A Spatio-Temporal Multidimensional Model and a Query Language for Seasons

Modelo Multidimensional Espacio-Temporal y Lenguaje de Consulta para Temporadas

by

Francisco Javier Moreno Arboleda

A Thesis Submitted to the

Universidad Nacional de Colombia, Sede Medellín

in Partial Fulfillment of the

Requirements for the degree of

DOCTOR EN INGENIERÍA-SISTEMAS

Major Subject: Data Warehouses

Universidad Nacional de Colombia,
Sede Medellín
2010

Examining Committee

Fernando Arango Isaza, Ph.D,

Thesis Adviser,

Universidad Nacional de Colombia, Sede Medellín.

Renato Fileto, Ph.D,

Member,

Universidade Federal de Santa Catarina, Brazil.

John Freddy Duitama Muñoz, Ph.D,

Member,

Universidad de Antioquia, Colombia.

Gloria Lucía Giraldo Gómez, Ph.D,

Member,

Universidad Nacional de Colombia, Sede Medellín.

To my mother, for her love
To Paddy McAloon, for his talent
To George Michael, for his voice
To Mikhail Tal, for his mind

Acknowledgments

I would like to thank all the people who made this thesis possible. First, my adviser Professor Fernando Arango for his brilliant guidance. Without his supervision, this work would have been a mess!

I would also like to thank Professor Renato Fileto for agreeing to be on my thesis committee and for his valuable help, especially during my stay (should I say *season*? Now everything makes sense!) in Universidade Federal de Santa Catarina, Florianopolis. I wish someday I could write with his precise and elegant style.

A special thanks to Lynn Schlecker for helping me with the English language, she is not only my editor but just a friend.

I cannot forget the University where I grew up as an engineer, my *Alma Mater*, Universidad de Antioquia. Professors (and friends) Freddy Duitama, who introduced me in the world of databases, and Professor Roberto Flórez who showed me the world of algorithms.

Thanks to Universidad Nacional de Colombia and Colciencias for making my doctoral studies a feasible experience.

Finally, although it might sound a bit foolish, my deep (so deep!) gratitude to Terpsichore, muse of *music*, my main source of inspiration, what can I do? Without it, *I am nothing*…I wish I could list all my favorites artists here but it would require another thesis! Just a few of them: The Brand New Heavies, Ce Ce Peniston, Curiosity Killed the Cat, Dina Carroll, George Michael (the Maestro!, the Entertainer!), Kirsty Hawkshaw (Opus III, the voice of an angel), Lisa Moorish, Lisa Stansfield, Mica Paris, Prefab Sprout, Republica, Sonia Evans, Soul II Soul, Sophie Ellis-Bextor, Sunscreem, Swing Out Sister…For robust and emotional reasons this work is dedicated to them.
Thanks for making me happy.

Francisco J. Moreno A.
2010

Abstract

A data warehouse (DW) is usually modelled using a multidimensional view of data. In a multidimensional model a set of dimensions is associated with a subject of analysis called fact. Each dimension is composed of a non-empty set of levels which are organized hierarchically. Usually, a dimension is considered static in a DW; however, the dimension schema and dimension data can evolve. In this thesis, we focus on a type of dimension data change known as *reclassification*, *i.e.*, when a member (instance) of a level changes its parent (a member of a higher level of the same dimension). The first contribution of this thesis is the development of a formal temporal multidimensional model, where different temporal units of reclassification are supported. For example, salespersons can be hired by stores for periods of *days*, whereas stores can be changed of status *annually*. As the second contribution, we introduce and formalize the notion of *season*, an interval during which a member of a level is associated with another one, and propose query language constructs to enable the formulation of *season queries*. For example, “What was the total value sold by a salesperson in his first season in a store?” We also extend season queries with spatial features enabling the formulation of queries such as “What was the total value sold by a salesperson in his n^{th} season in a given geographic region?”

Finally, we also make contributions on other related spatio-temporal DW issues: a) we propose a conceptual model to incorporate a trajectory as a first-class citizen in a DW, b) we extend a spatial OLAP operator with several spatial aggregate functions, and c) we address the problem of change in the degree of containment, another type of dimension data change, where a member of a level does not necessarily change its parent, but the degree (percentage) of association with it.

Resumen

Una bodega de datos (BD) se modela usualmente mediante una vista multidimensional de los datos. En un modelo multidimensional, un conjunto de dimensiones se asocia con un tema de análisis denominado hecho. Cada dimensión se compone de un conjunto no vacío de niveles los cuales están organizados jerárquicamente. Usualmente, una dimensión se considera estática en una BD; sin embargo, el esquema y los datos de una dimensión pueden evolucionar. Esta tesis se enfoca en un tipo de cambio de datos dimensional conocido como *reclasificación*, es decir, cuando un miembro (instancia) de un nivel cambia de padre (un miembro de un nivel superior en la misma dimensión). La primera contribución de esta tesis es la concepción de un modelo temporal multidimensional formal, donde se soportan diferentes unidades temporales de reclasificación. Por ejemplo, los vendedores pueden ser contratados por las tiendas por períodos de *días*, mientras que las tiendas pueden ser cambiadas de categoría *anualmente*. Como segunda contribución, se presenta y formaliza el concepto de *temporada*, un intervalo durante el cual un miembro de un nivel está asociado con otro, y se proponen operadores de consulta que permiten la formulación de *consultas de temporada*. Por ejemplo, "¿Cuál fue el valor total vendido por un vendedor en su primera temporada en una tienda?" También se extienden las consultas de temporada con elementos espaciales permitiendo la formulación de consultas tales como "¿Cuál fue el valor total vendido por un vendedor en su enésima temporada en una región geográfica dada?"

Finalmente, otras contribuciones de la tesis se relacionan con aspectos de BD espacio-temporales: a) se propone un modelo conceptual donde se incorporan trayectorias como elementos de primera clase en una BD, b) se extiende un operador OLAP espacial con diversas funciones de agregación espacial y c) se aborda el problema del cambio en el grado de inclusión, otro tipo de cambio de datos dimensional, donde un miembro de un nivel no necesariamente cambia de padre, sino el grado (porcentaje) de asociación con éste.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Research context

1.1.1 Data warehouse

A data warehouse (DW) [Inmon 2005], [Kimball 2008] is a specialized database for efficient querying and analysis of integrated information from a wide variety of sources [Yang 1998]. Since the past decade, DWs have enjoyed a remarkable and increasing popularity in both the research community and industry [Inmon 2005], [Kimball 2008]. DWs have proved their usefulness as systems for integrating information and supporting the decision-making process.

DWs are usually modelled using a *multidimensional* view of data. Although there are several multidimensional models [Agrawal 1997], [Gyssens 1997], [Vassiliadis 1998], [Golfarelli 1998], [Lehner 1998], [Pedersen 2001a], [Jensen 2004], [Timko 2005], [Kumar 2008]; they all share a set of key concepts such as dimension, fact, level, level attribute, hierarchy, and measure.

A set of dimensions is associated with a subject of analysis called fact. For example, in a retail sales scenario, SALESPERSON and PRODUCT are dimensions that are typically associated with a fact sale. A multidimensional collection of data arranged in this way is commonly referred to as a *data cube* [Jarke 2003] (to be referred to hereinafter in this thesis simply as *cube*).

A dimension is composed of a non-empty set of levels. For example, in Figure 1.1 *Salesperson*, *Store*, and *Status* are levels of the SALESPERSON dimension (the crowfoot connector represents a one-to-many relationship); *Product* and *Category* are levels of the PRODUCT dimension. A level in turn has attributes [Kumar 2008], which provide supplementary information about the level. For example, Name and Salary are typical attributes of the *Salesperson* level (for simplicity, we do not show attributes of levels in our diagrams).

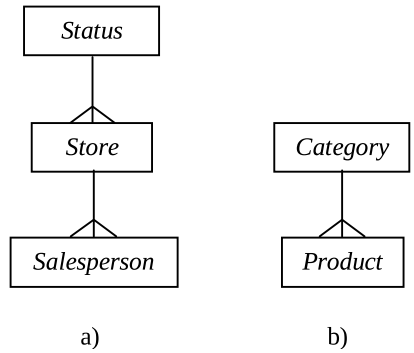


Figure 1.1. Some levels of: a) SALESPERSON dimension and b) a PRODUCT dimension.

The dimension levels are structured as a hierarchy according to the analysis needs in order to enable the data analysis at various levels of detail [Torlone 2003]. A hierarchy plays the role of a *classification* hierarchy (often called *roll-up* hierarchy [Golfarelli 2009b]), *i.e.*, it classifies (groups) members (instances) of a level at a higher member level of the hierarchy. For example, in our SALESPERSON dimension, salespersons are grouped into stores and stores are classified according to their statuses; in our PRODUCT dimension, products are classified according to their categories.

On the other hand, a fact has measures, *i.e.*, business metrics that analysts want to evaluate and report on, *e.g.*, number of units of a product sold and sale value are typical measures of a sale. This way of organizing the data allows us to perform some analytical queries in a flexible and intuitive way: What was the total value sold by each salesperson? By each store? By product in each store? (See Table 1.1) What was the average total number of units sold by product category in each store?

Table 1.1. Total value sold by product in each store.

Levels		**Measure**
Store	***Product***	**Sale_value**
st_1	pd_1	1500
st_1	pd_2	1800
st_1	pd_3	1700
st_2	pd_1	4000
st_2	pd_2	2000
st_3	pd_1	3000
st_3	pd_3	4000
st_4	pd_1	5000
…		

1.1.2 Temporality and spatiality

TIME is an omnipresent dimension in DWs [Malinowski 2006]. It allows the distribution and comparisons of facts in different periods and in different time granularities (days, months, years); thus the previous queries can be complemented with temporal data, making possible more detailed analysis, *e.g.*, total value sold by each store monthly, see Figure 1.2.

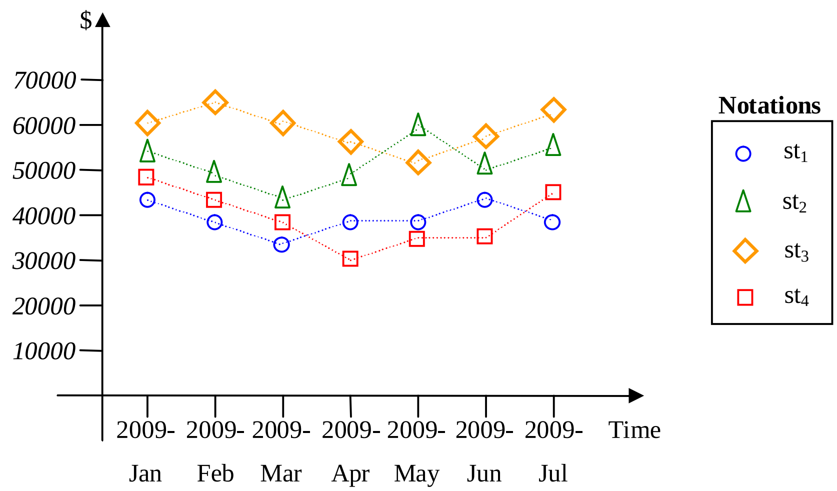


Figure 1.2. Monthly total value sold by store.

Note, however, that although DWs have a TIME dimension, this dimension is not used to manage other temporal changes that can occur in a DW. Further below, in this section, we present a short overview on this issue.

On the other hand, the explosion of technologies such as GIS (Geographic Information System) and GPS (Global Positioning System) [Turner 2010] demand the management of other data types and enable the formulation of other useful analytical queries. As a consequence, DWs have been enriched, *e.g.*, with spatial features [Malinowski 2008].

In an analogous way to the TIME dimension, the incorporation of spatial features in a DW could help to analyze the distribution and comparisons of facts through space (usually a geographic space). For example, in Figure 1.3 we show the location of the stores and the total value sold by each one.

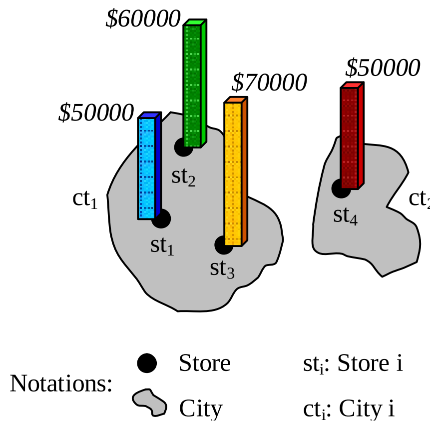


Figure 1.3. Location of the stores and the total value sold by each one.

Indeed, spatiality can be incorporated in the dimensions and/or the facts [Bimonte 2005]. For example, if we store the geographic coordinates of stores in the *Store* level of the SALESPERSON dimension, the users are now enabled to pose queries such as: What was the total value sold by the stores located in a certain geographic region? (A region that can be specified by a spatial window, as illustrated by the dashed box of Figure 1.4)

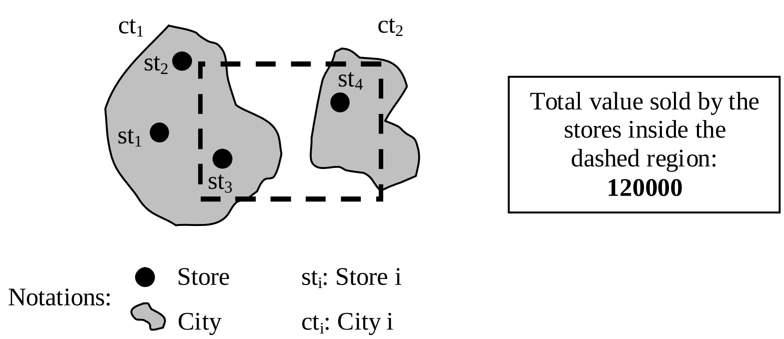


Figure 1.4. Spatial query window.

It is also possible to incorporate spatial measures to the facts [Han 1998], [Shekhar 2001] and perform their aggregation using spatial aggregate functions such as geometric union, minimum bounding rectangle (MBR), center of mass, convex hull, among others; as we propose in [Moreno 2009a] by extending the *map cube* operator: an operator that supports spatial aggregation in a spatial DW, but only using geometric union, and enables visualization of information through maps [Shekhar 2001], see Chapter 4. For example, in a DW for crimes, consider a measure Crime_points which represents the locations (points) where crimes were committed in a city, see Table 1.2 and Figure 1.5. In Figure 1.5 we show the center of mass and the MBR of all the crimes committed in city ct_1 according to the data from Table 1.2.

Table 1.2. Sample data of crimes (version 1).

Levels		Measure
Day	***City***	**Crime_points**
2009-Jan-01	ct_1	$\{p_1, p_2\}$
2009-Jan-02	ct_1	$\{p_3\}$
2009-Jan-03	ct_1	$\{p_4, p_5\}$
2009-Jan-04	ct_1	$\{p_6\}$
…		
2009-Jan-01	ct_2	$\{p_{889}\}$
…		

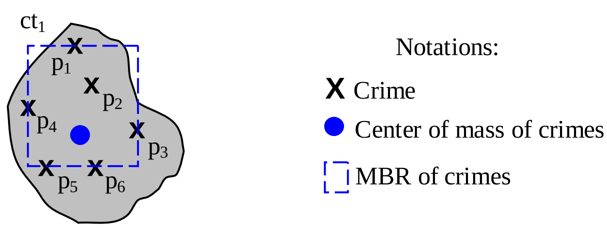


Figure 1.5. Center of mass and MBR of crimes committed in city ct_1.

While in multidimensional models such as those proposed in [Agrawal 1997], [Gyssens 1997], [Vassiliadis 1998], [Golfarelli 1998], [Lehner 1998], [Pedersen 2001a], [Kumar 2008]; the hierarchical relationship between the levels captures their *full containment*, *partial containment* is prevalent in *spatial data*. The partial containment allows us to represent situations where a dimension value is not fully contained in another one. For example, in our SALESPERSON dimension, a salesperson is fully contained in a store and a store is fully contained in a status; in our PRODUCT dimension, a product is fully contained in a category; while in a LOCATION dimension with *Highway* and *Department* levels, a highway is not necessarily fully contained in a department. For instance, 0.2 (20%) of a highway hw_1 could be contained in a department and 0.8 (80%) in another department (this percentage is termed *degree of containment* [Jensen 2004]). Thus, if a fact, *e.g.*, an accident is associated with highway hw_1, we cannot assure in which department the accident occurred (unless more information is given, *e.g.*, the coordinates of the accident); therefore, a query language that is intended to be used in this scenario must deal with this uncertainty [Jensen 2004], [Timko 2005]. Another scenario where partial containment might be useful would be to analyze how a jungle is shrinking in a country, *e.g.*, the Amazon rainforest shrinking in Brazil, Peru, and Colombia, among other countries.

1.1.3 Trajectories and reclassification

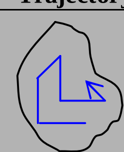
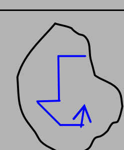
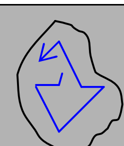
The conjunction of temporal and spatial features in a DW can lead to a richer dynamics. A survey on this subject was presented in [Moreno 2007a], [Moreno 2007b]. In particular, in our work, we consider that the notion of *trajectory* can be incorporated as a first-class citizen (a complex data type) in a DW.

Informally, a trajectory is the evolving position of an object travelling in a space (it could be an *abstract space*) during an interval [Spaccapietra 2008], a definition that entails the inherent spatio-temporal nature of a trajectory.

The incorporation of a trajectory as a first-class citizen in a DW enables the formulation of valuable queries for decision-makers. For example, consider the trajectory followed by a taxi during a day and consider the following queries: what were the three most profitable taxi trajectories in the last month? How many taxi trajectories intersected a given region within the last two hours? The trajectory aggregation problem could also be addressed, *e.g.*, what is the meaning of adding, averaging trajectories?

We believe that, just as temporal and spatial features, a trajectory can be incorporated in the dimensions and/or the facts. In our work, we propose the conceptual modelling of trajectories as complex measures [Moreno 2010d], see Chapter 5. Thus, we can consider a whole trajectory as a measure, *e.g.*, the trajectory followed by a taxi in a city during a day, see Table 1.3.

Table 1.3. Taxis trajectories.

Levels		**Measure**
Day	***Taxi***	**Trajectory**
2009-Jan-01	tx_1	
2009-Jan-02	tx_1	
…		
2009-Jan-01	tx_2	
…		

Note that a trajectory could be stored explicitly or implicitly, *e.g.*, we could infer a trajectory from the facts: facts are almost always associated with both spatial and temporal dimensions (a fact occurs in a specific place and at a specific time). For example, if we consider the dimensions SUSPECT, TIME, PLACE associated with crime facts, we could generate the trajectory of a potential serial killer, see Table 1.4 and Figure 1.6. In this example, for simplicity we assume that the serial killer moves from one point to another in a straight line. This issue is in an exploratory phase of development as an extension of our current work [Moreno 2009a], see Chapter 4.

Table 1.4. Sample data of crimes (version 2).

Levels			Measure
Suspect	***Day***	***Place***	**#Victims**
$susp_1$	2009-Feb-11	pl_1	1
$susp_1$	2009-Feb-22	pl_2	1
$susp_1$	2009-Mar-13	pl_3	2
$susp_1$	2009-Mar-25	pl_4	1
…			
$susp_2$	2009-Jan-15	pl_{29}	2
…			

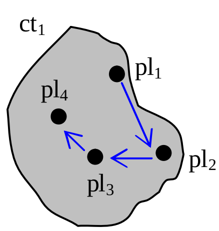


Figure 1.6. Trajectory of the crimes of suspect $susp_1$.

As we mentioned before, we shall now pass on to present a short overview regarding temporal changes that can occur in a DW. In practice, due to changing requirements, *dimension schema* and *dimension data* can evolve [Hurtado 1999], [Golfarelli 2009a] although usually in a slow way [Kimball 2008]. For example, in the SALESPERSON dimension the *Status* level could be dropped and a *City* level could be added, some salespersons can change of store, some salespersons can retire, while others are hired. Thus, the usual assumption that dimensions are static in a DW does not hold in these situations.

Several works deal with dimension changes. Hurtado [1999] and Blaschka [1999] propose operators to delete, insert, and update the dimension data and the dimension schema. In [Kaas 2004] operators to change the DW schema are considered, among them operators to insert and delete dimension and levels. Other authors [Eder 2001], [Body 2002], [Morzy 2004], [Golfarelli 2006], [Ravat 2006], [Rechy-Ramirez 2006], [Wrembel 2007] focus on *DW versioning*, *i.e.*, how to transform and/or query data that span several DW versions caused by dimension changes. For example, how to answer consistently a query such as: What was the total value sold by each city in each month of 2009? (Considering that the *City* level was added to the DW only since May 2009). For a recent survey on temporal issues related to DWs, the reader is referred to [Golfarelli 2009a]. Malinowski [2008] also addresses other DW temporal changes, including time-varying measures.

In our work, we focus on an interesting type of dimension data change known as *reclassification*, *i.e.*, when a member of a level changes its parent (a member of a higher level of the same dimension). For example, when a salesperson changes of store, or a store changes of status. Although the management of reclassifications in a multidimensional model has been considered [Chamoni 1999], Mendelzon [2000], Pedersen [2001a], [Vaisman 2004], [Malinowski 2008]; there still remain several problems to be solved. For example, salespersons can be hired by stores for periods of *days*, whereas stores can be changed of status *annually*. In order to devise an accurate model for this situation, we propose a formal multigranular temporal multidimensional model [Moreno 2009b], see Chapter 2, where different temporal units of reclassification are supported.

In addition, reclassifications require, from the query point of view, a careful handling in order to avoid inconsistent results. For example, consider reclassifications of salespersons through stores and suppose a user wants to know the total value sold by a salesperson sp_1 when he/she has worked in store st_1; the query system must have the ability to find for each sale of sp_1 the store where he/she was working when the sale was made [Mendelzon 2000], [Vaisman 2004]. Although a query like the previous one can be formulated in TOLAP [Mendelzon 2000], a temporal query language for OLAP (On-Line Analytical Processing), reclassifications can lead to other interesting queries as we shall see in the following subsection.

Note also that from the succession of reclassifications of a member of a level, we can infer a trajectory: a *reclassification trajectory*. For example, the trajectory of a salesperson through stores during his staying in an organization, the trajectory of a product through the different categories for which it has been classified (in this last example, the space where the object moves is abstract).

1.1.4 Seasons

We want to note that in the context of trajectories arises the notion of *season*. We consider a *season* an interval during which a moving object is associated with another object, *e.g.*, a region. For example, in the case of taxi trajectories, we can consider seasons of a taxi in a given region, and in the case of the trajectory of a suspect, we can consider his/her seasons in a neighborhood or a city. In our work (see Chapters 6 and 7) *we focus on seasons resulting from reclassification trajectories*.

Thus, in the context of these trajectories, each reclassification represents an interval during which a member of a level is associated with another one, *i.e.*, a season of association between two members of a dimension.

We believe that queries referring to seasons, called *season queries* in our work, can be valuable for decision-makers. For example, consider the queries: What was the total value sold by each salesperson in each season in each store? What was the total value sold by salesperson sp_1 in his first season in store st_1? (See Table 1.5). To the best of our knowledge there is no language or operator that facilitates the formulation of season queries in a compact and intuitive way. In [Moreno 2010b], see Chapter 6, we propose query constructs to facilitate the formulation of this type of queries.

Table 1.5. Total value sold by each salesperson in each season in each store.

Levels		**Measure**
Salesperson	***Season***	**Sale_value**
sp_1	First season in st_1	2000
sp_1	First season in st_2	2000
sp_1	Second season in st_1	1750
sp_1	First season in st_4	1750
sp_1	First season in st_3	2500
sp_1	Second season in st_4	2100
…		
sp_2	First season in st_2	2000
…		

Moreover, season queries could also be enriched with spatial features enabling the formulation of queries such as: What was the total value sold by sp_1 in his n^{th} season in a given geographic region? In Figure 1.7, *e.g.*, we show the total value sold by sp_1 in his first season in the stores contained in the dashed region. According to Table 1.5, the sales made by sp_1 during his first and second season in st_1, and his first season in st_2, contribute to the total requested (note that *second season* of sp_1 in st_1 contributes to the answer because sp_1 has not left the region). We refer to this type of query as *spatial season queries* [Moreno 2010c], see Chapter 7.

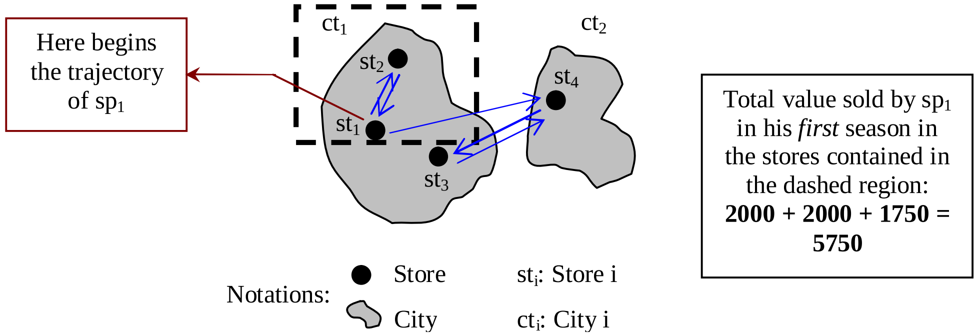


Figure 1.7. Trajectory of a salesperson sp_1 through stores (in blue) and spatial query window.

1.1.5 Change in the degree of containment

Another type of dimension data change we are interested in is the *change in the degree of containment*. For example, at a time t_i the degree of containment of a highway in a department is 0.2, but at a time t_{i+1}, this degree may change due to construction or destruction of highway sections, or boundaries change between the departments. Note that this type of change can be considered a reclassification, where a member of a level does not necessarily change its parent, but the degree of association with it. Unfortunately, proposals which consider partial containment [Jensen 2004], [Timko 2005] do not consider the change in the degree of containment between two dimension values. Note that in order to obtain consistent results over time, the degree of containment *at the time* when the facts occurred must be considered. We address this issue in [Moreno 2009c], [Moreno 2010a], see Chapter 3.

We summarize our research topics in Figure 1.8.

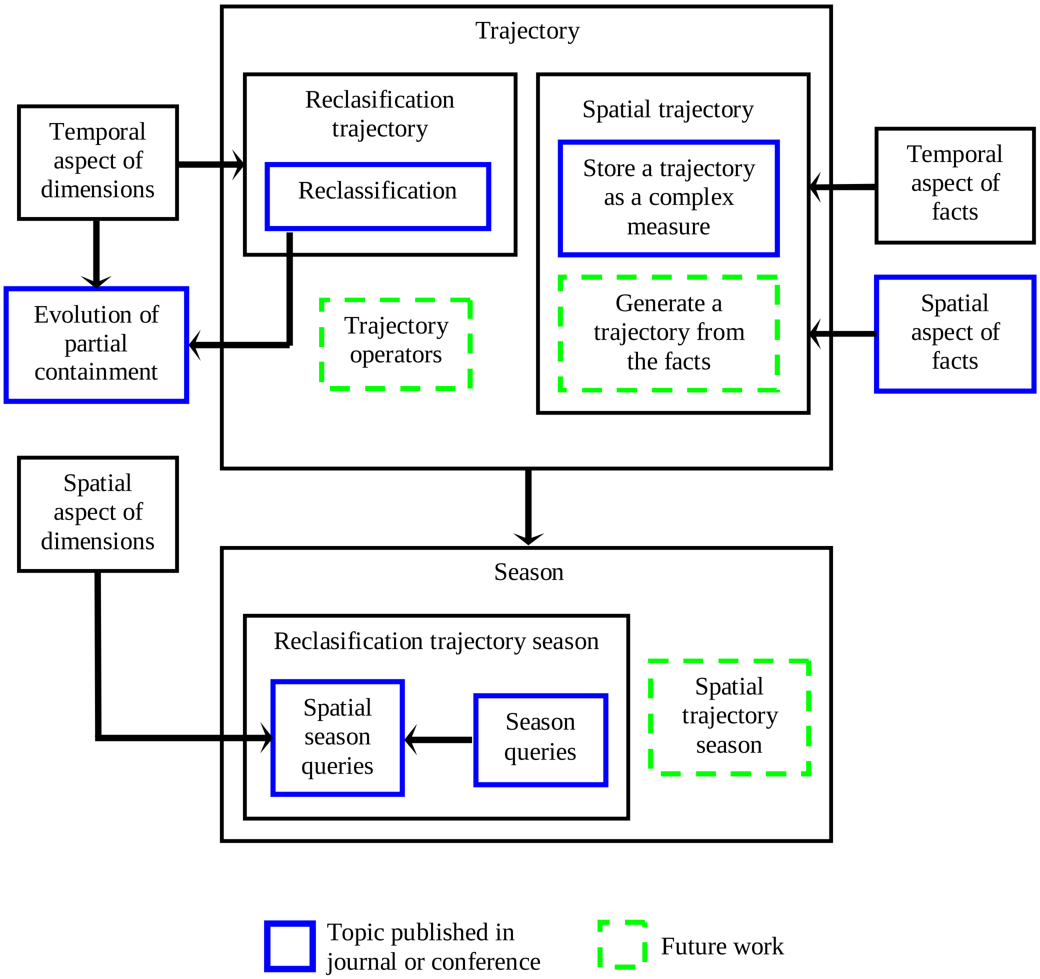


Figure 1.8. Research topics and their relationships.

1.2 Thesis organization

The outline of this thesis is as follows. In accordance with the previous section and Figure 1.8, we have covered the following topics.

Part I. Preliminar works:

- In Chapter 2 we propose a formal temporal multidimensional model which allows the representation of reclassifications that can occur with different temporal units in a dimension.
- In Chapter 3 we propose an extension to support the change in the degree of containment in a formal multidimensional model.
- In Chapter 4 we extend the map cube operator in order to support different spatial aggregate functions.

These three chapters form the first part of the thesis. They consider a variety of issues related with spatial and temporal DWs.

Part II. Trajectories:

- In Chapter 5 we extend a conceptual spatial multidimensional model by incorporating a trajectory as a first-class citizen in a DW.

This chapter forms the second part of the thesis. It incorporates a trajectory as first-class citizen in a DW. The notion of trajectory is later specialized in Part III, where the notion of reclassification trajectory is introduced.

Part III. Seasons:

- In Chapter 6 we introduce and formalize the notion of season of reclassification around the model of Chapter 2 and propose an operator for season queries.
- In Chapter 7 we extend our work from Chapter 6 in order to support spatial season queries.

These two chapters form the third part of the thesis. They focus on seasons, which can be considered the core of the thesis.

Finally, we present conclusions and future work. In Figure 1.9 we outline the structure of the thesis. Solid arrows show prerequisites, whereas dashed arrows show preferred, but not-mandatory, order among chapters.

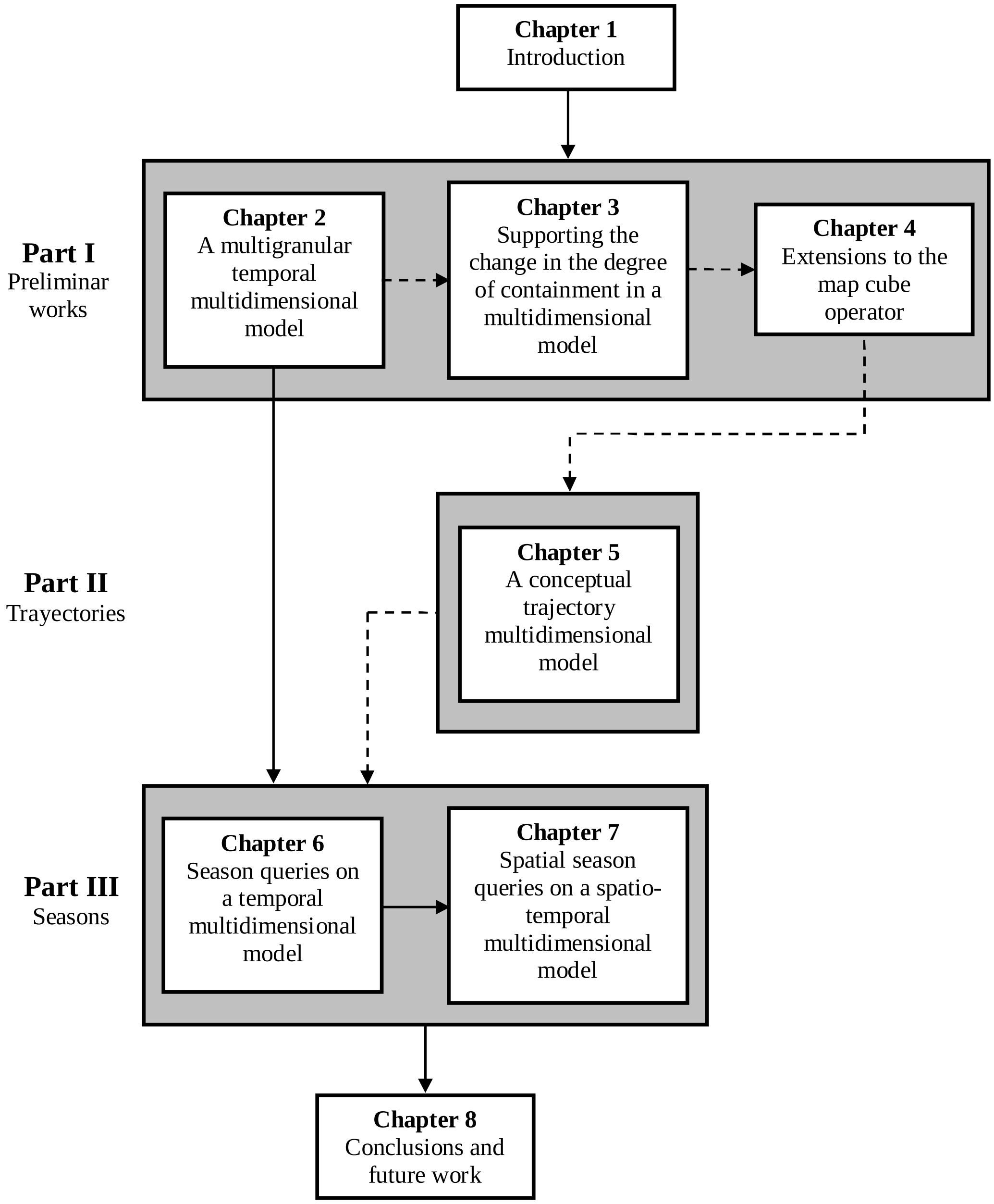


Figure 1.9. Structure of the thesis.

