Università
Ca'Foscari
Venezia

Dottorato di ricerca
in Informatica
Scuola di dottorato in Scienze e Tecnologie
Ciclo XXIV
(A.A. 2010 - 2011)

A Framework for Trajectory Data Warehousing and Visual OLAP Analysis

SETTORE SCIENTIFICO DISCIPLINARE DI AFFERENZA: INF/01
Tesi di dottorato di Luca Leonardi, matricola 955609

Coordinatore del Dottorato	**Tutore del dottorando**
Prof. Antonino Salibra	**Dott.ssa Alessandra Raffaetà**

Università Ca' Foscari di Venezia

Dipartimento di Scienze Ambientali, Informatica e Statistica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

A Framework for Trajectory Data Warehousing and Visual OLAP Analysis

Luca Leonardi

Supervisor
Alessandra Raffaetá

PhD Coordinator
Antonino Salibra

February, 2012

Author's Web Page: http://www.dsi.unive.it/~leonardi

Author's e-mail: leonardi@dsi.unive.it

Author's address:

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino, 155
30172 Venezia Mestre – Italia
tel. +39 041 2348411
fax. +39 041 2348419
web: http://www.dsi.unive.it

To my grandmothers
Antonia and Giannina

Abstract

This thesis is aimed at designing a Trajectory Data Warehouse (TDW) model, having the ability to storing and analysing trajectories data. In particular a trajectory data warehouse is a data warehouse able to store aggregate information related to trajectories of moving objects, which also offers visual OLAP operations for data analysis.

The data warehouse model includes both a temporal and a spatial dimensions, while permits also for several other dimensions. This ensures a flexibility of the model that results to be general enough to deal with objects that are either completely free or constrained in their movements. In particular, the spatial dimension and the associated concept hierarchy reflect the structure of the environment in which the objects travel. The temporal dimension, on the other hand, reflects the passing of time, while the other dimensions describe features of the studied moving objects. The TDW allows one to analyse the behaviour of objects inside a given area as well as movements of objects between areas in the same neighbourhood. We investigate in depth some issues related to the computation of corresponding aggregates, which are useful for the efficient implementation of roll-up operations.

We insert our TDW in a more general framework that offers functionalities in order to reconstruct trajectories data starting from raw time-stamped locations received from external devices, and then use these reconstructed trajectories in order to feed-up the data warehouse. The framework also provides a visual interface for easily navigate aggregate measures obtained from OLAP queries at different granularities. The user can get overall views in time and in space of the measures, as well as a focused view on specific measures, spatial areas, or temporal intervals.

To highlight the usefulness of the entire framework we propose two different case studies. The first one applies the framework to some trajectory data related to cars moving in the city of Milan, while the second one applies the system to trajectories obtained by monitoring some fishing boats sailing on the Adriatic Sea. The mainly differences between the two cases are related to the type of moving objects under observation, the available information about the objects, and their movement constraints, i.e. the road network for cars and freely movements for boats.

Acknowledgments

Research can be defined as the search for knowledge, and there is no better ways of doing it than analysing new data. But as every other activity, the group you team with makes the difference on the results you can achieve. For this reason, my first thank goes to my advisor, *Alessandra Raffaetà*, who supported me during this years with her advices, corrections and encouragements when I got lost.

Special thanks go also to the other people of the team I took part, Salvatore Orlando, Alessandro Roncato and Claudio Silvestri (in a strictly alphabetical order!), who worked with me and helped finding solution to lots of issues. Thanks also to the reviewers of my thesis, for their valuable suggestions.

I want also to express my gratitude to all my Ph.D colleagues, that share with my the last three years. A special mention must be given to Matteo Zanioli, Matteo Centenaro, Gian-Luca Dei Rossi and Luca Rossi for all the good moments and the good advices given during these years. I hope guys that your knowledge on Milan road traffic will be useful to you, one day!

Finally, last but most important, a great thank goes to my family, who has always helped me and gives me their unconditional support during all my studies, without them nothing of this would have been possible.

