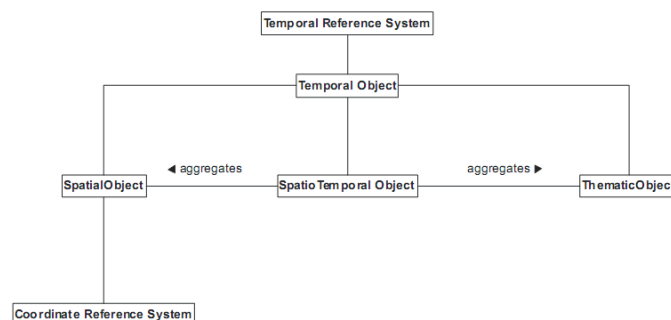


Figure 21 Spatio temporal object (Roosmann, Busch, Gorczyk, & Mauersberger, 2003)

Main Part -Time as Geometry

The scope of this section is to combine the relational model defined by Codd (Codd, 1970), the simple feature specification (OpenGIS_Consortium, 2006a) defined by the Open Geospatial Consortium and concepts related to temporal relations into a concept of spatio temporal relations. In literature common analogies between space and time had been identified when conceiving these dimensions into relational data bases. Further the International Standardisation Organisation published a standard called “ISO 19108 Geographic Information – Temporal Schema” which defined the geometry of time. Therefore the introduced concept of spatio temporal relations will be based on the extension of the relational model where space is implemented as simple feature geometry but also time is implemented as simple feature geometry. As a result spatial operations are used to answer both spatial and temporal questions.

7. Time as Geometry – A formal description

This chapter describes the steps which are necessary to model time as geometry. As a first step chapter 7.2 introduces a discrete view of time as the basis for this work. This view uses the Gregorian calendar in combination with the Coordinated World Time (UTC) as its temporal reference system. To model time as geometry it is further necessary to transform date specifications into something like coordinates. Therefore an interval scale based view of discrete time is introduced. Finally two functions are identified for the translation of date specifications into coordinates and coordinates into date specifications. The contents of this chapter are on a formal basis and therefore independent from an implementation. The implementation of the provided formal concepts is described in chapter 8.

7.1 Background

The main idea of this thesis applies to the idea of modelling time as geometry in order to integrate time into geo data bases. The main motivation behind that is to reuse the existing spatial framework within geo data bases for a temporal context. Therefore spatial data types are used to represent temporal values. For example an instant in time is represented as a point. A period of time is represented as a line geometry. As a result it is possible to apply other components of a spatial framework such as spatial operations to those transformed temporal values. The provided approach mainly follows the concepts of the ISO 19108 standard and the contents of the work of Roosman and others (Roosmann, Busch, Gorczyk, & Mauersberger, 2003). To model time as geometry is also based on some common analogies of those two dimensions which were identified in the past. In language there are many words which are used in the context of space and time (Peuquet, 2002). For example the words before and after are used in both, space and time. As Peuquet (Peuquet, 2002) mentioned, things happen in space over time. So there is fundamental relation between these two concepts. When something exists in the real world, it takes times and takes place (Peuquet, 2002). But there are also great differences between space and time as time can't be reused. Once something had happened it will never happen again at that time. The idea of modelling time as geometry also follows a traditional method to map time onto a time line constructed from integer or real numbers (Langran, 1992). Such an approach is useful and corresponds—with some imitations—to cognitive models of time (Frank, Egenhofer, & Colledge, 1998).

7.2 A Discrete View of Time

The goal of this chapter is to introduce a discrete view of time as a basis for this work. This view includes the following important assumptions:

- Time is absolute, but relative to an origin

The first assumption is that time is absolute and therefore it does not apply to Einstein's relativity theory. For this work it is assumed that time always goes by at same speed independent from an object's own motion. Hence time in the past and time in the future go by at same speed. The use of an origin is part of the first assumption. Time is relative to this origin. For example consider a date specification of January 1st 2008 as the origin and January 8th 2008 as another date specification. Hence January 8th 2008 is absolute but the relative time of January 1st 2008 to January 8th 2008 is 7 days.

- Time is linear and totally ordered

The second assumption is based on the first assumption and further defines that time is linear and totally ordered. Time starts at the origin and linearly extends from the origin to the future. A total order is assumed therefore for two date specifications t_1 and t_2 it is necessary that either $t_1=t_2$ or $t_1 < t_2$ respectively $t_1 > t_2$ but it is not possible that both $t_1 < t_2$ and $t_1 > t_2$. Therefore it is not possible that an event happens before and after a second event as it would be possible when cycling time is used (e.g. morning and evening). Once an event has had occurred at a time it will never occur again at that time. Time cannot be reused.

- Time is used within a temporal reference system

The third assumption is based on the first and second assumption and further includes that a temporal reference system is used to define the set of date specifications. For this work the Gregorian calendar is used to identify years, months and days. The Universal Coordinated Time (UTC) is used to identify time within a day. Hence it used to identify hours, minutes and seconds. The smallest time unit used in this work is one second. As a result this work assumes a date specification to be of the form {YYYY.MM.DD HH:MIN:SEC}. For example consider an event had happened on January 1st 2008 at 1pm the according date specification of this is {2008.01.01 13:00:00}.

For the provided discrete view of time a formal definition of any date specification is as follows:

$$D := \{YYYY.MM.DD HH:MIN.SEC \mid$$

$$YYYY \in \mathbb{N},$$

$$MM \in \{1,2, \dots, 12\},$$

$$DD \in \{1,2, \dots, 31\},$$

$$HH \in \{0,1, \dots, 24\},$$

$$MIN \in \{0,1, \dots, 60\},$$

$$SEC \in \{0,1, \dots, 60\}\}$$

Years are described as positive natural numbers where the minimum is the year 0 and the maximum is the year 9999. Months have a value range from 1 to 12 the same applies for days they have a value range from 1 to 31. Hours have a value range of 0 to 24 where 00:00 is assumed as the beginning of a new day and 24:00 is assumed as the end of the last day. Minutes and seconds have a value range from 0 to 60.

Figure 22 is a graphical representation of the provided discrete view of time including two date specifications and a distance between them.

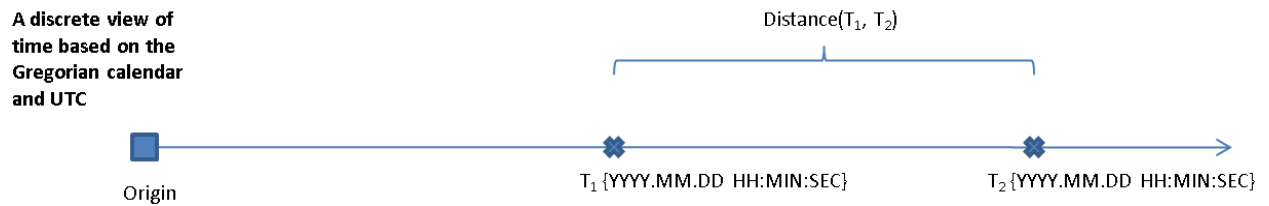


Figure 22 Discrete view of time (Gregorian calendar + UTC)

To allow measurements and distance calculations a metric must be defined. For any date specification the following assumptions are made:

- $\text{Distance}(T_1, T_1) = 0$
- $\text{Distance}(T_1, T_2) = 0 \rightarrow T_1 = T_2$
- $\text{Distance}(T_1, T_2) = \text{Distance}(T_2, T_1)$
- $\text{Distance}(T_1, T_3) \leq \text{Distance}(T_1, T_2) + \text{Distance}(T_2, T_3)$

The first assumption is that the temporal distance between two equal date specifications is 0. As a result there are no unequal date specifications with distance 0 between them. The third assumption is that the temporal distance from T_1 to T_2 is equal to the temporal distance from T_2 to T_1 . It is necessary that the temporal distance between T_1 and T_3 is less than or equal to the sum of the temporal distances of T_1 to T_2 and T_2 to T_3 .

7.3 A discrete view of time based on an interval scale

The previous chapter introduced a discrete view of time whose temporal values are date specifications. The goal of this chapter is to introduce a discrete view of time based on an interval scale whose values are the natural numbers. Further it is the goal of this chapter to describe a transformation between the discrete view of time using date specifications and the introduced discrete view of time based on an interval scale. To model time as geometry it is necessary to transform date specifications into something like coordinates. Because coordinates are numbers date specifications must be transformed into numbers first. An interval scale is used as it allows meaningful measurements between values which are necessary to calculate distances. When transforming date specifications into numbers it is further necessary that the previously identified assumptions apply to the interval scale based view too. Hence this view again assumes an origin as well as a total order. As a result a date specification relatively later to an origin is transformed into a greater number than a date specification relatively earlier to an origin. Further the temporal distance between two date specifications must be equal to the distance of the related numbers.

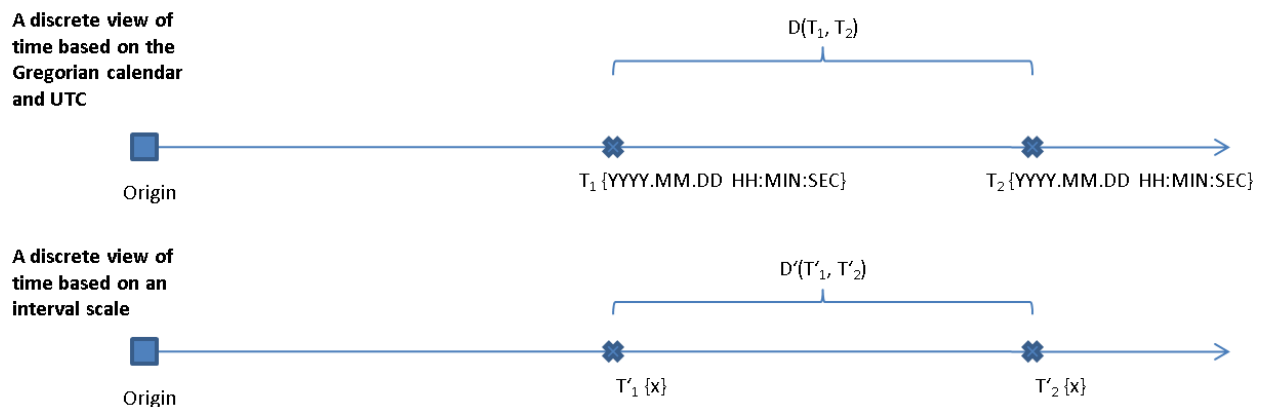


Figure 23 Calendar time vs. Interval scale

Figure 23 shows a comparison between the discrete view of time based on the Gregorian calendar and discrete view of time based on an interval scale. The two date specifications T_1 and T_2 have their transformed counterpart on an interval scale called T'_1 and T'_2 . When transforming date specifications into numbers it is necessary that $T_1 < T_2$ it implies that $T'_1 < T'_2$. Further it is necessary that the temporal distance between T_1 and T_2 is equal to the distance between T'_1 and T'_2 .

The formal definition of any transformed date specification into a number is as follows:

$$T' \in \mathbb{N} \forall T'$$

This formal definition says that after the transformation any date specification is member of the natural numbers. The date specification of the origin is transformed into 0 and is therefore the least number on the interval scale. Any date specification after the origin is transformed into a number greater than 0.

Again a metric is necessary to allow distance calculations. The following assumptions still apply:

- $\text{Distance}(T'_1, T'_1) = 0$
- $\text{Distance}(T'_1, T'_2) = 0 \rightarrow T'_1 = T'_2$
- $\text{Distance}(T'_1, T'_2) = \text{Distance}(T'_2, T'_1)$
- $\text{Distance}(T'_1, T'_3) \leq \text{Distance}(T'_1, T'_2) + \text{Distance}(T'_2, T'_3)$

The first assumption is that if the distance of two equal date specifications is 0 then the distance of their transformed numbers must be 0 too. As a distance between two unequal date specifications is not 0 the distance between their transformed numbers is not 0 too. The temporal distance from T'_1 to T'_2 is equal to the temporal distance from T'_2 to T'_1 . If $T_1 < T_2 < T_3$ then it is necessary that $T'_1 < T'_2 < T'_3$. Hence the temporal distance between T'_1 and T'_3 is less than or equal to the sum of the temporal distances of T'_1 to T'_2 and T'_2 to T'_3 .

This chapter had introduced a discrete view of time based on an interval scale whose values are the natural numbers. This view of time is more appropriate when modelling time as geometry as date specifications are transformed into numbers which corresponds to geometric coordinates consisting of numbers too. To accomplish a transformation from date specifications into natural numbers this chapter provided a description of the necessary formal aspects to consider. The next chapter describes the needed functions for the transformation on a formal basis.

7.4 Functions

Based on all previously defined assumptions this chapter describes two functions necessary for the transformation of date specifications into natural numbers. Hence this chapter describes how a discrete view of time based on calendar time is transformed into a discrete view of time based on an interval scale. Figure 24 compares both approaches and points out the necessary transformation functions.

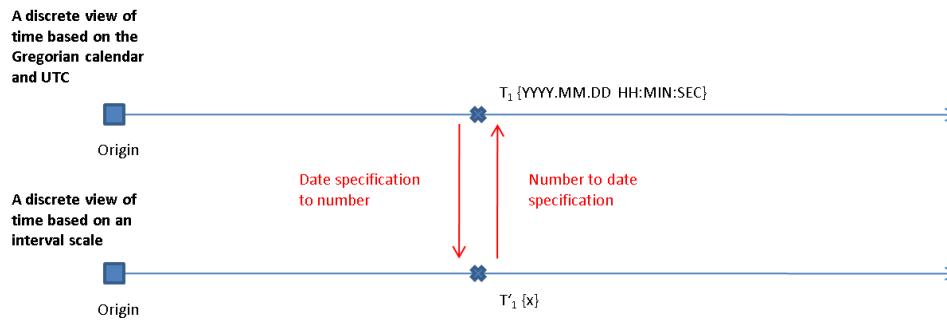


Figure 24 Functions needed for transformation

The first function transforms a date specification of the form {YYYY.MM.DD HH:MIN:SEC} into a number. This function is called TimestampToNumber. The second function transforms a number to a date specification. This function is called NumberToTimestamp. Before describing both functions in detail it is necessary to introduce an interval date specification of the form {TTTTTTTTT.HH.MIN.SEC}. It is needed to represent temporal distances between an origin and other date specifications. The formal definition of such an interval is as follows:

$$\begin{aligned}
 I(O, D) := & \{TTTTTTTTT.HH.MIN.SEC \mid \\
 & TTTTTTTTT \in \mathbb{N} \{0, \dots, 3649635\}, \\
 & HH \in \{0, \dots, 23\}, \\
 & MIN \in \{0, \dots, 59\}, \\
 & SEC \in \{0, \dots, 59\} \}
 \end{aligned}$$

An interval from an origin to any other date specification is defined as the number of whole days plus the number of left hours, minutes and seconds. The minimum number of days is 0 and the maximum number of days is 3649635 which is the number of days between the year 0 and the year 9999. For example the interval between a date specification of {2008.01.01 13:00:00} and {2008.01.02 14:38:52} is an interval of the length {000000001.01.38.52}. In other words the interval's length is one day, one hour, 38 minutes and 52 seconds.

In the next subchapters the above identified functions called `TimeStampToNumber` and `NumberToTimestamp` are described in detail.