1.3 Objetives

1.3.1 General objective

To improve the expressivity of DW query languages in order to facilitate the formulation of season queries and spatial season queries.

1.3.2 Specific objectives

a. To define and characterize the notion of season, season queries, and spatial season queries.

b. To propose a multidimensional model that integrates the necessary spatio-temporal concepts to solve the types of queries identified.

c. To propose query language constructs to facilitate the formulation of the types of queries identified.

d. To compare the proposed query language constructs with a language, such as TOLAP or SQL, in order to show its expressivity.

e. To develop a prototype and make basic experiments to validate the propose model and query language constructs.

Through Chapters 2, 6 and 7, we develop a spatio-temporal multidimensional model, formalize the notion of season and propose query language constructs for season and spatial season queries (along with a basic prototype, experiments, and language comparisons); achieving in this way all the proposed objectives. Although the rest of the chapters address issues that are somewhat beyond the foreseen objectives, we consider these issues to be of particular relevance in the context of our thesis research.

Part I: Preliminar Works

Chapter 2: A Multigranular Temporal Multidimensional Model

2.1 Introduction

As we mentioned in Chapter 1, dimension schema and dimension data can evolve. In this chapter, we focus on a specific type of dimension data change, the *reclassification*, *i.e.*, when a member (instance) of a level changes its parent (a member of a higher level of the same dimension). Reclassifications are frequent in several situations: a salesperson is rotated through stores, a store changes of status, a product is recategorized, a player changes its team, a team changes its division, a hurricane moves from one region or city to another.

A few works deal with reclassifications in DW dimensions. In Chamoni [1999], Pedersen [2001a], and Malinowski [2008], valid time intervals are used to keep track of reclassifications. In Mendelzon [2000], a multidimensional model supporting structural and data changes in dimensions, and a temporal multidimensional query language called TOLAP are proposed. This approach is extended later [Vaisman 2004] by introducing TSOLAP, an OLAP server supporting dimension updates. One aspect that is common to all these works is that they use only one temporal unit for keeping track of evolving associations of members in each dimension. This prevents the accurate representation of reclassifications that can occur with different temporal units. For example, salespersons can be hired by stores for periods of days, whereas stores can change of status annually or biannually. In order to deal with this situation, we extend a formal temporal multidimensional model.

The rest of the chapter is organized as follows. In Section 2.2, we present our formal temporal multidimensional model and in Section 2.3, we end the chapter.

2.2 Temporal multidimensional model

Our model is based on the work of Mendelzon [2000] that, in turn, was built on the work of Cabibbo [1997]. In the following, we represent the set of natural numbers including zero with $\mathbb{N}_0$ and the set of natural numbers not including zero with $\mathbb{N}$.

2.2.1 Dimensions

A dimension schema is a 5-tuple (D, L, $\preccurlyeq$, *All*, $\bot$) where:

i) D is the name of the dimension schema,

ii) L is a set of levels; each level $l \in$ L has a name, *Lname*, and is associated with a set of members (values), *i.e.*, a domain, denoted by dom(l),

iii) $\preccurlyeq$ is a partial order in the set L; we denote $\preccurlyeq'$ as the transitive reduction of $\preccurlyeq$. Let l_1, $l_2 \in$ L; $l_1 \preccurlyeq l_2$ means that l_1 rolls up to l_2,

iv) *All* $\in$ L is the top level of $\preccurlyeq$, *i.e.*, $\forall\ l \in$ L, $l \preccurlyeq$ *All*; dom(*All*) = {*all*}, and

v) $\bot \in$ L is the bottom level of $\preccurlyeq$, *i.e.*, $\forall\ l \in$ L, $\bot \preccurlyeq l$.

Example 2.1. Figure 2.1 presents the dimension schema (SALESPERSON, {*Salesperson*, *Sex*, *Store*, *All*}, $\preccurlyeq$, *All*, *Salesperson*), in which $\preccurlyeq'$ = {(*Salesperson*, *Sex*), (*Salesperson*, *Store*), (*Sex*, *All*), (*Store*, *All*)}. A member of the *Salesperson* level may include attributes such as salesperson name, date of birth, and salary.

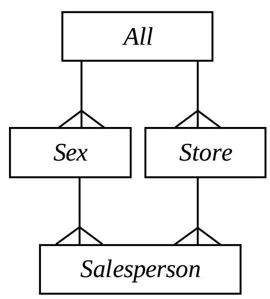


Figure 2.1. The SALESPERSON dimension schema.

2.2.2 Handling time

We consider time as discrete, *i.e.*, a point in the timeline that corresponds to a positive integer [Mendelzon 2000]. A positive integer represents an instance of a temporal unit, *e.g.*, an hour, a day, a month. For clarity, we will write ‘day 1’ (or an equivalent value such as ‘2009-Jan-01’) instead of just ‘1’. In addition, $I = [i, j]$, $i, j \in \mathbb{N}$, $i \leq j$; represents an interval that corresponds to a set of contiguous integers: $\{k \mid k \in \mathbb{N}$ AND $i \leq k \leq j\}$. A variety of functions can be applied to intervals [Allen 1983]. For example, Start(I) and End(I) return the first and the last positive integer of an interval I, respectively.

There is a *finer than* relation among temporal units. For example, each day is included in a specific month. Let μ_1 and μ_2 be temporal units, if μ_1 is finer than μ_2, we write $\mu_1 \rightsquigarrow \mu_2$. Note that the temporal units can be used to define a TIME dimension schema where they play the role of levels and the *finer than* relation corresponds to the *rolls up* relation.

Example 2.2. Consider the temporal units Day, Week, and Month. Day $\rightsquigarrow$ Day, Day $\rightsquigarrow$ Week, Day $\rightsquigarrow$ Month. For instance, the day 2009-Jan-11, is included in the month 2009-Jan. On the other hand, Week is not finer than Month, and Month is not finer than Week, *i.e.*, some temporal units are incomparable with regard to the finer than relation.

2.2.3 Adding time to dimensions

We add time to other dimensions (other than the TIME dimension) in two ways. The first one is timestamping each member in a particular level of a dimension with its valid time, in order to capture its lifespan. The second way is timestamping the association between members with their valid time, in order to capture the periods of their associations.

Consider a dimension schema (D, L, $\preccurlyeq$, *All*, $\perp$). A pair of levels $(l_1, l_2) \in \preccurlyeq'$, $l_2 \neq All$, can be associated with a temporal unit μ, that defines the Temporal Reclassification Granularity (TRG) between l_1 and l_2. If so, we say that the pair (l_1, l_2) is temporal. Note that in Mendelzon’s model [2000], unlike in ours, a unique TRG is defined for the whole dimension.

Example 2.3. Consider the dimension schema of Example 2.1. In real life, a salesperson is assigned to a store for a period of days. Therefore, we associate a TRG μ = Day with the pair (*Salesperson*, *Store*) to track the associations between salespersons and stores, as is shown in Figure 2.2.

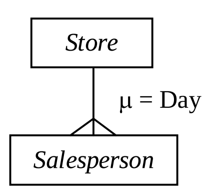


Figure 2.2. TRG between the *Salesperson* level and the *Store* level.

A dimension schema instance is a 2-tuple (D, RF) where D is a dimension schema, and RF is a set of rollup functions. Let L be the set of levels belonging to D; l_1, $l_2 \in$ L, and $\preccurlyeq$ the partial order on L then:

i) for each temporal pair $(l_1, l_2) \in \preccurlyeq'$ with TRG μ, there exists a rollup function RUP_l_1_l_2: $\mathrm{dom}(l_1) \times \mathrm{dom}(\mu) \rightarrow \mathrm{dom}(l_2)$, and

ii) for each non-temporal pair $(l_1, l_2) \in \preccurlyeq'$, there exists a rollup function RUP_l_1_l_2: $\mathrm{dom}(l_1) \rightarrow \mathrm{dom}(l_2)$.

Note that RUP_l_1_l_2 is a metaname, *i.e.*, l_1 and l_2 refer to level names.

Example 2.4. Consider an instance of the dimension schema of Example 2.1. Suppose the following domains: dom(*Salesperson*) = {sp_1, sp_2}, dom(*Sex*) = {Male, Female}, dom(*Store*) = {st_1, st_2, st_3}, dom(*All*) = {*all*}, and dom(Day) = $\mathbb{N}$. The rollup functions are shown in the right column of Table 2.1. For example, RUP_*Salesperson_Store* (sp_1, day 2) = st_1, and RUP_ *Salesperson_Sex*(sp_1) = Male.

Table 2.1. Examples of rollup functions for a SALESPERSON dimension schema instance.

Pair of ordered levels	Rollup function
(*Salesperson*, *Sex*)	{(sp_1, Male), (sp_2, Female)}
(*Salesperson*, *Store*)	{((sp_1, day 1), st_1), ((sp_2, day 1), st_2), ((sp_1, day 2), st_1), ((sp_2, day 2), st_2), ..., ((sp_1, day 46), st_2), ((p_2, day 46), st_2), ...}
(*Sex*, *All*)	{(Male, *all*), (Female, *all*)}
(*Store*, *All*)	{(st_1, *all*), (st_2, *all*), (st_3, *all*)}

2.2.4 Dimensions constraints

Summarizability is a desirable property in a multidimensional model. Summarizability refers to the correct aggregation of measures in higher levels considering existing aggregations in lower levels [Malinowski 2008]. To guarantee summarizability, dimension hierarchies must meet *disjointness* and *completeness* conditions [Lenz 1997]. Informally, disjointness states that a member of a level can only be associated with a member of a higher level (a member can only have one ancestor member), and completeness states that in a hierarchy, each member of a level must be associated with a member of its immediate parent level.

In order to guarantee the disjointness condition in our model, we enforce the following conditions. Let l_1, l_2, l_3, …, l_n be levels of a dimension schema, n > 1, where $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_n$. Let $U \neq \varnothing$ be the set of TRGs along the path $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_n$; then, the mapping from l_1 to l_n with an arbitrary temporal granularity μ is possible if $\forall\ \mu' \in U$ then $\mu \rightsquigarrow \mu'$. If $U = \varnothing$ then the mapping from l_1 to l_n is possible because all the mappings along the path $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_n$ are non-temporal.

Example 2.5. Consider Figure 2.3. Suppose that dom(*Salesperson*) and dom(*Store*) are as in Example 2.4, dom(*Status*) = {A, B}, and dom(μ_1) = dom(μ_2) = $\mathbb{N}$. A mapping from *Salesperson* to *Status* with μ = Day is possible because U = {Day, Semester}, Day $\rightsquigarrow$ Day, and Day $\rightsquigarrow$ Semester. This mapping is shown in the last row of Table 2.2. For example, if salesperson sp_1 was in the store st_1 on day 1, and st_1 was in status A in semester 1; then, sp_1 was in status A on day 1 because day 1 belongs to semester 1. On the other hand, a mapping from *Salesperson* to *Store* with μ = Month is impossible because U = {Day} and Month is not finer than Day. It means that a salesperson could be associated with several stores during a month.

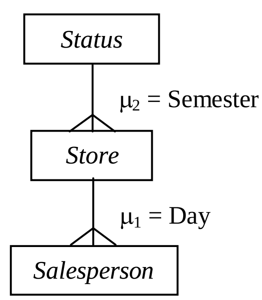


Figure 2.3. TRGs among *Salesperson*, *Store*, and *Status* levels.

Table 2.2. Mapping example.

Pair of ordered levels	**Rollup function**
(*Salesperson*, *Store*)	{((sp_1, day 1), st_1), ((sp_2, day 1), st_2), ((sp_1, day 2), st_1), ((sp_2, day 2), st_2), …}
(*Store*, *Status*)	{((st_1, semester 1), A), ((st_2, semester 1), B), ((st_1, semester 2), A), ((st_2, semester 2), A), ...}
(*Salesperson*, *Status*)	{((sp_1, day 1), A), ((sp_2, day 1), B), ((sp_1, day 2), A), ((sp_2, day 2), B), ...}

With regard to the completeness condition, let us consider, *e.g.*, the relationship between a salesperson and a store. There exist periods when a salesperson is not hired by any store. In order to guarantee completeness, we adopt Jensen's technique [2004]. The essential idea is to introduce "dummy" parent values to member levels with no parents, *e.g.*, a store value "No_store".

There is a third necessary condition for summarizability, but this depends on the correct use of measures and aggregation functions [Lenz 1997]; therefore, it will not be discussed here. In addition to summarizability conditions, we adopt Mendelzon's consistency condition [2000], *i.e.*, if there are different paths from one level to another, composing the rollup functions along the different paths must produce the same function.

2.2.5 Facts

A fact represents a subject of decision-oriented analysis [Torlone 2003]. A fact typically includes attributes called measures, *i.e.*, indicators to evaluate specific activities of an organization [Malinowski 2008]. Measures can be aggregated along the dimensional levels to facilitate data analysis. Formally, a fact schema is a 3-tuple (F, L_F, M) where:

i) F is the name of the fact schema,

ii) $L_F = \{l_1, \ldots, l_n\}$ is a set of levels. Each $l_i \in L_F$ is the bottom level ($\perp$) in a dimension schema, and

iii) $M = \{m_1, \ldots, m_m\}$ is a set of measures (note that in Mendelzon's model only one measure is considered). Each measure m_i is associated with a domain dom(m_i).

Example 2.6. Consider the fact schema (SALES, {*Salesperson*, *Product*, *Day*}, {Units_sold, Sale_value}), see Figure 2.4. To represent our multidimensional model, we use essential notations from [Malinowski 2008], see Figure 2.5, based on the entity-relationship graphical notations; however, we add the representation for TRGs.

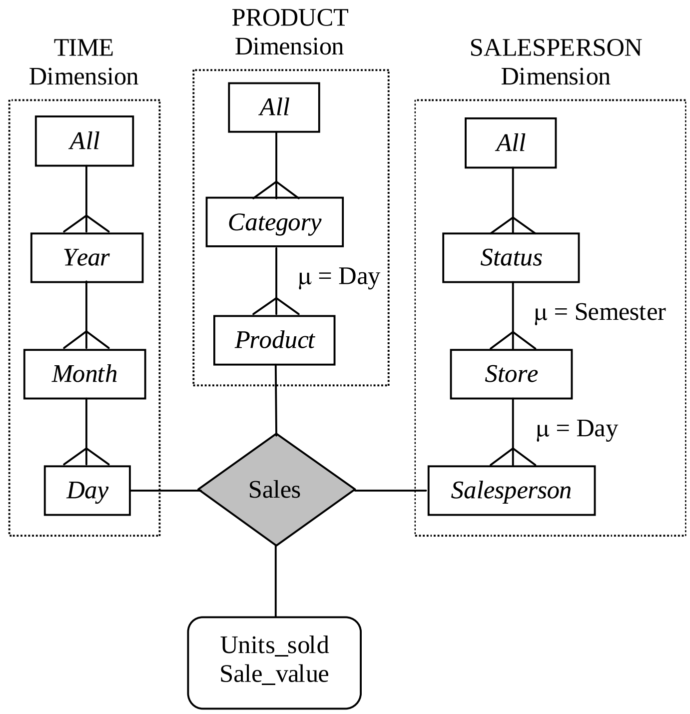


Figure 2.4. A temporal multidimensional model for sales.

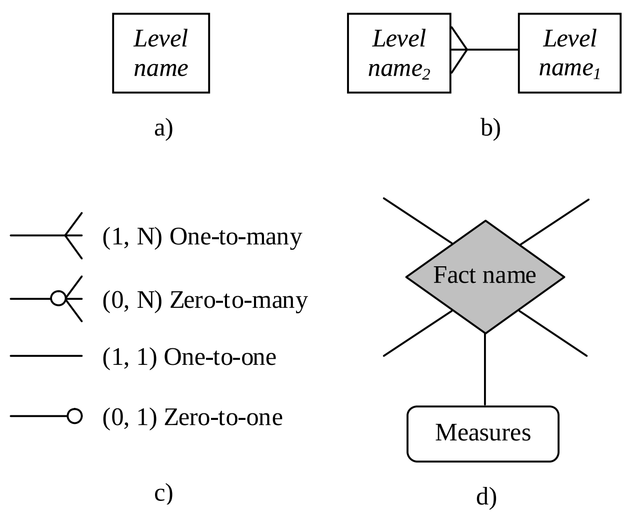


Figure 2.5. Notations to represent our multidimensional model: a) level, b) hierarchy, c) cardinalities, and d) fact relationship. Source [Malinowski 2008].

A fact instance of a fact schema (F, L_F, M), $L_F = \{l_1, \ldots, l_n\}$, M = $\{m_1, \ldots, m_m\}$ is a 2-tuple (l_F, m) where $l_F = \{\text{value}(l_1), \ldots, \text{value}(l_n)\}$ is a set of values where value(l_i) $\in$ dom(l_i); each value(l_i) is a member of a bottom level in a dimension schema instance, and m = $\{\text{value}(m_1), \ldots, \text{value}(m_m)\}$ is a set of values where value(m_i) $\in$ dom(m_i). A *fact table* is a set of fact instances.

Example 2.7. Suppose the following domains: dom(*Salesperson*) = {sp_1, sp_2}, dom(*Product*) = {pd_1, pd_2}, dom(*Day*) = ℕ, and dom(Units_sold) = dom(Sale_value) = ℕ. A fact table of the fact schema SALES of Example 2.6 is shown in Table 2.3, the first fact instance that appears there is ({sp_1, pd_1, day 1}, {2, 2000}).

Table 2.3. A fact table of the fact schema SALES.

Bottom levels			**Measures**	
Salesperson	***Product***	***Day***	**Units_sold**	**Sale_value**
sp_1	pd_1	1	2	2000
sp_1	pd_1	8	1	1000
sp_1	pd_2	8	1	500
sp_1	pd_2	28	1	500
sp_2	pd_1	1	3	3000

2.2.6 Fact constraints

Consider the multidimensional model for sales of Figure 2.4. Suppose the facts record weekly sales instead daily ones; therefore, we could find, *e.g.*, weekly units sold by salespersons, but we could not find weekly units sold by stores, because the TRG of the associations between salespersons and stores is Day. Let U_D be the set of all TRGs in a dimension schema D; then, measures can be aggregated in any level of D if ▪ $\forall \mu' \in U_D$ then $\mu_F \rightsquigarrow \mu'$, where μ_F is the bottom level of the TIME dimension associated with the fact schema F.

Example 2.8. In the temporal multidimensional model for sales of Figure 2.4, the set of TRGs in the SALESPERSON dimension is $U_{SALESPERSON}$ = {Day, Semester} and μ_{SALES} = Day. Day $\rightsquigarrow$ Day and Day $\rightsquigarrow$ Semester; then, measures can be aggregated in any level of the SALESPERSON dimension.

2.3 Conclusion

Motivated by the reclassifications of members of dimension levels, we extended a formal temporal multidimensional model in order to allow different temporal units in a dimension, *i.e.*, making it a temporal multidimensional model supporting different time granularities for associations between members of different pairs of levels in a dimension. Our extension helps to represent some situations from the real world with more accuracy. We also provide rules to guarantee disjointness and completeness conditions in our model. These conditions are required in order to guarantee correct aggregation of measures through dimensional hierarchies, *i.e.*, summarizability.

As a future work, we plan to extend our model in order to support many-to-many relationships between dimension values, *i.e.*, relaxing the disjointness condition. For example, a product may belong to several categories simultaneously.

Based on our model, in Chapter 6 we formalize the notion of *season*, a notion that leads to interesting queries (*season queries*), useful for decision-makers in several situations and application domains.

Chapter 3: Supporting the Change in the Degree of Containment in a Multidimensional Model

3.1 Introduction

In Chapter 2 we proposed a formal multigranular temporal multidimensional model in order to deal with a type of dimension change, the reclassification. In this chapter, we focus on another type of dimension change, the *change in the degree of containment.*

As we explained in Chapter 1, the hierarchical organization between the dimension levels captures their *full containment* relationship. For example, consider a LOCATION dimension with *Highway*, *Department*, *Country*, and *All* levels, see Section 3.2. A department is fully contained in a country; however, a highway is not necessarily fully contained in a department. In order to manage this situation, Jensen [2004] proposed a generalization of full containment, the *partial containment*.

The partial containment allows us to represent situations in which a dimension value is not fully contained in another. For example, a highway can be contained only 0.2 (20%) in a department. However, the model of Jensen [2004] does not support a possible change in the degree (percentage) of containment between two dimension values. For example, at a time t_i the degree of containment of a highway in a department is 0.2, but at a time t_{i+1}, this degree may change due to construction or destruction of highway sections.

Other examples where evolution of the degree of containment can arise are the containment of a jungle in a country, the containment of a group of animals in a geographic region, the containment of a tumour in an organ. In order to support this type of change, we extend the model of Jensen [2004]. To the best of our knowledge, this aspect has not yet been examined in previous works. Our extension is incorporated into a multidimensional query language as well, which enables what-if analysis (hypothetical queries), a very important decision support process as stated in Balmin [2000].

This chapter is organized as follows. In Section 3.2, we present a motivating example. In Section 3.3, we present Jensen's multidimensional model that supports partial containment. Next, in Section 3.4, we introduce the extension to support the change in the degree of containment and in Section 3.5, we incorporated our extension into a multidimensional query language, give examples, and present some basic experiments. Finally, in Section 3.6, we draw conclusions and outline future work.

3.2 Motivating example

Consider the road infrastructure of a country composed of highways that run through its departments (states). Figure 3.1 illustrates a situation where three highways (hw_1, hw_2, and hw_3) run through three departments (dep_1, dep_2, and dep_3).

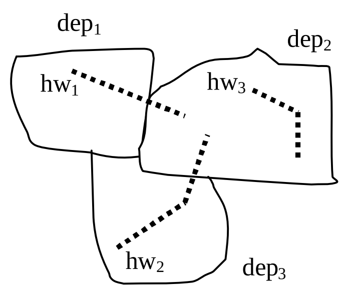


Figure 3.1. Road infrastructure of a country.

The traffic authorities are interested in analyzing such things as car accidents, *e.g.*, to identify what highways have a higher accident rate in order to improve their control, change its route, or take other measures to reduce accidents. In this scenario, accidents are the phenomena of interest, *i.e.*, they are the facts, that occur in one place and at a certain date (geographical and temporal dimensions). Figure 3.2 presents a multidimensional model to represent this situation (the notation of Jensen is used [Jensen 2004] to indicate full and partial containment) and Table 3.1 shows a sample data of the fact table of accidents. Note that each fact instance corresponds to the *set of accidents* that occurred in a highway at a particular date.

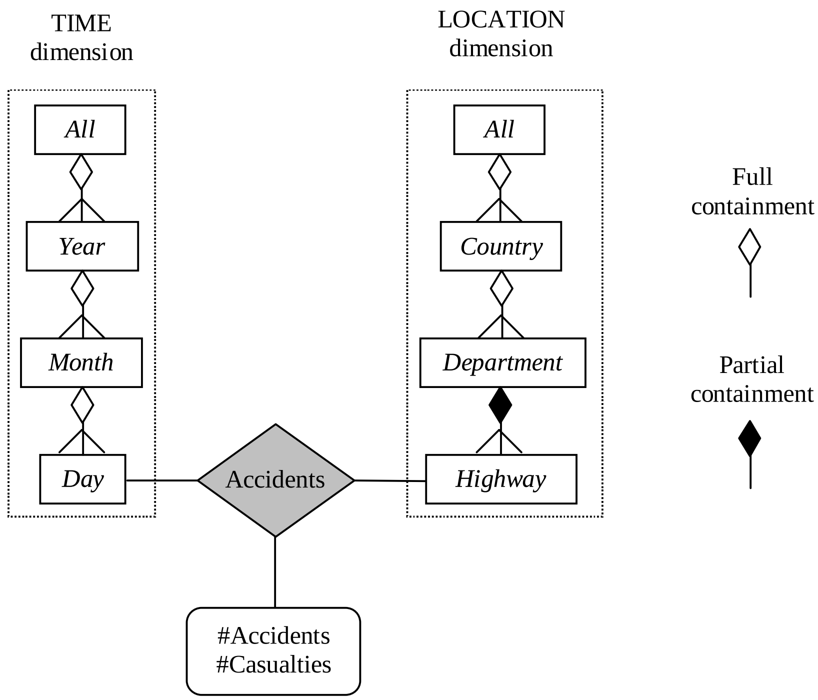


Figure 3.2. Multidimensional model for the analysis of accidents.

Table 3.1. Sample data of the fact table of accidents.

Bottom levels		**Measures**	
Day	***Highway***	**#Accidents**	**#Casualties**
…			
2008-Jan-01	hw_1	2	5
2008-Jan-01	hw_2	1	2
2008-Jan-02	hw_1	3	9
2008-Jan-02	hw_2	1	2
2008-Jan-03	hw_3	1	3
2008-Jan-04	hw_2	2	4
…			
2008-Jan-20	hw_2	3	3
…			

Suppose that the degree of containment of the highway hw_2 in the department dep_2 is 0.2 and in the department dep_3 is 0.8. Consider the query: What is the total number of accidents that have occurred in the department dep_2?

From Figure 3.1 it is noted that the facts associated with the highway hw_3 contribute to the total requested since that highway is fully contained in the department dep_2; however, with regard to the facts associated with the highway hw_2 there is not such certainty.

Nevertheless, it is possible to give an approximate answer to this query, see Table 3.2, if we consider the degree of containment of a highway in a department and the data are distributed proportionately.

Table 3.2. Calculation of the total number of accidents in the department dep_2 (a degree of containment equal to 0.2 of the highway hw_2 in the department dep_2 is considered).

Highway	**Total number of accidents**	**Degree of containment in the department dep_2**	**Estimated number of accidents in the department dep_2**
hw_1	5	0.2	5 * 0.2 = 1
hw_2	7	0.2	7 * 0.2 = 1.4
hw_3	1	1	1 * 1 = 1
		Total	**3.4**

Suppose now that the degree of containment of the highway hw_2 in the departments dep_2 and dep_3 changes as shown in Figure 3.3. The degree of containment of the highway hw_2 in both departments is now 0.5 due to the addition of a highway section.

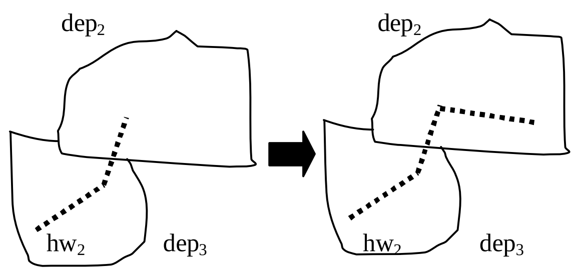


Figure 3.3. Change in the partial containment: growth of the highway hw_2.

Consider again the query raised and suppose that the new highway section will be available for vehicle traffic from 2008-Jan-15. Note that we must keep the evolution of changes in the degrees of containment of the highways in the departments, in order to obtain consistent results over time. Otherwise, all the facts prior to 2008-Jan-15 associated with the highway hw_2, would give the impression that they occurred when the degree of containment of the highway hw_2 in both departments is 0.5.

Table 3.3 shows the results that we obtain by applying the *current* degree of containment to all the data, *i.e.*, without considering the degree of containment when the facts occurred (5.5 accidents).

Conversely, the results of Table 3.4 are consistent with regard to the degree of containment when the facts occurred (4.3 accidents).

Table 3.3. Calculation of the total number of accidents in the department dep_2 (current degree of containment of the highway hw_2 in the department dep_2 is considered).

Highway	**Total number of accidents**	**Degree of containment in the department dep_2**	**Estimated number of accidents in the department dep_2**
hw_1	5	0.2	5 * 0.2 = 1
hw_2	7	0.5	7 * 0.5 = 3.5
hw_3	1	1	1 * 1 = 1
		Total	**5.5**

Table 3.4. Calculation of the total number of accidents in the department dep_2 (the degree of containment when the facts occurred is considered).

Highway	**Total number of accidents**	**Degree of containment in the department dep_2**	**Estimated number of accidents in the department dep_2**
hw_1	5	0.2	5 * 0.2 = 1
hw_2	4	0.2	4 * 0.2 = 0.8
hw_2	3	0.5	3 * 0.5 = 1.5
hw_3	1	1	1 * 1 = 1
		Total	**4.3**

In the model of Jensen [2004] the history of such changes is not preserved. In Section 3.4, we present the corresponding extension in order to support this situation.

3.3 Multidimensional model with partial containment

We present next the essential concepts of the multidimensional model of Jensen [2004], which supports partial containment.

3.3.1 Multidimensional schema

A *multidimensional schema* is a 2-tuple S = (F, DT), where F is a *fact type* and DT = {D_i, i = 1,…, n} is a set of *dimension types*. A dimension type D is a 4-tuple (LT_D, $\preccurlyeq$, *All*, $\perp$), where LT_D = {Lt_i, *i* = 1,…, k} is a set of *level types*. $\preccurlyeq$ is a partial order on the set LT_D. *All* is the top element of the partial order and $\perp$ represents the bottom element of the partial order. *All* represents the highest grouping level of the dimensional values and $\perp$ the lowest. The domain of *All* is a single value: dom(*All*) = {*all*}.

Example 3.1. Let Accidents = {A, DT} be a multidimensional schema, where A is a fact type for representing accidents and DT = {TIME, LOCATION}:

- TIME = (LT_{TIME}, $\preccurlyeq$, *All*, $\perp$), LT_{TIME} = {*Day*, *Month*, *Year*, *All*}, and $\perp$ = *Day*. The corresponding partial order is shown in Figure 3.4 (a).
- LOCATION = ($LT_{LOCATION}$, $\preccurlyeq$, *All*, $\perp$), $LT_{LOCATION}$ = {*Highway*, *Department*, *Country*, *All*}, and $\perp$ = *Highway*. The corresponding partial order is shown in Figure 3.4 (b).

Note that to represent a partial order, its transitive reduction is used (Hasse diagram [Freese 2004]).

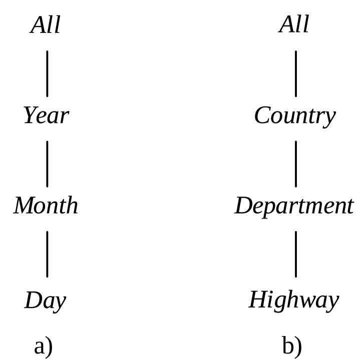


Figure 3.4. Dimension types: a) TIME and b) LOCATION.

3.3.2 Dimension instance

Given a multidimensional schema S = (F, DT), a dimension instance $\underline{d}$ of dimension type D $\in$ DT, is a 2-tuple $\underline{d}$ = ($L_{\underline{d}}$, §), where $L_{\underline{d}}$ = {lev_i, i = 1,…, k} is a set of levels. Each level *lev* is of level type *Lt* $\in$ LT_D, *i.e.*, a level *lev* is a set of values of level type *Lt*. § is a partial order on $\cup_i lev_i$ (union of all the values of the levels of a dimension instance). For simplicity, we henceforth write Dim instead of $\cup_i lev_i$.

Example 3.2. Let <u>time</u> be an instance of the dimension type TIME and <u>location</u> an instance of the dimension type LOCATION, see Example 3.1:

- <u>time</u> = {$L_{\underline{time}}$, §}, $L_{\underline{time}}$ = {*day*, *month*, *year*, *all_time*}, where *day* is of level type *Day*, *month* is of level type *Month*, *year* is of level type *Year*, and *all_time* is of level type *All*. *day* = {2007-Jan-01, 2007-Jan-02,…, 2008-Dec-31}, *month* = {2007-Jan, 2007-Feb,…, 2008-Dec}, *year* = {2007, 2008}, and *all_time* = {*all*}. The corresponding partial order is shown in Figure 3.5 (a).
- <u>location</u> = {$L_{\underline{location}}$, §}, $L_{\underline{location}}$ = {*highway*, *department*, *country*, *all_location*}, where *highway* is of level type *Highway*, *department* is of level type *Department*, *country* is of level type *Country*, and *all_location* is of level type *All*. *highway* = {hw_1, hw_2, hw_3}, *department* = {dep_1,

dep_2, dep_3}, *country* = {ct_1}, and *all_location* = {*all*}. The corresponding partial order is shown in Figure 3.5 (b).