Contents

List of Figures

1

Introduction

Physical objects have all in common an interesting characteristic. They have a position in space in every time instance, and this applies to countries, land parcels, rivers, glaciers, lakes, forests, to cite only some examples. However, for these types of objects the location in the space changes very slowly, or do not change at all, along the time. But beside them we can find other types of objects that change position continuously. Just to name few of them, we can have cars, taxis, boats, air planes, but also birds, bears, fishes, persons, and so on. An interesting fact related to these kinds of objects is that while they move around, they follow a path through space and time, that if stored permits to reconstruct their movements. Each of these paths is what it is called an object *trajectory*.

In the past few years the interest in location-aware services has constantly raised thanks to the increment of GPS-enabled devices available in the market, such as mobile phones, anti-thefts satellite systems, GPS car navigation systems, and RFID tags adoption, making this kind of technology available to a vast amount of people. These technologies allow for continuously tracking moving objects. If you think at cellular phones, their networks permit to track users with different precisions depending on the amount of GSM cells available in a certain zone (i.e. GSM network permits to detect the presence of a device in a certain cell, hence higher is the number of cells, higher is the precision of the positioning of the device that can be achieved through triangulation techniques). Car navigation systems can keep track of the movements of the devices by constantly storing the exact coordinates position of the device itself. Tracking devices, as for example GPS collars used on wild animals for monitoring purposes, can constantly register the positions of entire herds or single individuals, freely moving around open areas, forests and so on.

The analysis of such trajectories data raises opportunities for discovering behavioural mobility patterns that can be exploited in many innovative applications. Analysing GSM trajectories could help in better understanding human behaviours, or services accessibility. Tracking cars offers new methodologies for the study of traffic management and monitor. The use of tracking collars on animals can make it easier the study of migration routes or habits of some species. Monitoring boats sailing on the sea can be useful for control surveillance to prevent boats collision or to determine whether a boat has problems, i.e. engine failures.

An interesting aspect related to trajectories collected in a given scenario is that they usually have some similar behaviours. Take cars movements as an examples. Cars move on road networks, that are usually predefined, so either if every object has different paths, different starting points and different destinations, probably large parts of the trajectories of these objects behave in the same way. The same thing could be observed on animal movements, that even if they move on freely areas, usually follow migrating routes and move in large herds or following a leader. This characteristic means that trajectories data can somehow be aggregated together in order to obtain a more interesting comprehension of the phenomena concerning moving objects.

In this context, we argue that data warehousing technologies can play an important role in granting very fast, accurate and understandable responses to queries for analysing mobility data. Data warehouses [33, 10, 34, 35, 41, 42, 77], i.e. tools designed for handling large amount of data, are nowadays common tools in order to perform analysis over them. Data warehouses are used for analysing data by means of OLAP (On-Line Analytical Processing) [12, 41] tools which provide sophisticated features for aggregating, analysing, and comparing data, converting them into useful knowledge. These systems differ from traditional databases in the sense they are designed and tuned for answering complex queries rather than for high throughput of a mix of updating transactions, and they typically have a longer memory, i.e., they do not only contain the actual values (snapshot data) but also historical data. In data warehouses data are organized as a set of dimensions and fact tables. Dimensions contain the analysis axes, whether fact tables contain measures, i.e. the numerical attributes being analysed against the different dimensions. Thus, data are perceived as a cube, divided in cells, where each cell contains a measure or a set of measures of interest. Data warehouse dimensions are further organized into hierarchies that favour the data aggregation process. A hierarchy is divided into levels, each of which having a different granularity, going from coarser level, to most detailed one. The members of one level can be aggregated to form the members of the next higher level.

Reports generated by OLAP queries on data warehouses are usually in the form of tables, graphs, texts, numbers. However, trajectories data may not be easy to understand by using such representations. This kind of data is related to the territory where they have been collected, so their visualization over a map is not an optional requirement. Geographical Information Systems (GISs) [64, 79] have been extensively used in various application domains, ranging from ecological, economical and demographic analysis, to city planning. Typically a GIS maintains information in several thematic layers. In these systems, each layer is composed of purely spatial data, on the one hand, that is combined with classical alpha-numeric attributes on the other hand. Two main models are used for the representation of the spatial part of the information within a layer: *vector model* and *raster model*. In the vector model spatial information is represented in the form of geometries like points, lines, polylines and polygons, defined by some data structures. Non-spatial information

is associated with these geometries in the form of attributes. The raster model represents spatial data as pixels or cells, each one having an associated attribute or set of attributes. Usually these cells form a uniform grid in the plane, and for each of them the attribute value is associate with, e.g., a number or a colour, denoting a sample value of some computed function. Often raster layers and vector layers are used together in order to give different information.