7.4.1 `TimeStampToNumber(fpOrigin, fpDate)`

This function transforms a date specification of the form `{YYYY.MM.DD HH:MIN:SEC}` into a natural number on the interval scale. This natural number is the transformed counterpart of the original date specification. The function requires two date specifications as its input parameters. The first is a date specification of the origin and the second is a date specification of the date to be transformed. The function returns a natural number which is the number of seconds between the origin and the date. The functionality is to calculate the number of whole days plus the number of the remaining hours, minutes and seconds between the origin and the date to be transformed. The outcome of this calculation is an interval of the length `{TTTTTTTTT.HH.MIN.SEC}`. This length is then converted into the number of seconds via the following formula:

$$\text{NumberOfSeconds} := ((TTTTTTTTT * 86400) + (HH * 3600) + (MIN * 60) + SEC)$$

As a result this function returns the calculated number of seconds between the origin and the date to be transformed. When applying this number on the interval scale based view it is the transformed counterpart of the original date specification relative to the origin (which is 0). The following figure provides an overview of the calculation scenario.

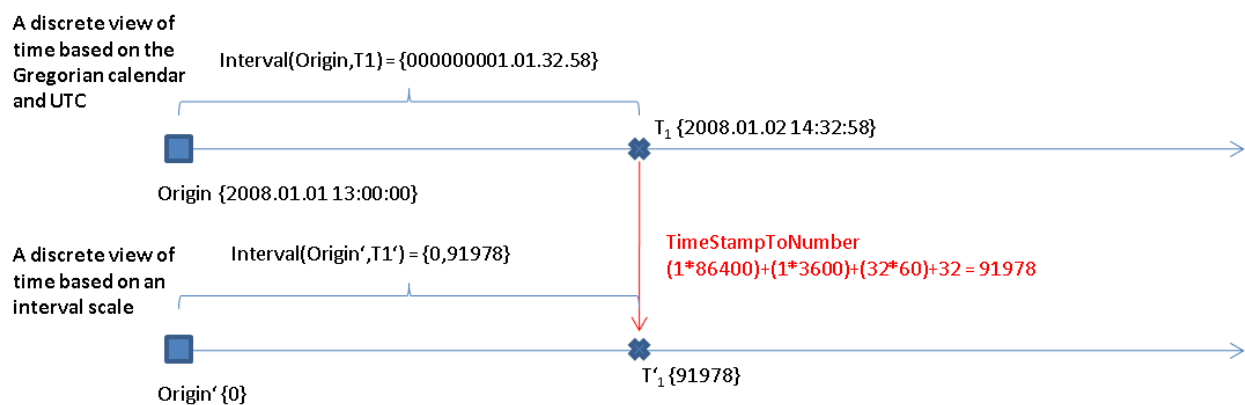


Figure 25 Date specification to number

This function is the basis for modelling time as geometry as it transforms date specifications into natural numbers. As a result these numbers are then used as coordinates of geometry.

7.4.2 NumberToTimeStamp(fpOrigin, NumberOfSeconds)

This function transforms a natural number into a date specification of the form {YYYY.MM.DD HH:MIN:SEC}. The function requires one date specifications and one number as its input parameters. The first is a date specification of the origin and the second is the number of seconds from the origin. The function returns a date specification. The functionality is to calculate the number of whole days plus the number of the remaining hours, minutes and seconds out of the number of seconds. The outcome of this calculation is an interval of the length {TTTTTTTTT.HH.MIN.SEC}. This length is then added to the date specification of the origin. The formula of the function is as follows:

$$D := (Origin + \{TTTTTTTTT.HH.MIN.SEC\})$$

$$TTTT := INTEGER\left(\frac{NumberOfSeconds}{86400}\right)$$

$$HH := INTEGER\left(\frac{NumberOfSeconds - (TTTT * 86400)}{3600}\right)$$

$$MIN := INTEGER\left(\frac{NumberOfSeconds - (TTTT * 86400) - (HH * 3600)}{60}\right)$$

$$SEC := INTEGER(NumberOfSeconds - (TTTT * 86400) - (HH * 3600) - (MIN * 60))$$

As a result this function returns the calculated date specification. This date specification is the transformed counterpart of the original number of seconds. The following figure provides an overview of the calculation scenario.

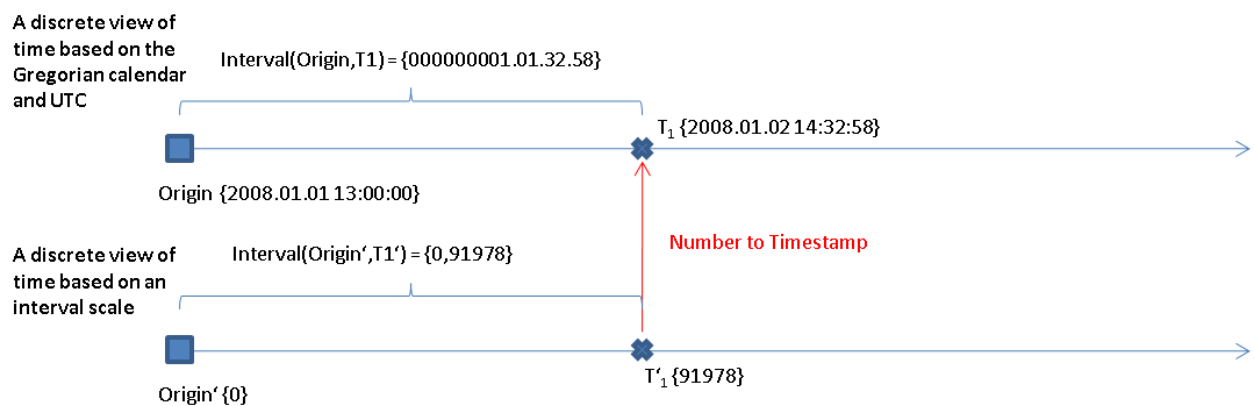


Figure 26 Number to date specification

This function is the logical counterpart to the TimeStampToNumber function.

7.5 Temporal geographic primitives

The scope of this chapter is to describe temporal geometric primitives as a basis for modelling time as geometry. There are two base types of temporal geometric primitives:

- Instant is a “0-dimensional geometric primitive representing position in time.” (ISO, 2002).
- Period is a “1-dimensional geometric primitive representing extent in time.” (ISO, 2002).

Temporal geometric primitives are the necessary geographic primitives to represent time as geometry. Figure 27 shows the class hierarchy of temporal geometric primitives.

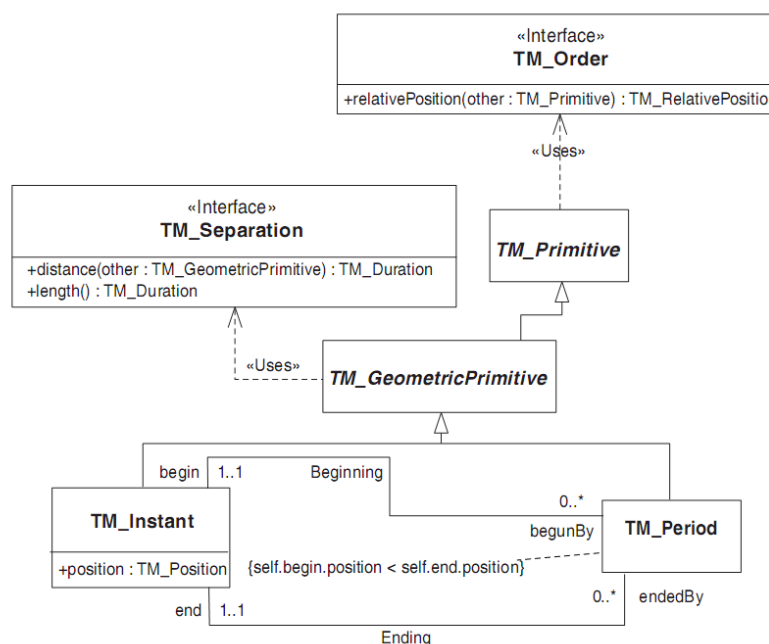


Figure 27 Temporal geometric primitives (ISO, 2002)

Temporal geographic primitives are temporal primitives represented as geographic primitives. As an instant represents position in time its geographic primitive represents position in space. Hence a period represents extent in time thus its geometric primitive is a curve in space. A period is defined by one instant where it starts and one instant where it ends. The duration of a period is equal to the temporal difference of its instants (ISO, 2002). The previous chapter described the function `TimeStampToNumber` which transforms date specifications into numbers. These numbers are the input values for the coordinates of temporal geometric primitive when time is modelled as geometry.

7.5.1 Instant

An instant represents position in time it is analogous to a geographic point representing position in space. An instant is the temporal counterpart of a geographic point. Therefore a point is the geometric primitive of an instant. To transform an instant into a point first the formal definition of a point in space is considered as:

$$Point(x, y) := \{ x \in R, y \in R \}$$

As a result the formal definition of an instant modelled as point geometry is as follows:

$$Instant(x, y) := \{ x \in \{0, \dots, 3.15328464^{11}\} \subseteq \mathbb{N}, \quad y \in \{0\} \subseteq \mathbb{N} \}$$

An instant represented as a point is defined by its coordinates. The minimum value of x is 0 which is the origin and applies to the return value of `TimeStampToNumber(Origin, Origin)`. The maximum value of x is 3.15328464^{11} which applies to the return value of `TimeStampToNumber(Origin, {9999.12.31 24:00:00})`. Moreover the `TimeStampToNumber` function returns any possible x coordinate which is considered as the number of seconds from the origin to the date specification of the instant. The y coordinate of an instant is always 0.

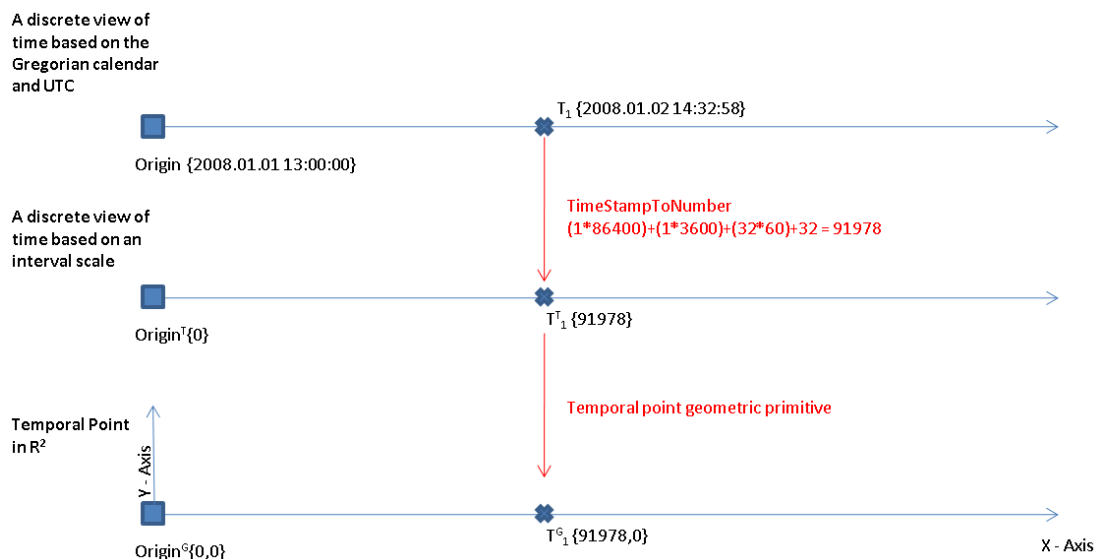


Figure 28 Process of transforming an instant into a point

Figure 28 summarises the complete process of transforming a date specification into a natural number which is the x coordinate of an instant. The date specification of {2008.01.02 14:32:58} is transformed into the natural number 91978 via the `TimeStampToNumber` function. This number is the x coordinate of a point P with an y coordinate of 0.

7.5.2 Period

A period represents extent in time hence it is analogous to a curve in space. A period therefore is the temporal counterpart of a curve in space. The formal definition of a curve in space is as follows (OpenGIS_Consortium, 2006a):

$$D(a, b) = \{t \in R \mid a \leq t \leq b\} \quad f: [a, b] \rightarrow \mathbb{R}^2$$

According to that definition a curve is a geometric object that is the homeomorphic image of a closed interval. Based on this abstract definition a line is defined as a curve consisting of two bounding points where linear interpolation is used to represent all points between them. A line geometry is therefore a straight line between a start point and an end point. A period is considered as the temporal counterpart of a line geometry which is a simple type of curve. To transform a period to a line geometry the formal definition of a line geometry is considered as:

$$Line(x^{start}, y^{start}, x^{end}, y^{end}) := \{x^{start}, y^{start}, x^{end}, y^{end} \in R\}$$

The formal definition of a period modelled as line geometry is as follows:

$$Period(x^{start}, y^{start}, x^{end}, y^{end}) := \{ \\ x^{start}, x^{end} \in \{0, \dots, 3.15328464^{11}\} \subseteq \mathbb{N} \wedge x^{start} < x^{end}, \\ y^{start}, y^{end} \in \{0\} \subseteq \mathbb{N}\}$$

A period represented as line geometry consists of the coordinates of its instants at the start and the instant at the end. The value range for all x coordinates is from 0 which is the origin to 3.15328464^{11} which is the return value of `TimeStampToNumber(Origin, {9999.12.31 24:00:00})`. As a period must have at least a duration of one second and has a direction from start to end it is necessary that $x^{start} < x^{end}$. A period with a duration of 0 seconds is considered as an instant. The duration of a period is equal to the distance between its start and end instant. As a period is represented as a line geometry the concepts of interior, exterior and boundary apply. The boundary of a period consists of the instant at the start and the instant at the end. All points between them are linear interpolated and considered as the interior. As a curve is a closed interval a line geometry representing a period includes the start and end instant.

Figure 29 summarises the process of transforming a period of time into a line geometry.

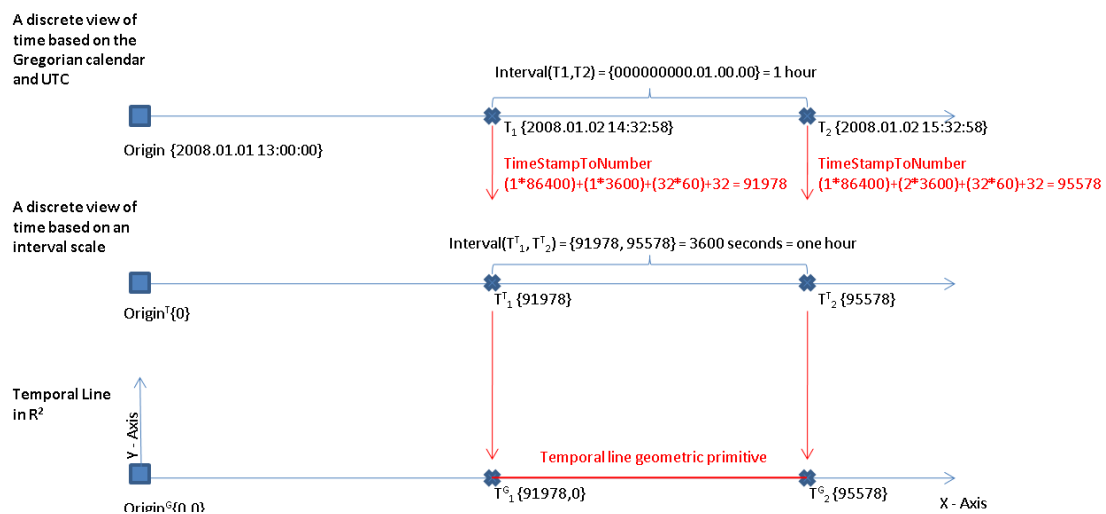


Figure 29 Process of transforming a period into a line

Within Figure 29 there are two date specifications given which are delimiting a period of time between them. The period in this example goes from January 2nd 14:32:58 to January 2nd 15:32:58 and therefore lasts for one hour. As a first step the two date specifications are transformed into natural numbers via the `TimeStampToNumber` function. The first date specification is transformed into 91978 and the second one is transformed into 95578. The difference of those two numbers is 3600 which is the number of seconds between them. This difference equals the original duration. As the last step the period is represented as a line geometry. Such a geometry consists of two points. The x coordinates of those points are the transformed numbers whereas the y coordinates are 0. The length of the line geometry is 3600 which is equal to the duration of the original calendar time. As a result the measures of the original date specification are obtained throughout all steps of this transformation.