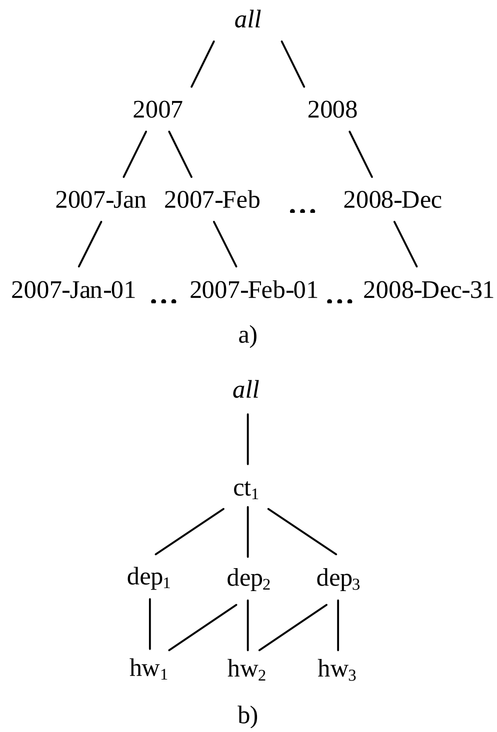


Figure 3.5. Dimension instances: a) time and b) location.

3.3.3 Degree of containment

Given two dimension values $v_1 \in$ Dim and $v_2 \in$ Dim, and a number $g \in [0; 1]$, the notation $v_1 \S_g v_2$ means that v_1 is contained in v_2 at least in g * 100%. g is the *degree of containment* of v_1 in v_2. If $g = 1$ means that v_1 is fully contained in v_2 and if $g = 0$ means that v_1 *may* be contained in v_2 (if containment does exist, the value of the degree is unknown).

Jensen [2004] presents several transitivity rules to infer degrees of containment between dimension values. In the following $v_3 \in$ Dim, $p \in [0; 1)$, and $q \in [0; 1)$.

i) Transitivity of full containment: if $v_1 \S_1 v_2$ and $v_2 \S_1 v_3$ then $v_1 \S_1 v_3$,
ii) Transitivity between partial and full containment: if $v_1 \S_p v_2$ and $v_2 \S_1 v_3$ then $v_1 \S_p v_3$,
iii) Transitivity between full and partial containment: if $v_1 \S_1 v_2$ and $v_2 \S_p v_3$ then $v_1 \S_0 v_3$, and
iv) Transitivity of partial containment: if $v_1 \S_p v_2$ and $v_2 \S_q v_3$ then $v_1 \S_0 v_3$.

For example, the rule iii) states that if v_1 is fully contained in v_2 and v_2 is contained in v_3 in p * 100% ($p < 1$), then it can only be inferred that v_1 *may* be contained in v_3 ($v_1 \S_0 v_3$).

3.3.4 Fact-dimension relation

A *fact-dimension relation* r is defined as $r \subseteq f \times Dim$, where f is a set of facts of fact type F, see Subsection 3.3.1. Each fact of f must be related to at least one value of each dimension. For simplicity, we assume that each fact is related to only a value of each dimension and the corresponding dimension value belongs to the bottom level of the dimension.

Example 3.3. Consider again Example 3.1. Let accidents = $\{ac_1, ac_2, ac_3, ac_4, ac_5\}$ be a set of facts of fact type A. Let the fact-dimension relations be:

- r_1 = {(ac_1, 2008-Jan-01), (ac_2, 2008-Jan-01), (ac_3, 2008-Jan-02), (ac_4, 2008-Jan-02), (ac_5, 2008-Jan-03)}.
- $r_2 = \{(ac_1, hw_1), (ac_2, hw_2), (ac_3, hw_1), (ac_4, hw_2), (ac_5, hw_3)\}$.

The relations r_1 and r_2 associate the set of facts accidents with the values of dimension instance <u>time</u> as well as with the dimension instance <u>location</u> from Example 3.2, respectively.

3.3.5 Fact characterization

The term fact characterization is defined from a fact-dimension relation r. It is said that a fact is characterized by a dimension value, if the fact is associated directly or indirectly (by transitivity in the partial order § of the dimension values) with such value, *i.e.*, a fact $f_1 \in f$ is characterized by a value $v_1 \in Dim$, written $f_1 \rightarrow v_1$, if: $(f_1, v_1) \in r$ or if there exists a value $v_2 \in Dim$ such that $(f_1, v_2) \in r$ and $v_2 \S v_1$.

Example 3.4. In Figure 3.6: $ac_1 \rightarrow hw_1$, $ac_1 \rightarrow dep_2$, $ac_1 \rightarrow dep_3$, $ac_1 \rightarrow ct_1$, $ac_5 \rightarrow hw_3$, $ac_5 \rightarrow dep_2$, and $ac_5 \rightarrow ct_1$.

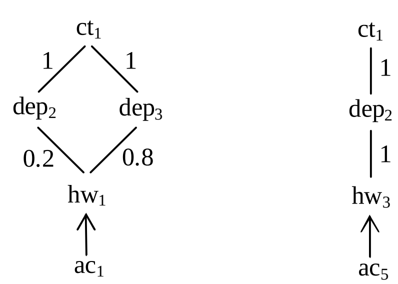


Figure 3.6. Facts ac_1 and ac_5 associated with dimension values.

3.3.6 Multidimensional object

After specifying the dimensions, the fact-dimension relation, and the fact characterization; the Multidimensional Object (**MO**) is then defined. Informally, a **MO** is a cube [OLAP Council 2009], *i.e.*, a group of cells (that contain the measures) associated with a set of dimension values. Formally, a **MO** is a 4-tuple **MO** = (S, f, DI, R), where S = (F, DT) is a multidimensional schema, f is a set of facts of fact type F, DI is a set of dimension instances each one of dimension type D $\in$ DT, and R is a set of fact-dimension relations.

Example 3.5. Let **AccidentsCube** = (Accidents, accidents, {$\underline{\text{time}}$, $\underline{\text{location}}$}, {$r_1$, r_2}) be a **MO**, where Accidents is the multidimensional schema of Example 3.1, accidents the set of facts of Example 3.3, {$\underline{\text{time}}$, $\underline{\text{location}}$} is the set formed by the dimension instances from Example 3.2, and {r_1, r_2} is the set formed by the fact-dimension relations from Example 3.3.

3.4 Support of the change in the degree of containment

The degree of containment between two dimension values may change over time. For example, in Figure 3.3 is shown the change in the degree of containment between a) the highway hw_2 and the department dep_2 and b) the highway hw_2 and the department dep_3.

In order to support the change in the degree of containment, the following extension to the model of the previous section is proposed. Let (LT_D, $\preccurlyeq$, *All*, $\perp$, μ) be a dimension type, where μ is a temporal unit (hours, days, months, years, among others). μ defines the temporal accuracy required (granularity) for the application to record the degree of containment between the dimension values.
Consider a pair of level types (Lt_1, Lt_2) $\in$ LT_D. Let $\underline{d}$ = ($L_{\underline{d}}$, §) be a dimension instance of dimension type D. Let the level $lev_1 \in L_{\underline{d}}$ be of level type Lt_1 and the level $lev_2 \in L_{\underline{d}}$ be of level type Lt_2. For the pair (lev_1, lev_2) a DC (Degree of Containment) function is defined with signature: $lev_1 \times lev_2 \times \text{dom}(\mu) \rightarrow [0;1]$. The DC function returns the degree of containment at a given time of a value of lev_1 with regard to a value of lev_2.

Example 3.6. Let LOCATION = ($LT_{LOCATION}$, $\preccurlyeq$, *All*, $\perp$, μ) be a dimension type, where μ = Day. Consider the pair of level types (*Highway, Department*) from Example 3.1. Let $\underline{\text{location}}$ = {$L_{\underline{location}}$, §} be an instance of the dimension type LOCATION, $L_{\underline{location}}$ = {*highway*, *department*, *country*, *all_location*}, *highway* is of level type *Highway* and *department* is of level type *Department.* For the pair (*highway*, *department*) a DC function is defined; some of their values are shown in Table

3.5 and are illustrated in Figure 3.7. For example, DC(hw_2, dep_3, 2008-Jan-01) = 0.8 and DC(hw_2, dep_3, 2008-Jan-15) = 0.5.

Table 3.5. Sample data of the DC function for (*highway*, *department*).
hw $\in$ *highway*, dep $\in$ *department*, and t $\in$ dom(*Day*).

hw	**dep**	**t**	**DC**
…			
hw_2	dep_2	2008-Jan-01	0.2
hw_2	dep_3	2008-Jan-01	0.8
hw_2	dep_2	2008-Jan-02	0.2
hw_2	dep_3	2008-Jan-02	0.8
…			
hw_2	dep_2	2008-Jan-15	0.5
hw_2	dep_3	2008-Jan-15	0.5
…			

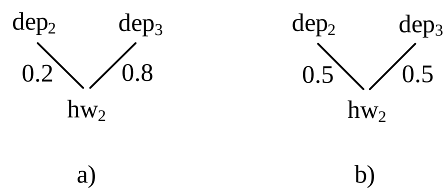


Figure 3.7. Degree of containment of the highway hw_2 in the departments dep_2 and dep_3: a) between 2008-Jan-01 and 2008-Jan-14 and b) from 2008-Jan-15.

For calculating the degree of containment between two dimension values that are not adjacent in the hierarchy, the rules of transitivity from the Subsection 3.3.3 are applied.

Example 3.7. Consider Figure 3.1 and suppose that the DC(hw_1, dep_1, 2008-Jan-31) = 0.8, see Figure 3.8 (a). Suppose that from 2008-Feb-01, the section of the highway hw_1 in the department dep_2 is eliminated, thus DC(hw_1, dep_1, 2008-Feb-01) = 1, see Figure 3.8 (b). Therefore, by applying the transitivity rules, it is obtained that DC(hw_1, ct_1, 2008-Jan-31) = 0.8 and DC(hw_1, ct_1, 2008-Feb-01) = 1.

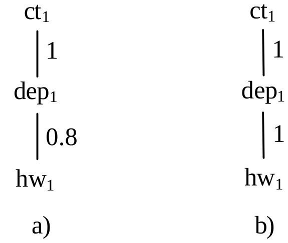


Figure 3.8. Degree of containment of the highway hw1 in the department dep1: a) in 2008-Jan-31 and b) in 2008-Feb-01.

3.5 Integration into a multidimensional language

This section illustrates how our proposed extension can be incorporated into a multidimensional query language. We present also some basic experiments related to accidents in Mexican highways.

3.5.1 Language

Although MDX (Multidimensional Expressions) [Whitehorn 2005] is a language which in recent years has become a *de facto* standard to query multidimensional data, we use the multidimensional query language proposed by Datta [1999], because of its similarity to the relational algebra. We use the operators of selection (σ) and aggregation (α). We give next a brief description of these operators. For details, refer to Datta [1999].

i) σ: allows us to select values from dimensions.

Notation: $\sigma_P(\mathrm{Cube}_1) = \mathrm{Cube}_2$, where P is a predicate, and

ii) α: applies aggregate functions to measures with one or more dimension levels specified as grouping attributes.

Notation: $\alpha_{[AL,\ GDL]}(\mathrm{Cube}_1) = \mathrm{Cube}_2$. AL is a list of elements $g_i(m_i)$ where g_i is an aggregate function applied to measure m_i, and GDL is a set of grouping dimensions levels.

For all the queries, the **AccidentsCube** cube from the Example 3.5 is used.

Query 3.1. What is the total number of accidents that have occurred in the department dep_2?

$\alpha_{[\mathrm{SUM(\#Accidents * DC(highway, 'dep2', day))}]}(\textbf{AccidentsCube})$

That is, all the facts from the **AccidentsCube** cube are selected. Then for each fact, the degree of containment of the corresponding highway in the department dep_2 is found, and this value is then multiplied by the number of accidents. Next, the total requested is obtained using the aggregate function SUM. The same query formulated in an SQL-like way is

```
SELECT SUM(#Accidents * DC(highway, 'dep2', day))
FROM AccidentsCube
```

Note that to calculate the degree of containment, the date (day) associated with the fact is used, *i.e.*, the degree of containment when the facts occurred is used. However, it is possible to formulate hypothetical queries in order to analyze past behaviors and make predictions, as exemplified in the following queries.

Query 3.2. What would have been the total number of accidents occurred in the department dep_2 if the existing degree of containment in the highways in such department in 2007-Jan-01 were considered?

$\alpha_{[\text{SUM(\#Accidents * DC(highway, 'dep2', '2007-Jan-01'))}]}((\textbf{AccidentsCube}))$

In this query, all the facts from the **AccidentsCube** cube are considered, *e.g.*, facts from 2007 and from 2008, but the degree of containment corresponding to 2007-Jan-01 is used.

Query 3.3. What would have been the total number of accidents occurred in the department dep_2 in 2007 given the current degree of containment of highways in that department? The current date is represented by *now*.

$\alpha_{[\text{SUM(\#Accidents * DC(highway, 'dep2', now))}]}(\sigma_{\text{day > '2007-Jan-01' AND day < '2007-Dec-31'}}(\textbf{AccidentsCube}))$

In this query, only the facts from the **AccidentsCube** cube from 2007 are selected, but the degree of containment corresponding to the current date is used.

3.5.2 Some basic experiments

In order to make some basic experiments, we built our multidimensional model for the analysis of accidents in a relational way using Oracle. We built the DC function using a many-to-many relationship between highway and department and a stand-alone Oracle function that was invoked from SQL queries.

We took data about accidents, highways, and departments (states) from Instituto Mexicano del Transporte [IMT 2009]. In Figure 3.9 we show the configuration of some highways in 2002 and in 2005. In Table 3.6 we present data about the number of accidents in these highways and in Table 3.7 we show the degree of containment of each highway in each department. Finally, in Table 3.8 we present the corresponding calculations of the total number of accidents in each department:

i) applying the corresponding degree of containment when the accidents occurred,

ii) applying to all the accidents, the degree of containment of the highways in 2002, and

iii) applying to all the accidents, the degree of containment of the highways in 2005.

For example, the calculations for highway M-002D and department Baja California in Table 3.8 are made as follows. Column i) 84 * 0.33 + 206 * 0.26 = 81, column ii) (84 + 206) * 0.33 = 96, and column iii) (84 + 206) * 0.26 = 75.

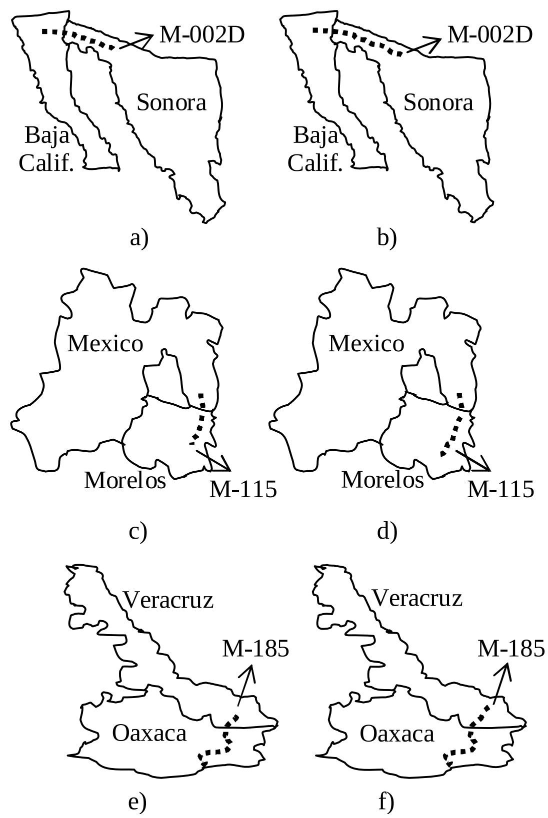


Figure 3.9. Configuration of highways: a) highway M-002D in 2002, b) highway M-002D in 2005, c) highway M-115 in 2002, d) highway M-115 in 2005, e) highway M-185 in 2002, and f) highway M-185 in 2005.

Table 3.6. Total number of accidents in 2002 and 2005.

Highway	**Year**	**#Accidents**
M-002D	2002	84
M-002D	2005	206
M-115	2002	263
M-115	2005	269
M-185	2002	26
M-185	2005	45

Table 3.7. Degree of containment of each highway in each department in 2002 and 2005.

Highway	Year	*Department*	Length (km)	Degree of containment
M-002D	2002	Baja Calif.	46.46	0.33
M-002D	2002	Sonora	92.94	0.67
M-002D	2005	Baja Calif.	46.46	0.26
M-002D	2005	Sonora	134.84	0.74
M-115	2002	Mexico	50.21	0.38
M-115	2002	Morelos	80.69	0.62
M-115	2005	Mexico	50.21	0.31
M-115	2005	Morelos	110.24	0.69
M-185	2002	Oaxaca	168.49	0.71
M-185	2002	Veracruz	68.11	0.29
M-185	2005	Oaxaca	168.49	0.67
M-185	2005	Veracruz	84.21	0.33

Table 3.8. Calculations of the total number of accidents: i) using the degree of containment when the accidents occurred, ii) using the degree of containment in 2002, and iii) using the degree of containment in 2005.

Highway	*Department*	i)	ii)	iii)
M-002D	Baja Calif.	81	96	75
M-002D	Sonora	209	194	215
M-115	Mexico	183	202	165
M-115	Morelos	349	330	367
M-185	Oaxaca	49	50	48
M-185	Veracruz	22	21	23

3.6 Conclusions and future work

In this chapter, we adopted a multidimensional model that supports partial containment. This model was extended in order to allow the possible change in the degree of containment between dimension values. The extension was also incorporated into a multidimensional query language. This enables the formulation of queries that are consistent with time. Furthermore, it allows the formulation of hypothetical queries (What if? What would have happened if?), which can help decision-makers.

As future work, we plan to incorporate our proposal into a platform such as Pentaho [2009] or Microsoft Analysis Server [Microsoft 2009]. However, since these platforms are oriented to multidimensional models that support full containment, the introduction of our extension poses interesting challenges. On the other hand, from the point of view of language, both platforms support MDX. However, since MDX is also oriented to the management of full containment, the incorporation of our proposal into this language poses challenges as well.

Finally, more extensive experiments and analysis of results are needed in order to try to identify possible behaviors. It would be interesting to analyze other domains where partial containment arises, *e.g.*, facts as crimes and fish catches, associated with regions that are located among several countries or departments (states).

Chapter 4: Extensions to the Map Cube Operator

4.1 Introduction

A GIS [Tomlin 1990], [Longley 2005] can integrate, store, edit, analyze, share, and display geographically referenced information. Although a GIS can be used for managing geographic data for decision support, a GIS usually works with geographic data separately from other business data [Pestana 2005] and it offers minimal analytical capabilities for non-geographic data [Ferri 2000], [Yin 2000], [Bédard 2001], [Rivest 2001].

On-Line Analytical Processing (OLAP) [Codd 1993] allows querying, browsing, and summarizing information in an efficient, interactive, and dynamic way. OLAP provides an aggregation approach to analyze large amounts of detailed data (usually represented in an alphanumeric format) typically over a DW. Thus, while OLAP offers powerful analytic capabilities, GIS offers spatial functionality.

We believe that OLAP-GIS integration is very promising. Other authors [Yin 2000], [Scotch 2005], [Cely 2006] also recognize the need for integrating these technologies. From an architectural functional point of view, we classify OLAP-GIS works into two groups: middleware and DW proposals. Middleware proposals [Ferri 2000], [Yin 2000], [Kouba 2000], [Miksovský 2001], [Ferreira 2001], [Da Silva 2004], [Scotch 2005]; make it possible to query geographic and business data together without changing the physical organization of data in both environments [Pourabbas 2005]. DW extension proposals [Rivest 2001], [Han 1998], [Pedersen 2001b], [Rao 2003], [Fidalgo 2004], [Sampaio 2006], [Jensen 2004], [Bimonte 2005], [Timko 2005], [Damiani 2006], [Malinowski 2008]; store and manage geographic data inside the DW. It means that the DW should offer spatial capabilities, such as a spatial storage engine, robust spatial data access, and a set of spatial functions in order to facilitate the spatial multidimensional analysis and to mimic GIS capabilities.

An OLAP-GIS integration provides business analysts with the opportunity to see strategic business data from a geographic point of view in a friendly and intuitive way. This can contribute to the detection of implicit and valuable spatial associations and patterns that otherwise would be very difficult to recognize. Thus, business analysts could see geographic data from different perspectives and various hierarchical levels. For example, in a crime scenario, police analysts could i) identify the places in each neighborhood where crimes concentrate, year by year, by type of crime, and ii) perform a spatial *roll-up* operation to view crime data at a more aggregated level, *e.g.*, going from the *Neighborhood* level to the *City* level.

Other scenarios where an OLAP-GIS integration can be useful are:

- **Health.** To identify the zones affected by different types of diseases. This could indicate points to relocate health centers or create new ones.
- **Agriculture.** To find the cultivated regions for different types of crops. This could indicate, *e.g.*, land parcels where some type of crop should be replaced in order to improve irrigation and fumigation controls.
- **Traffic control.** To find the route map in each neighborhood by type of transport (buses, trucks, trains). This could indicate zones where more routes are needed or zones with an excess of routes.

The map cube operator [Shekhar 2001] can accomplish tasks such as the previous ones. Map cube supports spatial aggregation in a spatial multidimensional database and enables visualization of information through maps. For example, in the crime scenario, we can use the map cube operator to aggregate the points where crimes were committed, and show the resulting maps: grouping of crime points by neighborhood and type of crime, by neighborhood regardless of type of crime, by type of crime regardless of neighborhood, and for a whole city, regardless of the neighborhood or type of crime.

Unfortunately, map cube only supports spatial aggregation using geometric union function; however, other spatial aggregate functions could be used. For example, in the crime scenario, functions such as center of mass, convex hull, and area-of-influence polygons (Voronoi diagram) could be appropriate to identify places where crimes concentrate.

In this chapter, we extend map cube in order to support spatial aggregate functions other than geometric union. In addition, we extend map cube for supporting several aggregate functions simultaneously and to overlay its results with maps. For example, in the crime scenario, we could apply the map cube operator using center of mass and convex hull as aggregate functions, and overlay its results with a map of hospitals and police stations. To the best of our knowledge, there are no previous works that have extended the map cube operator this way.

The remainder of the chapter is organized as follows. In Section 4.2, we describe spatial DWs. In Section 4.3, we present the map cube operator, point out some of its shortcomings and grammatical inconsistencies, and propose some improvements. In Section 4.4, we describe spatial aggregate functions and in Section 4.5, we illustrate our proposal with a case study about crimes. In Section 4.6, we end the chapter and outline future work.

4.2 From a conventional DW to a spatial DW

In order to illustrate how spatiality can be useful for business analysts, consider a DW model for crimes as shown in Figure 4.1. A sample data of the fact table Crimes is shown in Table 4.1. Each fact corresponds to the *set of crimes* of a particular type that happened in a given neighborhood on a specific date (day).

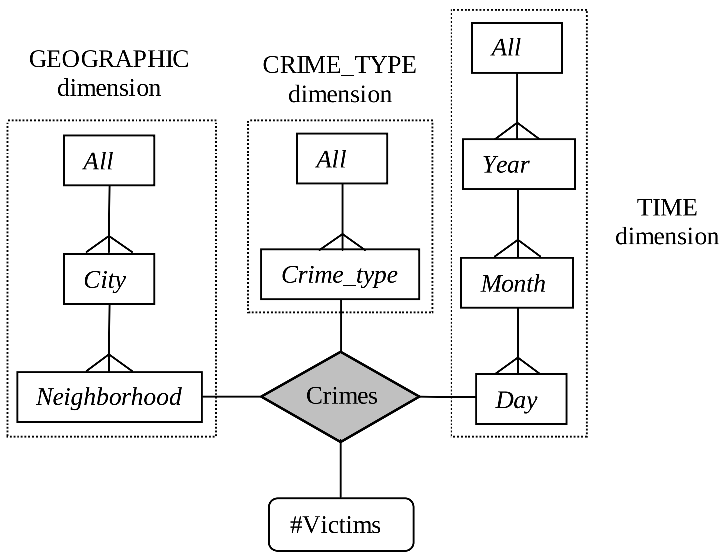


Figure 4.1. A conventional DW model for crimes.

Table 4.1. Crimes table.

Bottom levels			Measures
Neighborhood	***Crime_type***	***Day***	**#Victims**
East Garfield Park	Assault	2008-Oct-01	5
East Garfield Park	Vandalism	2008-Oct-01	3
East Garfield Park	Assault	2008-Oct-02	7
Logan Square	Vandalism	2008-Oct-01	2
Logan Square	Burglary	2008-Oct-02	3

Now, consider the query “find the total number of victims in each neighborhood”. The results are fifteen in East Garfield Park and five in Logan Square. Next, we add to our DW the geographic extent (region) of each neighborhood, see Figure 4.2. Such spatiality enhancement allows us to display the results of the previous query on a map, see Figure 4.2 (c).

Spatiality can also be added to the facts and spatial measures can arise. For example, suppose the points where crimes were committed are known, see Figure 4.2 (b). In Figure 4.2 (a) and Table 4.2 crime points (Crime_points) are handled as a spatial measure. Now, police analysts are enabled to formulate a query such as: What was the total number of victims and the center of mass of crimes in each neighborhood? The results are shown in Figure 4.2 (c). In the next section, we present and extend map cube, an operator that provides a simple way to formulate this type of queries.

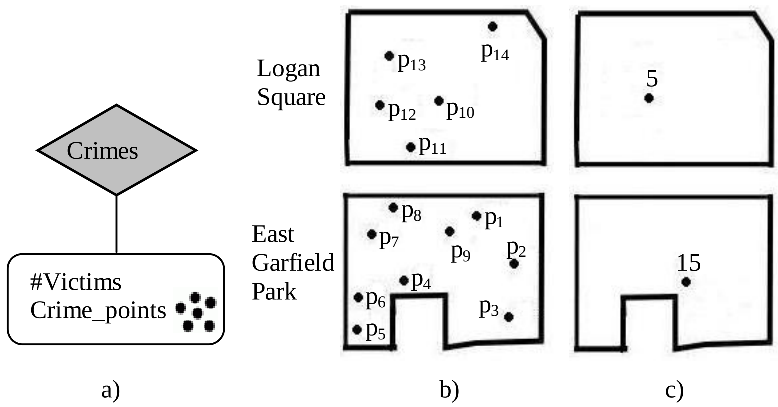


Figure 4.2. Adding a spatial measure: a) Crime_points measure, b) the points where crimes were committed, and c) the total number of victims and the center of mass of crimes in each neighborhood.

Table 4.2. Crimes table with spatial measure Crime_points.

Bottom levels			**Measures**	
Neighborhood	***Crime_type***	***Day***	**#Victims**	**Crime_points**
East Garfield Park	Assault	2008-Oct-01	5	$\{p_1, p_2, p_3\}$
East Garfield Park	Vandalism	2008-Oct-01	3	$\{p_4, p_5, p_6\}$
East Garfield Park	Assault	2008-Oct-02	7	$\{p_7, p_8, p_9\}$
Logan Square	Vandalism	2008-Oct-01	2	$\{p_{10}, p_{11}\}$
Logan Square	Burglary	2008-Oct-02	3	$\{p_{12}, p_{13}, p_{14}\}$
…				

4.3 The map cube operator

4.3.1 Overview

The map cube operator was developed by Shekhar [2001], [Lu 2003] as a spatial extension of the data cube operator [Gray 1997]. In turn, data cube is a generalization of the SQL GROUP BY clause. Given n grouping columns, data cube generates subtotals for all the possible combinations of these columns, *i.e.*, 2^n subtotals. Each combination is called a cuboid [Agarwal 1996]. For example, consider Table 4.2; a data cube by *Neighborhood* and *Crime_type* columns generates subtotals (*e.g.*, the total number of #Victims) by (*Neighborhood*, *Crime_type*), (*Neighborhood*), (*Crime_type*), and (*All*); *i.e.*, the grand total.

On the other hand, map cube enables spatial aggregation, *e.g.*, besides the sum of the number of victims, if we had the geographic points where crimes were committed, these points could be spatially aggregated. Map cube then associates a map with each cuboid generated by data cube and integrates data and maps in a single view. The corresponding map cube sentence is shown in Table 4.3.

Table 4.3. Example of map cube sentence.

Map cube sentence		Output
Base-Map	Crimes_Map	- Cuboid (*Neighborhood*, *Crime_type*) with its corresponding map. - Cuboid (*Neighborhood*) with its corresponding map. - Cuboid (*Crime_type*) with its corresponding map. - Cuboid (*All*) with its corresponding map.
Base-Table	Crimes	
Aggregate by	**SUM:** #Victims	
Reclassify by	Neighborhood, Crime_type	
Data cube dimension	Neighborhood, Crime_type	
Cartographic preference	Thickness = 1, Color = Blue	

Next we briefly describe the terms of this sentence: i) *Base-Map* represents the map where the spatial information lies, ii) *Base-Table* specifies the fact table, iii) *Aggregate by* specifies the column (measure) to be aggregated along with an aggregate function, iv) *Reclassify by* and *Data cube dimension* specify the grouping columns (dimensions). For additional details about them refer to [Shekhar 2001], and v) *Cartographic preference* specifies visualization parameters.

An *implicit* geometric union is performed over the spatial column Crime_points of the Crimes table. This column is related to the Crimes_Map. Thus, in this example, a geometric union of Crime_points is performed.

Unfortunately, after reviewing the map cube operator, we identified the following shortcomings, that we overcome in Subsection 4.3.3.

- Map cube does not allow spatial aggregate functions other than geometric union. This prevents the use of functions such as center of mass, convex hull, Voronoi diagram, and intersection, among others, that can be useful in different domains, see Section 4.1.
- It is impossible to perform more than one spatial aggregation in a single map cube sentence.
- It is impossible to overlay maps with the map cube results.

The original map cube grammar [Shekhar 2001], written in Yacc [Levine 1995], is shown in Figure 4.3.

```
Base-Map =                 <base-map name>
Base-Table =               <base-table name>
                           (Where <join attribute list>
                           (And <join attribute list>)*)?
Aggregate by               <aggregate list>
Reclassify by              <attribute list>
Data cube dimension        <attribute list>
Cartographic preference <carto attribute list>

<base-map name> → <name> (, <name>)*
<base-table name> → <name>
<aggregate list> → <aggregate unit> (<operator> <aggregate unit>)?
<aggregate unit> → <aggregate func> : <name>
<aggregate func> → SUM | MAXN | MINN | COUNT | MEDIAN
<join attribute list> → <name> <operator> <name>
<attribute list> → <name>? | <name> (, <name>)*
<carto attribute list> → <carto-attribute-value pair>  (, <carto-attribute-value pair>)*
<carto-attribute-value pair> → <carto-attribute> = <carto-value>
<carto-attribute> → Color | Thickness | Texture | Annotation |
                    Text | Symbol | Layout | Legend | Title |
                    No-of-map-per cuboid | Normalize
<carto-value> → <name> | <num>
<num> → <digit>+ (. <digit>+)? ( E (+ | -)? <digit>+)?
<name> → <letter> (<letter> | <digit> | <symbol>)*
<letter> → A | B | ... | Z | a | b | ... | z
<digit> → 0 | 1 | 2 | 3 | 4 | ... | 9
<symbol> → - | _ | , | . | :
<operator> → = | > | < | + | - | * | /
```

Figure 4.3. Original map cube grammar.

4.3.2 Grammar review

After reviewing the map cube grammar, we identified the following shortcomings:

- The **<aggregate list>** element does not allow us to specify multiple aggregate functions. For example, we cannot express: '**SUM:** column1**, COUNT:** column2'.
- There is no way to specify the spatial aggregate function to be used. The spatial aggregation is implicitly performed using geometric union.
- It is impossible to specify maps to be overlaid with the map cube results.
- The **Where** clause only supports logical conjunctions.

In addition, we identified the following inconsistencies:

- The **<name>** element allows us to include symbols that can generate confusions. For example, a valid name in this grammar is 'variable,variable'; if we use such a name in **<join attribute list>** element, we get an invalid comparison expression.
- The **<base-table name>** element should be defined in the same way as **<base-map name>** element, *i.e.*, as a list of names separated by commas. This suggests a lack of uniformity in the grammar.

- **<operator>** element allows us arithmetic and comparison operators. This can generate errors, *e.g.*, **<join attribute list>** element only makes sense for comparison operations; however, the grammar allows us arithmetic operations here. Similarly, **<aggregate list>** element only makes sense for arithmetic operations; however, the grammar allows us comparison operations here.