1.1 Main Contributions of the Thesis

The main achievement of the thesis is the definition of a data warehouse model having the aim to manage spatio-temporal object observations, in the form of trajectories, for an efficiently storing and analysis of these data in order to help users in mining new knowledge.

The Trajectory Data Warehouse (TDW) model has been developed in order to be flexible and very adaptive to various scenarios (i.e. sailing boats, people moving, animals migrating, cars). The main dimensions of analysis in the context of moving objects are the spatial and the temporal one. More specifically, the spatial domain can be structured according to the application requirements, from simple sets of nested grids (like in [55, 50]), to regions with arbitrary shapes, which can also be used to model a road network. On the other hand, the temporal domain will represent the data at different time intervals, that can be set to be bigger or smaller depending on the user needs.

The Trajectory Data Warehouse is the core module of a general framework for handling trajectories data. The framework collects streams of spatio-temporal observations related to the position of moving objects. In general these observations are collected at different or irregular sampling rates, at different times, and data can be missing or can contain errors. In between time-stamps there is no knowledge of the movement of the entities, so the information must be inferred in some ways. This is the task of the first module of the framework, that will reconstruct trajectories data starting from the raw locations received by the system. These reconstructed trajectories will then be used in order to feed-up our TDW. During the ETL phase, reconstructed trajectories are aggregated together, and the measures are calculated for each cell of the spatio-temporal data cube.

In order to allow for OLAP operations on the Trajectory Data Warehouse, adequate aggregation functions need to be defined for the data warehouse measures over the defined dimensions. We provide a novel formalisation measures and aggregate functions. Then we use this formalism in order to prove some analytical properties of the measures itself. Among the other we introduce the measure *Visits*, counting the number of times each base cell of the data warehouse is visited by trajectories. This measure result to be important since give a good approximation of the *holistic* measure *Presence*, which counts the number of *distinct* trajectories occurring in a granule.

Measures are stored as TDW facts associated with base cells of our TDW model. A base cell, called base granule in the context of TDW, represents the base elements in the partition of the spatial, temporal and eventually other available dimensions domains, and can be aggregated along the available dimensions according to the associated hierarchies. These measures already represent aggregate information, as they report some significant features of the sets of trajectories crossing the granules, whereas single trajectory details, like object identifiers, are not kept in the TDW at all. There are, in general, good reasons for this design choice. Often individual data can be highly volatile and require huge memory space. More importantly, in some cases they cannot be stored due to legal or privacy issues, and anonymization might not suffice to guarantee privacy of tracked people [69]. In addition, for common spatio-temporal applications aggregate measures are typically much more relevant than information about individual moving objects [71].

Visual representation of data are essential for enabling a human analyst to understand the data, extract relevant information and derive knowledge, that is the aim of the work proposed in this thesis. It is generally recognized that visual displays facilitate effective perception and cognition [51], promote ideation [13] and support analytical thinking [72]. For these reasons, in order to offer suitable tools for OLAP analysis on the defined Trajectory Data Warehouse, we develop, as last module of the proposed framework, a visual OLAP interface, able to perform OLAP analysis, queries and operation in a visual manner. It allows for multidimensional and interactive analysis, and it permits to overcome the limits of the usual OLAP operations provided by traditional data warehouses. In fact, the ordinary data warehouse representation based on relational tables makes it very difficult for the user to grasp the relationships between areas in the same neighbourhood, the evolution in time of spatial measures, or the correlations of different values. We believe that, as for spatio-temporal data, visualisation is crucial: it can be seen simultaneously as the output and endproduct of a knowledge discovery cycle and as the starting point for further, interactive and visual, analysis.