7.6 Summary

This chapter provided a formal basis for modelling time as geometry. A discrete view of time using date specifications was introduced. Those date specifications are transformed into natural numbers on an interval scale as basis for coordinates. The necessary functions for this transformation are described. To model time as geometry the relevant geometric primitives were identified. Based on the formal description the next chapter describes a prototypical implementation inside a geodatabase.

8. Time as Geometry – A prototypical implementation

The scope of this chapter is to describe an implementation of the previously provided formal description of transforming time into geometry. The outcome of this implementation should be a prototypical implementation of a spatio temporal application.

8.1 Scope

The goal of the implementation is to integrate time into a geo database. It is the scope of the implementation to develop the necessary functions to transform time into geometry in order to reuse spatial data structures and operations within a geo database. The formal description of modelling time as geometry from previous chapters is implemented within database functionality such as functions, procedures or triggers. The outcome of integrating time into geo databases are database tables consisting of conventional attributes, spatial attributes and temporal attributes. Spatial attributes describe the location on the earth's surface whereas temporal attributes describe when a fact was observed in the real world. Both spatial as well as temporal attributes are represented as geometry. If the observation period has no duration then it is considered as an instant in time represented as point geometry. If the observation has at least a duration of one second then it is considered as a period which is represented as line geometry.

As time is represented as geometry the implementation points out that spatial operations such as overlap, meet, contains apply to temporal operations. Moreover spatial analytical functions like distance calculations apply to temporal functions too. As a result time is integrated into a geo database based on a geometric representation. This work underlines that spatial functionality within geo databases provides meaningful operations applied to time. It is the scope of this work to prove spatial functionality of its significance in the temporal case.

8.2 Functional Specification

This section provides a functional specification of the implementation. It defines the necessary functionality in order to integrate time into geo databases represented as geometry.

8.2.1 Need to have

- TimeStampToNumber function as a database function
- NumberToTimeStamp function as a database function
- A database function transforming instants into point geometries
- A database function transforming periods into line geometries

8.2.2 Nice to have

- Uniqueness constraints considering time as part of a primary key
- Referential integrity constraints considering time as part of a foreign key
- Temporal modification operations such as sequenced insert, update or delete
- Provide spatial metadata applying to the temporal case

8.2.3 Not to have

- Visualisation techniques
- Complex geometries such as Multipoint or Multiline
- Topology

8.3 Base technology

This section shortly introduces the used base technology on which the implementation is based on. To integrate time into geo databases there are three main software components necessary. The first is a geo database which is the destination of the integration. Further a SQL client application to express SQL statements is needed. To visualise the results in an appropriate manner a desktop GIS client application is used in order to display time represented as geometry.

8.3.1 Oracle Express Edition with Oracle Locator

Oracle Express Edition³ is a data base for developing purposes. It is free to use as its limited to one CPU, one gigabyte of memory and four gigabyte of hard disk space. It is based on the 10g R2 release. Oracle Locator⁴ is the package which provides the necessary spatial functionality and is included in Oracle Express Edition. It includes spatial data type as well as spatial operations, indexing methods and some analytical functions. Further it is based on the simple feature specification of the Open Geospatial consortium (OpenGIS_Consortium, 2006b) as it provides functions transforming its native geometry type into a well-know text or well-known binary representation.

8.3.2 Oracle SQL Plus Worksheet

The Oracle SQL Plus Worksheet is a client application which enables a simple SQL interface to formulate SQL expressions. In this work it is used to formulate SQL expressions as well as showing their results in a tabular manner.

8.3.3 Esri ArcView

Esri ArcView⁵ is used as a desktop GIS client application visualising spatial information as maps. As in this work time is represented as geometry this application is further used to visualise temporal information on a map.

³ <http://www.oracle.com/technology/products/database/xe/index.html>

⁴ <http://www.oracle.com/technology/products/spatial/index.html>

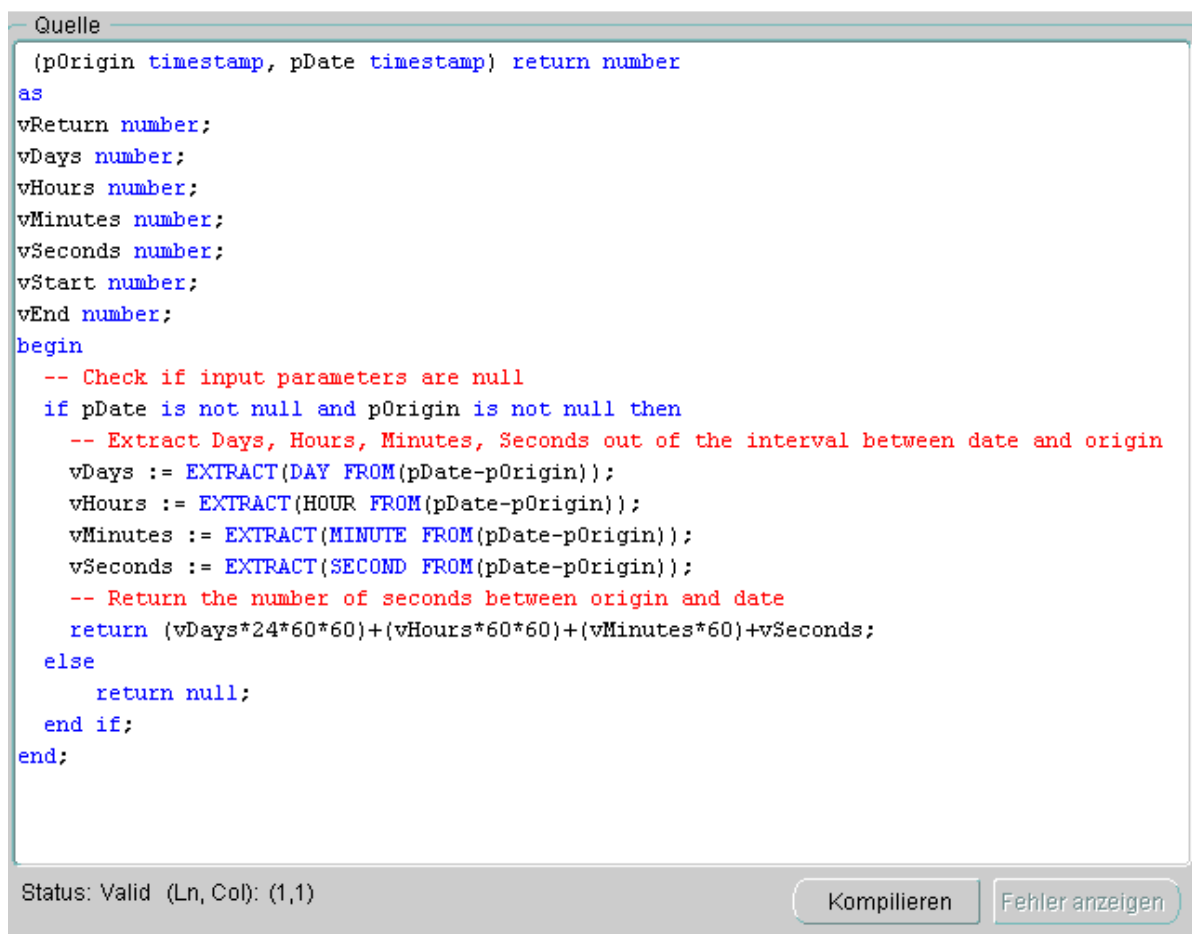
⁵ <http://www.esri.com/software/arcgis/arcview/about/features.html>

8.4 Functions

The scope of this section is to provide a description of implemented functions. The source code of each function is provided as a figure in order to maintain syntax colouring. Developer comments are red. The syntax applies to the Oracle PL/SQL standard.

8.4.1 TimeStampToNumber

The source code of the TimeStampToNumber function as defined in chapter 7.4.1 is shown in the following figure.



```

Quelle
(p0origin timestamp, pDate timestamp) return number
as
vReturn number;
vDays number;
vHours number;
vMinutes number;
vSeconds number;
vStart number;
vEnd number;
begin
  -- Check if input parameters are null
  if pDate is not null and p0origin is not null then
    -- Extract Days, Hours, Minutes, Seconds out of the interval between date and origin
    vDays := EXTRACT(DAY FROM(pDate-p0origin));
    vHours := EXTRACT(HOUR FROM(pDate-p0origin));
    vMinutes := EXTRACT(MINUTE FROM(pDate-p0origin));
    vSeconds := EXTRACT(SECOND FROM(pDate-p0origin));
    -- Return the number of seconds between origin and date
    return (vDays*24*60*60)+(vHours*60*60)+(vMinutes*60)+vSeconds;
  else
    return null;
  end if;
end;

```

Status: Valid (Ln, Col): (1,1) Kompilieren Fehler anzeigen

Figure 30 TimeStampToNumber source code

The goal of the TimeStampToNumber function is to transform a date specification to a number relative to an origin. Therefore the return value of this function is the number of seconds between the origin and the date. As input this function takes an origin and a date to transform. For both a timestamp data type is used. The functionality is to calculate the interval between the origin and the date which is accomplished by a subtraction. It returns a DAY TO SECOND type of interval which then is converted into seconds.

The following figure shows an example of using the implemented function.

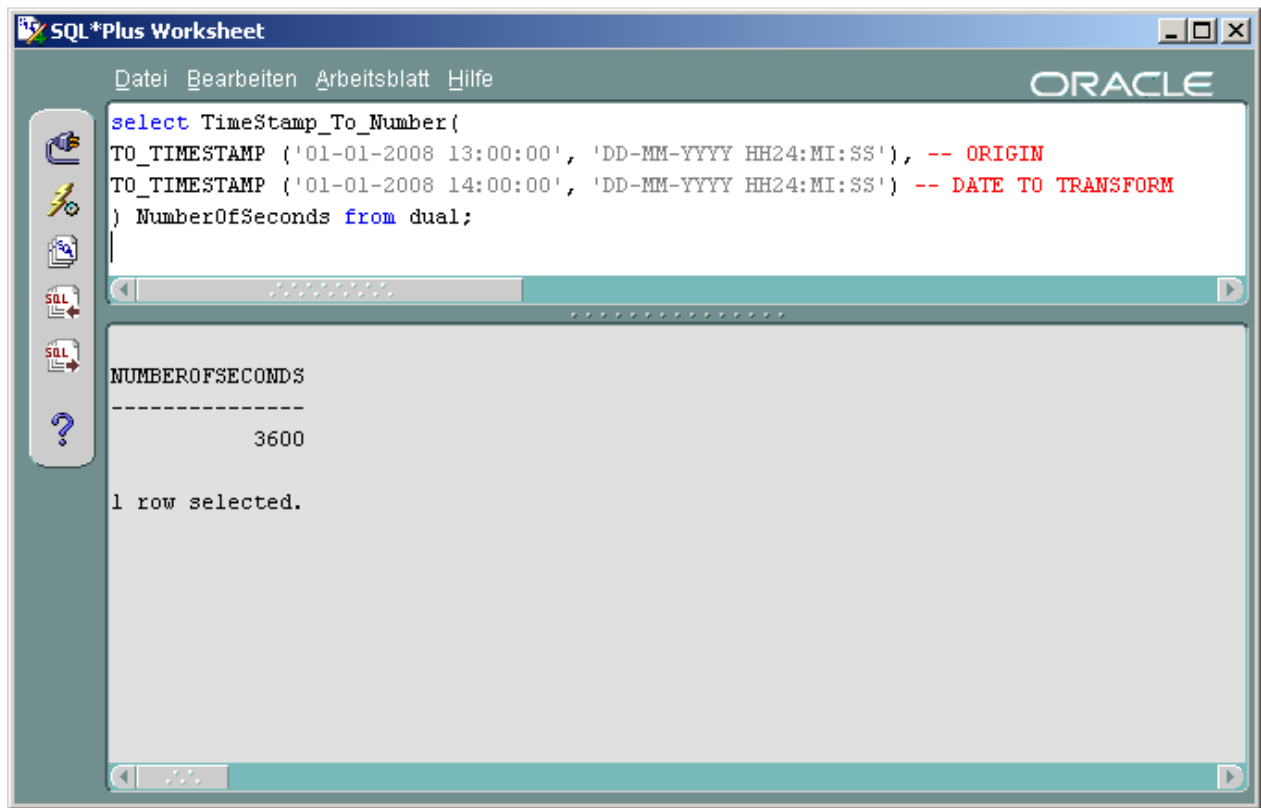
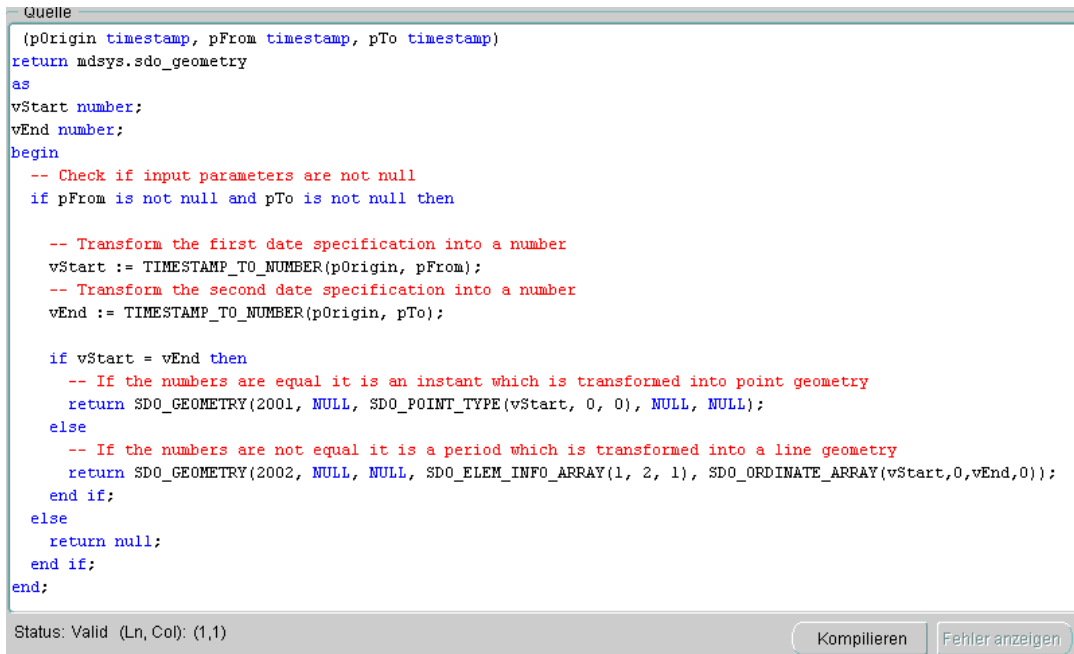


Figure 31 Transforming a date specification into a number

In this example January 1st at 1pm is used as the origin and January 1st 2pm is used as the date to transform. The result of the function is 3600 which is the number of seconds between the origin and the date which is equal to one hour.