4.3.3 Proposed grammar changes

Following is the new grammar for map cube, see Figure 4.4. The changes are in blue.

```
Base Map                 <name list>
Base Table               <name list> (Where <condition>)?
Aggregate by             <aggregate list>
Reclassify by            <attribute list>
Data cube dimension      <attribute list>
Cartographic preference  (<overlay clause>)? <carto attribute list>

<name list> → <name> (, <name>)*
<condition> → <join attribute pair> (<logical operator> <join attribute pair>)*
<join attribute pair> → <column name> <comparison operator>
                        (<column name> | <value>)
<aggregate list> → <aggregate type> (, <aggregate type>)*
<aggregate type> → <simple aggregation> | <spatial aggregation>
<simple aggregation> → <simple aggregation unit> (<arithmetic operator>
                       (<simple aggregation unit> | <num>) )* (AS <name>)?
<simple aggregation unit> → (<simple aggregate function> |
                            <special aggregate function>): <column name>
<special aggregate function> → <numeric spatial function>: <spatial aggregate function>
<spatial aggregation> → <spatial aggregation unit> (AS <name>)?
<spatial aggregation unit> → <spatial aggregate function>: <column name>
<attribute list> → <column name> (, <column name>)*
<carto attribute list> → <carto-attribute-value pair> (, <carto-attribute-value pair>)*
<overlay clause> → Overlay: (<name list>)
<carto-attribute-value pair> → <carto-attribute> = <carto-value>
<carto-attribute> → Color | Thickness | Texture | Annotation |
                    Text | Symbol | Layout | Legend | Title |
                    No-of-map-per cuboid |Normalize
<carto-value> → <name> | <num>
<column name> → <name> | <compound column>
<compound column> → <name> . <name>
<user function> → <name>
<value> → '<name>' | <num>
<name> → <letter> (<letter> | <digit> | _ )*
<num> → (-)? (<digit>)+ ( . (<digit>)+)?
<letter> → A | B | ... | Z | a | b | ... | z
<digit> → 0 | 1 | 2 | 3 | 4 | ... | 8 | 9
<arithmetic operator> → + | - | * | /
<comparison operator> → =| > | < | >= | <=  | <>
<logical operator> → AND | OR
<simple aggregate function> → SUM | MAXN | MINN | COUNT | MEDIAN |
                              AVG | MAX | MIN | <user function>
<numeric spatial function> → AREA | PERIMETER | LENGTH | <user function>
<spatial aggregate function> → GEOMETRIC_UNION | INTERSECTION |
```

CENTER_OF_MASS | CONVEX_HULL | MBR |
MBC | VORONOI_DIAGRAM | <user function>

Figure 4.4. New map cube grammar.

The main changes are:

- **Base Map** and **Base Table** can contain a list of names, *i.e.*, **<table list>** element. A name can only contain letters, digits, and underscores.
- Arithmetic and comparison operators are separated into **<arithmetic operator>** and **<comparison operator>** elements respectively. We have also broadened the set of comparison operators with: **>=**, **<=**, and **<>**; and add a **<logical operator>** element that allows us to specify conjunctions and disjunctions.
- **<condition>** element is added to support join conditions and simple comparisons (a comparison between a column and a numeric or string value).
- **<aggregate list>** element is modified to support a list of simple and spatial aggregate functions. A *simple aggregate* function returns a numeric or a string value and can be of two types: a) a conventional aggregate function such as **COUNT**, **SUM**, **MAX**, and b) a combination of a numeric spatial function such as **AREA**, **LENGTH**, **PERIMETER**, with a spatial aggregate function, *e.g.*, '**AREA: GEOMETRIC_UNION:** spatialcolumn'. A spatial aggregate function returns a geometry or set of geometries, see Section 4.4. If there are several spatial aggregations in the same sentence, they are overlaid using a simple union process [Tomlin 1990], [Longley 2005].
- **<overlay clause>** element is added. It allows us to specify a list of maps to be overlaid with the map cube results. The overlay process is performed using a simple union process [Tomlin 1990], [Longley 2005].
- **<user function>** element allows us to specify user-defined functions. These can be of three types: simple aggregate, spatial aggregate, and numeric spatial.

4.4 Spatial aggregate functions

The main contribution of our approach is that the user can specify in the map cube operator the spatial aggregate functions needed for a particular application. Some of the most common spatial aggregate functions for a set of geometries *G* are the following:

- **Geometric union** returns the geometry or set of geometries covered by the geometries in G.
- **Intersection** returns the geometry or set of geometries shared by the geometries in G.

- **Center of mass** is a point at which the mass of the geometries in G may be considered to be concentrated. For example, the center of mass of a set P of points (with equal masses at each point) is the arithmetic mean of each coordinate of the points.
- **Convex hull** is the smallest convex polygon c that surrounds the geometries in G, *i.e.*, each geometry in G is either on the boundary or inside c.
- **Minimum bounding rectangle** (**MBR**) is the bounding geometry formed by the minimum and maximum X and Y coordinates in a geometry. This definition can be extended to a set G of geometries, as Figure 4.5 (c) shows.
- **Minimum bounding circle** (**MBC**) is the smallest circle that contains the geometries in G.
- **Voronoi diagram** for a set of points P is the partition of the plane that associates a region R(p) with each point p ∈ P in such a way that all points in R(p) are closer to p than to any other point in P. R(p) is the region of influence of p.

Figure 4.5 show examples of some of these functions for points. For each spatial aggregate function to be used in a map cube sentence, a **<spatial aggregation unit>** is required:

<spatial aggregation unit> → <spatial aggregate function>: <column name>

Where **<spatial aggregate function>** is the name of a spatial aggregate function and **<column name>** is the name of a column that contains spatial data. If the application requires a spatial aggregate function other than those listed above, the user can specify it in the **<user function>** element.

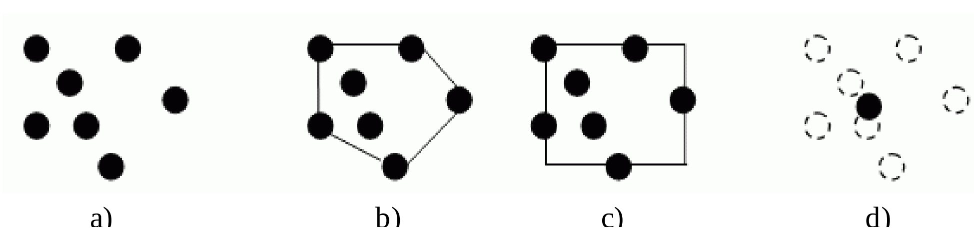


Figure 4.5. Examples of spatial aggregate functions for points: a) input set, b) convex hull, c) MBR, and d) center of mass.

4.5 Case study – analyzing crimes

We consider six Chicago Northwest neighborhoods (community areas): Logan Square, Hermosa, West Humboldt Park (WHP), Humboldt Park (HP), West Garfield Park (WGP), and East Garfield Park (EGP). We analyze data about three types of crimes: assault, burglary, and vandalism. As a source we use SpotCrime [2009]. Figure 4.6 shows our DW model. A sample data of Crimes table is shown in Table 4.2.

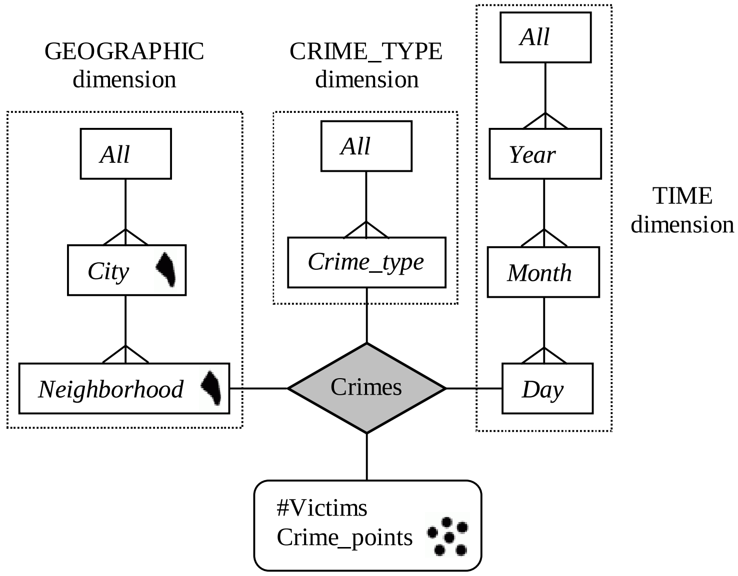


Figure 4.6. A spatial DW for crimes.

Crime_points is a spatial measure that represents the points where crimes were committed. We adopt Han's definition of spatial measure [Han 1998]. A spatial measure contains a collection of pointers to spatial objects; in our model, a collection of pointers to points. A map of the neighborhoods is shown in Figure 4.7 (a), and a map of crimes is shown in Figure 4.7 (b); the data correspond to the period from 2008-Oct-01 to 2008-Oct-08.

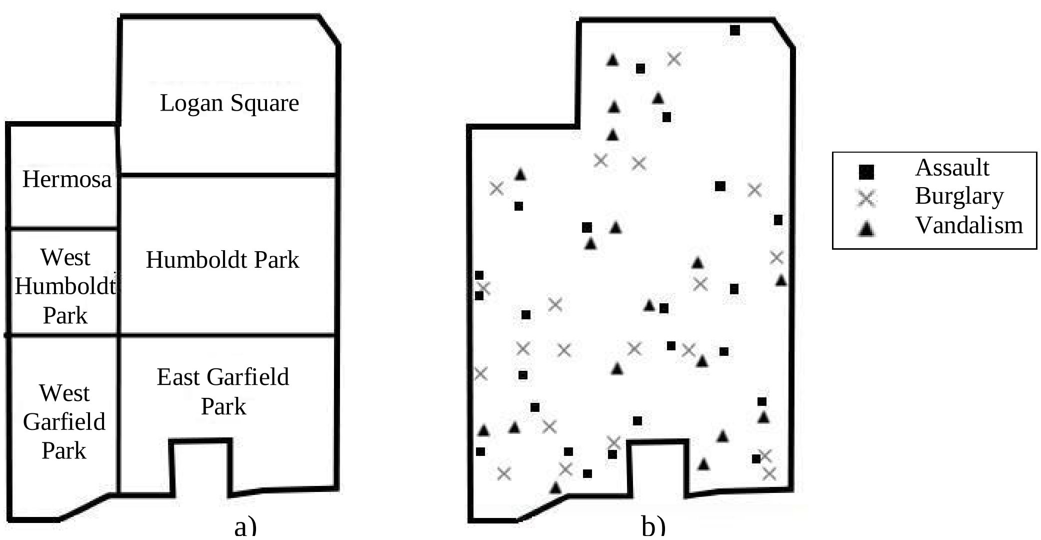


Figure 4.7. Maps: a) Neighborhoods_Map and b) Crimes_Map.

Now, suppose police analysts want to know where crimes concentrate: i) by neighborhood, ii) by type of crime, iii) by neighborhood and type of crime, and iv) by the whole city. We will use center of mass and convex hull to find the crimes concentration; however, other functions such as MBR could be used. Police analysts also want to know the total number of victims in accordance with previous groups and overlay the results with neighborhoods map. To solve this request, we can apply map cube as follows.

Base Map **Crimes_Map**
Base Table **Crimes**
Aggregate by **CONVEX_HULL: Crime_points AS Conv_hull,**
CENTER_OF_MASS: Crime_points AS Cent_mass,
SUM: #Victims AS Sum_victims
Reclassify by **Neighborhood, Crime_type**
Data cube dimension **Neighborhood, Crime_type**
Cartographic preference **Overlay: (Neighborhoods_Map)**

Map cube generates results for the cuboids (*Neighborhood*, *Crime_type*), (*Neighborhood*), (*Crime_type*), and (*All*). Table 4.4 shows the results by neighborhood and type of crime (only results for Logan Square and Hermosa are shown). Spatial aggregations of crimes are shown in Figure 4.8. Table 4.4 and Figure 4.8 make up cuboid (*Neighborhood*, *Crime_type*).

Table 4.4. Cuboid (*Neighborhood, Crime_type*).

Bottom levels		**Measures**		
Neighborhood	***Crime_type***	**Conv_hull**	**Cent_mass**	**Sum_victims**
Logan Square	Assault	CH_1	CM_1	8
Logan Square	Burglary	CH_2	CM_2	6
Logan Square	Vandalism	CH_3	CM_3	7
Hermosa	Assault	CH_4	CM_4	3
Hermosa	Burglary	CH_5	CM_5	2
Hermosa	Vandalism	CH_6	CM_6	2
…				

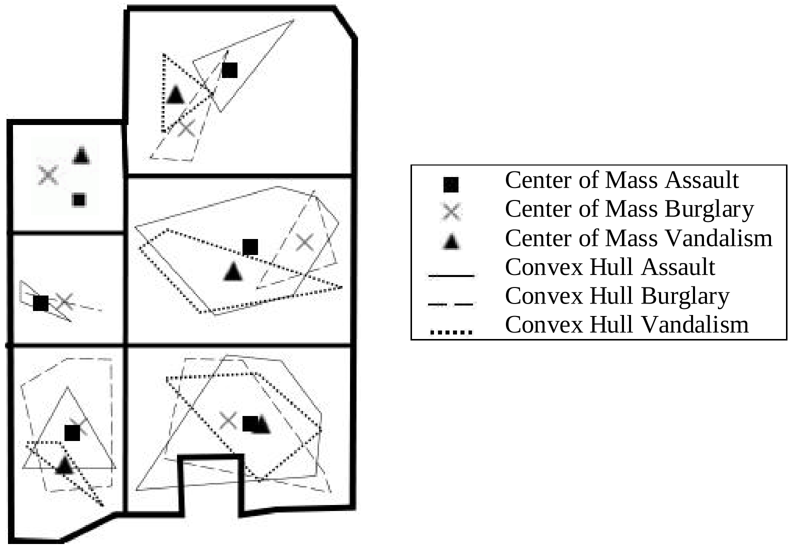


Figure 4.8. Concentration of crimes by neighborhood and type of crime.

Table 4.5 and Figure 4.9 show the results of cuboid (*Neighborhood*). In Figure 4.9, the center of mass in each neighborhood represents a potential point for implementation of security policies, *e.g.*, placing patrols around these points. Similarly, in the region defined by each convex hull in each neighborhood, more police officers could be assigned. Analogous results are generated for cuboids (*Crime_type*) and (*All*).

Table 4.5. Cuboid (*Neighborhood*).

Bottom levels		**Measures**		
Neighborhood	***Crime_type***	**Conv_hull**	**Cent_mass**	**Sum_victims**
Logan Square	*all*	CH_1	CM_1	21
Hermosa	*all*	CH_2	CM_2	7
West Humboldt Park	*all*	CH_4	CM_4	11
Humboldt Park	*all*	CH_3	CM_3	27
West Garfield Park	*all*	CH_6	CM_6	23
East Garfield Park	*all*	CH_5	CM_5	36

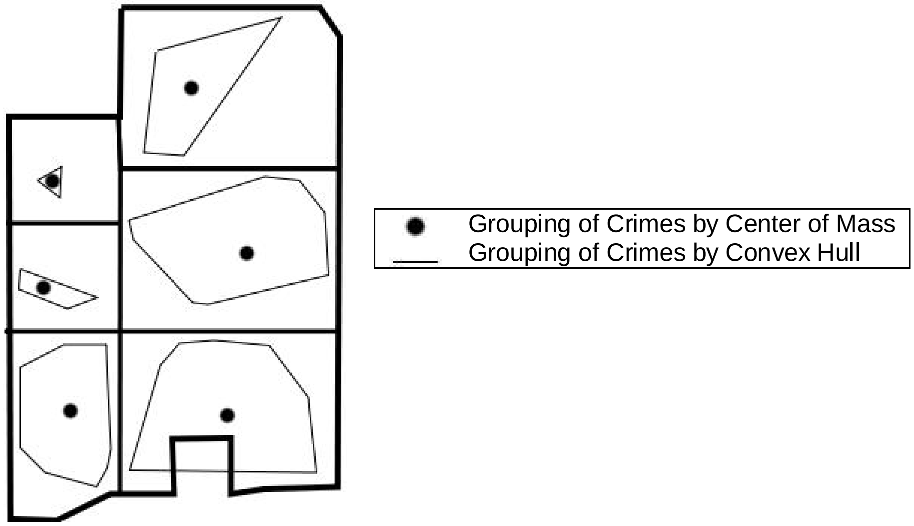


Figure 4.9. Concentration of crimes by neighborhood.

Finally, an example of the Voronoi diagram is shown in Figure 4.10, the Voronoi diagram of crimes for the cuboid (*Neighborhood*). For example, each region of the Voronoi diagram could help to assign patrols and police officers in order to attend more quickly crime reports.

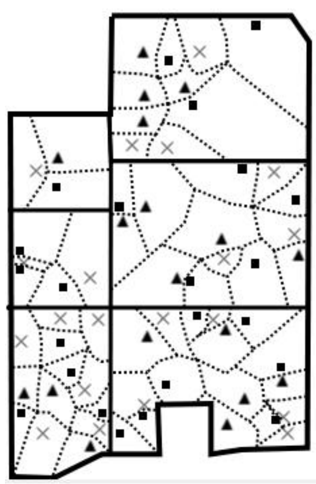


Figure 4.10. Voronoi diagram of crimes by neighborhood.

4.6 Conclusions and future work

In this chapter, we extended the functionality of the map cube operator. Our main contribution is to allow the user to choose the spatial aggregate functions appropriate for his domain. In addition, we extend this operator for supporting several spatial aggregate functions simultaneously and overlay its results with maps. We also fixed some inconsistencies of the map cube grammar.

To illustrate the convenience of our proposal, we presented a case study about crimes. The results presented in maps can help police analysts to identify spatial patterns at different levels of detail, *e.g.*, in the whole city and in each of its neighborhoods. Consequently, policies could be formulated in order to create or relocate, *e.g.*, police stations and hospitals and to place, *e.g.*, patrols and police officers across the city.

While the original version of map cube provides visualization facilities such as color, width of lines, and gray scale, among others, more features could be incorporated to this operator. For example, a symbol or color system that allows users to specify how they want certain regions and points to be depicted as in the case study about crimes, *e.g.*, squares and bold lines to represent assaults, crosses and dashed lines to represent burglaries, and triangles and dotted lines to represent vandalisms. However, the operator should not be overcrowded with features such as these, because a visualization tool could be more suitable for this purpose.

In addition, the visualization of some spatial aggregate functions may be difficult to understand. For example, consider the Voronoi diagram for geographical points where crimes occurred for the cuboid (*Crime_type*). Three Voronoi diagrams, one for each type of crime, are generated and overlaid in a single map. Unless we offer the user a way to distinguish them, we may end drawing an obfuscated diagram.

Another work is the incorporation of temporal elements. For example, in the case study about crimes, suppose we have data about the evolution of neighborhoods shapes. For police analysts, it might be interesting to see the map cube results according to these spatial changes.

Finally we are currently working in the incorporation of a Trajectory function to the map cube operator, a spatio-temporal aggregate function. The essential idea is to infer a trajectory from the facts as we explained in Chapter 1.

Part II. Trajectories

Chapter 5: A Conceptual Trajectory Multidimensional Model

5.1 Introduction

Conventional DWs mainly manage alphanumeric data; however, in recent years DWs have been enriched, *e.g.*, with spatial data that can be useful to discover patterns that otherwise would be difficult to recognize [Han 1998], [Bédard 2001], [Jensen 2004], [Bimonte 2005], [Damiani 2006], [Malinowski 2008].

Support for temporal data has also been incorporated in DWs as explained in Chapter 1 (a survey can also be seen in Golfarelli [2009a]). In fact, although DWs include a TIME dimension, this dimension is not oriented to keep track of changes in other dimensions [Malinowski 2008]; therefore, additional temporal support is required.

On the other hand, with the advance of technologies such as sensors and GPS, other types of data are becoming available in huge quantities, *e.g.*, trajectory data about movements of people, animals, vehicles, ships, airplanes. "The concept of trajectory is rooted in the evolving position of some object travelling in some space during a given time interval" [Spaccapietra 2008]. This definition entails the spatio-temporal nature of a trajectory. We believe that the incorporation of this new type of data into a DW can help decision-makers to discover interesting spatio-temporal behaviors. In this chapter, we extend a conceptual spatial multidimensional model by incorporating a trajectory as a first-class concept.

Although there are specialized works related with trajectory DWs [Braz 2007], [Orlando 2007a], [Orlando 2007b], [Marketos 2008]; none of them is devoted to conceptual modelling. They focus on operators for analyzing trajectory data and some of them also address ETL (Extract, Transform, and Load) issues [Braz 2007], [Orlando 2007a], [Marketos 2008].

There are a few proposals [Brakatsoulas 2004], [Spaccapietra 2008] that address conceptual modelling of trajectories but in a *non-multidimensional* context. In [Brakatsoulas 2004], the authors present a specialized non-multidimensional model for a traffic management system, focusing on trajectories, vehicles, and roads. In [Spaccapietra 2008], two non-multidimensional conceptual modelling approaches for trajectories of moving points are proposed. The first one uses a design pattern, *i.e.*, a predefined schema that can be adjusted to meet specific trajectory requirements. The second one uses dedicated trajectory data types equipped with a set of methods to manipulate trajectories. Methods can be added to the data types to meet specific trajectory requirements.

This chapter is organized as follows. In Section 5.2, we present a motivating example. In Section 5.3, we discuss trajectories and their components, and introduce our multidimensional trajectory modelling approaches. Finally, in Section 5.4, we end the chapter and outline future research.

5.2 Motivating example

Consider a taxi company that needs to analyze its daily taxi journeys. Taxis are classified according to fuel type, *e.g.*, gasoline, compressed natural gas (CNG), or E85 (85% bioethanol and 15% petrol). Data about the total number of passengers, the total number of gallons of fuel consumed, and the total fares collected by a taxi during a working day are recorded. A multidimensional model to represent this scenario is shown in Figure 5.1. A sample data of Taxi_journeys fact relationship is shown in Table 5.1.

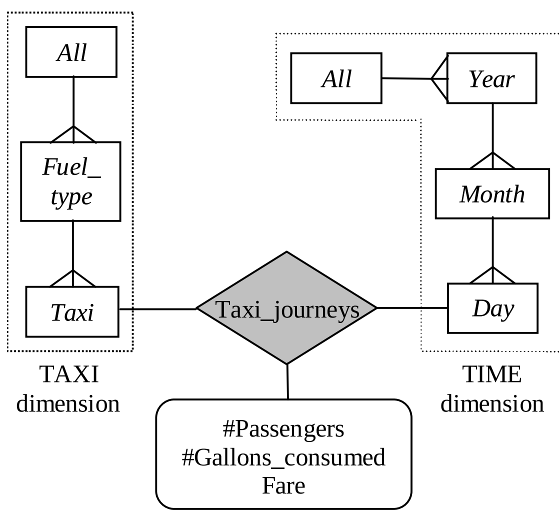


Figure 5.1. A conventional multidimensional model for analyzing taxi journeys.

Table 5.1. Sample data of Taxi_journeys fact relationship.

Bottom levels		Measures		
Taxi	***Day***	**#Passengers**	**#Gallons_consumed**	**Fare ($)**
tx_1	2008-Jan-01	25	12	500
tx_1	2008-Jan-02	20	11	600
tx_2	2008-Jan-01	31	12	450
tx_2	2008-Jan-02	30	13	400

Taxi_journeys fact relationship facilitates data analysis. For example, analysts can formulate queries such as: What is the total number of gallons consumed monthly by fuel type? What are the days of the week where on average more passengers were transported in 2008? What are the top three most profitable taxis in each month? (Where profitability could be computed based on fuel consumption and taxi fares). These queries can be solved using current OLAP tools.

However, suppose that the taxi company also records information about the routes followed by the taxis during a day, *i.e.*, their trajectories. In order to track a taxi's trajectory, a sensor sends several data packages. Each data package contains information about the position of the taxi at a specific minute, along with other information, *e.g.*, weather conditions, the speed and fuel level (if the taxi is moving), the number of gallons of fuel purchased (if the taxi stopped to fill up), the fare (if the taxi completed a ride).

This information enables trajectory data analysis. For example, given a set of taxi trajectories, analysts could formulate the following queries:
i) Find the common points of the taxi trajectories that occurred in the previous month. For that purpose, spatial and temporal thresholds could be considered: two taxi trajectories could have points separated just for one or two blocks and their trajectories could be separated in time for at most two hours. In practice such points could be considered common, see Figure 5.2,
ii) Give a quantitative indicator of similarity [Pelekis 2007] of the taxi trajectories that occurred on business days and that use gasoline, *e.g.*, how similar in shape is a set of trajectories, see Figure 5.3, direction, average speed, or profit (where the trajectories' profits could be calculated based on gallons of gasoline purchased and taxi fares),
iii) Compose a larger trajectory, see Figure 5.4. For example, we could assemble all the trajectories of a taxi during January 2008 and generate a single trajectory for this month. In Figure 5.4, we connect the end of the first trajectory (End_1) with the begin ($Begin_2$) of the second trajectory. We assume that the object moves along a straight line from End_1 to $Begin_2$ at a constant speed, and
iv) Find the number of taxi trajectories that intersect a given region, *e.g.*, the downtown area, during the day. This number is called *presence* [Braz 2007], [Orlando 2007a], see Figure 5.5.

The answers to these questions could help to identify, *e.g.*, profitable routes, points to place speed controls and taxi stations, regions of intense traffic.

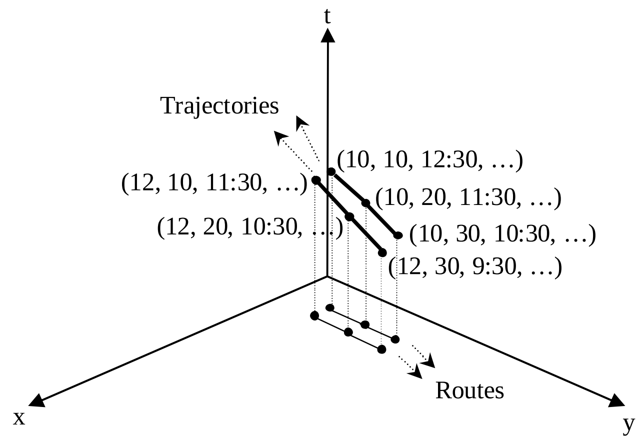


Figure 5.2. Two trajectories considered common within specific temporal and spatial thresholds.

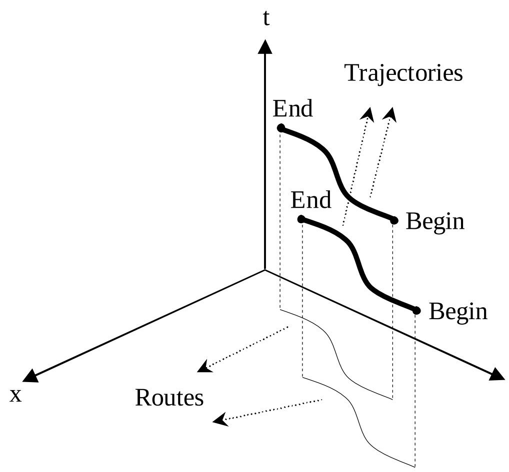


Figure 5.3. Two trajectories similar in shape.

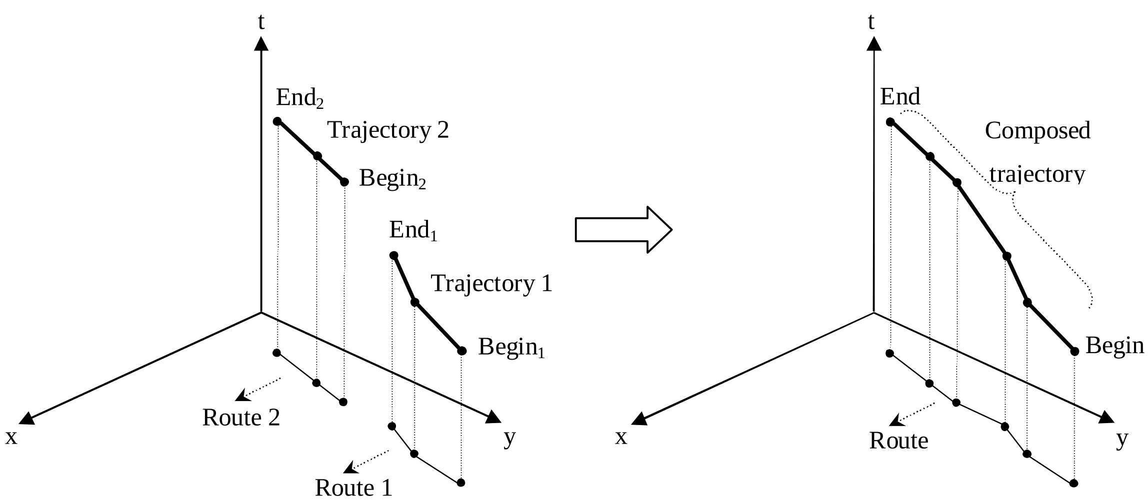


Figure 5.4. Assembling two trajectories. We assume that the object moves along a straight line from End_1 to $Begin_2$ at a constant speed.

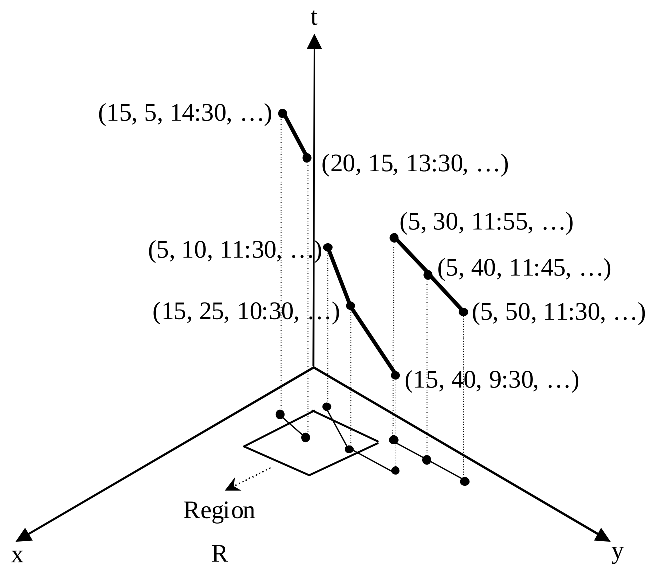


Figure 5.5. Three trajectories, two of them passed through region R during the same day.

5.3 Trajectories

A trajectory is the record of the evolution of the location of an object that is moving in space during a specific interval [t_1, t_n] [Spaccapietra 2008]. This interval can be defined by the user or be application-dependent, *e.g.*, we could consider daily or weekly trajectories for a taxi. The definition of trajectory allows an object to make several trajectories during its lifespan, each with its specific interval. The trajectories of an object are disjoint and are not necessarily consecutive in time.

We represent a trajectory T as a sequence of observations (generated by a sensor), *i.e.*, time-stamped locations that can include complementary semantic data about the trajectory. $T = \langle o_1, o_2, \ldots, o_n \rangle$ where each $o_i = (loc_i, t_i, sem_i)$, *i.e.*, the travelling object is at location loc_i at time t_i ($t_i < t_{i+1}$) and semantic data sem_i can be associated with each observation. For example, consider a taxi trajectory, in addition to the location and time of each of its observations, we could include semantic data such as temperature, speed, and fuel level.

Note that for a moving region the projection on the plane of its trajectory locations gives us its *traversed area* [Güting 2005]. For a moving point the projection on the plane of its trajectory locations gives us its *route* [Vazirgiannis 2001], [Frentzos 2005]. For simplicity, we restrict the discussion hereafter on moving points. Unless more information becomes available, the object is assumed to move along a straight line from location (x_i, y_i) to location (x_{i+1}, y_{i+1}) [Güting 2005]. Figure 5.6 shows the trajectory of a moving point with four observations and its corresponding route.

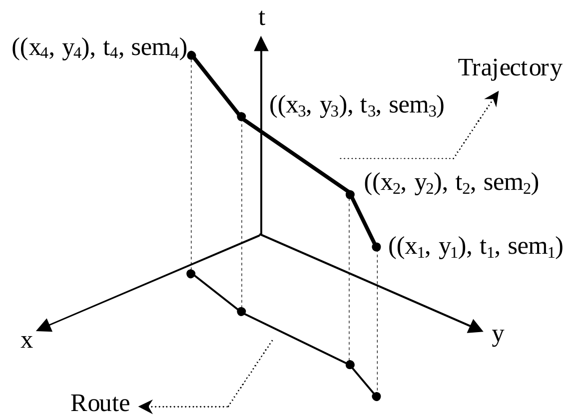


Figure 5.6. Trajectory of a moving point.

Note that we attach semantic information to trajectories, which is of fundamental importance for their analysis [Alvares 2007], [Guc 2008]. However, not necessarily the same type of semantic data is included in all the observations. For example, consider again a taxi trajectory: when the taxi stops to fill up, we could collect data about the number of gallons of fuel purchased; when the taxi stops to pick up passengers, we could collect data about the fare; when the taxi is moving, we could collect data about its speed and fuel level; see Figure 5.7. Therefore, depending on the requirements of a particular application, trajectory observations can be classified into types. In the previous example, we could define three types of observation: fill-ups, pick-ups, and moves. There could be some semantic data common to all or some of the types of observation defined. For example, data about weather conditions could be included in the three types of observation previously defined, as illustrated in Figure 5.7 (temperature).

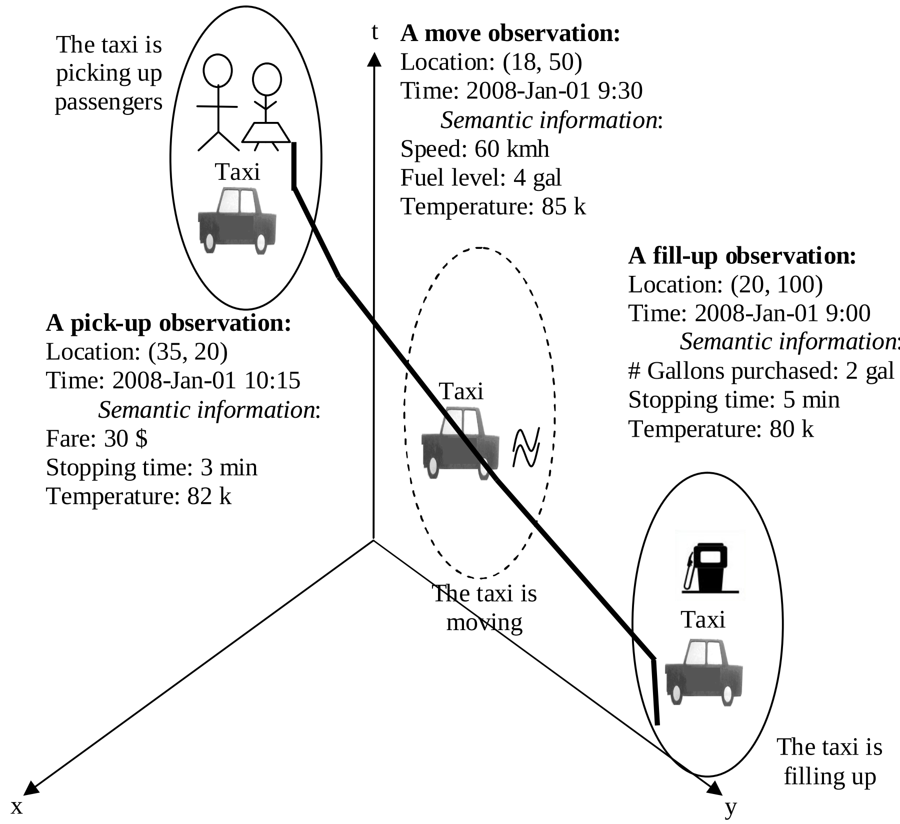


Figure 5.7. Three types of observation for a taxi trajectory.

To represent a trajectory in our multidimensional model, we propose the icons of Figure 5.8. Figure 5.8 (a) represents the trajectory of a moving generic geometry *Geo*. A *Geo* can be replaced by a simple or a complex geometry (spatial data types), see Figure 5.9. For example, Figure 5.8 (b) represents the trajectory of a moving point (*e.g.*, a taxi), Figure 5.8 (c) the trajectory of a moving line (*e.g.*, a train), Figure 5.8 (d) the trajectory of a moving region (*e.g.*, a hurricane, an oil spill), and Figure 5.8 (e) the trajectory of a moving group of regions (*e.g.*, a group of clouds).

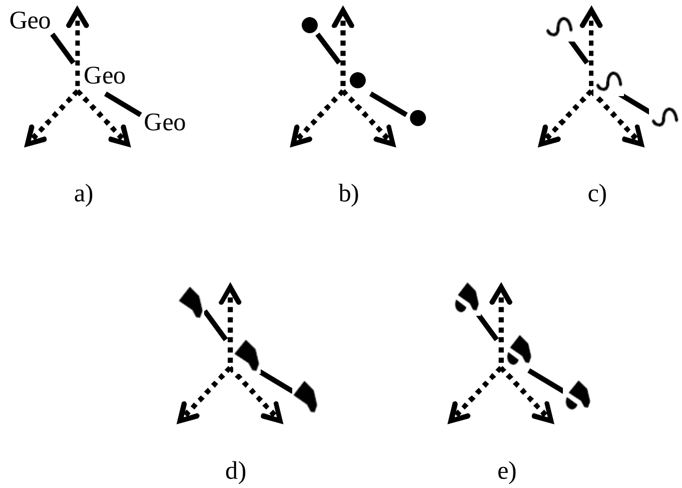


Figure 5.8. Notations for a trajectory of a moving: a) generic geometry, b) point, c) line, d) region, and e) group of regions.

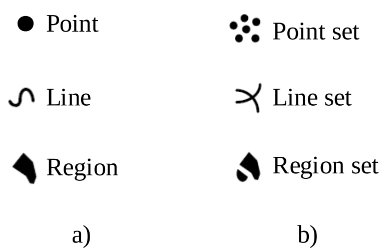


Figure 5.9. Notations for: a) simple geometries and b) complex geometries.
Source: [Parent 1999], [Malinowski 2008].

In order to specify types of observation and their corresponding semantic fields, we propose the notation shown at Figure 5.10. Note that each observation type implicitly includes the object's location (in accordance with the geometry associated with the trajectory) and its corresponding timestamp. For example, consider the icon of Figure 5.8 (b); an instance of an observation type of this trajectory is represented as $((x, y), t$, semantic fields). Now consider Figure 5.8 (c); an instance of an observation type of this trajectory is represented as $((p_1, p_2), t$, semantic fields), where p_1 and p_2 are points that define, *e.g.*, a straight line.

In the following section, we incorporate a trajectory into a multidimensional model. To facilitate this task, we propose two modelling approaches: composed multivalued timestamped measures and composition of facts.

Trayectory_name**{**
Observation_type$_1$(semantic fields),
Observation_type$_2$(semantic fields),
…
Observation_type$_n$(semantic fields)
}

a)

Taxi_trajectory**{**
Fill_up(#Gallons_purchased),
Move(Speed, Fuel_level),
Pick_up(Fare)
}

b)

< ((x_1, y_1), t_1, 3 gal), → Fill-ups
((x_2, y_2), t_2, 60 kmh, 6 gal), → Moves
((x_3, y_3), t_3, 75 kmh, 5 gal), → Moves
((x_4, y_4), t_4, 2 gal), → Fill-ups
((x_5, y_5), t_5, 25$), → Pick-up
...
>

c)

Figure 5.10. Representation of types of observation: a) a trajectory of a moving point with n types of observation, b) a taxi trajectory with three types of observation, and c) instances of types of observation of b).

5.3.1 Composed multivalued timestamped measures

Continuing with the example of taxi trajectories, we classify taxi observations into three types: fill-ups, pick-ups, and moves. The following semantic data are associated with them: stopping time and number of gallons of fuel purchased with fill-ups, stopping time and fare with pick-ups, and fuel level and speed with moves. Note that we consider observations as sensor snapshots. In this example, we assume a minute as the temporal granularity of an observation.

We define one fact relationship, Taxi_journeys, see Figure 5.11. Observations are represented by three composed multivalued timestamped measures: Fill_up, Pick_up, and Move; they are described in Table 5.2. Table 5.3 shows a sample data of Taxi_journeys fact relationship.

Although this solution is natural and compact, it has some drawbacks: i) aggregate functions must deal with multivalued measures, which could prevent their use in current OLAP systems, ii) handling of the relationship between the observations' timestamps and the TIME dimension is required in order to enable time hierarchy navigation, because these implicit timestamps are not connected to a time level, *e.g.*, *Minute* (dimension levels are connected to fact relationships, but not to measures), and iii) time consistency checkings are required, *e.g.*, the observations' timestamps must "rollup" to the same day associated with their taxi journey, and the timestamp of an observation cannot intersect the interval made up by the timestamp of any fill-up (or pick-up) observation plus its stopping time. In order to overcome some of these difficulties, we propose an alternative modelling approach in the following section.

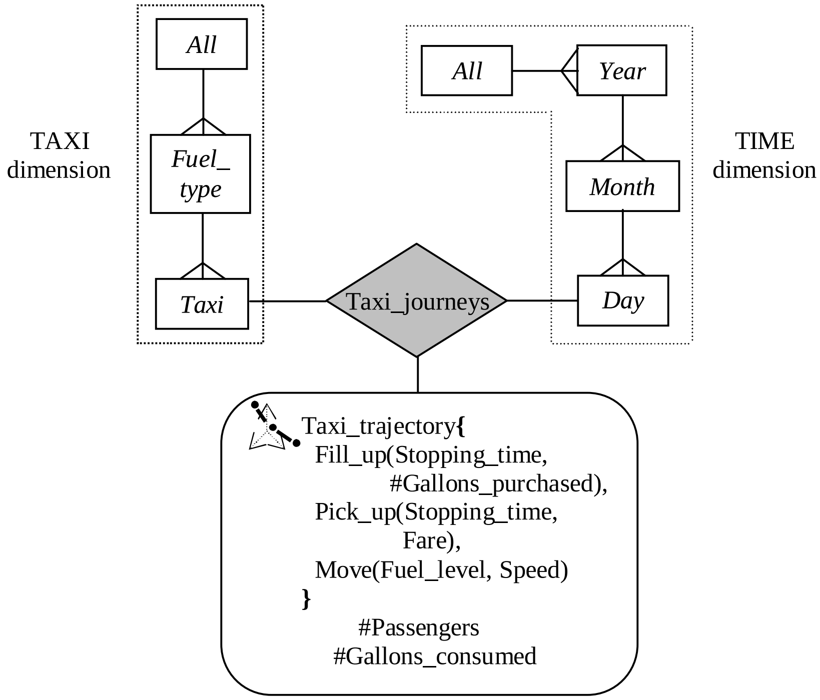
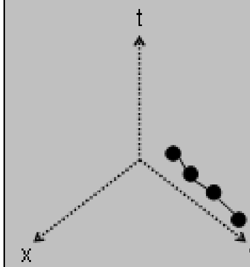
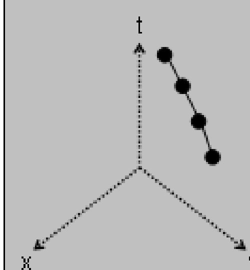


Figure 5.11. A multidimensional model for analyzing taxi trajectories using composed multivalued timestamped measures.

Table 5.2. Measures of our multidimensional model of taxi trajectories.

Measure	**Description**	**Associated observation type**	**Data type**
Taxi_trajectory	Represents the taxi's trajectory.	-	Spatio-temporal
Timestamp	Represents the time of an observation	Implicit in all the observations	Temporal
Location	Represents the spatial position of the taxi.	Implicit in all the observations	Spatial
Stopping_time	Registers how much time the stop lasted.	Fill_up, Pick_up	Numeric
#Gallons_purchased	Records the number of gallons of fuel purchased.	Fill_up	Numeric
Fare	Represents the money paid for a taxi ride.	Pick_up	Numeric
Fuel_level	Represents the current fuel level in the taxi's fuel tank.	Move	Numeric
Speed	Represents the current speed of the taxi.	Move	Numeric
#Passengers	Records the number of passengers transported.	-	Numeric
#Gallons_consumed	Records the number of gallons of fuel consumed.	-	Numeric

Table 5.3. Sample data of Taxi_journeys fact relationship.

Bottom levels		Measures		
Taxi	***Day***	**Taxi_trajectory**	**#Passen-gers**	**#Gallons_ consumed**
tx_1	2008-Jan-01	 { A move ← ((10, 95), 2008-Jan-01 7:10, 2 gal, 50 kmh), A fill-up ← ((10, 80), 2008-Jan-01 7:20, 8 min, 3 gal), A pick-up ← ((12, 70), 2008-Jan-01 8:30, 2 min, 30 $), …}	25	12
tx_1	2008-Jan-02	 { A move ← ((30, 75), 2008-Jan-02 7:20, 2 gal, 80 kmh), A move ← ((25, 65), 2008-Jan-02 8:30, 1 gal, 40 kmh), A fill-up ← ((20, 50), 2008-Jan-02 8:50, 10 min, 4 gal), …}	20	11

5.3.2 Composition of facts

We define four fact relationships: Taxi_journeys, Fill_ups, Pick_ups, and Moves; see Figure 5.12. In this approach, the Taxi_trajectory measure is derived from the fact relationships Fill_ups, Pick_ups, and Moves, that represent the trajectory observations. A derived measure is generated from other measures and is shown by preceding its name with a slash (/).

Each taxi journey includes a set of observations; to represent such a composition, we propose a dotted relationship, see Figure 5.12. A composition such as this implies that if a taxi makes a taxi journey on a day (*e.g.*, 2008-Jan-01), there must be a non-empty set of observations associated with this journey. In addition, the minute values of those observations must rollup to the same day (2008-Jan-01).

This approach, unlike the previous one, does not require the handling of multivalued measures, and the observations' timestamps are explicitly connected to a time level, enabling time hierarchy navigation. However, this solution also has some drawbacks: i) an operation that relates fact relationships is required in order to combine a taxi journey with its observations, *i.e.*, a type of drill-across operation [Golfarelli 1998], and ii) handling of several fact relationships can become complex, *e.g.*, to the formulation of queries. Because a fact relationship is created for each observation type, if the number of types of observation is high, we would have to deal with a proliferation of fact relationships. In Table 5.4, we compare our trajectory modelling approaches.

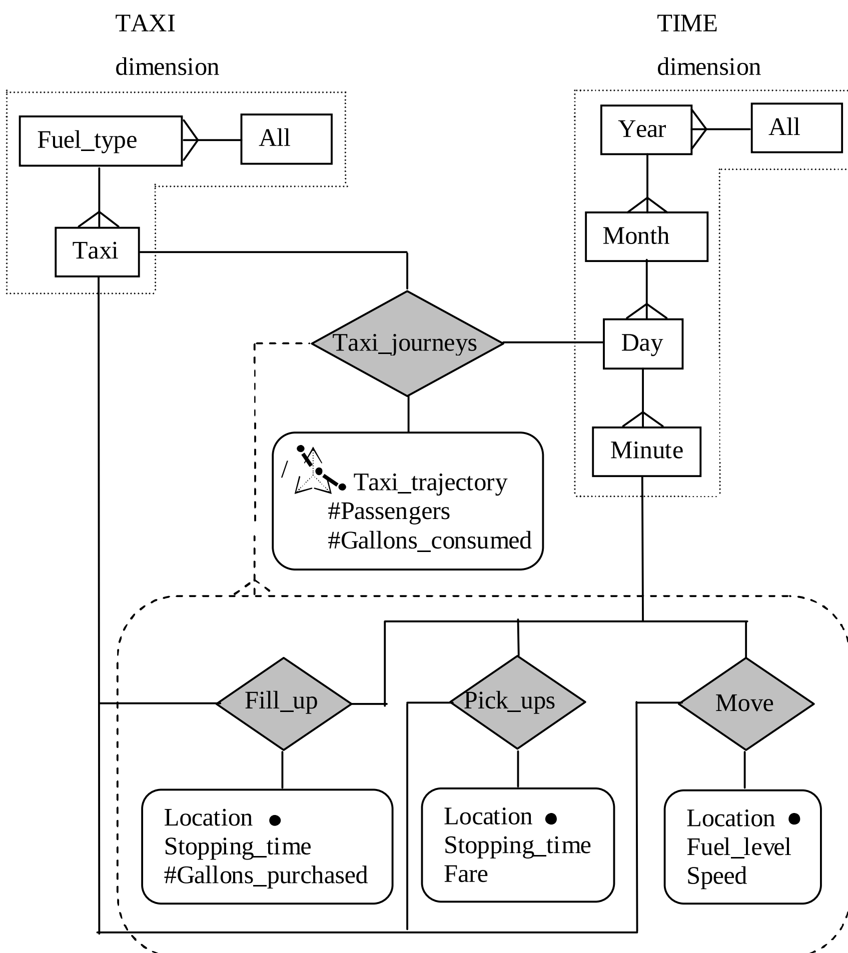


Figure 5.12. A multidimensional model for analyzing taxi trajectories using composition of facts.

Table 5.4. Comparison of our trajectory modelling approaches.

Trajectory	**Composed multivalued timestamped measures**	**Composition of facts**
Representation	Trajectories are explicitly represented and play the role of a measure.	Trajectories are explicitly represented and play the role of a derived measure. A type of drill-across operation is required to combine the observations of a trajectory.
Aggregation	Operators for trajectory aggregation could be used. Parts of the trajectory (observations) can also be aggregated. Aggregate functions must deal with multivalued measures.	Operators for trajectory aggregation could be used. Parts of the trajectory (observations) can also be aggregated.

Observations	Types of observation are represented in a compact and natural way in just one fact relationship. The observations' timestamps are not connected to a time level, implying additional time consistency checkings and navigational capabilities.	A fact relationship is created for each observation type. If the number of types of observation is high it results in a proliferation of fact relationships. The observations' timestamps are connected to a time level, enabling time hierarchy navigation.

5.3.3 Granularity and aggregation of measures

It is a design decision to determine the level of detail of a measure, *e.g.*, we could represent #Gallons_purchased as a Taxi_journeys measure instead of a Fill_ups measure, or we could represent #Passengers as a Pick_ups measure instead of a Taxi_journeys measure.

Note that Fill_ups, Pick_ups, and Moves measures have a lower time granularity (minute) with regard to the time granularity (day) of Taxi_journeys measures. However, as usual in a multidimensional model, we can aggregate Fill_ups, Pick_ups, and Moves measures in order to generate aggregates at a coarser granularity, *e.g.*, we can find the total money collected by a taxi on a day by adding all its corresponding taxi fares. In particular, the aggregation of Location, which plays the role of a spatial measure, must be performed using a spatial aggregate function, such as geometric union, center of mass, convex hull, and others. For example, suppose we select the locations of taxis where they stopped to fill up, the center of mass of these locations could suggest a place to set up a gas station. As usual in DWs, some of these aggregates could be precalculated in order to speed up time response of queries.

On the other hand, the aggregation of the Taxi_trajectory measure leads to interesting questions, such as those described in Section 5.2. For example, suppose we want to compose the trajectories of each taxi. Next, we give the reader an idea of how this query could be formulated in an SQL-like way:

```
SELECT Taxi, Compose_trajectory(Taxi_trajectory) AS Comp_traj
FROM Taxi_journeys
GROUP BY Taxi;
```

Where Compose_trajectory() is an aggregate function to assemble trajectories as illustrated in Figure 5.4. Obviously, such a function must be formally defined.

5.4 Conclusions and future work

We proposed a notation to represent trajectories as a first-class concept in a conceptual spatial multidimensional model. We stressed the semantic nature of a trajectory by classifying its observations in accordance with their semantic data. Two modelling approaches were presented. The first one is based on composed multivalued measures. The second one is based on composition of facts relationships.

A preliminary judgement suggests that the first approach could be more suitable than the second one when the number of types of observation is high. However, other criteria, such as handling of aggregation, implementation issues, storage, among others, must be considered in order to evaluate both approaches.

As future work, we plan to transform our conceptual model into a logical one. From a physical point-of-view, a related issue is how to store and efficiently retrieve a trajectory in a multidimensional context. Data structures and indexing schemes must be designed for this purpose. We also plan to develop a query language in order to express analytical trajectory queries, such as the ones of Section 5.2. Operators related to trajectory aggregation should also be addressed. The works of [Braz 2007], [Orlando 2007a], [Orlando 2007b], [Marketos 2008] are points of departure for these issues.

As we explained in Chapter 1, the notion of *season* arises in the context of trajectories. Informally, a season is an interval during which a moving object is associated with another object. For example, in the case of taxi trajectories, we can consider seasons of a taxi in a given region. In the following chapter, we specialized the notion of season, where we focus specifically on seasons that arise in the context of *reclassification trajectories*.

Part III. Seasons

Chapter 6: Season Queries on a Temporal Multidimensional Model

6.1 Introduction

In Chapter 2 we presented a formal multigranular temporal multidimensional model that keeps track of reclassifications. Based on this model, we introduce in this chapter the notion of *season of reclassification* (hereinafter, simply called *season*), *i.e.*, an interval during which two members of a dimension are associated with each other, *e.g.*, a salesperson sp_1 is associated with a store st_1 during the interval [day 1, day 45], a product pd_1 is associated with a category cat_1 during the interval [day 1, day 180]. Note that throughout his lifespan a salesperson can experience several (disjoint) seasons in the same store as well as a product can return to a previous category several times.

If we consider the evolution (a series of assignments) of a salesperson through the stores and the history of the reclassifications of a product through the different categories, we could establish a link with the notion of trajectory, see Section 5.3: "The term trajectory is sometimes used in a metaphorical sense to describe an evolution, although the evolution at hand is not related to physical movement" [Spaccapietra 2008]. This metaphorical use relies on the idea of an object moving in an abstract space whose points are the different values for which the object passes. In this sense, we could consider the trajectory of a product through the categories as a *metaphorical trajectory*. On the other hand, the trajectory of a salesperson through the stores has a more geographical connotation. In either case, we can speak of a *reclassification trajectory*, *i.e.*, the evolution of a member of a dimension with regard to its reclassifications, where each of its reclassifications holds for an interval, *i.e.*, for a season, see Figure 6.1.

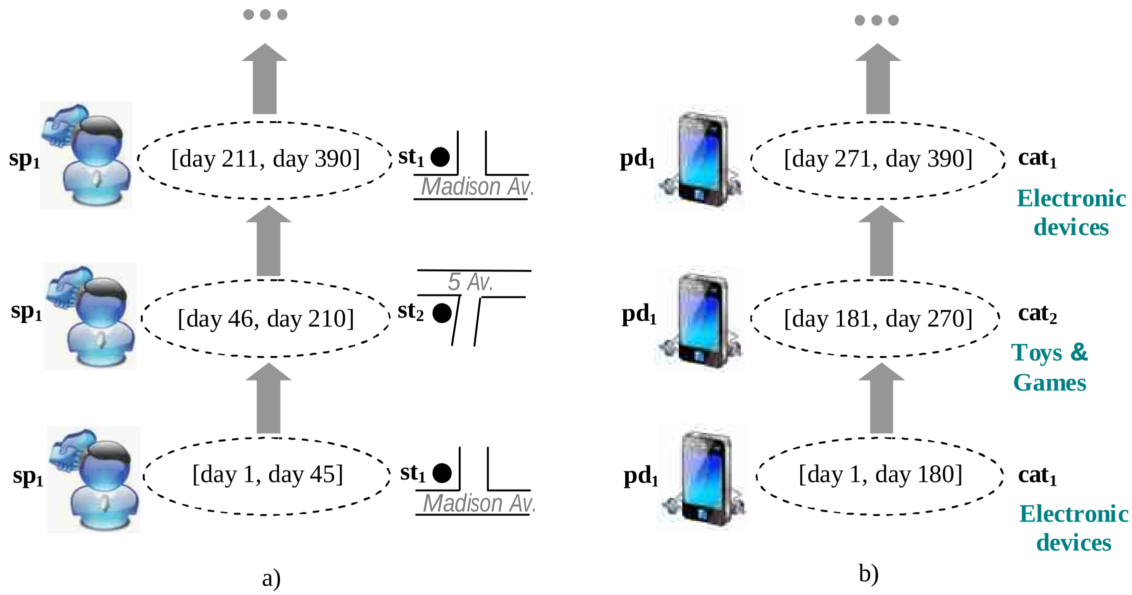


Figure 6.1. Reclassification trajectories of: a) a salesperson sp_1 and b) a product pd_1.

We believe that the notion of season can lead to interesting queries (called *season queries* in our work), that can be useful for decision-makers in several situations and application domains. Consider, *e.g.*, the following queries referring to associations between salespersons and stores: What was the total number of units sold by salesperson sp_1 in his first season in store st_1? What was the season and store when the total number of units sold by salesperson sp_1 was the highest? What was the total number of units sold by each salesperson in each season in each store?

The previous queries can help to adjust rotation policies of staff. In this same scenario, analogous queries can help to identify periods for recategorizing products and reclassifying customers. Season queries can also be useful in other scenarios. For example, in a soccer competition scenario they can help to assess the players' performance in face of the dynamics of their transfers; this type of analysis can help to understand the economics of soccer, a field that is still in its infancy [Torgler 2007]. In environmental sciences analogous season queries can be useful to evaluate the impact of recurrent phenomena, such as hurricanes over a region.

To the best of our knowledge, there is no language or operator that allows one to formulate this type of query in a concise and simple way. In fact, the notion of season is not present in any work we have found in the literature. Although, in TOLAP (a temporal multidimensional query language) [Mendelzon 2000] it is possible to formulate queries such as: What was the total number of units sold by salesperson sp_1 when *he has worked* in store st_1?; TOLAP is not oriented to formulate season queries.

In order to deal with season queries, we start from our formal multigranular temporal multidimensional model, see Chapter 2. Then, we derive the formal notion of season and an operator to express season queries. Our operator receives a cube, *i.e.*, a multidimensional collection of data [Jarke 2003], and returns a new cube, thus facilitating its integration into a multidimensional query language, and enabling the composition of queries and integration of their results.

The rest of the chapter is organized as follows. In Section 6.2, we introduce and formalize the notion of season around the model of Chapter 2. In Section 6.3, we propose and exemplify an operator for season queries. Finally, in Section 6.4, we present conclusions and future work.

6.2 Seasons

Informally, a *season* is an interval during which a member of a level is associated with a member of a higher level, *e.g.*, a season of a salesperson in a store, a season of a store in a status, a season of a product in a category. Although we use the word "season", there are other more precise words referring to periods during which some kind of association applies in some domains, *e.g.*, a "term" of a president in a department, a "shift" of a worker at a machine, a "spell" of an atmospheric phenomenon in a region, among others.

Next, we give a formal definition of a season of association between two members of consecutive levels in a dimension schema. Let l_1, l_2 be levels, $a \in dom(l_1)$, $b \in dom(l_2)$, and the pair $(l_1, l_2) \in \preccurlyeq'$ is temporal with TRG μ. A season of a in b is an interval S with temporal granularity μ', where:

i) $\mu' \rightsquigarrow \mu$,

ii) $\forall t \in S$, RUP_l_1_l_2(a, t) = b,

iii) if RUP_l_1_l_2(a, Start(S) − 1) is defined then RUP_l_1_l_2(a, Start(S) − 1) ≠ b, and

iv) if RUP_l_1_l_2(a, End(S) + 1) is defined then RUP_l_1_l_2(a, End(S) + 1) ≠ b.

Note that conditions iii) and iv) guarantee that S is a *maximum* interval during which a is associated with b. A general definition of a season of association between a member a of a level l_1 and a member b of a higher level l_j of l_1 is given next. Let $l_1, l_2, l_3, \ldots, l_j$ be levels of a dimension schema, $j > 1$, where $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_j$. Let $U \neq \varnothing$ be the set of TRGs along the path $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_j$,

$a \in dom(l_1)$, and $b \in dom(l_j)$. A season of a in b is an interval S with temporal granularity μ', where:

i) $\forall \mu \in U$, $\mu' \rightsquigarrow \mu$,

ii) $\forall t \in S$, RUP_ l_1_$l_j(a, t) = b$,

iii) if RUP_l_1_$l_j(a$, Start(S) − 1) is defined then RUP_l_1_$l_j(a$, Start(S) − 1) $\neq b$, and

iv) if RUP_l_1_$l_j(a$, End(S) + 1) is defined then RUP_l_1_$l_j(a$, End(S) + 1) $\neq b$.

If $U = \varnothing$ then during its lifespan a is always associated with b; as a consequence, we consider S as the unique season of a in b, where Start(S) and End(S) correspond to the lifespan of a.

Example 6.1. Consider again the Example 2.5. The rollup values (stores) for a salesperson sp_1 are shown in Table 6.1, and the rollup values (status) for stores st_1 and st_2 are shown in Table 6.2. For the sake of simplicity, we assume months of 30 days.

Table 6.1. Rollup values (stores) for salesperson sp_1.

Salesperson	sp_1	sp_1	sp_1	sp_1	sp_1
Day	1 - 45	46 - 210	211 - 390	391 - 480	481 - 540
Store	st_1	st_2	st_1	No_store	st_2

Table 6.2. Rollup values (status) for stores st_1 and st_2.

Store	st_1	st_1	st_1	st_2	st_2	st_2
Semester	1	2	3	1	2	3
Status	A	A	B	B	A	A

From Tables 6.1 and 6.2 we can see that [day 1, day 45] and [day 211, day 390] are seasons of sp_1 in st_1, [day 46, day 210] and [day 481, day 540] are seasons of sp_1 in st_2. [semester 1, semester 2] is a season of st_1 in status A, [semester 3, semester 3] is a season of st_1 in status B, [semester 1, semester 1] is a season of st_2 in status B, and [semester 2, semester 3] is a season of st_2 in status A. Consequently, [day 1, day 45], [day 181, day 360], and [day 481, 540] are seasons of sp_1 in status A, and [day 46, day 180] and [day 361, day 390] are seasons of sp_1 in division B, see Figure 6.2.

Next, we consider the ordering of the seasons of association between two members of a dimension and define the notion of the n^{th} season. Let S be a season of a in b. S is the *first* season of a in b if it does not exist a season S' of a in b such that End(S') < Start(S). S is the *second* season of a in b, if there exists just one season S' of a in b such that End(S') < Start(S). In general, let $\mathbf{S} = \{S_1, S_2, \ldots,$

$S_n\}$ be a set of seasons of a in b; then, $S_i \in \mathbf{S}$ is the n^{th} season of a in b, where $n = |\{S_j \in \mathbf{S} \mid \text{End}(S_j) < \text{Start}(S_i)\}| + 1$. We refer to this number as the *season number*.

Example 6.2. Consider again Figure 6.2. From Example 6.1, [day 1, day 45] is the first season of sp$_1$ in st$_1$, and [day 211, day 390] is the second season. [day 46, day 210] is the first season of sp$_1$ in st$_2$, and [day 481, day 540] is the second season. In addition, [day 1, day 45] is the first season of sp$_1$ in status A, [day 181, day 360] is the second season, and [day 481, 540] is the third season. [day 46, day 180] is the first season of sp$_1$ in status B, and [day 361, day 390] is the second season.

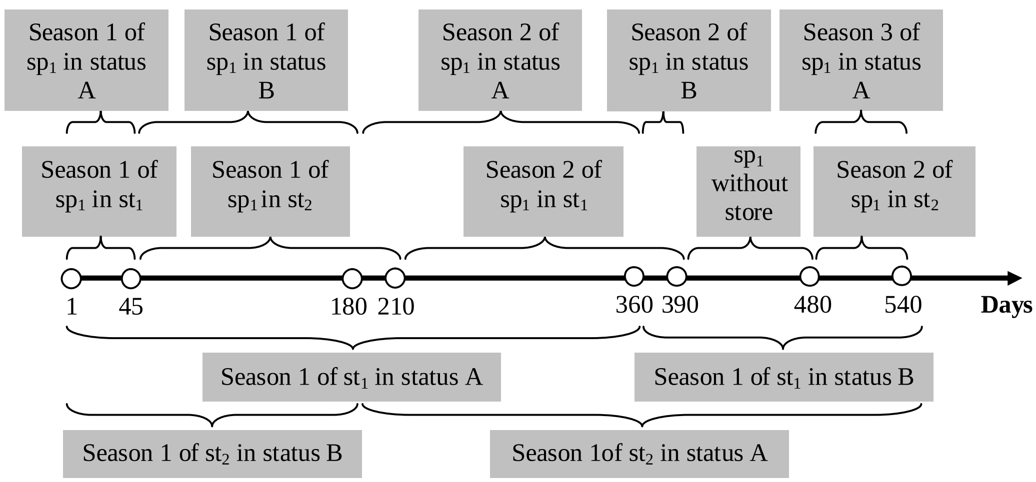


Figure 6.2. Examples of seasons.

6.3 A new operator for season queries

Consider the Example 6.1 and the rollup function extended with valid time. The rollup function RUP_*Salesperson_Store*(sp$_1$, day 7), *e.g.*, returns the store in which sp$_1$ was on day 7, *i.e.*, st$_1$. However, this function does not return the corresponding season of sp$_1$ in st$_1$ or the season number. To accomplish these and other tasks, we define a season query operator. First, we consider some guidelines in order to design our operator.

Our operator must form or be part of a closed language. The closure property requires that the results of an operator are again elements of the data model [Haase 2004]; thus enabling the combination among query results. In our case, these elements are cubes, *i.e.*, fact tables (hereafter, we use these terms interchangeably). Hence, our operator must be designed so that it can be embedded into a multidimensional query language, either theoretical [Cabibbo 1997], [Vassiliadis

1998], [Datta 1999], [Mendelzon 2000], [Pedersen 2001a], or practical, such as MDX [Whitehorn 2005].

Our operator must capture the interesting phenomena, in our case seasons, and must be based on genuine applications and users' requirements about seasons, see Section 6.1 and Table 6.3. Our proposal also attempts to be minimalist, *i.e.*, to minimize the extensions required in a multidimensional query language in order to support our season queries.

Table 6.3. User requests and query language requirements.

User request	Query language requirements
What was the total number of units sold by salesperson sp_1 in all his seasons in store st_1, *i.e.*, when he has worked in st_1?	i) Conventional aggregate functions, *e.g.*, SUM, AVG. ii) Temporal rollup**:** find the corresponding store value associated with a salesperson in a specific time [Mendelzon 2000]. iii) Restriction and projection of facts with aggregated measures.
What was the total number of units sold by salesperson sp_1 in his first season in store st_1?	i) The same as the previous request. ii) Season number**:** find the corresponding season number of a salesperson in a store in a specific time.
What was the total number of units sold by each salesperson in each season in each store?	i) The same as the second request. ii) Intervals of the corresponding seasons. iii) Grouping of facts.
What was the season (including season number) and store when the total number of units sold by salesperson sp_1 was the highest?	i) The same as the previous requests.

Table 6.3 summarizes the query language requirements (right column) for some season queries (left column). In order to meet these user requests and consequent language requirements, we define the operator Seasons_l_1_l_j(*ft*) = *ft'* that receives a fact table *ft* and returns a new one *ft'*. Informally, Seasons_l_1_l_j groups facts based on seasons: for each season of a member of l_1 in a member of l_j, the measures of the corresponding facts are aggregated. Each aggregate function, *e.g.*, SUM, AVG, is applied to each measure. An example is shown in Figure 6.3. Although this may become a very demanding task, in a concrete query only aggregations requested could be materialized (see examples in Subsection 6.4.3), *i.e.*, our operator is situated on a declarative level, it does not address optimization issues.