8.4.2 TimeToGeometry

The goal of the TimeToGeometry function is to transform time into geometry. Hence instants are transformed to points and periods are transformed into lines. The source code of the function is as follows:



```
Quelle
(pOrigin timestamp, pFrom timestamp, pTo timestamp)
return mdsys.sdo_geometry
as
vStart number;
vEnd number;
begin
  -- Check if input parameters are not null
  if pFrom is not null and pTo is not null then

    -- Transform the first date specification into a number
    vStart := TimestampToNumber(pOrigin, pFrom);
    -- Transform the second date specification into a number
    vEnd := TimestampToNumber(pOrigin, pTo);

    if vStart = vEnd then
      -- If the numbers are equal it is an instant which is transformed into point geometry
      return SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(vStart, 0, 0), NULL, NULL);
    else
      -- If the numbers are not equal it is a period which is transformed into a line geometry
      return SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(vStart,0,vEnd,0));
    end if;
  else
    return null;
  end if;
end;
```

Status: Valid (Ln, Col): (1,1) Kompilieren Fehler anzeigen

Figure 32 TimeToGeometry source code

The function takes as input three date specifications one for the origin and two for delimiting a period. The parameter pFrom is the date specification where a period starts. Hence the parameter pTo is the date specification where a period ends. If pFrom is equal to pTo then the period is considered as an instant. In the case of a period the return value of the function is a simple line geometry. In the case of an instant the return value is a simple point geometry. The functionality is to transform the input date specification into numbers via the TimestampToNumber function. Geometries are created through geometry constructors which take the calculated numbers as coordinates.

Time as Geometry – A prototypical implementation

The following figure shows an example of transforming an instant into a point geometry.

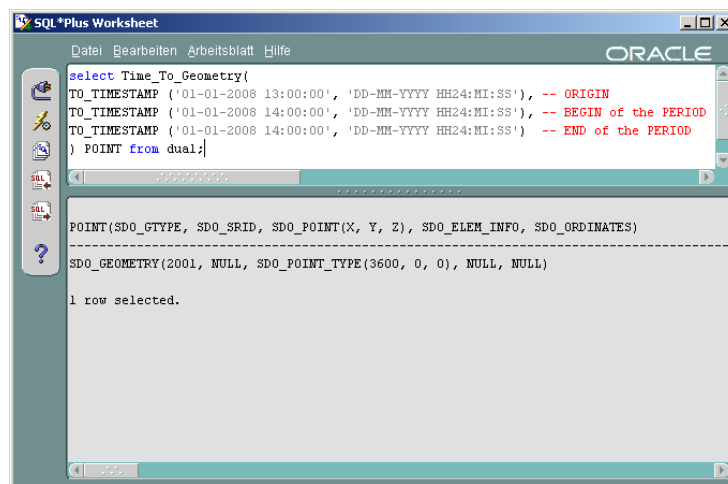


Figure 33 Transforming an instant into a point geometry

In this example January 1st at 1pm is used as the origin and January 1st 2pm is used as the start and end of a period which is therefore considered as an instant. The result of the function is a point geometry with 3600 as its x coordinate and 0 as its y coordinate.

The following figure shows an example of transforming a period into a line geometry.

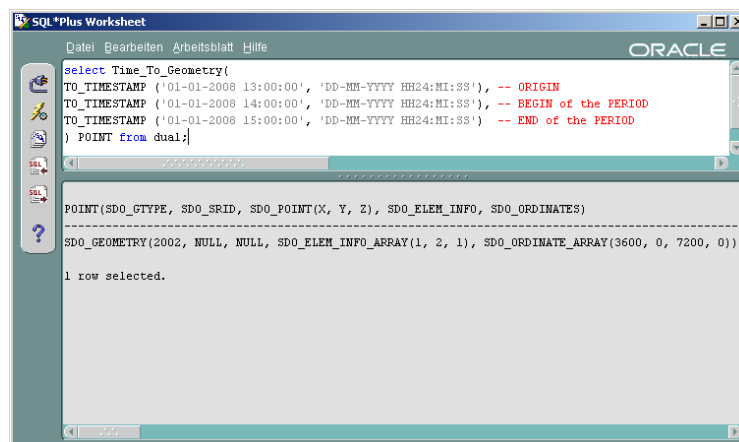
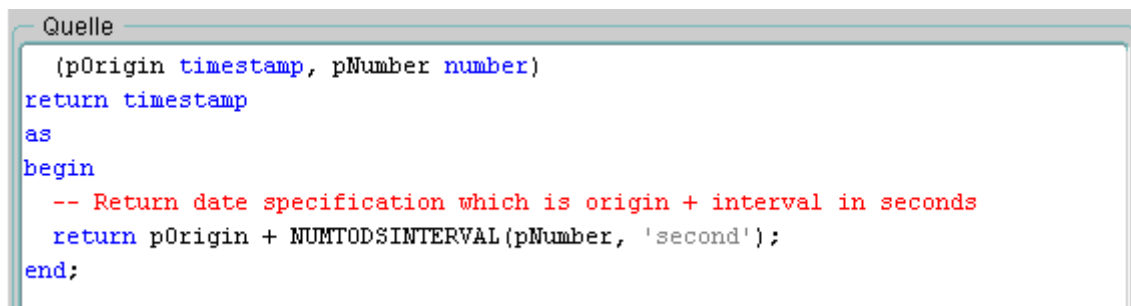


Figure 34 Transforming a period into a line geometry

In this example January 1st at 1pm is used as the origin and January 1st 2pm is used as the start and January 1st 3pm as the end of a period. The result of the function is a line geometry consisting of a start point with a coordinate pair of {3600,0} and an end point with a coordinate pair of {7200,0}.

8.4.3 NumberToTimeStamp

The goal of this function is to transform a number relative to an origin into a date specification. It is the natural counterpart of the TimeStampToNumber function. The following figure provides the source code of the function.



```

(pOrigin timestamp, pNumber number)
return timestamp
as
begin
  -- Return date specification which is origin + interval in seconds
  return pOrigin + NUMTODSINTERVAL(pNumber, 'second');
end;
```

Figure 35 NumberToTimeStamp source code

The following figure shows an example of transforming a number into a date specification.

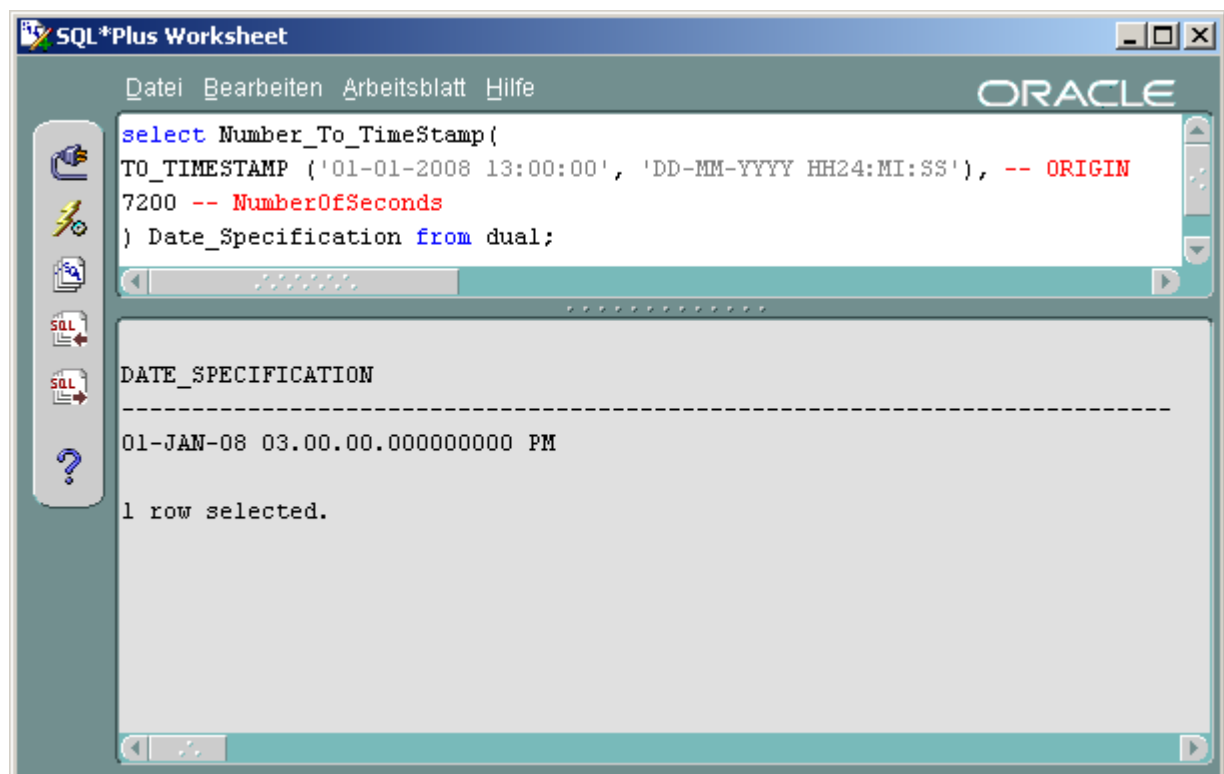
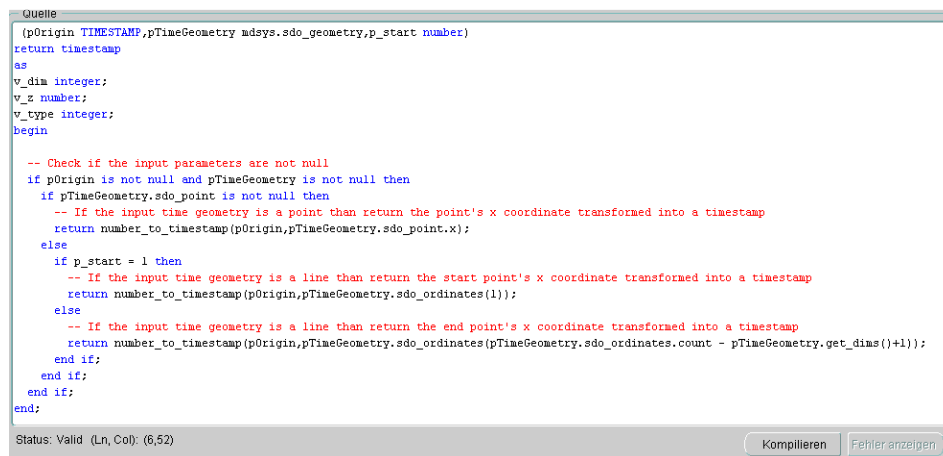


Figure 36 Transforming a number into a date specification

In this example January 1st at 1pm is the origin and the number to transform is 7200. The result is a date specification of January 1st at 3pm.

8.4.4 GeometryToTime

The goal of the function is to transform a geometry into a date specification. It is the natural counterpart to the TimeToGeometry function. This function is only necessary for displaying date specifications in a tabular manner rather than coordinates of the used geometries. It provides a better understanding for the results of ongoing examples. The following figure provides the source code of the function.

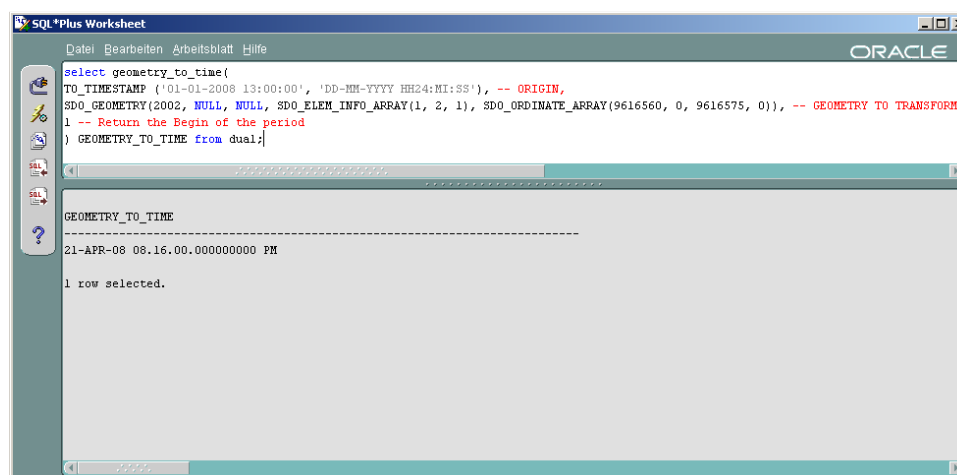


```
Quelle
(pOrigin TIMESTAMP,pTimeGeometry mdsys.sdo_geometry,p_start number)
return timestamp
as
v_dim integer;
v_z number;
v_type integer;
begin
-- Check if the input parameters are not null
if pOrigin is not null and pTimeGeometry is not null then
if pTimeGeometry.sdo_point is not null then
-- If the input time geometry is a point then return the point's x coordinate transformed into a timestamp
return number_to_timestamp(pOrigin,pTimeGeometry.sdo_point.x);
else
if p_start = 1 then
-- If the input time geometry is a line then return the start point's x coordinate transformed into a timestamp
return number_to_timestamp(pOrigin,pTimeGeometry.sdo_ordinates(1));
else
-- If the input time geometry is a line then return the end point's x coordinate transformed into a timestamp
return number_to_timestamp(pOrigin,pTimeGeometry.sdo_ordinates(pTimeGeometry.sdo_ordinates.count - pTimeGeometry.get_dims()+1));
end if;
end if;
end if;
end;
```

Status: Valid (Ln, Col): (6,52) Kompilieren Fehler anzeigen

Figure 37 GeometryToTime source code

The function takes as input a date specification of the origin, a geometry to transform and a number indicating whether begin or the end of the period should be returned. The following figure shows an example of the function.



```
SQL*Plus Worksheet
Datei Bearbeiten Arbeitsblatt Hilfe
ORACLE

select geometry_to_time(
TO_TIMESTAMP ('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'), -- ORIGIN,
SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(9616560, 0, 9616575, 0)), -- GEOMETRY TO TRANSFORM
1 -- Return the Begin of the period
) GEOMETRY_TO_TIME from dual;
```

GEOMETRY_TO_TIME

21-APR-08 08.16.00.000000000 PM

1 row selected.

Figure 38 Transforming geometry into a date specification

8.5 Spatial operations in temporal context

The scope of this chapter is to use spatial operations to determine relationships in temporal context. For each relationship two periods are transformed into line geometries via the `Time_To_Geometry` function. Afterwards the spatial relationship of those line geometries is determined via the Oracle Locator *relate* function. As a result the determined spatial relationships are evaluated of their expressiveness in the temporal case.

As an introduction to this topic Ott and Swiaczny (Ott & Swiaczny, 2001) provided a mapping between spatial and temporal operators.

spatial operators	temporal operators
disjoint	early/late
meets	meets
equals	equals
contains	during
covers	starts/finishes
overlaps	overlaps
intersects	intersects

Figure 39 Spatial and temporal operators (Ott & Swiaczny, 2001)

The following table provides a mapping of the identified temporal operations in literature to their counterparts within Oracle Locator.

Temporal operator	Oracle Locator Operator
before / after	
meets	SDO_TOUCH
equals	SDO_EQUAL
during	SDO_CONTAINS
starts/finishes	SDO_COVERS
Overlaps	SDO_OVERLAPS

Table 2 Temporal operators vs. Oracle Locator operations

The following table provides a mapping of temporal set operations and their counterparts within Oracle Locator.