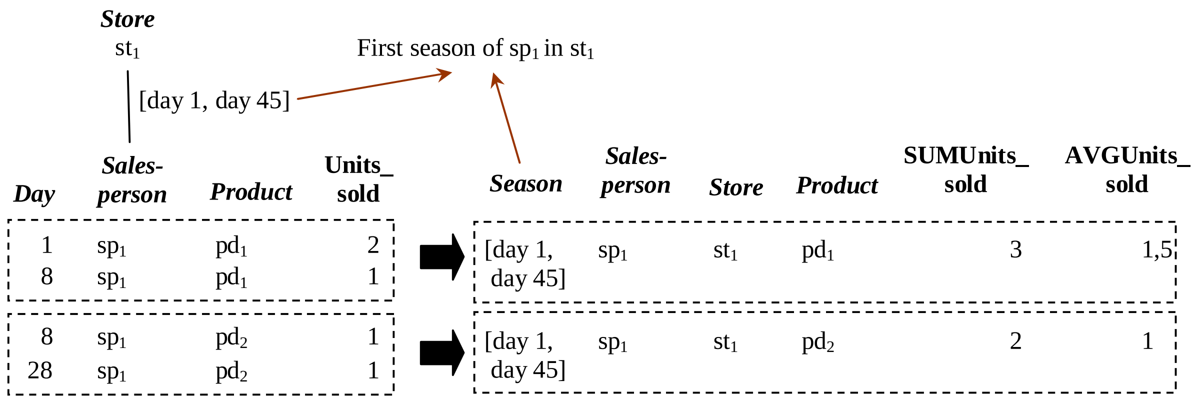


Figure 6.3. Grouping the facts of the first season of salesperson sp_1 in store st_1.

6.3.1 Seasons_l_1_l_j operator: resulting fact schema

We assume a fact schema (F, L_F, M) where $L_F = \{\mu_f, l_1, fl_1, fl_2, ..., fl_k\}$ is a set of bottom levels, μ_f is the bottom level of the TIME dimension, and M = $\{m_1, …, m_m\}$ is a set of measures. $ft = \{fi_1, fi_2, ..., fi_l\}$ is a fact table where each fact instance fi_i = ({member(μ_f), member(l_1), member(fl_1), ..., member(fl_k)}, {value(m_1), ..., value(m_m)}).

Let $l_1, l_2, l_3, …, l_j$ be levels of a dimension schema Y, $j > 1$, where $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 … \preccurlyeq' l_j$, $l_1 \in L_F$. Y is the dimension on which Seasons_l_1_l_j operartor will be applied. Seasons_l_1_l_j is defined as Seasons_l_1_l_j(ft) = ft', where ft' is a fact table. The resulting fact schema is (F', $L_{F'}$, M') where $L_{F'} = L_F - \{\mu_f\} \cup \{Season, l_j\}$. That is:

i) we preserve all the bottom levels of L_F except the bottom level of the TIME dimension,

ii) *Season* is a bottom level of a homonymous dimension schema. The SEASON dimension has two levels: *Season* and *All*, where *Season* $\preccurlyeq$ *All*. The *Season* level includes three attributes: SeasonNumber, SeasonStart, and SeasonEnd; they are explained in Table 6.4,

iii) l_j is a bottom level of a dimension schema $SUBY_1$. $SUBY_1$ has a set Z of levels where Z is the set of levels in Y such that $\forall z \in Z$, $l_j \preccurlyeq z$, see Figure 6.4, and

iv) l_1 in $L_{F'}$ is a bottom level of a dimension schema $SUBY_2$. $SUBY_2$ preserves all the hierarchies in Y where l_1 is a bottom level, except the hierarchy where $l_1 \preccurlyeq l_j$. If there are no other hierarchies where l_1 is a bottom level, a level *All* is generated, where $l_1 \preccurlyeq$ *All*, see Figure 6.4.

M' is a set of aggregated measures. Let G = {g_1, g_2, …, g_p} be a set of aggregate functions. Our season operator applies each aggregate function $g_i \in G$ to each measure $m_j \in M$. A level name g_im_j for each aggregated measure is generated, *i.e.*, M' = {g_1m_1, ..., g_1m_m, ..., g_pm_1, ..., g_pm_m}. For example, if g_i = SUM and m_j = Units_sold, then g_im_j = SUMUnits_sold. Figure 6.4 outlines the original and the resulting schema generated by Seasons_l_1_l_j.

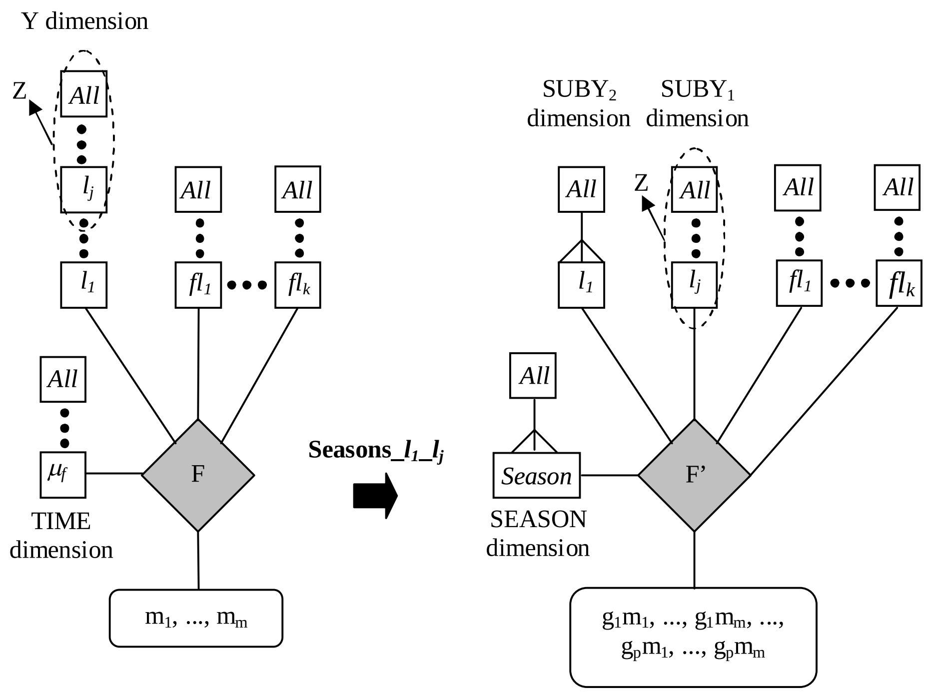


Figure 6.4. Original (left) and resulting schema (right) generated by Seasons_l_1_l_j.

6.3.2 Seasons_l_1_l_j operator: resulting fact table

Next, in Table 6.4 we outline an algorithm to generate the resulting fact table *ft'*. We describe how *ft* is transformed into *ft'*; however, we emphasize that in an actual implementation *ft* should remain intact.

Table 6.4. Seasons_l_1_l_j operator algorithm.

Seasons_l_1_l_j(*ft*)

Input: Fact table *ft*
Output: Fact table *ft'*
Procedure:

Step 1. Compute the seasons for each member of l_1 with regard to the members of l_j. The results make up a SEASON dimension schema instance. Let *a* be a member of l_1, *b* a member of l_j, and *U* the set of TRGs along the path $l_1 \preccurlyeq' l_2 \preccurlyeq' l_3 \ldots \preccurlyeq' l_j$. A member of *Season* level includes the following attributes: SeasonNumber is the season number of *a* in *b*, *dom*(SeasonNumber) = $\mathbb{N}$. If $U = \emptyset$ then SeasonNumber = 1. SeasonStart and SeasonEnd define the season of *a* in *b*, *dom*(SeasonStart) = *dom*(SeasonEnd) = *dom*(μ_f). If $U = \emptyset$ then SeasonStart = Start(LS) and SeasonEnd = End(LS), where LS is the lifespan interval of *a*.

Step 2. Insert the SEASON dimension schema instance, generated in Step 1, into the fact table *ft*. Each fact instance $fi \in ft$ is associated with a member of *Season* level as follows. Let *W* be the set of members of *Season* level corresponding to a member of $fi.l_1$. The member $w \in W$ associated with *fi* is $\{w \mid w \in W$ AND w.SeasonStart <= $fi.\mu_f$ <= w.SeasonEnd$\}$.

Step 3. Generate the SUBY_1 dimension schema instance. SUBY_1 is copied (set Z of levels) from the Y dimension schema instance as illustrated in Figure 6.4.

Step 4. Insert the SUBY_1 dimension schema instance, generated in Step 3, into the fact table *ft*. Each fact instance $fi \in ft$ is associated with *r*, a member of l_j, such that RUP_l_1_l_j($fi.l_1$, $fi.\mu_f$) = *r*.

Step 5. Generate the SUBY_2 dimension schema instance as explained in Subsection 6.4.1.

Step 6. Remove the TIME dimension from *ft* and aggregate the measures for the rest of dimensions: $\alpha_{[\text{AL, GDL}]}(ft)$, where AL = $g_1(m_1)$, …, $g_p(m_1)$, …, $g_1(m_m)$, …, $g_p(m_m)$ and GDL = $L_{F'}$. The aggregation operator α [Datta 1999] (see Subsection 3.5.1) receives as parameters: i) a cube (*ft*), ii) a list of elements $g_i(m_j)$ where g_i is an aggregate function $\in$ G and m_j a measure $\in$ M, and iii) a set of grouping dimension levels ($L_{F'}$). The output is a fact table *ft'*.

Example 6.3. Consider a sample data of the fact table *Sales* shown in Table 6.5. The resulting fact schema of the Seasons_*Salesperson*_*Store*(*Sales*) operation is shown in Figure 6.5 and the resulting fact table in Table 6.6. We assume the seasons of Example 6.1 and G = {SUM, AVG}. The operation is outlined in Figure 6.6. We assume only two products in order to simplify the drawing of the four dimensions.

Note that in the resulting fact table, *Store* becomes a bottom level and the SEASON dimension plays the role of a TIME dimension. For example, the first fact instance in Table 6.6 shows that sp_1 in his first season in st_1, which took place between day 1 and day 45, sold three units of product pd_1. Note that the first two fact instances in Table 6.5 contribute to the generation of the first fact instance in Table 6.6. In Table 6.5, rows that are included in the season of a salesperson are shown with the same colour, the corresponding colour is used in Table 6.6.

As Table 6.6 shows, our operator provides a simple mechanism to find and calculate data specifically focused on seasons. In general, the farther l_1 and l_j are in the dimension hierarchy, the more complex the process accomplished by Seasons_l_1_l_j. For example, consider the Seasons_*Salesperson*_*Status*(*Sales*) operation; the corresponding season *S* for a salesperson sp_1 in a status sta_1 is computed checking that during *S*, sp_1 is associated only with stores that rollup to status sta_1. Refer to Example 6.1 and Figure 6.2.

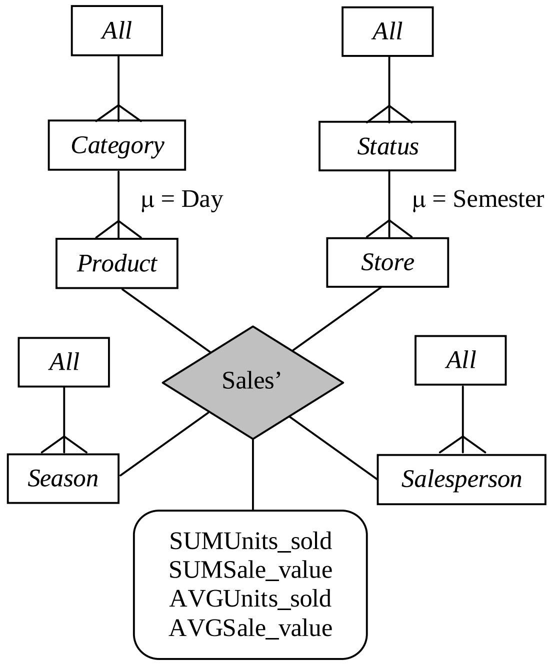


Figure 6.5. Resulting schema of Seasons_*Salesperson_Store*(*Sales*) operation.

Table 6.5. Sample data of *Sales* fact table.

Bottom levels			Measures	
Day	***Salesperson***	***Product***	**Units_sold**	**Sale_value**
1	sp_1	pd_1	2	2000
8	sp_1	pd_1	1	1000
8	sp_1	pd_2	1	500
28	sp_1	pd_2	1	500
50	sp_1	pd_1	1	1000
88	sp_1	pd_1	3	3000
320	sp_1	pd_1	1	1000
325	sp_1	pd_2	1	500
350	sp_1	pd_2	2	1000
500	sp_1	pd_1	2	2000
507	sp_1	pd_1	1	1000
521	sp_1	pd_1	1	1000
535	sp_1	pd_1	3	3000
1	sp_2	pd_1	3	3000
10	sp_2	pd_1	1	1000

Table 6.6. Resulting fact table of Seasons_*Salesperson_Store*(*Sales*) operation. (1) = SUMUnits_sold, (2)= SUMSale_value, (3) = AVGUnits_sold, and (4) = AVGSale_value.

Bottom levels				Measures			
***Season* (SeasonNumber, SeasonStart, SeasonEnd)**	***Salesperson***	***Store***	***Product***	**(1)**	**(2)**	**(3)**	**(4)**
s_1 = (1, 1, 45)	sp_1	st_1	pd_1	3	3000	1.5	1500
s_1 = (1, 1, 45)	sp_1	st_1	pd_2	2	1000	1	500
s_2 = (1, 46, 210)	sp_1	st_2	pd_1	4	4000	4	2000
s_3 = (2, 211, 390)	sp_1	st_1	pd_1	1	1000	1	1000
s_3 = (2, 211, 390)	sp_1	st_1	pd_2	1	1500	1.5	750
s_4 = (2, 481, 540)	sp_1	st_2	pd_1	7	7000	1.75	1750
s_5 = (1, 1, 150)	sp_2	st_2	pd_1	4	4000	2	2000

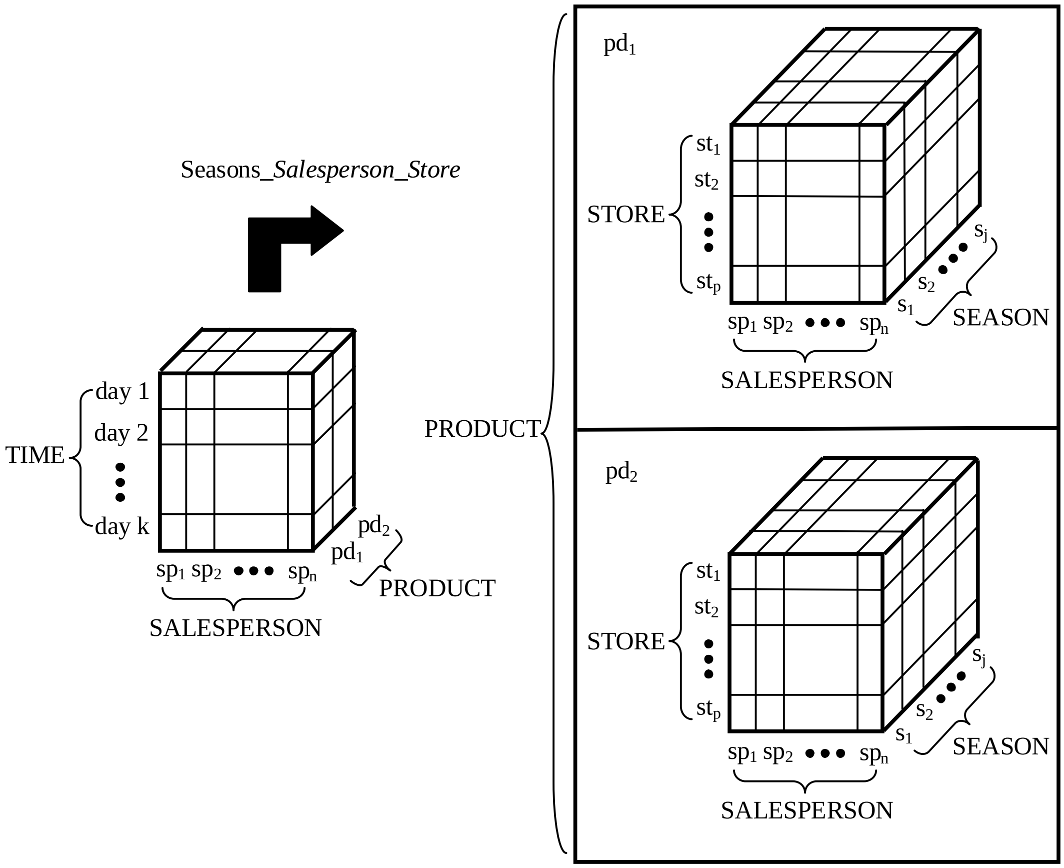


Figure 6.6. Outline of the Seasons_*Salesperson_Store*(*Sales*) operation.

6.3.3 Season queries examples

Table 6.7 presents solutions to the user requests of Table 6.3 using our season operator. We use the multidimensional query language of Datta [1999], which operates with cubes as the essential unit of input and output for all operators. In addition to the operators selection (σ) and aggregation (α), see Section 3.5.1, we use here the join of cubes ($\bowtie$):

$\bowtie$: relates two cubes having one or more dimensions in common.

Notation: $\text{Cube}_1 \bowtie \text{Cube}_2 = \text{Cube}_3$.

Besides, we propose the notation LevelName.AttributeName, *e.g.*, Season.SNumber, Salesperson.Salary; in order to reach the attributes of levels, since Datta's language lacks this feature.

Table 6.7. Season queries examples.

User request	Let $\text{Cube}_1 = \alpha_{[\text{SUM(SUMUnits\_sold) AS SumUnits, \{Season, Salesperson, Store\}}]}(\textbf{Seasons\_Salesperson\_Store}(\text{Sales}))$ **Query**
Total number of units sold by salesperson sp_1 in all his seasons in store st_1, *i.e.*, when he has worked in st_1.	$\alpha_{[\text{SUM(SumUnits), \{Salesperson\}}]}(\sigma_{\text{Salesperson = sp1 AND Store = st1}}(\text{Cube}_1))$
Total number of units sold by salesperson sp_1 in his first season in st_1.	$\sigma_{\text{Salesperson = sp1 AND Store = st1 AND Season.SeasonNumber = 1}}(\text{Cube}_1)$
Total number of units sold by each salesperson in each season in each store.	Cube_1
Season (including season number) and store when the total number of units sold by salesperson sp_1 was the highest.	i) $\text{Cube}_2 = \alpha_{[\text{MAX(SumUnits) AS MaxUnits, \{Salesperson\}}]}(\sigma_{\text{Salesperson = sp1}}(\text{Cube}_1))$ ii) $\sigma_{\text{SumUnits = MaxUnits}}(\text{Cube}_1 \bowtie \text{Cube}_2)$
Total value sold by each salesperson in his first two seasons in status B.	i) $\text{Cube}_3 = \textbf{Seasons\_Salesperson\_Status}(\text{Sales})$ ii) $\alpha_{[\text{SUM(SUMSale\_value), \{Salesperson\}}]}(\sigma_{\text{Status = B AND Season.SeasonNumber < 3}}(\text{Cube}_3))$

6.4 A brief comparison with SQL

In order to show the expressiveness [Gilman 1984] of our season operator, we formulate in SQL some of the queries from Table 6.7. We chose SQL because most of the database developers are familiar with this language. Another option is MDX, a multidimensional query language that has become a *de facto* standard for OLAP systems [Whitehorn 2005]. In Figure 6.7, we show some tables that correspond to a relational implementation of the temporal multidimensional model of Figure 2.4. The temporal relationship between *Salesperson* and *Store* levels is represented by a relational table Salesperson_Store.

Salesperson

CodSp	Name
--------	----------
sp_1	Lisa
sp_1	Andrew
sp_1	Kirsty
…	

Salesperson_Store

CodSp	CodSt	Start	End
--------	----------	------	----
sp_1	st_1	1	45
sp_1	st_2	46	210
sp_1	st_1	211	390
sp1	No_store	391	480
sp_1	st_2	481	540
sp_2	st_2	1	150
…			

Store

CodSt	Address
----------	---------------
st_1	Av. 5 - 99
st_2	221 Baker St.
st_3	South Mall-3
…	
No_store	No store

Sales

Day	Salesperson	Product	Units_sold	Sale_value
-----	--------------	---------	------------	------------
1	sp_1	pd_1	2	2000
8	sp_1	pd_1	1	1000
8	sp_1	pd_2	1	500
28	sp_1	pd_2	1	500
50	sp_1	pd_1	1	1000
88	sp_1	pd_1	3	3000
…				

Figure 6.7. Some tables of the relational implementation of our temporal multidimensional model for sales.

Next, we present an SQL formulation to find the total number of units sold by each salesperson in each season in each store, without using our season operator. The first step is to enumerate the seasons of each salesperson in each store. To accomplish this task, we use the analytic function ROW_NUMBER() of SQL-99 [Kline 2004].

```
CREATE VIEW Season_Salesperson_Store AS
SELECT CodSp, CodSt, ROW_NUMBER() OVER
        (PARTITION BY CodSp, CodSt ORDER BY Start) AS SeasonNumber,
        Start AS SeasonStart, End AS SeasonEnd
FROM Salesperson_Store;
```

Now, we can find the total number of units sold by each salesperson in each season:

```
SELECT CodSp, CodSt, SeasonNumber, SUM(Units_sold) AS SumUnits
FROM Sales, Season_Salesperson_Store
WHERE Salesperson = CodSp AND Day BETWEEN SeasonStart AND SeasonEnd
GROUP BY CodSp, CodSt, SeasonNumber;
```

Note that although the analytic function ROW_NUMBER() helps to formulate the query, the corresponding expression in Table 6.7 (third user request) is shorter and intuitive. On the other hand, the simulation of the last query from Table 6.7 is more complex, because we need to find the seasons between two non-consecutive levels in the hierarchy, *i.e.*, *Salesperson* and *Status*; we present an SQL solution in the Appendix.

6.5 Conclusions and future works

Motivated by the reclassifications of members of dimension levels and based on the formal temporal multidimensional model that we proposed in Chapter 2, we introduced the notion of *season*, a notion that gives rise to a family of queries that can support strategic decisions in several scenarios, such as sales, sports, the environment, customer management, and health care, among others. We also proposed an OLAP operator that allows us to express queries about seasons in a concise and simple way. We showed how it can be embedded in a typical multidimensional query language, and we presented its formal definition. We illustrated our approach through a case study about retail sales, where we identified and exemplified several season queries.

As a future work there are several issues to develop:

i) to relax the disjointness condition. This could lead to the management of overlapped seasons. For example, a salesperson could have a season from day 1 to day 90 with a store st_1 and another season from day 60 to day 150 with a store st_2,

ii) to split seasons. For example, suppose a salesperson signs two consecutive contracts with the same store, instead of managing a "big" season that covers the two contracts, a season could be defined for each contract in order to distinguish them. This could be useful for managing, *e.g.*, consecutive presidential terms,

iii) to merge seasons. For example, suppose a salesperson finishes a contract with a store and a week later renews it; we could ignore this "short" hiatus and define a single season that covers both contracts, and

iv) to experiment with real data in several domains and analyze the results in order to discover business trends that may be associated with seasons.

In the following chapter, we extend our operator in order to consider season queries that involve spatial features. For example, in a sales scenario, where salespersons are rotated through stores, we could formulate a query such as: What was the total number of units sold by a salesperson in his first season in a given region R_1? (Where R_1 is a spatial query window that contains a set of stores).

Chapter 7: Spatial Season Queries on a Spatio-temporal Multidimensional Model

7.1 Introduction

In Chapter 2 we proposed a formal multigranular multidimensional temporal model where a member (instance) of a level can be associated with several members of a higher hierarchical level throughout its lifespan, *i.e.*, a member can be reclassified, *e.g.*, a salesperson can rotate between the stores. These reclassifications originated the notion of *season* of reclassification, see Chapter 6. Informally, a season is a maximum interval during which a member of a level is associated with a member of a higher level. Note that throughout his lifespan a salesperson can experience several (disjoint) seasons in the same store. Thus the ordering of the seasons between two members originates the notion of the n^{th} season, *e.g.*, the first season of salesperson sp_1 in store st_1, the second season of sp_1 in st_1, the first season of sp_1 in st_2, and so on.

The seasons can originate queries such as: What was the total sales value made by sp_1 in his n^{th} season in st_1? This type of query is called *season queries* and were considered in Chapter 6. However, in that Chapter we did not consider season queries where spatial features could be involved, *e.g.*, what was the total sales value made by sp_1 in his n^{th} season in region R_1? (Where R_1 is a spatial query window that contains a set of stores). In this chapter, we extend our work in order to support this type of query, *i.e.*, *spatial season queries*. For this purpose, we propose a Spatial_Season operator in order to facilitate their formulation. Our operator receives a cube and returns a new one, thus facilitating its integration into a multidimensional query language, and enabling the composition of queries and the integration of their results. To the best of our knowledge, there is no language or operator that allows one to formulate spatial season queries in a concise and simple way.

Although over the last years both spatial and temporal DWs have been an active field of research [Malinowski 2008], [Golfarelli 2009a], the notion of spatial season queries is not present in any work we have found in the literature. The works closest to ours are the following. In [Rao 2003], the authors focus on solving queries such as: What was the total sales value of all stores that are inside a given region R_1? (Where R_1 is a spatial query window); however, they do not deal with members' reclassifications. Shekhar [2001] proposes an operator that supports spatial aggregation in the context of a spatial multidimensional database; however, this work also does not deal with members' reclassifications. Other works [Pedersen 2001a], [Chamoni 1999], [Mendelzon 2000], [Malinowski 2006], deal with reclassifications but they do not consider spatial features or season queries.

This chapter is organized as follows. In Section 7.2, we present a motivating example. In Section 7.3, we propose our Spatial_Season operator and in Sections 7.4 and 7.5, we give examples. In Section 7.6, we provide some basic experimental results and describe a prototype for the operator. Finally, in Section 7.7, we present conclusions and outline future work.

7.2 Motivating example

Consider a consortium with stores in the cities of a country. The country is divided territorially into departments (states), which group the cities. One of the most important subjects of analysis for the consortium are the sales of products, since from their behaviour may raise strategies for production, distribution, purchasing, inventory management, marketing, among others. The products are classified into categories, *e.g.*, cosmetics, meat and dairy products, and the customers are classified by gender and age groups.

A multidimensional model for representing this scenario is shown in Figure 7.1. Note that *Store*, *City*, and *Department* are *spatial levels*. A spatial level is a level that the application needs to keep its spatial characteristics [Malinowski 2008]. This is captured by its geometry represented using spatial types [Parent 1999], see examples in Figure 5.9. In addition, the symbol is used to represent the topological relationship *inside* [Parent 1999], [Schneider 2004] between spatial levels, *e.g.*, a store is inside a city and a city is inside a department. A sample data of Sales fact relationship is shown in Table 7.1.

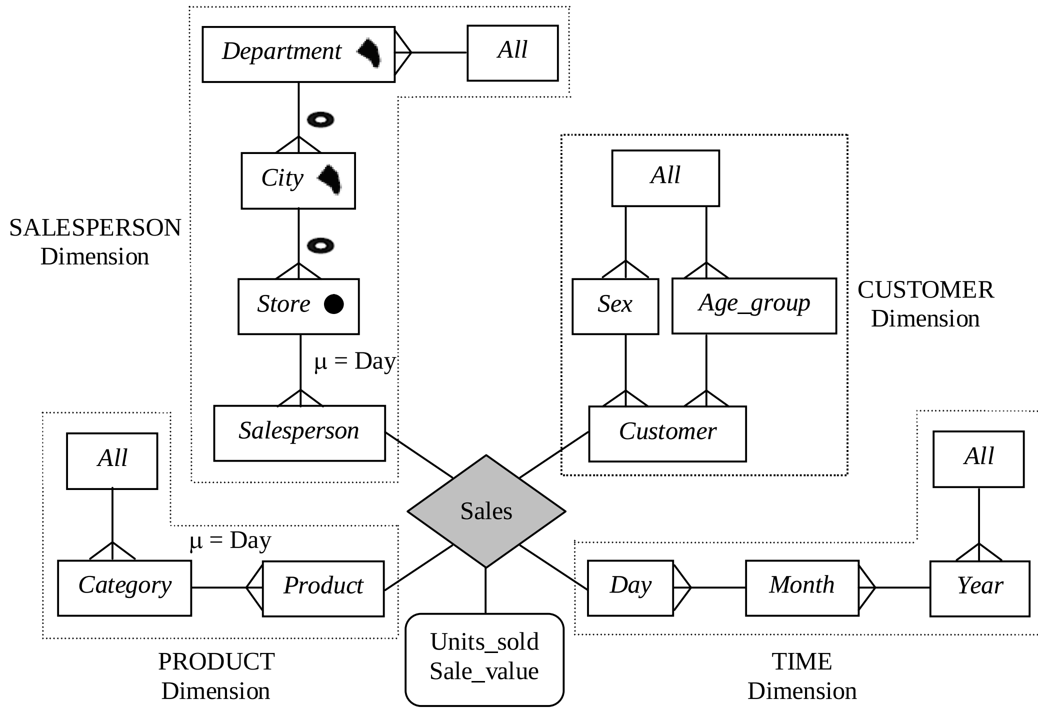


Figure 7.1. A multidimensional model for Sales.

Table 7.1. Sample data of Sales fact relationship.

Bottom levels				**Measures**	
Day	***Product***	***Salesperson***	***Customer***	**Units_sold**	**Sale_value**
1	pd_1	sp_1	$cust_1$	8	40
1	pd_2	sp_1	$cust_2$	7	70
2	pd_1	sp_1	$cust_1$	13	65
1	pd_2	sp_2	$cust_2$	6	60
2	pd_1	sp_2	$cust_1$	1	5

The salespersons of the consortium tend to rotate between the stores in periods of days. The rotation is due to factors such as salesperson's experience, skills, greater number of people in certain stores at certain times, distribution and launch of products, management of replacements due to vacation, permissions, sick leaves of the salespersons. Thus a salesperson associated with store st_1, may go on training, and later be associated with store st_2 and then return to store st_1. The temporary association between salespersons and stores is shown in Figure 7.1 using μ = Day (temporary unit to trace their assignments), see Chapter 2.

The consortium is interested in analyzing how the rotation affects the performance in sales of its salespersons, *e.g.*, analyzing the effect on the sales of a salesperson when he returns to the stores of a given region. For example, compare the total sales value of a salesperson in his n^{th} season in a region regarding his previous seasons in that region. Note that factors such as knowledge acquired in previous seasons or training received before returning to a region, can influence the performance of a salesperson. The results could help identify the training that is beneficial and when it should take place, the periods for launching products, and the stays (frequency and duration) of the salespersons in the stores, all with the aim of increasing sales.

Consider the dashed region R_1 shown in Figure 7.2. Let us suppose that R_1 represents the set of stores in the western region of a country and consider the query: obtain the total sales value made by salesperson sp_1 in his first season in the stores in the western region of the country. To answer this query, according to Figure 7.2, the sales made by sp_1 in his first and second seasons in st_1 and in his first season in st_2, st_3, and st_4 should be considered. Note that the sales made by sp_1 corresponding to his *second* season in st_1 are included because he *has not left* R_1. Thus while sp_1 rotates between the stores of R_1 without leaving this region, his sales will be part of the total requested. Eventually, when sp_1 leaves R_1 and then returns to some store in that region, he begins his *second* season in R_1 (in Figure 7.2, when sp_1 returns to st_3 from st_6). Moreover, more specialized queries can be formulated, *e.g.*, obtain the total sales value of cosmetics made to middle-aged women by sp_1 in his first season in the western stores.

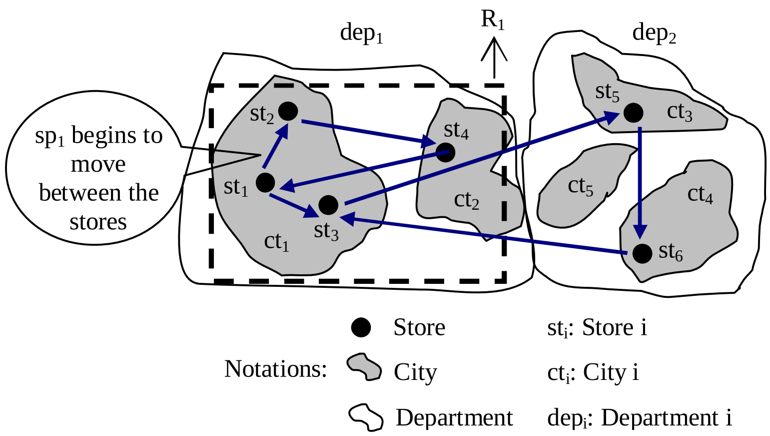


Figure 7.2. Rotation of salesperson sp_1 between the stores and spatial query window R_1.

Similar queries to the previous ones can be applied in other fields. In the military field, where the military units perform missions at strategic sites, the following query can be formulated: in its third season when the Red Unit performed missions in sites within the southern region, did the number of casualties decrease in comparison to the previous two seasons? In the fishing field, where the boats are regularly assigned to certain fishing spots, the following query may be formulated: What was the total number of salmon caught by all the boats in their first three seasons in the Polar region? (Where the Polar region contains a set of specific fishing spots). In the next section, we present an operator to facilitate the formulation of this type of query.

7.3 The Spatial_Season operator

In order to design an operator to facilitate the formulation of the queries outlined in Section 7.2, we identify the arguments required by such an operator. Consider Figure 7.3 and the query: obtain the total sales value made by salesperson sp_1 in his first season in the region R_2. Assume that the SALESPERSON dimension is formed as shown in Figure 7.4.

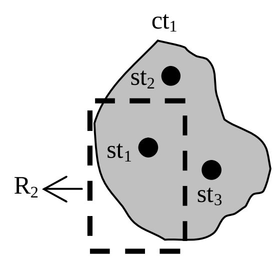


Figure 7.3. Spatial query window R_2.

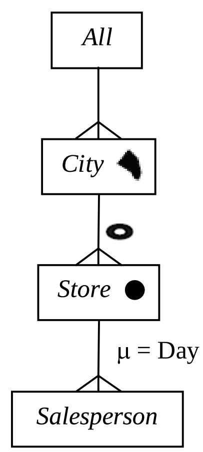


Figure 7.4. SALESPERSON dimension (first version).

Let Q be one sale made by sp_1 in his first season in st_1. Q contributes to the total requested since st_1 is inside R_2. Assume now that the SALESPERSON dimension is formed as shown in Figure 7.5 and that the Q sale was made by sp_1 when he *lived* in neighborhood n_1, see Figure 7.6. Thus the Q sale is characterized, from the geographic point-of-view, by store st_1 that is inside R_2 (and therefore, it *contributes* to the total requested) and by the neighborhood n_1 which is outside R_2 (and therefore, it does not c*ontribute* to the total requested). Therefore, the statement of the query should be clarified to avoid this ambiguity.

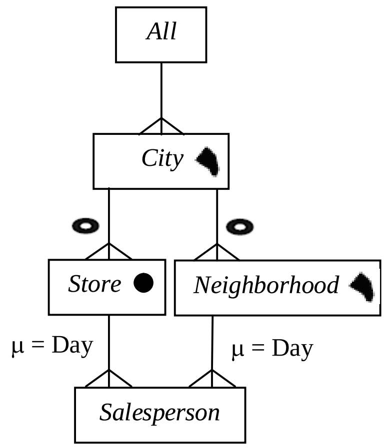


Figure 7.5. SALESPERSON dimension (second version).

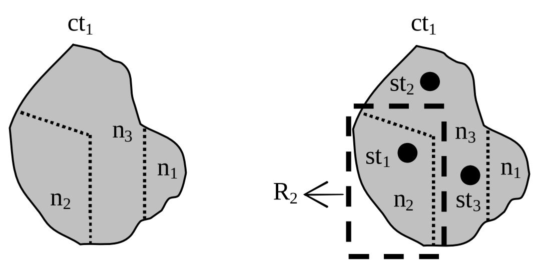


Figure 7.6. Neighborhoods of city ct_1 and spatial query window R_2.

Thus the user must specify, in the statement of his query, the corresponding *geographic context*: i) to obtain the total sales value made by sp_1 in his first season in the *stores* of R_2 or ii) obtain the total sales value made by sp_1 in his first season in the *neighborhoods* of R_2. That is, the geographic context indicates the geographic elements of interest associated with the facts and that are included in a given spatial query window. Note that in the second interpretation, and according to Figure 7.6, the sales made by sp_1 in st_1, st_2, and st_3, contribute to the total requested if they were made when sp_1 was in his first season in n_2.

On the other hand, note that it is possible that in a moment in time, a salesperson lives in a neighborhood of a city different from the city of the store where he works. That is, in time *t* the city associated with a salesperson along the path that goes by the *Store* level might be different from the city associated with the salesperson along the path that goes by the *Neighborhood* level. Thereby, *Salesperson.Store.City* and *Salesperson.Neighborhood.City* represent *different* geographic contexts. The first refers to the city where the salesperson *works* (store) and the second refers to the city where he *lives* (neighborhood). Thus a sale might be referred to two cities: the city of the store where the sale was made, and the city where the salesperson (that made the sale) lives.

In order to facilitate the formulation of this type of query, we define a Spatial_Season operator. Our operator receives as arguments: i) a spatial query window, ii) a geographic context, c) a list of aggregates, where an aggregate is an aggregate function applied to a measure, and iv) a cube. Our operator returns a cube as well. For example, consider a cube corresponding to the schema of Figure 7.1, the region R_2 of Figure 7.3 and the geographic context *Salesperson.Store*. For each salesperson in Sales his seasons in R_2 are calculated. Each season has its start and end time (SStart and SEnd attributes) and its corresponding order number (SNumber attribute). The facts are grouped into their respective seasons along with the aggregates requested.

For example, assume that the facts (day 5, pd_1, sp_1, $cust_1$, 10, 50), (day 28, pd_1, sp_1, $cust_1$, 15, 75), (day 19, pd_2, sp_1, $cust_1$, 8, 80), and (day 35, pd_2, sp_1, $cust_1$, 5, 50) are the only ones that are part of the first season of salesperson sp_1 in the region R_2, a season that takes place between day 1 and day 40. When applying the Spatial_Season operator with the aggregate list {SUM(Sale_value)} the operator generates two facts: (s_1, pd_1, sp_1, $cust_1$, 125) and (s_1, pd_2, sp_1, $cust_1$, 130). These results show that the salesperson sp_1 sold in his first season (s_1) in the region R_2 to customer $cust_1$, a total value of 125 of the product pd_1 and a total value of 130 of the product pd_2.

We define the following syntax for the operator Spatial_Season: $\text{Spatial\_Season}_{W,\ GC,\ AL}(C) = C'$, where:

i) W (Window): is the spatial query window, *e.g.*, region R_1 in Figure 7.2 and region R_2 in Figure 7.3,

ii) GC (Geographic Context): is a path expression that indicates the geographic context. The expression is formed by the level names separated by dots, it starts with the bottom level of one dimension and must end with a spatial level of the same dimension, *e.g.*, *Salesperson.Store* and *Salesperson.Store.City* are valid geographic contexts,

iii) AL (Aggregate List): is a list of elements g(m) where g is an aggregate function such as SUM, MAX, COUNT and m is a measure, *e.g.*, {SUM(Sale_value), MAX(Units_sold)}. Each g(m) generates a measure with name gm, *e.g.*, the previous list generates the names SUMSale_value and MAXUnits_sold,

iv) C (Cube): is the cube on which the Spatial_Season operator is applied, and

v) C': is the resulting cube.

Note that our Spatial_Season operator satisfies the closure property, *i.e.*, the results of an operator are again elements of the data model [Haase 2004]. Our operator takes a cube (C) as an argument and returns a new cube (C’), thus facilitating its integration into a multidimensional query language, and enabling the composition of queries and the integration of their results, see Section 7.5.

For simplicity, we have considered W as a single region; however, in a more general sense, W may represent a region set. The interpretation of a spatial season query in this situation remains the same, *e.g.*, consider the set R formed by the two search regions in Figure 7.7. Now consider a query such as: What was the total sales value made by a salesperson in his first season in R? To answer this query, we must consider the sales made by the salesperson from the moment he enters a store contained in R (st_1, st_5, or st_6), until the salesperson goes out of the stores in that region. Note that while the salesperson rotates between the stores contained in R, all his sales are part of his first season in that region. Eventually, when the salesperson leaves R and then returns to some store in that region, he begins his *second* season in R.

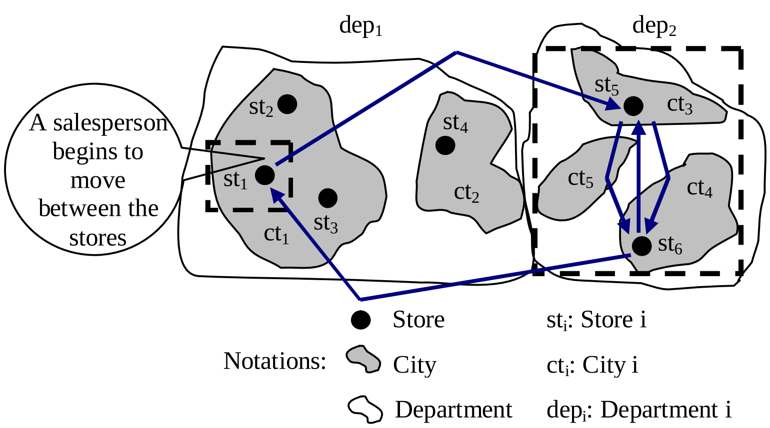


Figure 7.7. Rotation of a salesperson between the stores and two search regions (region set R).

The corresponding schema for the resulting cube C’ is generated as follows:

i) We preserve all the dimensions of the schema of the original cube (C) except the TIME dimension,

ii) The TIME dimension is replaced by a SEASON dimension with a homonymous level with attributes SStart, SEnd, and SNumber, and

iii) The measures are generated from the aggregate list as explained in iii) in the previous list.

In Figure 7.8 we outline the original and the resulting schema generated by Spatial_Season and in Table 7.2 we outline an algorithm to generate the resulting cube C’. We describe how C is transformed into C’; however, we emphasize that in an actual implementation C should remain intact.

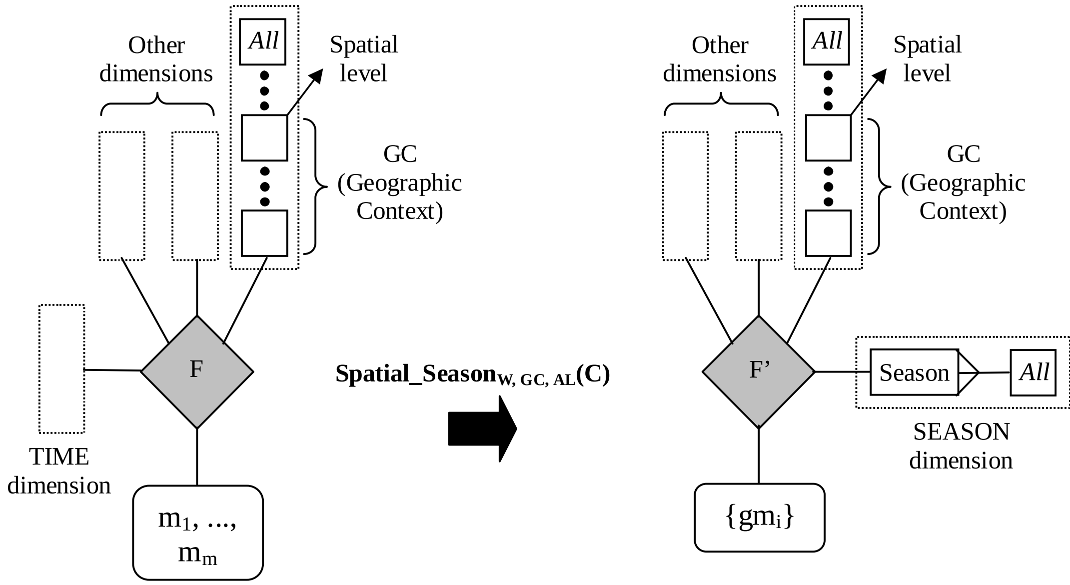


Figure 7.8. Spatial_Season operator: a) original schema and b) resulting schema.
AL = $\{g(m)_i\}$ where $m \in \{m_1, \ldots, m_m\}$.

Table 7.2. Spatial_Season operator algorithm.

Spatial_Season$_{W, GC, AL}$(C) = C'
Input: Cube C **Output:** Cube C' **Procedure:** **Step 1.** Let *highest_level* be the highest level of GC. Identify the members of *highest_level*, which is a spatial level, that are contained in the spatial query window W. **Step 2.** Let *lowest_level* be the lowest level of GC. For each member of *lowest_level* compute its seasons with regard to the region W and considering the members identified in the Step 1. For this purpose, consider the periods of association between the members of *lowest_level* and *highest_level*. The results make up a SEASON dimension with a level Season and attributes SStart, SEnd, and SNumber. **Step 3.** Insert the SEASON dimension, generated in Step 2, into the cube C. Each fact instance $f \in$ C is associated with a member of Season level as follows. Let SM be the set of members of Season level corresponding to the member *f.lowest_level*. The member $sm \in$ SM associated with the fact instance *f* is {*sm* | *sm* $\in$ SM AND *sm*.SeasonStart <= *f*.time_level <= *sm*.SeasonEnd}, where *f*.time_level refers to the member of the bottom level of the TIME dimension associated with *f* (remember that we use the symbol '.' to reach the attributes of a level). **Step 4.** Remove the TIME dimension from C and aggregate the measures, according to the aggregate list AL, for the rest of dimensions. The output is a cube C'.

7.4 Spatial_Season operator example

Consider the cube of Figure 7.9 corresponding to the schema of Figure 7.1, region R_1 of Figure 7.2, and the periods of association of sp_1 with the stores of Figure 7.10. The results of the operation Spatial_Season$_{R1,\ Salesperson.Store,\ \{SUM(Sale_value)\}}$(Sales) are shown in Figure 7.11 and in Figure 7.12. The algorithm is illustrated in Table 7.3.

Day	***Salesperson***	***Product***	***Customer***	**Units_sold**	**Sale_value**	
1	sp_1	pd_1	$cust_1$	8	40	Sales made by sp_1 in his *first* season in stores inside R_1.
1	sp_1	pd_2	$cust_2$	7	70	
2	sp_1	pd_1	$cust_1$	13	65	
			…			
225	sp_1	pd_2	$cust_2$	5	50	
226	sp_1	pd_1	$cust_{30}$	3	15	Sales made by sp_1 in stores *outside* R_1.
			…			
400	sp_1	pd_1	$cust_{35}$	11	55	
401	sp_1	pd_2	$cust_1$	2	20	Sales made by sp_1 in his *second* season in stores inside R_1.
401	sp_1	pd_1	$cust_2$	6	30	
402	sp_1	pd_2	$cust_1$	4	40	
			…			
613	sp_1	pd_1	$cust_2$	16	80	
614	sp_1	pd_1	$cust_{30}$	9	45	Sales made by sp_1 in stores *outside* R_1.
			…			
			…			Sales of other salespersons.

Figure 7.9. Sample data of Sales fact table.

Store	**From (day)**	**To (day)**		
st_1	**1**	40	Periods of the *first* season of sp_1 in stores inside region R_1.	*First* season of sp_1 in stores inside R_1: [**day 1**, **day 225**]
st_2	41	100		
st_4	101	150		
st_1	151	200		
st_3	201	**225**		
st_5	226	300	Periods of sp_1 in stores *outside* region R_1.	
st_6	301	400		
st_3	**401**	550	Periods of the *second* season of sp_1 in stores inside R_1.	*Second* season of sp_1 in stores inside R_1: [**day 401**, **day 613**]
st_4	551	**613**		
st_6	614	790	Periods of sp_1 in stores *outside* region R_1.	
	…			

Figure 7.10. Periods of association of salesperson sp_1 with the stores.

Season (SStart, SEnd, SNumber)	*Salesperson*	*Product*	*Customer*	**SUMSale_value**
s_1 = (day 1, day 225, 1)	sp_1	pd_1	$cust_1$	40 + 65 + …
s_1 = (day 1, day 225, 1)	sp_1	pd_2	$cust_2$	70 + … + 50
s_2 = (day 401, day 613, 2)	sp_1	pd_2	$cust_1$	20 + 40 + …
s_2 = (day 401, day 613, 2)	sp_1	pd_1	$cust_2$	30 + … + 80
		…		

Figure 7.11. Results of $\text{Spatial\_Season}_{\text{R1, Salesperson.Store, \{SUM(Sale\_value)\}}}$(Sales) = Sales'.

The results of Figure 7.11 show, *e.g.*, that salesperson sp_1 sold during his first season in the region R_1, that elapsed between day 1 and day 225, a total value of (70 + … + 50) of the product pd_2 to customer $cust_2$.

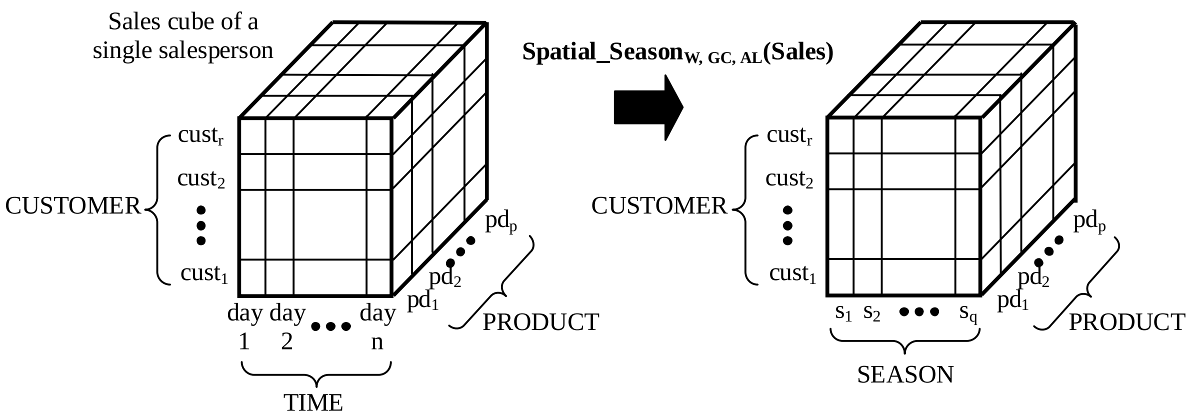


Figure 7.12. Spatial_Season operator: a) original cube and b) resulting cube.
(The operator is illustrated for a single salesperson).

Table 7.3. Example of Spatial_Season operator algorithm.

$\text{Spatial\_Season}_{\text{R1, Salesperson.Store, \{SUM(Sale\_value)\}}}$(Sales) = Sales'
Input: Cube Sales **Output:** Cube Sales' **Procedure:** **Step 1.** The highest level of GC = Salesperson.Store is Store. Stores contained in R_1 = {st_1, st_2, st_3, st_4}. **Step 2.** The lowest level of GC = Salesperson.Store is Salesperson. For example, to the salesperson sp_1 and considering his periods of association with the stores of Figure 7.10 his seasons are {s_1, s_2} where s_1 = (day 1, day 225, 1) and s_2 = (day 401, day 613, 2). **Step 3.** For example, consider the third fact instance (day 2, pd_1, sp_1, $cust_1$, 13, 65) of Figure 7.9. This fact instance is associated with the season s_1 because day 1 <= day 2 <= day 225. **Step 4.** The TIME dimension is removed from the cube C. The cube C is aggregated for AL = {SUM(Sale_value)} and for the rest of dimensions: {SEASON, SALESPERSON, PRODUCT, CUSTOMER}. The results are shown in Figure 7.11 and they make up the resulting cube Sales'. Note that a measure SUMSale_value is included there.

7.5 Spatial season queries examples

Table 7.4 presents formulations to some spatial season queries using our spatial season operator. We use the multidimensional query language of Datta [1999], which operates with cubes as the essential unit of input and output for all operators, see Subsections 3.5.1 and 6.3.3.

Table 7.4. Spatial season queries examples.

User request	Let $\textbf{Cube}_1 = \text{Spatial\_Season}_{\text{R1, Salesperson.Store, \{SUM(Sale\_value)\}}}(\textbf{Sales})$ **Query**
Total sales value made by sp_1 in his first season in the stores in the western region (R_1).	$\alpha_{\text{[SUM(SUMSale\_value), \{Salesperson\}]}}\sigma_{\text{Salesperson = sp1 AND Season.SNumber = 1}}(\textbf{Cube}_1)$
Total sales value of cosmetics made to middle-aged women by sp_1 in his first season in the stores in the western region (R_1).	$\alpha_{\text{[SUM(SUMSale\_value), \{Salesperson\}]}}\sigma_{\text{Salesperson = sp1 AND Season.SNumber = 1 AND Product.Category = Cosmetics AND Customer.Sex = Female AND Customer.Age\_group = Middle-aged}}(\textbf{Cube}_1)$
Total sales value made by sp_1 in all his seasons in the stores in the western region (R_1).	$\alpha_{\text{[SUM(SUMSale\_value), \{Salesperson\}]}}\sigma_{\text{Salesperson = sp1}}(\textbf{Cube}_1)$
Total sales value made by all salespersons in their three first seasons in the stores in the western region (R_1).	$\alpha_{\text{[SUM(SUMSale\_value), \{Salesperson\}]}}\sigma_{\text{Season.SNumber < 4}}(\textbf{Cube}_1)$
Total sales value made by sp_1 in his second season in the neighborhoods from the region R_2.	i) $\textbf{Cube}_2 = \text{Spatial\_Season}_{\text{R2, Salesperson.Neighborhood, \{SUM(Sale\_value)\}}}(\textbf{Sales})$ ii) $\alpha_{\text{[SUM(SUMSale\_value), \{Salesperson\}]}}\sigma_{\text{Salesperson = sp1 AND Season.SNumber = 2}}(\textbf{Cube}_2)$

7.6 Some basic experiments and prototype

In order to make some basic experiments, we analyzed data from a Colombian company that sells cosmetics. This company has branches in several towns in the department of Antioquia. Its salespersons are rotated between the branches. Usually a salesperson stays in a branch during a week. In his first visit to a branch, a salesperson gets in touch with potential costumers and paves the way for future sales in his subsequent visits. In Figure 7.13 we present the results of analyzing the first six seasons of six salespersons in a given region that contains several branches. Although, more extensive experiments and analysis are needed in order to try to identify possible behaviors, these results suggest that the total sales value made by a salesperson tends to increase in his first four seasons and then tends to stabilize.

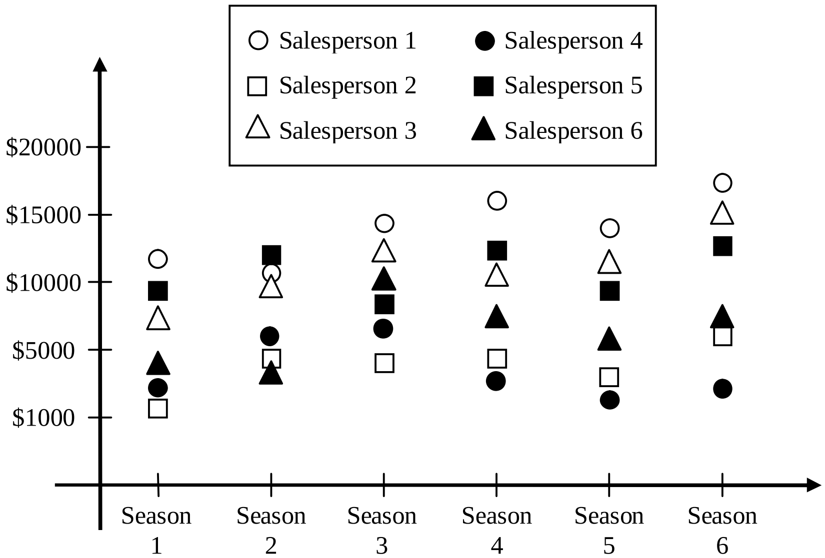


Figure 7.13. Total sales value made by six salespersons in their first six seasons.

In order to show the feasibility of our proposal, we developed a prototype where we simulated our Spatial_Season operator. We use Java and an Oracle 10g database with its spatial features (spatial data types and operators). We chose a relational implementation for our multidimensional model for Sales. Similarly, as we did in Section 6.4, we create tables to store information about sales (the fact table), salespersons, stores, cities, the temporal relationship between salesperson and stores, products, etc. The tables for stores and cities include an attribute of type SDO_GEOMETRY (the fundamental spatial data type in Oracle) to represent the location of a store (a point) and of a city (a region).

Figure 7.14 can give the reader a better idea about our interface. Salespersons are listed on the far left of the screen. Cities are represented by blue regions. Stores are represented by small squares of different colours; their names also appear with its corresponding colour in the top centre of the screen. There are five input boxes on the far right of the screen; these are intended to allow the user to set the values (season numbers) for his queries as described below.

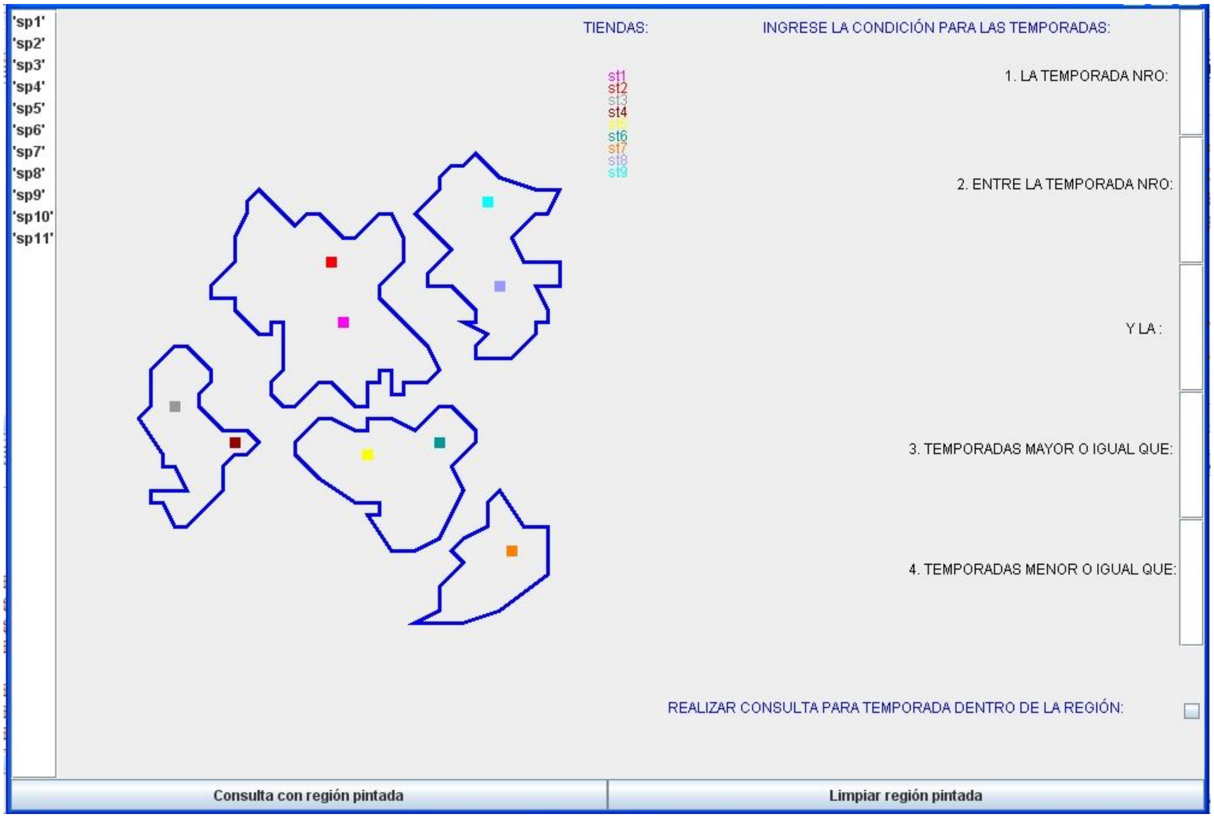


Figure 7.14. Prototype interface for spatial season queries.

To define a spatial season query using our graphical interface, the user must follow these steps: i) to choose a salesperson from the left list of salespersons, ii) to depict a spatial query window. In order to do that, the user must drag the mouse with the left button pressed, around the set of stores that he wants to enclose, iii) to set the values for season numbers. For example, if the user wants to focus on the first season of the salesperson that he chose, he must enter the number 1 in the top right input box, then press Enter key, iv) to mark the checkbox "Realizar consulta para temporada dentro de la region", and v) to press the bottom left button "Consulta con región pintada". Figure 7.15 and Figure 7.16 present an example of a user-defined spatial season query using our interface.

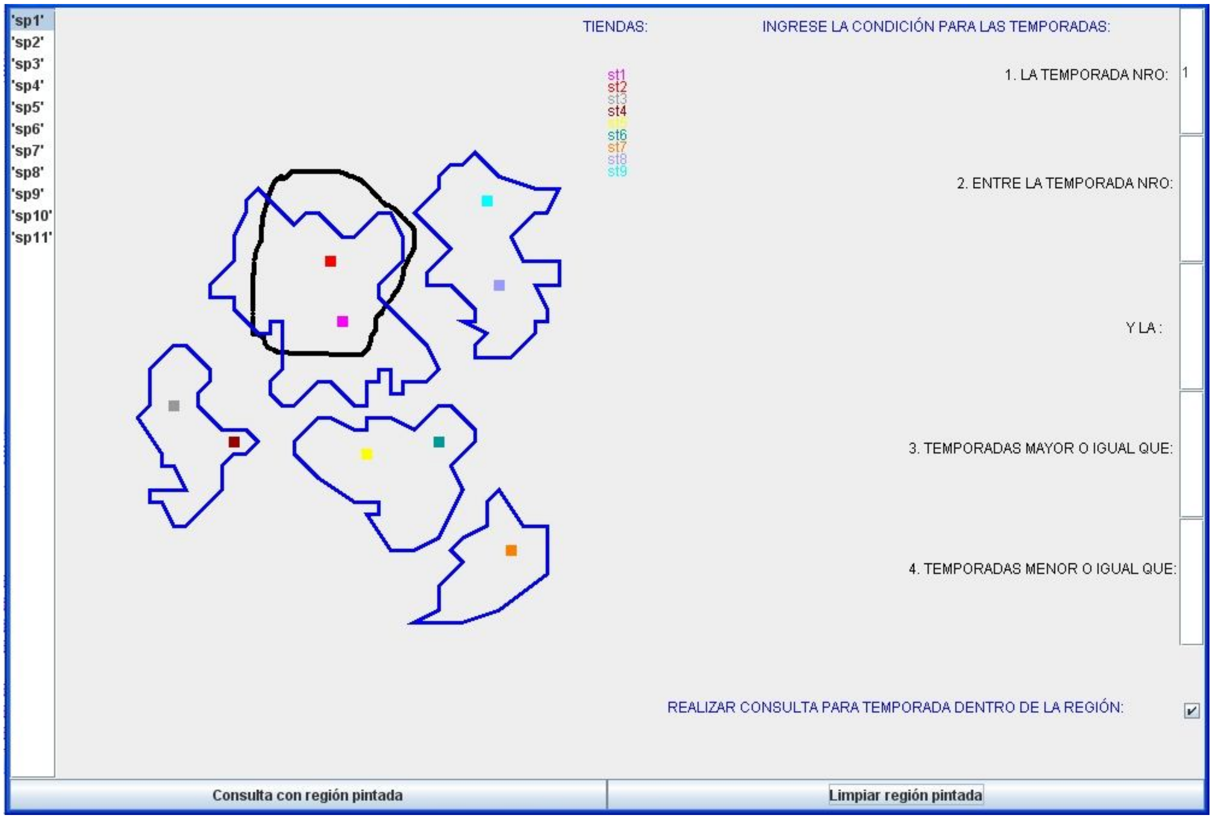


Figure 7.15. Definition of a spatial season query.

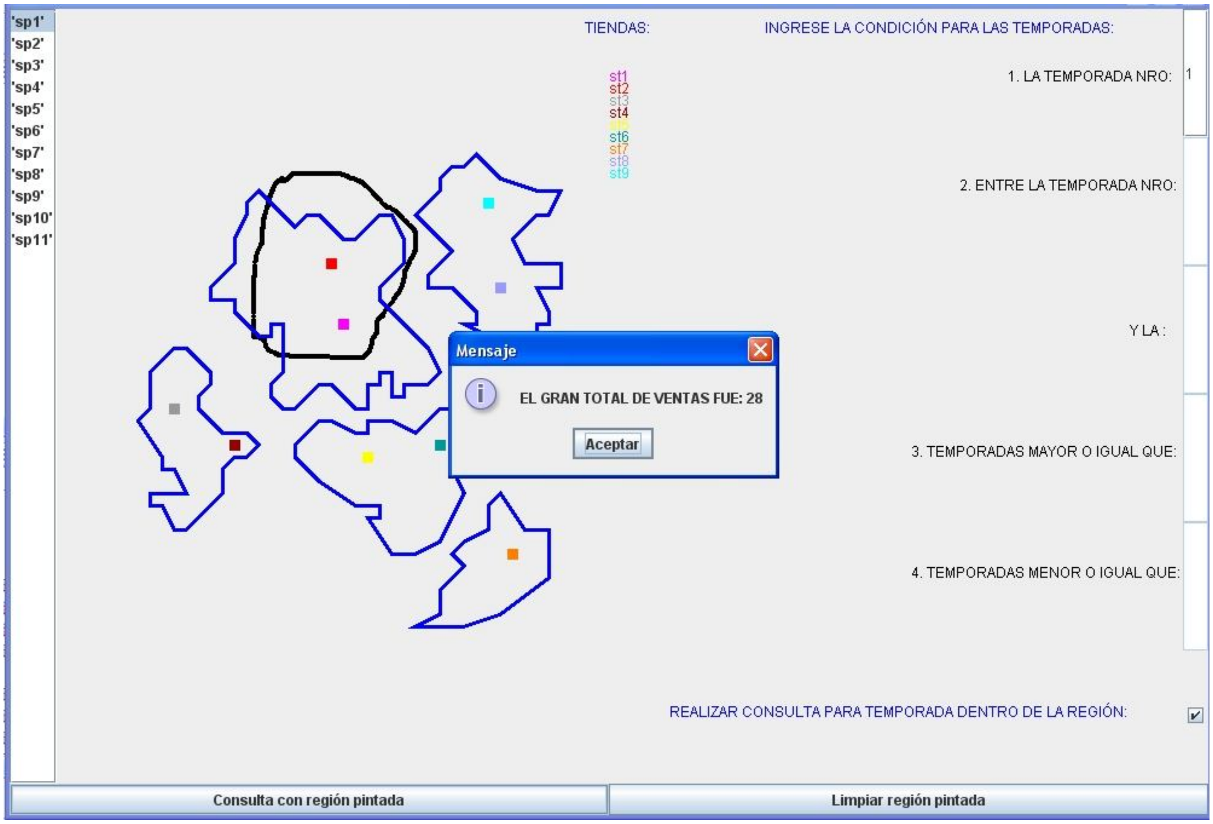


Figure 7.16. Result of a spatial season query.