Temporal set operations	Oracle Locator set operations
intersection	SDO_GEOM.SDO_INTERSECTION
union	SDO_GEOM.SDO_UNION
difference	SDO_GEOM.SDO_DIFFERENCE

Table 3 Temporal set operations vs. Oracle set operations

Time as Geometry – A prototypical implementation

8.5.1 Disjoint

The following figure shows the disjoint relationship as the two periods do not have any spatial interact. The first period lasts from January 1st 2pm to 5pm. The second period lasts from January 1st 7pm to 8pm. Both are converted into a line geometry via the Time_To_Geometry function. As they do not have any spatial interact they do not have any temporal interact as well and their relationship is disjoint.

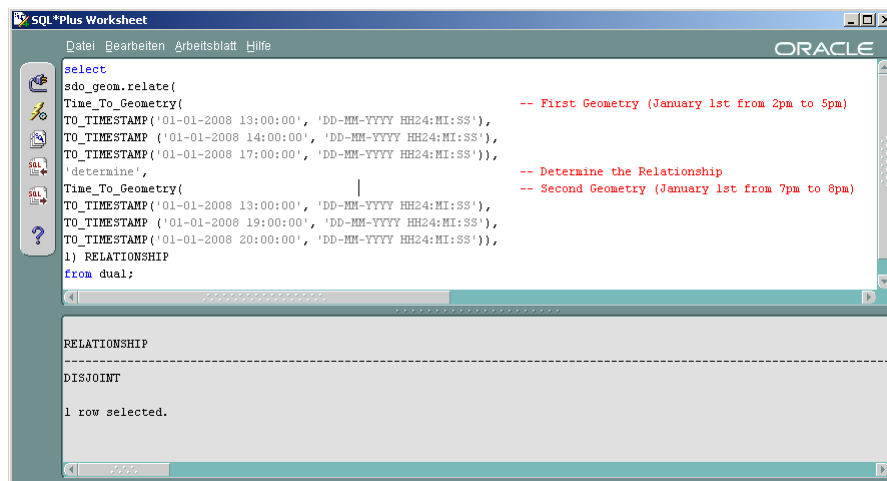


Figure 40 Disjoint relationship

8.5.2 Touch

The following figure shows the touch relationship as the two periods have a spatial interact at their boundary. The first period lasts from January 1st 2pm to 3pm. The second period lasts from January 1st 3pm to 4pm. Both are converted into a line geometry via the Time_To_Geometry function. At 3pm the first period ends and the second starts so their line geometries have this point in common. As a result the touch relationship applies.

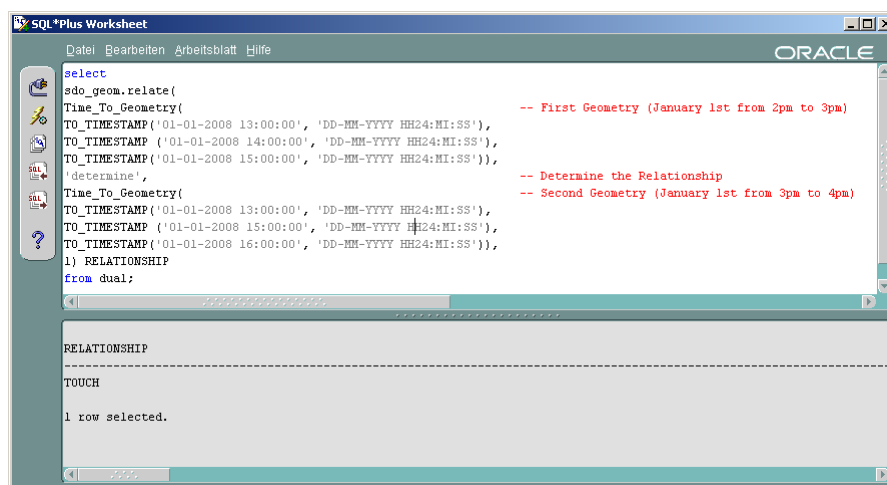


Figure 41 Touch relationship

8.5.3 Covers

The following figure shows the covers relationship as the two periods have a spatial interact at their interior and at their boundary. The first period lasts from January 1st 2pm to 5pm. The second period lasts from January 1st 4pm to 5pm. Both are converted into a line geometry via the Time_To_Geometry function. The line geometry of the second period is within the first period. Both geometries share the point at 5pm. As a result their interior and their boundary have spatial interact and therefore the covers relationship applies.

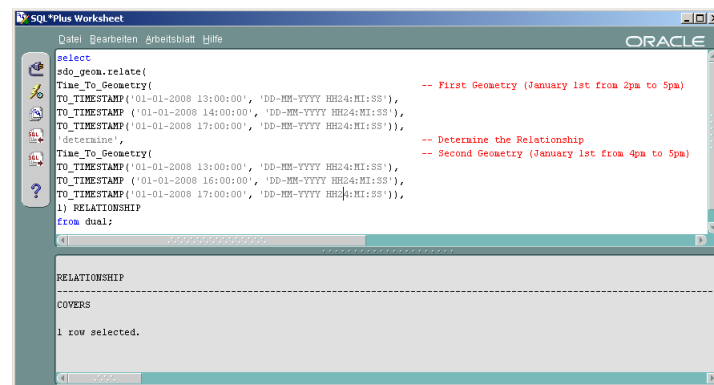


Figure 42 Covers relationship

8.5.4 Contains

The following figure shows the contains relationship as the two periods have a spatial interact at their interior. The first period lasts from January 1st 2pm to 5pm. The second period lasts from January 1st 3pm to 4pm. Both are converted into a line geometry via the Time_To_Geometry function. The line geometry of the second period is completely within the first period's line geometry therefore the contains relationship applies. The difference between contains and covers is that for a *covers* relationship the boundaries of the lines must interact.

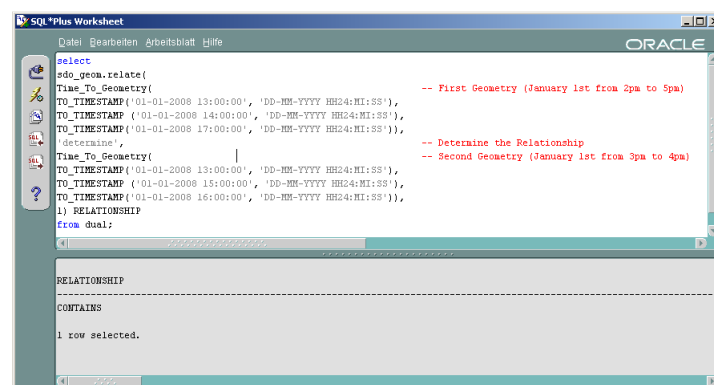


Figure 43 Contains relationship

8.5.5 Equal

The following figure shows the equal relationship as the two periods have the same interior and boundary. The first period lasts from January 1st 2pm to 3pm. The second period lasts from January 1st 2pm to 3pm too. Both are converted into a line geometry via the Time_To_Geometry function. The line geometries of both periods are the same therefore the equal relationship applies.

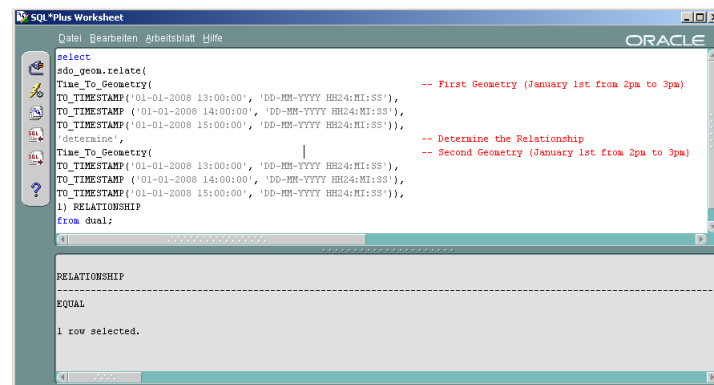


Figure 44 Equal relationship

8.5.6 Overlap

The following figure shows the overlap relationship as the two periods have a spatial interact at their interior and their boundary. The first period lasts from January 1st 2pm to 5pm. The second period lasts from January 1st 1pm to 4pm. Both are converted into a line geometry via the Time_To_Geometry function. Their line geometries share the period between 2pm and 4pm therefore the overlap relationship applies. In the case of line geometries the overlap relationship is called overlapbdydisjoint which means that an overlap applies but the boundaries are disjoint.

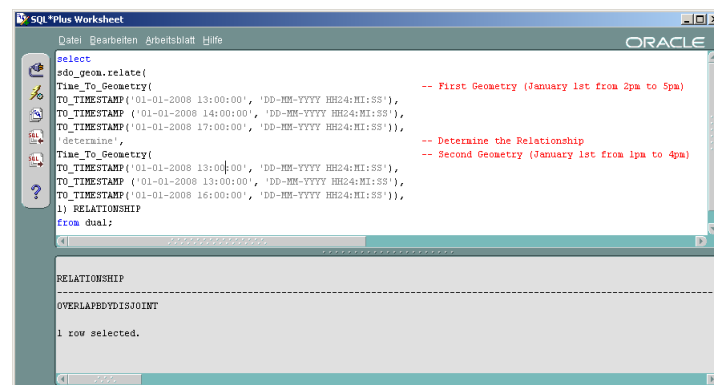
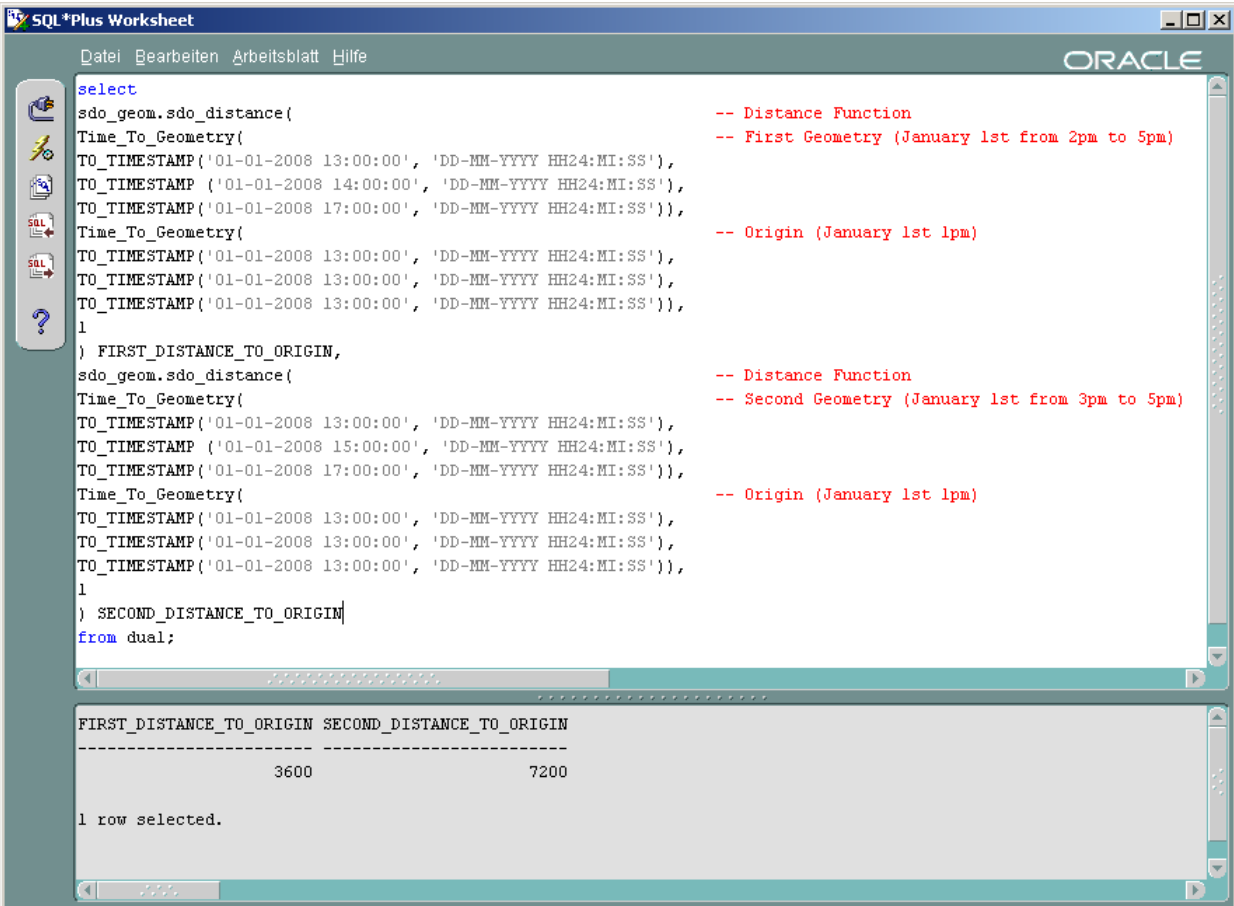


Figure 45 Overlap relationship

8.5.7 Before or After

Within Oracle Locator there exist no spatial operator considering a before or after relationship because in space there is basically no direction. But it is possible to provide a before or after relationship via the linear distance function. The first period lasts from January 1st 2pm to 5pm. The second period lasts from January 1st 3pm to 5pm. Both are converted into a line geometry via the Time_To_Geometry function. First the linear distance of both line geometries to the origin is calculated via the distance function. The origin is January 1st 1pm therefore the linear distance of the first period is 3600 spatial units (= 3600 seconds = 1 hour) and the distance of the second period is 7200 units (=7200 seconds = 2 hours). A period's line geometry with a smaller distance to the origin always starts before a period's line geometry with a greater distance. As a result it is possible to determine a before or after relationship by comparing the distances of periods to their origin.



```

select
sdo_geom.sdo_distance(
Time_To_Geometry(
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 14:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 17:00:00', 'DD-MM-YYYY HH24:MI:SS')),
Time_To_Geometry(
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS')),
1
) FIRST_DISTANCE_TO_ORIGIN,
sdo_geom.sdo_distance(
Time_To_Geometry(
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 17:00:00', 'DD-MM-YYYY HH24:MI:SS')),
Time_To_Geometry(
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS')),
1
) SECOND_DISTANCE_TO_ORIGIN
from dual;

```

FIRST_DISTANCE_TO_ORIGIN	SECOND_DISTANCE_TO_ORIGIN
3600	7200

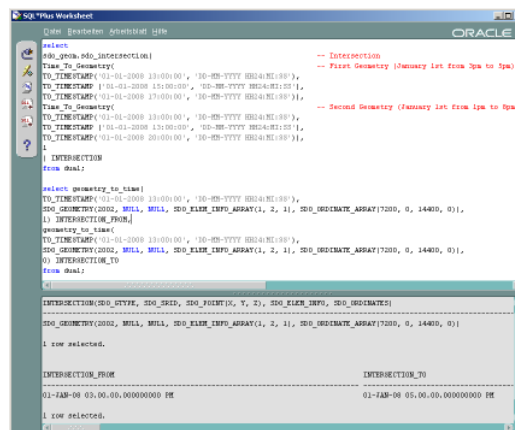
1 row selected.

Figure 46 Before or After relationship

Time as Geometry – A prototypical implementation

8.5.8 Intersection

The following figure shows an intersection of two periods. The first period lasts from January 1st 3pm to 5pm. The second period lasts from January 1st 1pm to 8pm. Both are converted into a line geometry via the Time_To_Geometry function. Their line geometries share the period between 3pm and 5pm which is the result of the intersection. The second statement in Figure 47 transforms the result into readable date specifications via the Geometry_To_Time function.



```
SQL> select
  sdo_geom_sdo_intersection(
    Time_To_Geometry(
      TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 20:00:00', 'DD-MM-YYYY HH24:MI:SS')
    ),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(7200, 0, 14400, 0)),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(7200, 0, 14400, 0))
  )
  from dual;

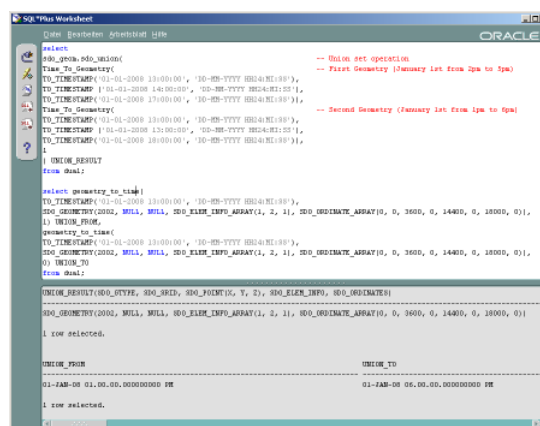
INTERSECTION
-----
select geometry_to_time(
  TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
  sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(7200, 0, 14400, 0)),
  1) INTERSECTION_FROM,
  geometry_to_time(
    TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(7200, 0, 14400, 0)),
    2) INTERSECTION_TO
  from dual;

INTERSECTION_FROM           INTERSECTION_TO
-----
01-JAN-09 03.00.000000000 PM 01-JAN-09 05.00.000000000 PM
1 row selected.
```

Figure 47 Intersection operation

8.5.9 Union

The following figure shows the union of two periods. The first period lasts from January 1st 3pm to 5pm. The second period lasts from January 1st 1pm to 6pm. Both are converted into a line geometry via the Time_To_Geometry function. The union of their line geometries is equal to the union of the periods which is from 1pm to 6pm. The second statement in Figure 48 transforms the result into readable date specifications via the Geometry_To_Time function.



```
SQL> select
  sdo_geom_sdo_union(
    Time_To_Geometry(
      TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-JAN-2009 20:00:00', 'DD-MM-YYYY HH24:MI:SS')
    ),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(0, 3600, 0, 14400, 0, 18000, 0)),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(0, 3600, 0, 14400, 0, 18000, 0))
  )
  from dual;

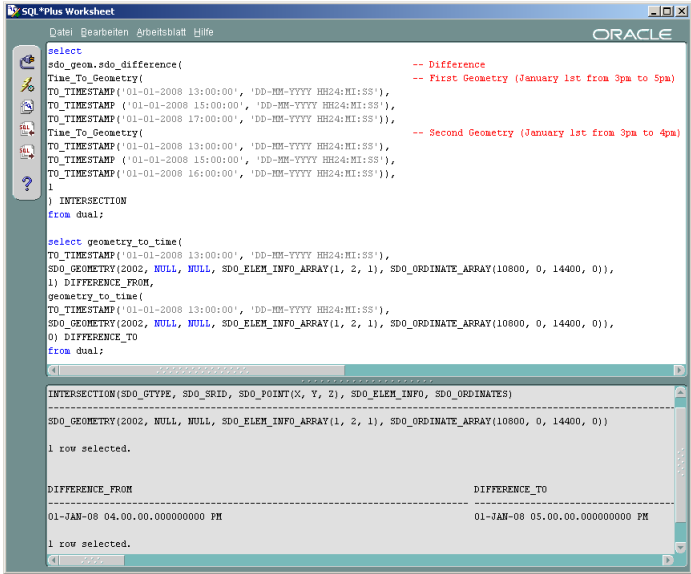
UNION_RESULT
-----
select geometry_to_time(
  TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
  sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(0, 3600, 0, 14400, 0, 18000, 0)),
  1) UNION_FROM,
  geometry_to_time(
    TO_TIMESTAMP('01-JAN-2009 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    sdo_geometry(2002, MULL, MULL, sdo_elem_info_array(1, 2, 1), sdo_ordinate_array(0, 3600, 0, 14400, 0, 18000, 0)),
    2) UNION_TO
  from dual;

UNION_FROM           UNION_TO
-----
01-JAN-09 01.00.000000000 PM 01-JAN-09 06.00.000000000 PM
1 row selected.
```

Figure 48 Union operation

8.5.10 Difference

The following figure shows the difference of two periods. The first period lasts from January 1st 3pm to 5pm. The second period lasts from January 1st 3pm to 4pm. Both are converted into a line geometry via the Time_To_Geometry function. Their line geometries do not share the period between 4pm and 5pm which is the result of the difference. The second statement in Figure 49 transforms the result of the difference operation into readable date specifications via the Geometry_To_Time function.



```

select
  sdo_geom.sdo_difference(
    Time_To_Geometry(
      TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 17:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      1
    ),
    Time_To_Geometry(
      TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 16:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      1
    )
  ) INTERSECTION
from dual;

select geometry_to_time(
  TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
  SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(10800, 0, 14400, 0)),
  1) DIFFERENCE_FROM,
  geometry_to_time(
    TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(10800, 0, 14400, 0)),
    0) DIFFERENCE_TO
from dual;

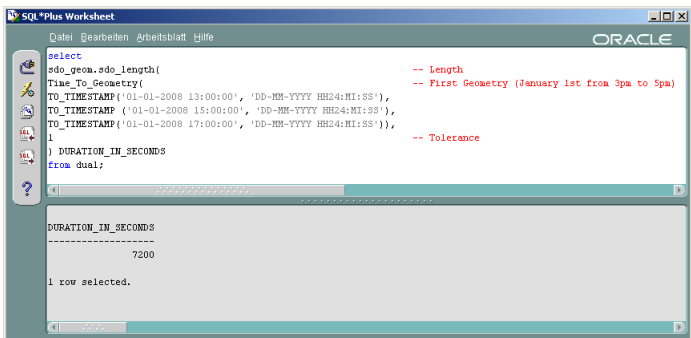
```

DIFFERENCE_FROM	DIFFERENCE_TO
01-JAN-08 04.00.00.000000000 PM	01-JAN-08 05.00.00.000000000 PM

Figure 49 Difference Operation

8.5.11 Length

The following figure shows a length calculation of a period. The period lasts from January 1st 3pm to 5pm. It is converted into a line geometry via the Time_To_Geometry function. The length of the period's line geometry is 7200 spatial units. In the temporal context these are 7200 seconds respectively the period's duration is 2 hours.



```

select
  sdo_geom.sdo_length(
    Time_To_Geometry(
      TO_TIMESTAMP('01-01-2008 13:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      TO_TIMESTAMP('01-01-2008 17:00:00', 'DD-MM-YYYY HH24:MI:SS'),
      1
    ),
    1
  ) DURATION_IN_SECONDS
from dual;

```

DURATION_IN_SECONDS
7200

Figure 50 Length calculation

8.6 The bus station scenario

The goal of this chapter is to provide interesting spatio temporal use cases based on a bus station scenario. For example a spatio temporal question on this scenario would be *“Find those stations where more than one bus line stop in the same period of time within a walking distance of 1km within the next half an hour?”*. The result of such a question gives information of when and where it is convenient to change the bus line. The following figure gives an overview of the bus lines and their stations.

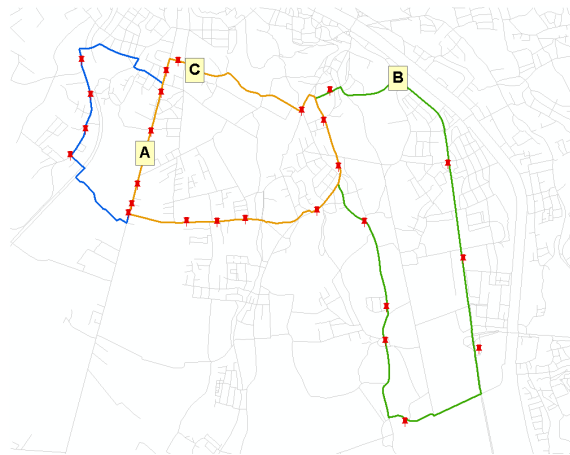


Figure 51 Bus station scenario

There are three bus lines called A,B and C. Each bus line has a number of stations where the bus stops. At each bus station there is a schedule providing the information when a bus arrives. There are stations where more than one bus line stops. The time a bus stops at a bus station is implemented as a period and not as an instant. It is closer to reality as a bus takes time at the station when passengers enter or exit. For this scenario it is assumed that a bus needs 15 seconds from the arrival to the departure at a station. At end stations this period may be longer. In order to reuse the functionality of spatial operations the period is transformed into a line geometry via the Time_To_Geometry function.

Chapter 8.6.1 provides an overview of the Entity Relationship Model of the bus station scenario. How the initial data loading was accomplished is described in chapter 8.6.2. In the following chapters spatial, temporal and spatio-temporal questions are formulated and translated into SQL statements. For spatial and temporal questions only spatial operations are used as time is modelled as geometry. At the end the results of the statements are discussed.

8.6.1 Entity Relationship Model

The following figure shows the Entity Relationship Model of the bus station scenario.

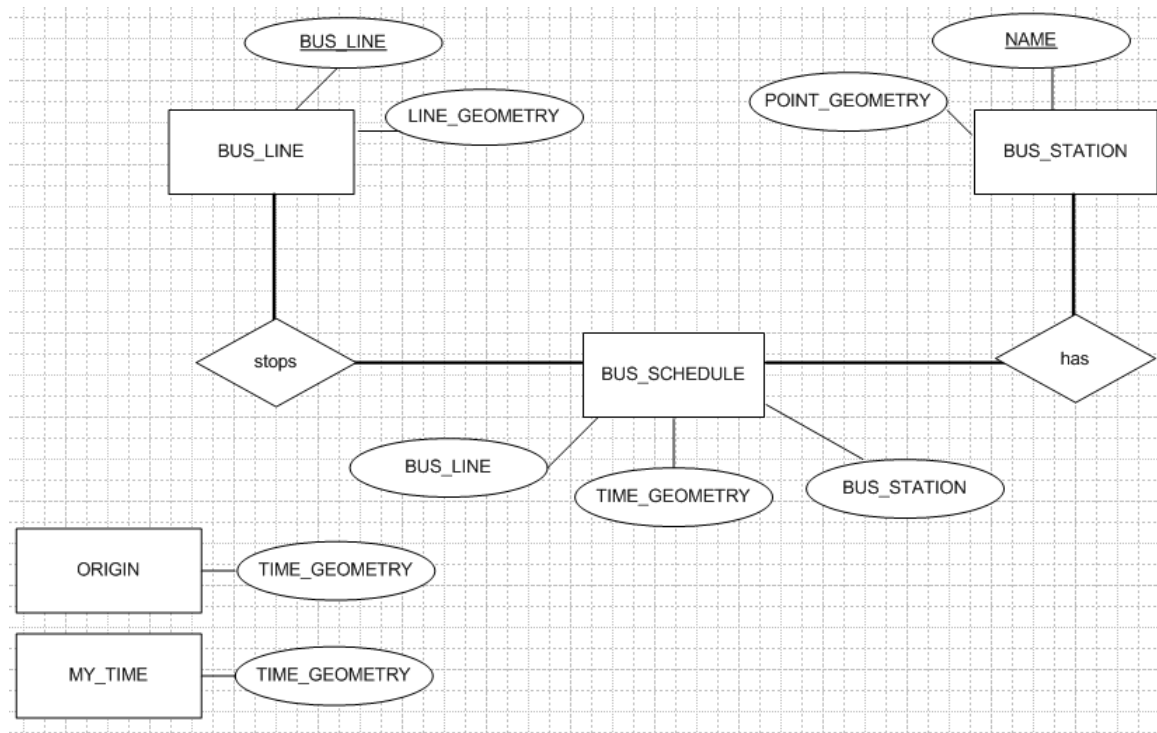


Figure 52 Bus station ERM

The BUS_LINE table consists of a BUS_LINE attribute for the name of the bus line which is also the primary key of that table. For simplicity it is assumed that there are no two bus lines with the same name. The LINE_GEOMETRY attribute is for a line geometry which represents the line of the bus connecting the bus stations. The BUS_STATION table consists of a name attribute for the name of the station and a point geometry for the location of the station. The most important table is the BUS_SCHEDULE table as it stores the relevant information when a bus stops at a station and is therefore the weak-entity between BUS_LINE and BUS_STATION. It consists of the attributes BUS_LINE and BUS_STATION and a line geometry representing the period of time a bus stops at a station. The ORIGIN table consists of one TIME_GEOMETRY attribute for the origin which is used in following SQL statements. The MY_TIME table consists of one TIME_GEOMTRY attribute for a local time which is used in following SQL statements. For both tables the TIME_GEOMETRY attribute represents an instant in time as a point geometry. Both tables are helper tables as following SQL statements are more readable when an origin and a local time can be used by joining those tables.

8.6.2 Initial Data Loading

The scope of this chapter is to give a short distinctive overview of the initial data loading of the provided data model. The BUS_STATION and BUS_LINE tables are filled with conventional SQL insert statements. The following figure shows an insert statement into the BUS_SCHEDULE table converting a period of time into a line geometry.

```
insert into bus_schedule(bus_line, bus_station, time)
values
(
  'A',
  'Volksschule Moos',
  TIME_TO_GEOMETRY(
    TO_TIMESTAMP('01-01-2008 00:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('21-04-2008 05:50:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('21-04-2008 06:00:00', 'DD-MM-YYYY HH24:MI:SS')
  )
);
```

-- bus line
-- bus station
-- time to geometry function
-- origin (January 1st 00:00)
-- arrival (April 21st 05:50)
-- departure (April 21st 06:00)

Figure 53 Inserting values into the bus schedule

Figure 53 shows the information that bus line 'A' arrives at station 'Volksschule Moos' at April 21st 5am and departs at 6am. This period lasts for ten minutes because this station is the end station. It is converted into a line geometry via the Time_To_Geometry function. In order to use the same origin throughout following SQL statements an origin of January 1st 00:00am is inserted into the origin table. Figure 54 shows the insert statement.

```
insert into origin values (
  Time_To_Geometry(
    TO_TIMESTAMP('01-01-2008 00:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('01-01-2008 00:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('01-01-2008 00:00:00', 'DD-MM-YYYY HH24:MI:SS')
  )
);
```

Figure 54 Definition of the origin

In chapter 0 a local time for a spatio temporal question is needed. In order to reuse this local time in SQL statements and make them more readable a date specification of April 21st 07:40am is inserted into the MY_TIME table. Figure 55 shows the insert statement.

```
insert into my_time values (
  Time_To_Geometry(
    TO_TIMESTAMP('01-01-2008 00:00:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('21-04-2008 07:40:00', 'DD-MM-YYYY HH24:MI:SS'),
    TO_TIMESTAMP('21-04-2008 07:40:00', 'DD-MM-YYYY HH24:MI:SS')
  )
);
```

Figure 55 Definition of my time

Time as Geometry – A prototypical implementation

As a result the following figure shows a graphical representation of some rows of the bus schedule table.



Figure 56 Bus schedule viewed on the time line

The period between a bus arrival and departure is transformed into a line geometry which is labelled with the name of the bus station. Bus line A is represented as a red line, bus line B as a green line and bus line C as a blue line. In order to avoid display conflicts the bus lines are displayed with different y offsets.

8.6.3 Spatial Questions

The following figure shows an example of a conventional spatial query within Oracle Locator. Therefore the aspect of time is not considered. For this example the originating location is at the station “Sinnhubstraße”. Now the spatial question is to find those stations which are within a linear distance of 200 meters of the originating location.