Note that in step iii) the user can specify, using the different right input boxes, the condition for season numbers in several ways: seasons between the second and the fourth one, seasons greater than the third one, seasons less than the fifth one. Although the prototype in its current state has several limitations, *e.g.*, it does not support complex Boolean conditions, it only aggregates a measure using SUM function, and it computes seasons only for a single salesperson; it allows us to glimpse the possibilities of building a robust system where users can take advantage of graphical capabilities, which simplify their work and at the same time, free them from the burden of remembering the query language syntax. In this sense, our prototype can be considered a first attempt to create a visual query tool for spatial query seasons.

7.7 Conclusions and future work

In this chapter, we proposed an operator to facilitate the formulation of spatial season queries within the context of a multidimensional model, *i.e.*, queries such as: What was the total sales value of cosmetics made by a salesperson to middle-aged women in his first season in the stores of a given geographic region? (A spatial query window). This type of query can help to evaluate the performance of the salespersons in the wake of their rotation between the stores. Furthermore, these queries can be useful in other domains too, where several phenomena are involved in a recurring manner in a geographic scenario, *e.g.*, analyze both the material and human losses caused by a hurricane in its n^{th} season in a city, department, or country. This can help not only to take preventive measures but also to evaluate their effectiveness.

As future work, we plan to incorporate our operator in MDX [Whitehorn 2005]. However, at first glance there are two drawbacks that ought to be considered: first MDX has no spatial features and second MDX does not support temporal relationships between levels.

On the other hand, the temporality that exists between two levels can generate a complex data type: a trajectory. For example, a salesperson rotation between the stores defines a trajectory. We believe that the management of a trajectory as a first-class element in a DW, see Subsection 1.1.3 and Chapter 5, can similarly generate interesting queries, *e.g.*, analyze the performance of the salespersons that have followed similar trajectories, where the notion of similarity of trajectories should be defined. For example, two salesperson trajectories could be considered similar if they have in common at least 75% of stores visited. The works [Brakatsoulas 2004], [Orlando 2007], [Marketos 2008], [Spaccapietra 2008] are points of departure for these issues.

Chapter 8: Conclusions, Future Work, and Publications

8.1 Conclusions and future work

This thesis was motivated by the dynamics of a variety of changes that can take place in a DW. In the first part, we considered three issues related with spatial and temporal DWs: i) management of reclassifications that can occur with different temporal units in a dimension, ii) support for the change in the degree of containment, and iii) extensions to the map cube operator. In the second part, we addressed the representation of a trajectory as a complex measure in a conceptual spatial multidimensional model. In the third part, we introduced the notion of *season of reclassification*, a notion derived from reclassification trajectories, and proposed query constructs to facilitate the formulation of *season queries* and *spatial season queries*. These issues summarized our contributions.

Our main contribution was the formalization of the notion of *season* and its related query constructs. Therefore, this can be considered the core of our work.

We also believe we have paved the way toward the incorporation of a trajectory as a first-class element into a DW. This issue opens a wide spectrum of research possibilities: query languages, uncertainty management, storage, physical structures, among others. In particular, the trajectory uncertainty management in a DW is a promising issue. The essential idea is to try to fill missing information in a trajectory relying on historical data related with other “similar” trajectories. Obviously, the notion of similarity of trajectories must be formally defined. In addition, scenarios where the objects move on a predefined path, as in the case of public transport systems, may help complete the missing information.

Visual capabilities are also required, not only because trajectories are inherently spatio-temporal complex elements, but also because decision-makers are used to graphic interfaces that allow the recognition, in a friendly way, of possible patterns. In particular, we showed how a basic graphic interface that mimics spatial season queries, hides the mathematical complexity associated with this type of query, thus facilitating the formulation of the user requests.

In the course of our work, we performed some basic experiments in order to show the expediency of our proposals. Although these basic experiments suggested some incipient behaviours, we are aware that more rigorous and detailed experiments must be conducted to assess the practical value of our work. In the same vein, the incorporation of our query constructs to commercial query languages is a must. Although we showed an incipient implementation using an SQL-like approach, a comparison with more specialized OLAP languages is needed to evaluate, *e.g.*, their expressiveness.

With the development of our multidimensional model, the formalization of the notion of *season* and season query constructs, along with our basic prototype, experiments, and language comparisons; we have achieved all the proposed objectives. We also have fulfilled the requirements regarding the number of journal papers that must be accepted for the completion of a doctoral degree in our University. The doctoral program requires 1 paper accepted in a journal (indexed in Colciencias, Category A). In addition to other publications, see Section 8.2 and references, we fulfil this requirement with 4 journal papers: [Moreno 2010a], [Moreno 2010b], [Moreno 2010c], and [Moreno 2010d].

Finally, the reader is referred at the end of each chapter where more specific future works are posed.

8.2 Publications

- [Moreno 2010b] *Season queries on a temporal multidimensional model.*
  Mathematical and Computer Modelling, Elsevier, 52(7-8), 2010.
  Indexed in Colciencias (**Category A2**) and in **ISI SCI** (Institute for Scientific Information, Science Citation Index).
- [Moreno 2010a] *Cambio en el grado de inclusión en un modelo multidimensional*.
  Revista Facultad de Ingeniería Universidad de Antioquia, accepted for publication, 2010.
  Indexed in Colciencias (**Category A1**) and in **ISI SCI**.
- [Moreno 2010c] *Reclassification queries in a geographical data warehouse*.
  Revista Técnica de la Facultad de Ingeniería Universidad del Zulia, accepted for publication, 2010.
  Indexed in Colciencias (**Category A1**) and in **ISI SCI**.
- [Moreno 2010d] *A conceptual trajectory multidimensional model*.
  DYNA, accepted for publication, 2010.
  Indexed in Colciencias (**Category A1**) and in **ISI SCI**.
- [Moreno 2009b] *A multigranular temporal multidimensional model*.
  1st **IEEE** MiproBIS Conference on Business Intelligence Systems, Opatija, Croatia, 2009.
- [Moreno 2007a] *Estado del arte de los modelos multidimensionales espacio temporales*.

Revista Avances en Sistemas e Informática, 4(1), 2007.
Indexed in Colciencias (Category C).

- [Moreno 2007b] *Un acercamiento a los modelos multidimensionales espacio temporales*.
  VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC), Lima, Perú, 2007.
  Appears in DBLP (Digital Bibliography & Library Project).
- [Moreno 2008a] *Una extensión espacial al operador data cube*.
  Revista Avances en Sistemas e Informática, 5(1), 2008.
  Indexed in Colciencias (Category C).
- A shorter version [Moreno 2008b] of the previous paper was also published in:
  III CCC, Congreso Colombiano de Computación, Medellín, Colombia, 2008.
- [Moreno 2009a] *Extending the map cube operator with multiple spatial aggregate functions and map overlay*.
  17th International Conference on Geoinformatics, Fairfax, USA, 2009.
  Appears in **IEEE Xplore® Digital Library**.
- [Moreno 2009c] *Supporting the change in the degree of containment in a multidimensional model*.
  Journal of Information Technology and Control, 38(4), 2009.
  Indexed in **ISI SCI**.

Appendix: Seasons Between Salesperson and Status: An SQL Solution

Next, we present an SQL solution to find the total sale value of each salesperson in his first two seasons in status B, the last user request in Table 6.7. The first step is to determine for each season of a salesperson in a store, the intersection with the seasons of stores in statuses. To accomplish this task, we create the following view:

```
CREATE VIEW V1 AS
SELECT CodSp, SpSt.Start AS Salesperson_Start, SpSt.End AS Salesperson_End,
       CodStatus, StSta.Start AS Status_Start, StSta.End AS Status_End
FROM Salesperson_Store AS SpSt, Store_Status AS StSta
WHERE SpSt.CodSt = StSta.CodSt AND
     ((SpSt.Start BETWEEN StSta.Start AND StSta.End) OR
      (SpSt.End BETWEEN StSta.Start AND StSta.End) OR
      (SpSt.Start < StSta.Start AND SpSt.End > StSta.End)
     );
```

Now, we create a second view to determine for each tuple of V1 exactly which days correspond with the salesperson in the corresponding status:

```
CREATE VIEW V2 AS
SELECT CodSp, CASE WHEN Salesperson_Start <= Status_Start THEN
                 Status_Start ELSE Salesperson_Start
                 END AS New_Status_Start,
       CASE WHEN Salesperson_End <= Status_End THEN
       Salesperson_End ELSE Status_End
       END AS New_Status_End,
       CodStatus
FROM V1;
```

The next step is to find the maximum continuous intervals of each salesperson in each status, *i.e.*, the seasons of salespersons in statuses. Although it is possible to formulate a cumbersome SQL query to accomplish this task, we preferred to use the procedural extensions for SQL, *i.e.*, SQL PSM (Persistent Stored Modules [ISO/IEC 2008]). In the following, we use Oracle's PL/SQL

[Oracle 2009], which is based on SQL PSM. Season_Salesperson_Status is an auxiliary table to store the seasons between salespersons and statuses.

```
DECLARE
CURSOR Sorted IS SELECT * FROM V2 ORDER BY CodSp, New_Status_Start;
Current_Row Sorted%ROWTYPE;
Next_Row Sorted%ROWTYPE;
Status_End V2.New_Status_End%TYPE;
Flag NUMBER(1) := 0;
BEGIN
OPEN Sorted;
FETCH Sorted INTO Current_Row;
IF Sorted%FOUND THEN
  Status_End := Current_Row.New_Status_End;
  LOOP
    FETCH Sorted INTO Next_Row;
    EXIT WHEN Sorted%NOTFOUND;
    WHILE Current_Row.CodSp = Next_Row.CodSp AND
              Current_Row.CodStatus = Next_Row.CodStatus LOOP
      Status_End := Next_Row.New_Status_End;
      FETCH Sorted INTO Next_Row;
      EXIT WHEN Sorted%NOTFOUND;
    END LOOP;
    INSERT INTO Season_Salesperson_Status VALUES
    (Current_Row.CodSp, Current_Row.New_Status_Start, Status_End, Current_Row.CodStatus);
    IF Sorted%NOTFOUND THEN Flag := 1;
    ELSE Flag := 0;
    END IF;
    Current_Row := Next_Row;
    Status_End := Next_Row.New_Status_End;
  END LOOP;
  IF Flag = 0 THEN
    INSERT INTO Season_Salesperson_Status VALUES
```

```
    (Current_Row.CodSp, Current_Row.New_Status_Start, Status_End, Current_Row.CodStatus);
  END IF;
END IF;
CLOSE Sorted;
END;
```

Now, it follows an analogous process as developed in Subsection 6.4.4, enumeration of the seasons of each salesperson in each status:

```
CREATE VIEW VSeason_Salesperson_Status AS
SELECT CodSp, CodStatus, ROW_NUMBER() OVER
           (PARTITION BY CodSp, CodStatus ORDER BY Start) AS SeasonNumber,
            Start AS SeasonStart, End AS SeasonEnd
FROM Season_Salesperson_Status;
```

Finally, we can find the total sale value of each salesperson in his first two seasons in status B:

```
SELECT CodSp, SUM(Sale_value) AS SumSale_value
FROM Sales, VSeason_Salesperson_Status
WHERE Salesperson = CodSp AND Day BETWEEN SeasonStart AND SeasonEnd AND
CodStatus = 'B' AND SeasonNumber < 3
GROUP BY CodSp;
```

This example illustrates the expressiveness of our season operator, which accomplishes the same task in a concise way as is shown in the last row of Table 6.7 (second column).

References

[Agarwal 1996] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi, On the computation of multidimensional aggregates, 22nd International Conference on Very Large Data Bases (VLDB), pp. 506-521, Mumbai, India, 1996.

[Agrawal 1997] R. Agrawal, A. Gupta, S. Sarawagi, Modeling multidimensional databases, 13th International Conference on Data Engineering (ICDE), pp. 232-243, Birmingham, UK, 1997.

[Allen 1983] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26(11), 1983, pp. 832-843.

[Alvares 2007] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. Fernandes de Macêdo, B.Moelans, A. A. Vaisman, A model for enriching trajectories with semantic geographical information, 15th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS), pp. 162-169, Seattle (Washington), USA, 2007.

[Balmin 2000] A. Balmin, T. Papadimitriou, Y. Papakonstantinou, Hypothetical queries in an OLAP environment, 26th International Conference on Very Large Data Bases (VLDB), pp. 220-231, Cairo, Egypt, 2000.

[Bédard 2001] Y. Bédard, T. Merrett, J. Han, Fundaments of spatial data warehousing for geographic knowledge discovery, chapter in: H. Miller (ed.), Geographic data mining and knowledge discovery, Taylor & Francis, UK, 2001, pp. 53-73.

[Bimonte 2005] S. Bimonte, A. Tchounikine, M. Miquel, Towards a spatial multidimensional model, 8th International Workshop on Data Warehousing and OLAP (DOLAP), pp. 39-46, Bremen, Germany, 2005.

[Blaschka 1999] M. Blaschka, C. Sapia, G. Höfling, On schema evolution in multidimensional databases, 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 153-164, Florence, Italy, 1999.

[Body 2002] M. Body, M. Miquel, Y. Bèdard, A. Tchounikine, A multidimensional and multiversion structure for OLAP applications, 5th International Workshop on Data Warehousing and OLAP (DOLAP), pp. 1-6, McLean, USA, 2002.

[Brakatsoulas 2004] S. Brakatsoulas, D. Pfoser, N. Tryfona, Modeling, storing, and mining moving object databases, 8th International Database Engineering and Applications Symposium (IDEAS), pp. 68-77, Coimbra, Portugal, 2004.

[Braz 2007] F. J. Braz, Trajectory data warehouses: Proposal of design and application to exploit data, 9th Brazilian Symposium on GeoInformatics (GeoInfo), pp. 61-72, Campos do Jordão, Brazil, 2007.

[Cabibbo 1997] L. Cabibbo, R. Torlone, Querying multidimensional databases, 6th International Workshop on Database Programming Languages (DBPL-6), pp. 319-335, Estes Park, USA, 1997.

[Cely 2006] J. Cely, Y. Bédard, El paradigma multidimensional: desarrollo de nuevas tecnologías para la gestión del territorio, 12th International Symposium of the Latinoamerican Society of Remote Sensing and Spatial Information Systems (SELPER), pp. 1-11, Cartagena, Colombia, 2006.

[Chamoni 1999] P. Chamoni, S. Stock, Temporal structures in data warehousing, 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 353-358, Florence, Italy, 1999.

[Codd 1993] E. F. Codd, Providing OLAP (On-Line Analytical Processing) to user analysts: an IT mandate, Technical Report, E.F. Codd Associates, USA, 1993.

[Damiani 2006] M. L. Damiani, S. Spaccapietra, Spatial data warehouse modeling, chapter in: J. Darmont (ed.), Processing and managing complex data for decision support, Idea Publishers, UK, 2006, pp. 1-27.

[Da Silva 2004] J. Da Silva, V. Times, R. Fidalgo, R. Barros, Towards a web service for geographic and multidimensional processing, 6th Brazilian Symposium on Geoinformatics (GeoInfo), pp. 1-18 , Campos do Jordão, São Paulo, Brazil, 2004.

[Datta 1999] A. Datta, H. Thomas, The cube data model: A conceptual model and algebra for on-line analytical processing in data warehouses, Decision Support Systems 27(3), 1999, pp. 289-301.

[Eder 2001] J. Eder, C. Koncilia, Changes of dimension data in temporal data warehouses, 3rd International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 284-293, Munich, Germany, 2001.

[Ferreira 2001] A. C. Ferreira, M. L. Campos, A. K. Tanaka, An architecture for spatial and dimensional analysis integration, 6th World Multiconference on Systemics, Cibernetics and Informatics (SCI), pp. 392-395, Orlando, USA, 2001.

[Ferri 2000] F. Ferri, E. Pourabbas, M. Rafanelli, F. L. Ricci, Extending geographic databases for a query language to support queries involving statistical data, 12th International Conference on Scientific and Statistical Database Management (SSDBM), pp. 220-230, Berlin, Germany, 2000.

[Fidalgo 2004] R. Fidalgo, V. Times, J. da Silva, F. da Fonseca de Souza, GeoDWFrame: a framework for guiding the design of geographical dimensional schemas, 6th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 26-37, Zaragoza, Spain, 2004.

[Freese 2004] R. Freese, Automated lattice drawing, 2nd International Conference on Formal Concept Analysis (ICFCA), pp. 112-127, Sydney, Australia, 2004.

[Frentzos 2005] E. Frentzos, K. Gratsias, N. Pelekis, Y. Theodoridis, Nearest neighbor search on moving object trajectories, 9th Symposium on Spatial and Temporal Databases (SSTD), pp. 328-345, Angra dos Reis, Brazil, 2005.

[Gilman 1984] L. Gilman, A.J. Rose, APL: An interactive approach, 3rd Edn., Wiley, New York, 1984.

[Golfarelli 1998] M. Golfarelli, D. Maio, S. Rizzi, The dimensional fact model: A conceptual model for data warehouses, International Journal of Cooperative Information Systems 7(2-3), 1998, pp. 215-247.

[Golfarelli 2006] M. Golfarelli, J. Lechtenbörger, S. Rizzi, G. Vossen, Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation, Data and Knowledge Engineering, 59(2), 2006, pp. 435-459.

[Golfarelli 2009a] M. Golfarelli, S. Rizzi, A survey on temporal data warehousing, International Journal of Data Warehousing and Mining 5(1), 2009, pp. 1-17.

[Golfarelli 2009b] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern principles and methodologies, 1st Edn., McGraw-Hill Osborne Media, New York, 2009.

[Gray 1997] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals, Data Mining and Knowledge Discovery 1(1), 1997, pp. 29-53.

[Guc 2008] B. Guc, M. May, Y. Saygin, C. Körner, Semantic annotation of GPS trajectories, 11th International Conference on Geographic Information Science (AGILE), pp. 1-9, Girona, Spain, 2008.

[Güting 2005] R. H. Güting, M. Schneider, Moving objects databases, 1st Edn., Morgan Kaufmann, San Francisco, 2005.

[Gyssens 1997] M. Gyssens, L. Lakshmanan, A foundation for multi-dimensional databases, 23rd International Conference on Very Large Data Bases (VLDB), pp. 106-115, Athens, Greece, 1997.

[Haase 2004] P. Haase, J. Broekstra, A. Eberhart, R. Volz, A comparison of RDF query languages, 3rd International Semantic Web Conference (ISWC), pp. 502-517, Hiroshima, Japan, 2004.

[Han 1998] J. Han, N. Stefanovic, K. Koperski, Selective materialization: An efficient method for spatial data cube construction, 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp. 114-158, Melbourne, Australia, 1998.

[Hurtado 1999] C. A. Hurtado, A. O. Mendelzon, A. A. Vaisman, Updating OLAP dimensions, ACM 2nd International Workshop on Data Warehousing and OLAP (DOLAP), pp. 60-66, Kansas City, USA, 1999.

[Inmon 2005] W. H. Inmon, Building the data warehouse, 4th Edn., Wiley, New York, 2005.

[IMT 2009] IMT: Instituto Mexicano del Transporte, Anuario estadístico de accidentes en carreteras federales 1997 – 2006, date of access July 2009, available in: http://www.imt.mx/Espanol/Publicaciones/

[ISO/IEC 2008] International Organization for Standardization (ISO)/International Electrotechnical Commission(IEC), ISO/IEC 9075-4:2008 SQL part 4: Persistent Stored Modules (SQL/PSM), ISO Standard, USA, 2008.

[Jarke 2003] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis, Fundamentals of data warehouses, 2nd Edn., Springer, New York, 2003.

[Jensen 2004] C. S. Jensen, A. Kligys, T. B. Pedersen, I. Timko, Multidimensional data modeling for location-based services, VLDB Journal 13(1), 2004, pp. 1-21.

[Kaas 2004] C. Kaas, T. B. Pedersen, B. Rasmussen, Schema evolution for stars and snowflakes, 6th International Conference on Enterprise Information Systems (ICEIS), pp. 425-433, Porto, Portugal, 2004.

[Kimball 2008] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, The data warehouse lifecycle toolkit, 2nd Edn., Wiley, New York, 2008.

[Kline 2004] K. E. Kline, D. Kline, B. Hunt, SQL in a nutshell, 1st Edn., O'Reilly, New York, 2004.

[Kouba 2000] Z. Kouba, K. Matousek, P. Miksovský, On data warehouse and GIS integration, 11th International Conference on Database and Expert Systems Applications (DEXA), pp. 604-613, London, UK, 2000.

[Kumar 2008] N. Kumar, A. Gangopadhyay, S. Bapna, G. Karabatis, Z. Chen, Measuring interestingness of discovered skewed patterns in data cubes, Decision Support Systems 46(1), 2008, pp. 429-439.

[Lehner 1998] W. Lehner, J. Albrecht, H. Wedekind, Normal forms for multidimensional databases, 10th International Conference on Scientific and Statistical Database Management (SSDBM), pp. 63-72, Capri, Italy, 1998.

[Lenz 1997] H. Lenz, A. Shoshani, Summarizability in OLAP and statistical data bases, 9th International Conference on Scientific and Statistical Database Management (SSDBM), pp. 132-143, Olympia (Washington), USA,1997.

[Levine 1995] J. R. Levine, T. Manson, D. Brown, Lex and Yacc, 2nd Edn., O'Reilly and Associates,1995.

[Longley 2005] P. A. Longley, M. F. Goodchild, D. J. Maguire, D. W. Rhind, Geographical information systems: Principles, techniques, management and applications, 2nd Edn., Wiley, New York, 2005.

[Lu 2003] C. T. Lu, Y. Kou, H. Wang, S. Shekhar, P. Zhang, R. Liu, Two Web-based spatial data visualization and mining systems: mapcube & mapview, 1st International Workshop on Next Generation Geospatial Information (N2GI), No pages, Cambridge (Massachusetts), USA, 2003.

[Malinowski 2006] E. Malinowski, E. Zimányi, A conceptual solution for representing time in data warehouse dimensions, 3rd Asia-Pacific Conference on Conceptual Modelling (APCCM 2006), pp. 45-54, Hobart, Tasmania, 2006.

[Malinowski 2008] E. Malinowski, E. Zimányi, Advanced data warehouse design: From conventional to spatial and temporal applications, 1st Edn., Springer, New York, 2008.

[Marketos 2008] G. Marketos, E. Frentzos, I Ntoutsi, N. Pelekis, A. Raffaetà, Y. Theodoridis, Building real world trajectory warehouses, 7th International ACM SIGMOD Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), pp. 1-8, Vancouver, Canada, 2008.

[Mendelzon 2000] A. O. Mendelzon, A. A. Vaisman, Temporal queries in OLAP, 26th International Conference on Very Large Data Bases (VLDB), pp. 242-253, Cairo, Egypt, 2000.

[Microsoft 2009] Microsoft, Microsoft SQL Server 2008, date of access June 2009, available in: http://www.microsoft.com/sqlserver/2008/en/us.

[Miksovský 2001] P. Miksovský, Z. Kouba, GOLAP - Geographical online analytical processing, 12th International Conference Database and Expert Systems Applications (DEXA), pp. 442-449, Munich, Germany, 2001.

[Moreno 2007a] F. Moreno, F. Arango, Estado del arte de los modelos multidimensionales espacio temporales, Revista Avances en Sistemas e Informática 4(1), 2007, pp. 11-16.

[Moreno 2007b] F. Moreno, F. Arango, Un acercamiento a los modelos multidimensionales espacio temporales, VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC), pp. 93-98, Lima, Perú, 2007.

[Moreno 2008a] F. Moreno, A. Urrea, Una extensión espacial al operador data cube, Revista Avances en Sistemas e Informática 5(1), 2008. pp. 19-28.

[Moreno 2008b] F. Moreno, A. Urrea, Una revisión al operador map cube, Tercer Congreso Colombiano de Computación (CCC), No pages, Medellín, Colombia, 2008.

[Moreno 2009a] F. Moreno, F. Arango, Extending the map cube operator with multiple spatial aggregate functions and map overlay, 17th International Conference on Geoinformatics (Geoinformatics), No pages, Fairfax, USA, 2009.

[Moreno 2009b] F. Moreno, F. Arango, R. Fileto, A multigranular temporal multidimensional model, 1st IEEE Conference on Business Intelligence Systems (miproBIS), pp. 1-6, Opatija, Croatia, 2009.

[Moreno 2009c] F. Moreno, F. Arango, I. Uribe, Supporting the change in the degree of containment in a multidimensional model, Journal of Information Technology and Control 38(4), 2009, pp. 311-318.

[Moreno 2010a] F. Moreno, F. Arango, I. Uribe, Cambio en el grado de inclusión en un modelo multidimensional, Revista Facultad de Ingeniería Universidad de Antioquia (53), 2010. pp. 236-244.

[Moreno 2010b] F. Moreno, F. Arango, R. Fileto, Season queries on a temporal multidimensional model, Mathematical and Computer Modelling 52(7-8), 2010. pp. 103-109.

[Moreno 2010c] F. Moreno, F. Arango, J. Echeverry, Reclassification queries in a geographical data warehouse, Revista Técnica de la Facultad de Ingeniería Universidad del Zulia, accepted to appear, 2010.

[Moreno 2010d] F. Moreno, F. Arango, A conceptual trajectory multidimensional model, DYNA, accepted to appear, 2010.

[Morzy 2004] T. Morzy, R. Wrembel, On querying versions of multiversion data warehouse, 7th ACM International Workshop on Data Warehousing and OLAP (DOLAP), pp. 92-101, Washington D.C., USA, 2004.

[OLAP Council 2009] OLAP Council, The OLAP glossary, date of access May 2009, available in: http://www.olapcouncil.org/research/resrchly.htm.

[Oracle 2008] Oracle Corporation, Oracle® Database PL/SQL language reference 11g release 1 (11.1), Oracle Publishers, San Francisco, 2008.

[Orlando 2007a] S. Orlando, R. Orsini, A. Raffaeta, A. Roncato, Trajectory data warehouses: Design and implementation issues, Journal of Computing Science and Engineering 1(2), 2007, pp. 211-232.

[Orlando 2007b] S. Orlando, R. Orsini, A. Raffaetà, A. Roncato, C. Silvestri, Spatio-temporal aggregations in trajectory data warehouses, 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 66-77, Regensburg, Germany, 2007.

[Parent 1999] C. Parent, S. Spaccapietra, E. Zimányi, Spatio-temporal conceptual models: Data structures + space + time, 7th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS), pp. 26-33, Kansas, USA, 1999.

[Pedersen 2001a] T. B. Pedersen, C. S. Jensen, C. E. Dyreson, A foundation for capturing and querying complex multidimensional data, Information Systems 26(5), 2001, pp. 383-423.

[Pedersen 2001b] T. B. Pedersen, N. Tryfona, Pre-aggregation in spatial data warehouses, 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD), pp. 460-480, Redondo Beach, USA, 2001.

[Pelekis 2007] N. Pelekis, I. Kopanakis, I. Ntoutsi, G. Marketos, Y. Theodoridis, Mining trajectory databases via a suite of distance operators, 23rd International Conference on Data Engineering (ICDE), pp. 575-584, Istanbul, Turkey, 2007.

[Pentaho 2009] Pentaho, Pentaho BI Suite Enterprise Edition, date of access June 2009, available in: http://www.pentaho.com.

[Pestana 2005] G. Pestana, M. Mira da Silva, Multidimensional modeling based on spatial, temporal and spatio-temporal stereotypes, 25th ESRI International User Conference (ESRI), No pages, San Diego, USA, 2005.

[Pourabbas 2005] E. Pourabbas, Cooperation of geographic and multidimensional databases, chapter in: M. Khosrow-Pour (ed.), Encyclopedia of information science and technology (I), Idea Group, UK, 2005, pp. 596-602.

[Rao 2003] F. Rao, L. Zhang, X. Yu, Y. Li, Y. Chen, Spatial hierarchy and OLAP-favored search in spatial data warehouse, 6th International Workshop on Data Warehousing and OLAP (DOLAP), pp. 48-55, New Orleans, USA, 2003.

[Ravat 2006] F. Ravat, O. Teste, Supporting data changes in multidimensional data warehouses, International Review on Computers and Software 1(3), 2006, pp. 251-259.

[Rechy-Ramirez 2006] E. Rechy-Ramirez, E. Benitez-Guerrero, A model and language for bitemporal schema versioning in data warehouses, 15th International Conference on Computing (CIC), pp. 309-314, Mexico City, Mexico, 2006.

[Rivest 2001] S. Rivest, Y. Bédard, P. Marchand, Toward better support for spatial decision making: defining the characteristics of spatial on-line analytical processing (SOLAP), Geomatica 55(4), 2001, pp. 539-555.

[Sampaio 2006] M. C. Sampaio, A. Gomes de Sousa, C. de Souza Baptista, Towards a logical multidimensional model for spatial data warehousing and OLAP, ACM 9th International Workshop on Data Warehousing and OLAP (DOLAP), pp. 83-90, Arlington, Virginia, USA, 2006.

[Schneider 2004] M. Schneider, Computing the topological relationship of complex regions, 15th International Conference on Database and Expert Systems Applications (DEXA), pp. 844-853, Zaragoza, Spain, 2004.

[Scotch 2005] M. Scotch, B. Parmanto, SOVAT: Spatial OLAP Visualization and Analysis Tool, 38th Annual Hawaii International Conference on Systems Sciences (HICSS), pp. 1-7, Big Island (Hawaii), USA, 2005.

[Shekhar 2001] S. Shekhar, C. T. Lu, X. Tan, S. Chawla, Map cube: A visualization tool for spatial data warehouses, chapter in: H. J. Miller, J. Han (eds.), Geographic data mining and knowledge discovery, Taylor and Francis, USA, 2001, pp. 73-108.

[Spaccapietra 2008] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. Fernandes de Macêdo, F. Porto, C. Vangenot, A conceptual view on trajectories, Data & Knowledge Engineering 65(1), 2008, pp. 126-146.

[SpotCrime 2009] SpotCrime: The most comprehensive online source of crime information, date of access February 2009, available in:
http://www.spotcrime.com

[Timko 2005] I. Timko, C. E. Dyreson, T. B. Pedersen, Probabilistic data modeling and querying for location-based data warehouses, 17th International Conference on Scientific and Statistical Database Management (SSDBM), pp. 273-282, Santa Barbara, USA, 2005.

[Tomlin 1990] C. D. Tomlin, Geographic information systems and cartographic modeling, 1st Edn., Prentice Hall, London, 1990.

[Torgler 2007] B. Torgler, S. Schmidt, What shapes player performance in soccer? Empirical findings from a panel analysis, Applied Economics Taylor and Francis Journals 39(18), 2007, pp. 2355-2369.

[Torlone 2003] R. Torlone, Conceptual multidimensional models, chapter in: M. Rafanelli (ed.), Multidimensional databases: Problems and solutions, Idea Group, UK, 2003, pp. 69-90.

[Turner 2010] K. Turner, S. J. Bespalko, Spatial data technologies, 89th Transportation Research Board Meeting (TRB), No pages, Washington, USA, 2010.

[Vaisman 2004] A. A. Vaisman, A. O. Mendelzon, W. Ruaro, S. G. Cymerman, Supporting dimension updates in an OLAP server, Information Systems 29(2), 2004, pp. 165-185.

[Vassiliadis 1998] P. Vassiliadis, Modeling multidimensional databases, cubes and cube operations, 10th International Conference on Scientific and Statistical Database Management (SSDBM), pp. 53-62, Capri, Italy, 1998.

[Vazirgiannis 2001] M. Vazirgiannis, O. Wolfson, A spatiotemporal model and language for moving objects on road networks, 7th Symposium on Spatial and Temporal Databases SSTD, pp. 20-35, Redondo Beach, USA, 2001.

[Whitehorn 2005] M. Whitehorn, R. Zare, M. Pasumansky, Fast track to MDX, 2nd Edn., Springer, New York, 2005.

[Wrembel 2007] R. Wrembel, B. Bebel, Metadata management in a multiversion data warehouse, Journal of Data Semantics 8(1), 2007, pp. 118-157.

[Yang 1998] J. Yang, J. Widom, Maintaining temporal views over non-temporal information sources for data warehousing, 6th International Conference on Extending Database Technology (EDBT), pp. 389-403, Valencia, Spain, 1998.

[Yin 2000] S. Yin, L. Hui, Integration of web-based GIS and online analytical processing, 21st Asian Conference on Remote Sensing (ACRS), No pages, Taipei, Taiwan, 2000.