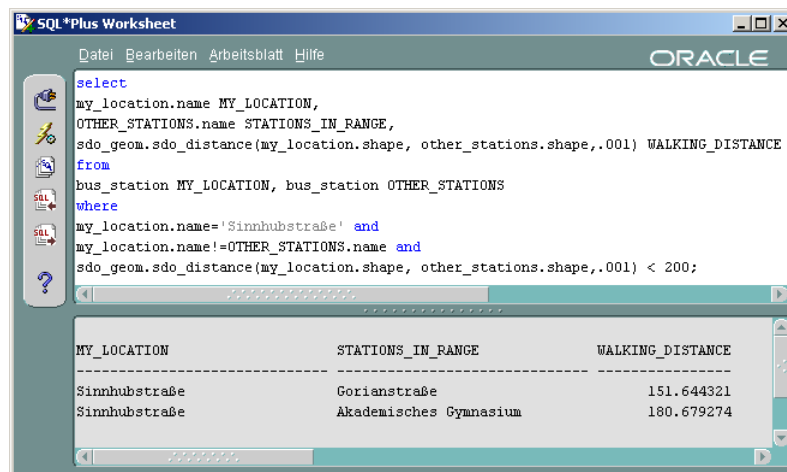


Figure 57 Spatial question

Figure 57 shows the solution to the above stated question. The linear distance is calculated via the SDO_GEOM.SDO_DISTANCE function and is compared to be less than 200 meters. As a result there are two stations within a distance of 151.644 meters respectively 180.679 meters away from “Sinnhubstraße”.

8.6.4 Temporal Questions

The following figure shows an example of a temporal query within Oracle Locator. Therefore the aspect of space is not considered. For this example the period of interest is between April 21st 2008 07:45am and 08:00 am. Now the temporal question is to find those stations at which a bus arrives or departures within that period of time.

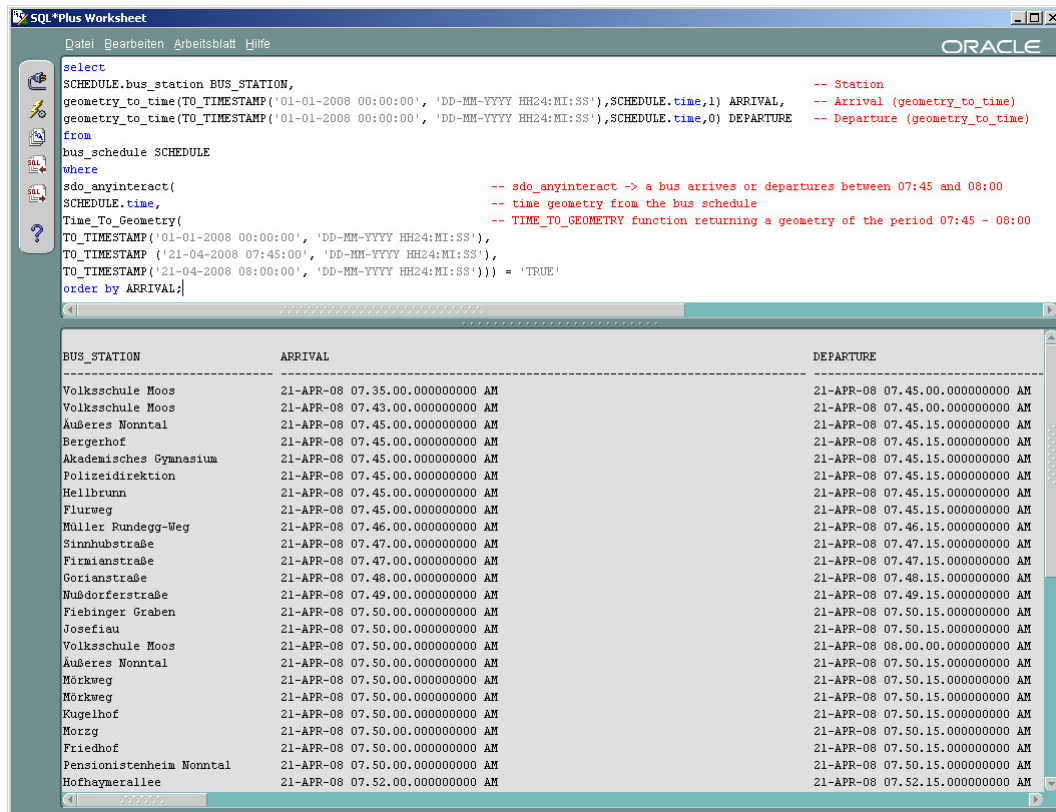


Figure 58 Temporal question

Figure 58 shows the SQL statement to the above stated question. The period of interest is converted into a line geometry via the Time_To_Geometry function. It is then used within a SDO_ANYINTERACT spatial operator in order to find spatially interacting periods of the BUS_SCHEDULE table which are represented as line geometries too. As the spatial operator SDO_ANYINTERACT searches for any spatial relationship it returns all bus stations where a bus arrives or departures within the period of interest. In order to provide a readable result the periods' line geometry is converted back into two timestamps via the Geometry_To_Time function. The result shows that there are many bus stations where a bus stops or arrives between 07:45 and 08:00.

8.6.5 Spatio-temporal questions

The goal of this chapter is to provide interesting questions as space and time are considered together. For this example the originating location is at the bus station “Sinnhubstraße”. The time of interest is April 21st 2008 at 07:40am. Now the spatio temporal question is to find those stations within a linear distance of 500 meters where a bus stands at a bus station at 07:40am.

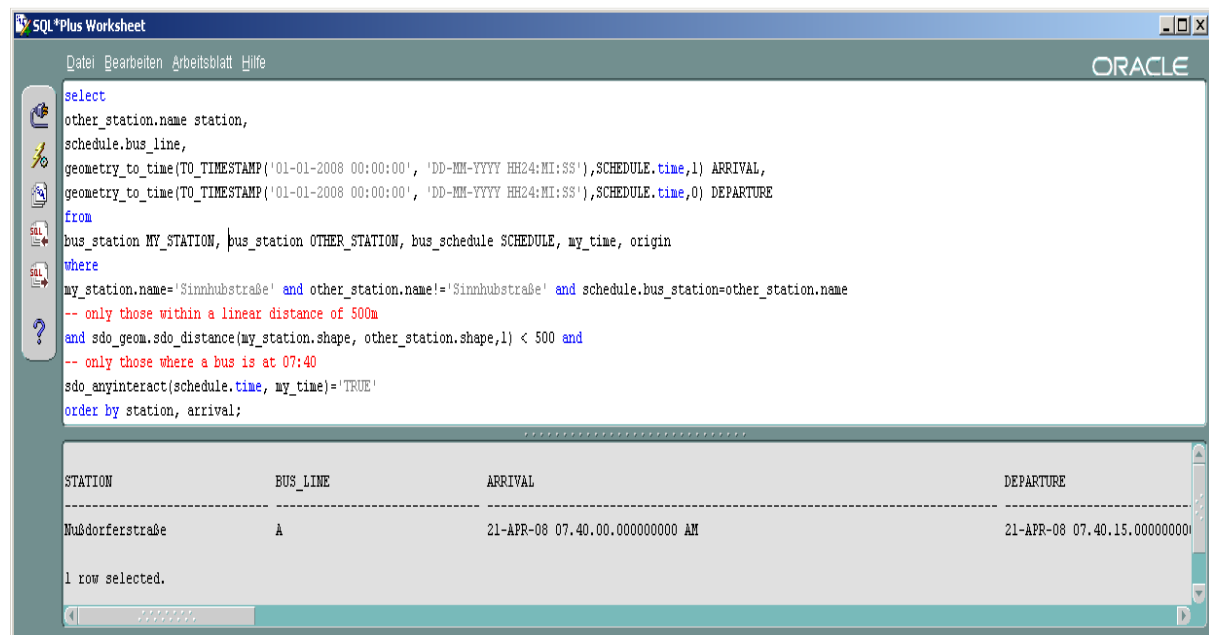
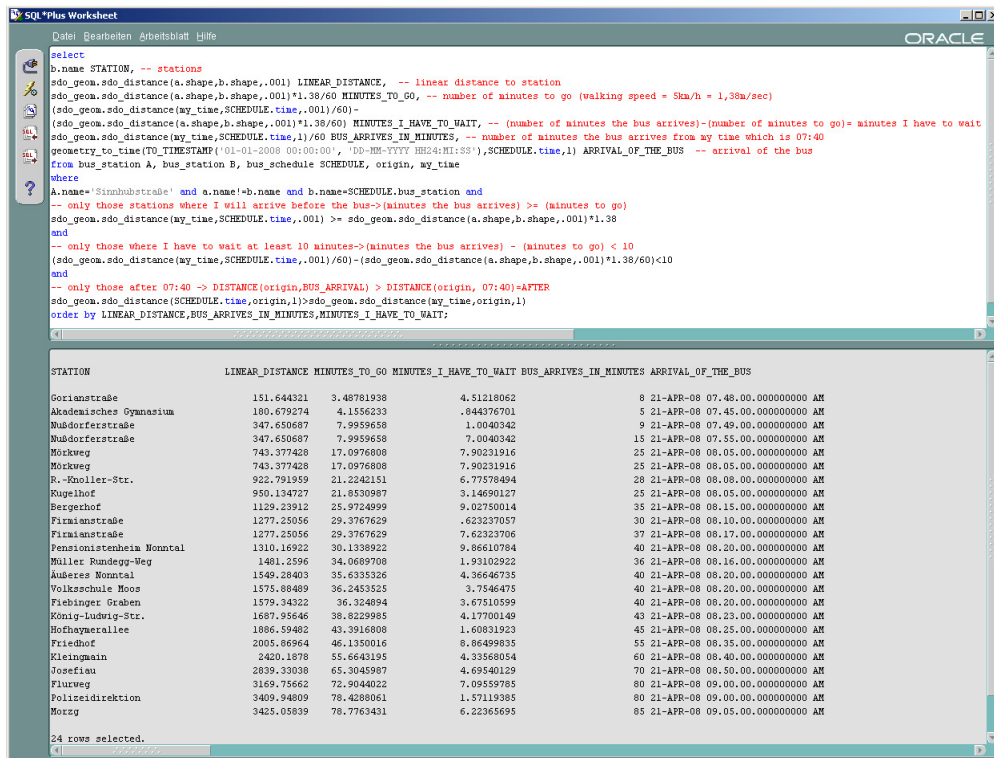


Figure 59 Spatio temporal Question I

Figure 59 shows the SQL statement of the above stated question. As the time of interest is a time instant it is converted into a point geometry via the Time_To_Geometry function. Afterwards it is used within a SDO_ANYINTERACT spatial operator in order to find spatially interacting periods of the BUS_SCHEDULE table which are represented as line geometries. As the spatial operator SDO_ANYINTERACT searches for any spatial relationship it returns all bus stations where a bus stands at the station at 07:40am. In order to provide a readable result the bus schedule periods' line geometry is converted back into two timestamps via the Geometry_To_Time function. The result shows that there is only one bus station where bus line “A” stops at 07:40am. As mentioned earlier it is assumed that it takes a bus at least 15 seconds to handle incoming and leaving passengers.

Time as Geometry – A prototypical implementation

The example points out the power of combining space and time into spatio temporal questions. Again the originating location is at “Sinnhubstraße” and the time of interest is April 21st 2008 at 07:40am. Now the question is to find those stations where one will arrive before a bus and must not wait longer than for ten minutes at that station. Further for each station the result should provide the information how long one has to go to arrive there. A walking speed of 5km/h is assumed.



```
select
b.name STATION, -- stations
sdo_geom.sdo_distance(a.shape,b.shape,.001) LINEAR_DISTANCE, -- linear distance to station
sdo_geom.sdo_distance(a.shape,b.shape,.001)*1.38/60 MINUTES_TO_GO, -- number of minutes to go (walking speed = 5km/h = 1,38m/sec)
(sdo_geom.sdo_distance(my_time,SCHEDULE.time,.001)/60)-
(sdo_geom.sdo_distance(a.shape,b.shape,.001)*1.38/60) MINUTES_I_HAVE_TO_WAIT, -- (number of minutes the bus arrives)-(number of minutes to go)= minutes I have to wait
sdo_geom.sdo_distance(my_time,SCHEDULE.time,1)/60 BUS_ARRIVES_IN_MINUTES, -- number of minutes the bus arrives from my time which is 07:40
geometry_to_time(TO_TIMESTAMP('01-01-2008 09:00:00','DD-MM-YYYY HH24:MI:SS'),SCHEDULE.time,1) ARRIVAL_OF_THE_BUS -- arrival of the bus
from bus_station A, bus_station B, bus_schedule SCHEDULE, origin, my_time
where
A.name='Sinnhubstraße' and a.name!=b.name and b.name=SCHEDULE.bus_station and
-- only those stations where I will arrive before the bus-->(minutes the bus arrives) >= (minutes to go)
sdo_geom.sdo_distance(my_time,SCHEDULE.time,.001) >= sdo_geom.sdo_distance(a.shape,b.shape,.001)*1.38
and
-- only those where I have to wait at least 10 minutes-->(minutes the bus arrives) - (minutes to go) < 10
(sdo_geom.sdo_distance(my_time,SCHEDULE.time,.001)/60)-(sdo_geom.sdo_distance(a.shape,b.shape,.001)*1.38/60)<10
and
-- only those after 07:40 -> DISTANCE(origin,BUS_ARRIVAL) > DISTANCE(origin, 07:40)=AFTER
sdo_geom.sdo_distance(SCHEDULE.time,origin,1)>sdo_geom.sdo_distance(my_time,origin,1)
order by LINEAR_DISTANCE,BUS_ARRIVES_IN_MINUTES,MINUTES_I_HAVE_TO_WAIT;
```

STATION	LINEAR_DISTANCE	MINUTES_TO_GO	MINUTES_I_HAVE_TO_WAIT	BUS_ARRIVES_IN_MINUTES	ARRIVAL_OF_THE_BUS
Gorjanstraße	151.644321	3.48781938	4.51218062	8	21-APR-08 07.48.00.000000000 AM
Alteanisches Gymnasium	180.679274	4.1556233	.844376701	5	21-APR-08 07.45.00.000000000 AM
Waldorferstraße	347.650687	7.9959658	1.0040342	9	21-APR-08 07.49.00.000000000 AM
Waldorferstraße	347.650687	7.9959658	7.0040342	15	21-APR-08 07.55.00.000000000 AM
Morkweg	743.377428	17.0976808	7.90231916	25	21-APR-08 08.05.00.000000000 AM
Morkweg	743.377428	17.0976808	7.90231916	25	21-APR-08 08.05.00.000000000 AM
R.-Knoller-Str.	922.791959	21.2242151	6.77578494	28	21-APR-08 08.08.00.000000000 AM
Kugelhof	950.134727	21.8530987	3.14690127	25	21-APR-08 08.05.00.000000000 AM
Bergerhof	1129.23912	25.9724999	9.02750014	35	21-APR-08 08.15.00.000000000 AM
Falkenstraße	1277.25056	29.3767629	.623237057	30	21-APR-08 08.10.00.000000000 AM
Falkenstraße	1277.25056	29.3767629	7.62323706	37	21-APR-08 08.17.00.000000000 AM
Pensionistenheim Nontal	1310.16922	30.1338922	9.86610784	40	21-APR-08 08.20.00.000000000 AM
Müller Runderweg-Weg	1481.2596	34.0689708	1.93102922	36	21-APR-08 08.16.00.000000000 AM
Außeres Nontal	1549.28403	35.6335326	4.36646735	40	21-APR-08 08.20.00.000000000 AM
Volkschule Moos	1575.88489	36.2453525	3.7546475	40	21-APR-08 08.20.00.000000000 AM
Friedhof Gröben	1579.34322	36.324894	3.67910599	40	21-APR-08 08.20.00.000000000 AM
König-Ludwig-Str.	1687.95646	38.8229985	4.17700149	43	21-APR-08 08.23.00.000000000 AM
Hofmayerallee	1886.59482	43.3916808	1.60831923	45	21-APR-08 08.25.00.000000000 AM
Friedhof	2005.86964	46.1350016	8.86499835	55	21-APR-08 08.35.00.000000000 AM
Kleinmain	2420.1878	55.6643195	4.33568054	60	21-APR-08 08.40.00.000000000 AM
Josefau	2839.33038	65.3045987	4.69540129	70	21-APR-08 08.50.00.000000000 AM
Flurweg	3169.75662	72.9044022	7.09559785	80	21-APR-08 09.00.00.000000000 AM
Polizeidirektion	3409.94809	78.4288061	1.57119385	80	21-APR-08 09.00.00.000000000 AM
Morsg	3425.05839	78.7763431	6.22365695	85	21-APR-08 09.05.00.000000000 AM

24 rows selected.

Figure 60 Spatio temporal question II

Figure 60 shows the SQL statement to the above stated question. The time one has to go to the station is calculated via the SDO_GEOM.SDO_DISTANCE function which returns the distance in meters to a station. This distance is multiplied by 1.38 (5km/h= 1,38m/sec) to calculate the seconds to go which are then converted into minutes. The minutes one has to wait are calculated by subtracting the minutes to go from the number of minutes the bus arrives relative to 07:40. To find only those stations where one arrives before a bus is accomplished by comparing if the number of minutes the bus arrives is greater than the minutes one has to go there. The limitation that one must not wait longer than ten minutes at a station is accomplished by ensuring that the number of minutes a bus arrives minus the number of minutes to go there is less than 10 minutes. As a last step the statement ensures that the result only consists of bus stations where a bus stops after 07:40 because one

Time as Geometry – A prototypical implementation

cannot arrive at a bus station in the past. The result shows that there are 24 stations where one arrives before a bus and must not wait longer than ten minutes. Considering the last line of the result where one has to cover a distance of more than 3km and go for 78 minutes in order to wait at least ten minutes the query should be extended to limit the linear distance a station is away. Another lack of the query is that within the result a bus station is included more than once showing several bus arrivals. Maybe only the next bus arrival within ten minutes is of interest. This can be accomplished by including a minimum function. The following figure shows the same example as before but the identified limitations are corrected.

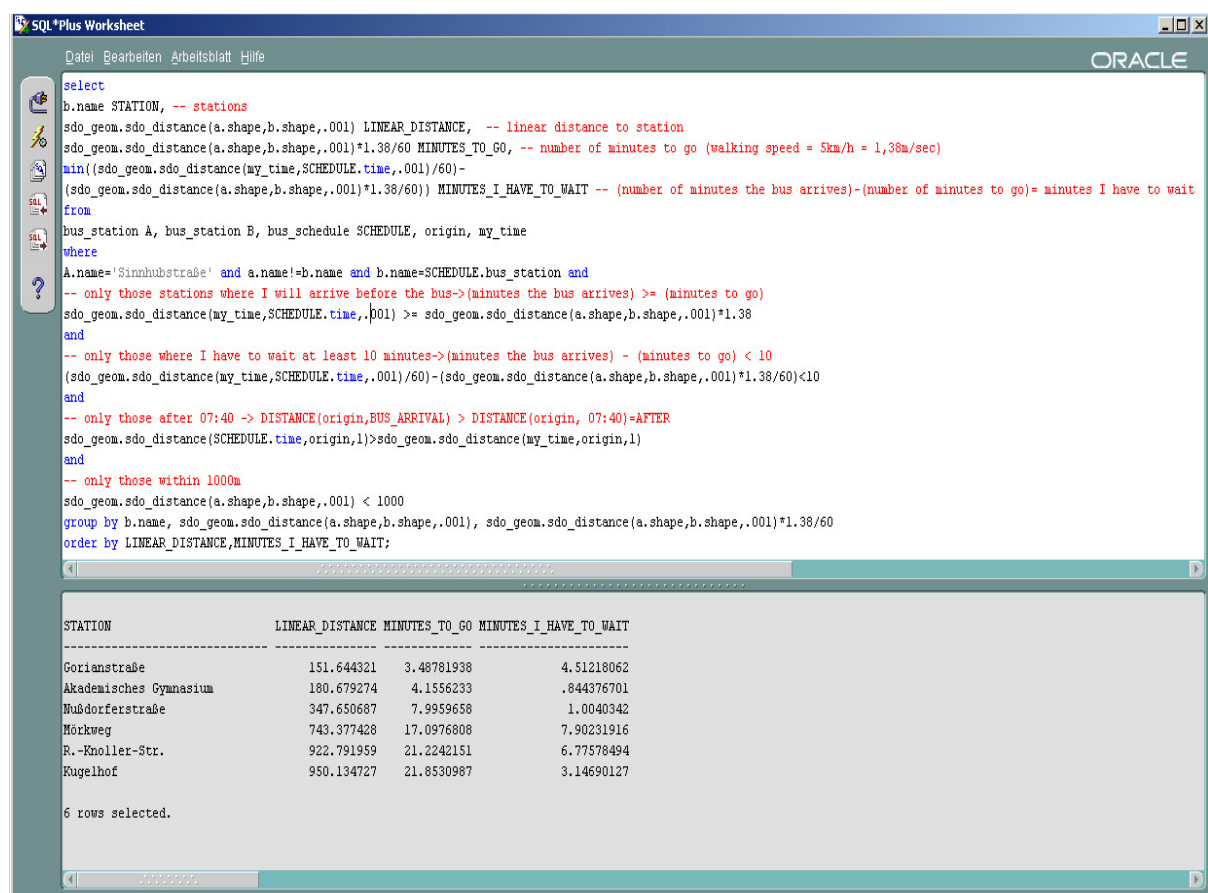


Figure 61 Spatio temporal question III

As a result there are 6 bus stations where one arrives before a bus and must not wait longer than for ten minutes. The station with the greatest distance is 950 meters where one has to go for 21 minutes. No station in the result is listed more than once as only the next arrival of a bus within ten minutes is of interest.

9. Summary

In the section of the background research this work identified the main contributions in the field of integrating time into geodatabases. The main foundation for relational database management systems is the relational model defined by Codd (Codd, 1970). This thesis reminded of the importance of this early work as it contributes to all spatial and temporal databases modelling paradigms. Afterwards this work described the main standards of the International Standardisation Organisation and the Open Geospatial Consortium concerning the integration of space into databases which resulted in geodatabases storing information about features. Besides that the temporal domain and its previous works defining the basic structure of time served as a foundation of how time is conceived within this thesis. Concepts of temporal relations were evaluated in order to identify problems and their solutions. For some of the identified solutions only minor adoptions were necessary to apply them to the ideas of this work. Finally the background research introduced the ISO 19018 standard with its innovative chapter describing the geometry of time.

The idea of transforming time into geometry in order to integrate it into a geodatabase is described in the main part. First formal descriptions of a discrete view of time based on calendar time and a discrete view of time based on an interval scale were provided. To transform time into geometry it was necessary to transform date specification into something like coordinates. Hence formal descriptions of functions were provided which transformed date specifications into numbers. A mapping between temporal primitives and geometric primitives resulted in temporal geometric primitives which supported a basic understanding of the geometry of time. The goal of the implementation was to evaluate the feasibility of integrating time into a geodatabase. Database Functions inside a geodatabase were developed in order to implement the functionality of the formal descriptions. Sample SQL statements using geometry constructors for creating instants and periods were provided. Based on a mapping between temporal and spatial operations this work applied spatial operations in temporal context. Finally a bus station scenario provided use cases for spatial, temporal and spatio temporal questions. This work showed how temporal questions are transformed into spatial questions and therefore solved via spatial operations. Finally interesting spatio-temporal questions were applied to the bus station scenario and delivered insights into data considering space and time together.

9.1 Conclusion

The main goal of this work was to transform time into geometry in order to integrate it into geodatabases. This goal had been achieved on a formal basis and also by the prototypical implementation within a geodatabase. Further it was of interest if spatial operations allow meaningful operations in temporal context applied to time represented as geometry. This work pointed out that spatial operations determining the relationship of geometries have an appropriate meaning in temporal context. Each of the spatial operation available in a geodatabase produced a meaningful temporal result. Therefore the implementation confirms what an expected result was.

The bus station scenario served as a basis for complex spatio temporal questions. As it includes space and time at an equivalent level it follows the idea of the ISO 19108 standard of spatio temporal objects. In the introduction of this work spatio temporal question had been formulated. What was achieved with the bus station scenario is that all temporal problems are solved only using functionality of a spatial framework. It turns out that regardless of the complexity of the question this work gets along without any other functionality than spatial for solving temporal problems.

The major conclusion of this work is that transforming time into geometry is an appropriate way to integrate time into geodatabases. Not only it is a practicable approach because the extensive functionality of a spatial framework can be reused but also time represented as geometry corresponds to our perception of time. It is a natural approach to consider an instant of time as a point and a period of time as line geometry. It turned out that time represented as geometry helps to understand temporal relationships of periods as we tend to imagine that a period has a linear representation on a time line. Finally it must be mentioned that the findings of earlier papers which had identified analogies between space and time apply to the findings of this work in many ways.

9.2 Future Research

Upon the provided concepts of modelling time as geometry further research is needed in the field of integrity constraints. Snodgrass (Snodgrass, 2000) identified that when integrating time into relations it is obvious that it should be considered as part of the primary key. This also applies to referential integrity. As a result there are concepts needed to include time which is represented as geometry into integrity constraints. In geodatabases geometry is implemented as a user defined data type which cannot be part of a primary key on a declarative level. Therefore the uniqueness of a primary key including geometry must be implemented as database functionality such as row triggers or assertions. Snodgrass had provided the basic concepts for that which may be adapted to work with geometry.

Another direction for future research are complex date specifications. It is an interesting question how a date specification *“Every Monday at 7 AM”* can be represented as geometry. A multipoint geometry for example may be an appropriate geometric representation of the above mentioned complex date specification. A multiline geometry for example may be an appropriate geometric representation of a date specification *“At weekends from 1pm to 3pm”*.

How to query the current state out of a temporal relation is another issue which is left open by this work. Existing approaches are using the maximum date available in a system as the end of a valid time period. This approach may be adapted to time represented as geometry where the line geometry of a valid time period extends to the maximum date the system provides. Therefore the maximum date must be defined which would be the third major metadatum besides the origin and the granularity.

The topic of metadata needs further investigation too. It is an important question whether spatial metadata of a period's line geometry apply in temporal context. An appropriate start point for metadata of time represented as geometry is the ISO 19108 standard (ISO, 2002) as it defines temporal reference systems.

10. Bibliography

Alsheikh, A. A. (n.d.). *Developing a Spatio-Temporal Database For Moving Objects*. Retrieved 03 20, 2008, from Developing a Spatio-Temporal Database For Moving Objects: http://www.gisdevelopment.net/proceedings/mapindia/2006/transportation/mi06tran_117.htm

Armstrong, W. A. (1974). Dependency Structures of Data Base Relationships. *IFIP Congress 1974*, (pp. 580-583).

Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A Structured English Query Language. *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control*, (pp. 249-264).

Chomicki, J. (2005). Time in Database Systems. In e. M. Fisher et al., *Handbook of Time in Artificial Intelligence* (pp. 1-58). Elsevier.

Clifford, J., & Tansel, A. U. (1985, May). On an algebra for historical relational databases: two views. *ACM SIGMOD Record* (volume 14 issue 4), pp. 247 - 265.

Codd, E. F. (1970). A Relational Model of Data for larged shared Data Banks. *Communications of the ACM* , pp. 64 - 69.

Codd, E. F. (1972). Relational Completeness of Data Base Sublanguages. In R. Rustin, *Database Systems* (pp. 65-98). California.

Frank, A., Egenhofer, M. J., & Colledge, R. G. (1998). Different Types of „Times“ in GIS. Spatial and Temporal Reasoning in GIS. *Oxford University Press* , pp. 40-61.

Güting, R. H., & Schneider, M. (2005). *Moving Objects Databases*. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Elsevier.

ISO. (2003, 05 01). *ISO 19107 Geographic information — Spatial Schema (draft version)*. Retrieved 03 20, 2008, from <https://committees.standards.org.au>.

ISO. (2002, 02 13). *ISO 19108 Geographic information - Temporal schema (draft version)*. Retrieved 03 20, 2008, from https://www.seegrid.csiro.au/twiki/pub/CGIModel/GeologicTime/19108_FDIS.pdf.

Jensen, C. S. (2000, April). *Temporal Database Management*. Retrieved 03 20, 2008, from Temporal Database Management: <http://www.cs.aau.dk/~csj/Thesis/>

Jensen, C. S., Snodgrass, R. T., & Soo, M. D. (1996, August). Extending Existing Dependency Theory to Temporal Databases. *IEEE Transactions on Knowledge and Data Engineering* (volume 8 - issue 4), pp. 563 - 582.

Jensen, C. S., Soo, M. D., & Snodgrass, R. T. (1994). Unifying temporal data models via a conceptual model. *Information Systems* (volume 19 - number 7), pp. 513-547.

<Bibliography

Kaiser, A. (1998). Neuere Entwicklungen auf dem Gebiet temporaler Datenbanken - eine kritische Analyse. *Proceedings des 6. Internationalen Symposiums für Informationswissenschaft (ISI 1998), Prag, 3. – 7. November 1998* (pp. 329 – 343). Prag: Konstanz: UVK Verlagsgesellschaft mbH.

OpenGIS_Consortium. (2006a, 10 05). *Simple feature access - Part 1: Common architecture*. Retrieved 03 20, 2008, from <http://www.opengeospatial.org/standards/sfa>.

OpenGIS_Consortium. (2006b, 10 05). *Simple feature access - Part 2: SQL option*. Retrieved 03 20, 2008, from <http://www.opengeospatial.org/standards/sfs>.

OpenGis_Consortium. (1999, 03 24). *Topic 5 - Features*. Retrieved 03 20, 2008, from <http://www.opengeospatial.org/standards/as>.

Ott, T., & Swiaczny, F. (2001). *Time-Integrative Geographic Information Systems*. Berlin, Deutschland: Springer-Verlag Berlin.

Peuquet, D. J. (2002). *Representations of Space and Time*. 72 Spring Street, New York, NY 10012: Guilford Publications.

Roosmann, R., Busch, W., Gorczyk, J., & Mauersberger, F. (2003). MODELLING SPATIOTEMPORAL OBJECTS AND PROCESSES AS A BASIS FOR MONITORING. *Proceedings 4th MATHMOD* (pp. 598-608). Wien: ARGESIM-Verlag.

Snodgrass, R. T. (2000). *Developing Time-Oriented Database Applications in SQL*. San Francisco, CA 94104-3205: Morgan Kaufmann Publishers.

Snodgrass, R. T. (1992). Temporal Databases. In I. C. A. U. Frank, *Theory and Methods of Spatio-Temporal Reasoning in Geographic Space* (pp. 22-64). Italy: Springer.

Somayeh, D., & Alesheikh, A. A. (n.d.). *Evaluating different approaches of spatial database management for moving objects*. Retrieved 03 20, 2008, from Evaluating different approaches of spatial database management for moving objects: http://www.gisdevelopment.net/technology/gis/me05_021.htm

Steiner, A. (1998). A Generalisation Approach to Temporal Data Models and their Implementations, PhD Thesis. ETH Zürich, Schweiz.

Worboys, M. F. (1994). A unified model of spatial and temporal information. *The computer journal*, vol 37 no. 1, pp. 26-34.