To highlight the usefulness of the proposed framework, and in particular of the TDW and the visual OLAP tool, we present two case studies. The first use case consists of using our framework in order to analyse data related to cars moving in the road network of the city of Milan, in Italy. Data have been shared inside the GeoPKDD[1] European project, during which part of the research of this thesis has been done. The goal of the project was to develop theory, techniques and systems for geographic knowledge discovery, based on new privacy-preserving methods for extracting knowledge from large amounts of raw data referenced in space and time. This dataset contains information about a week of movements of a fleet of cars monitored by an insurance company, and has no other information on the users despite their spatio-temporal location. The second use case is based on a dataset containing information about fishing boats sailing in the Adriatic sea. The dataset

[1]Geographic Privacy-aware Knowledge Discovery and Delivery - http://www.geopkdd.eu/

has been given to us during a collaboration with some environmental scientists from Università Ca' Foscari Venezia, that were studying habits of the marine wildlife and fishermen. This dataset is far richer of information with respect to the first one, and gives us the possibility to instantiate our TDW model with different dimensions, other than the spatial and temporal ones. Using our tool the environmental scientists have been able to obtain meaningful information for their studies, while we had a confirmation of the flexibility of the proposed system.

1.2 Structure of the thesis

Chapter 2 describes some related work. It will be briefly presented some base concepts on spatio-temporal data and tools to manage them, as moving object databases and spatial data warehouses.

Chapter 3 introduces the general framework we proposed, and discusses some of the problems related to the trajectory reconstruction task. It will also present the ETL phase needed in order to store aggregated trajectory data into the trajectory data warehouse. We introduce in this chapter also two application scenarios to which the framework has been applied.

Chapter 4 describes the abstract TDW conceptual model. The chapter illustrates the spatio-temporal hierarchies, some stored measures and the associated aggregate functions. Moreover, in this chapter we also discuss the measure $\mathcal{V}$, and devise an algebraic aggregate function for this measure that is able to exactly answer roll-up queries, independently from the specific discretization and hierarchies of the spatio-temporal dimensions. Finally, we discuss the issues related to approximating measure *Presence*. The chapter provides propositions and formal proofs in order to prove the validity of the model and the soundness of the given aggregate functions.

Chapter 5 is devoted to the presentation of the visual framework we developed in order to permit visual analysis on the TDW and to the illustration of some of its visual functionalities.

Chapter 6 proposes two applicative scenarios in which we have tested our framework, illustrating the obtained practical results.

Finally Chapter 7 contains some concluding remarks and proposes some topics for future work.

2

Related work

A *data warehouses* (DW) is a repository of subject-oriented, integrated, and non-volatile information aimed at supporting knowledge workers (executives, managers, analysts) to make better and faster decisions [33]. Data warehouses contain a large amount of information, which is collected from a variety of independent sources and are often maintained separately from the operational databases. Traditionally operational databases are optimized for on-line analytical processing (OLTP), where consistency and recoverability are critical. Transactions typically access a small number of individual records based on primary key. Operational databases maintain current state information. In contrast, data warehouses maintain historical, summarized, and consolidated information, and are designed for on-line analytical processing (OLAP) [12]. The data in the warehouses are often modelled as a multidimensional space to facilitate the query engines for OLAP, where queries typically aggregate data across many dimensions in order to detect trends and anomalies [54]. These aggregated data are called *facts* and each of them consists on set of numeric measures that are the subject of analysis in a multidimensional data model. Each of the numeric measures is determined by a set of dimensions. In a census data warehouse, for example, the measure is population, and the dimensions of interest are age group, ethnicity, income type, time (year), location (census tract), and so on. Given N dimensions the measures can be aggregated in 2^N different ways. The SQL aggregate functions and the group-by operator only produce one out of 2^N aggregates at a time. A *data cube* [27] is an aggregate operator which computes all 2^N aggregates in one shot.

Spatial data warehouses (SDWs) contain geographic data in addition to non-spatial data. A major difference between conventional and spatial data warehouses lies in the visualization of the results. Conventional data warehouses OLAP results are often shown as summary tables or spread sheets of text and numbers, whereas in the case of spatial data warehouses the results may be albums of maps. It is not trivial to convert the alpha-numeric output of a data cube on spatial data warehouses into an organized collection of maps.

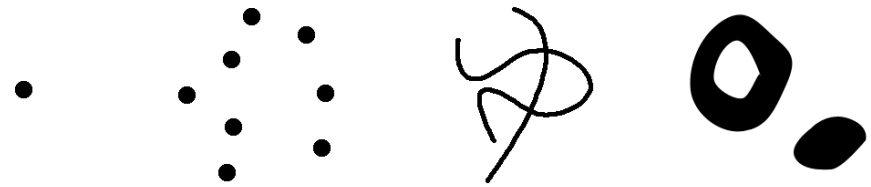
Another research field that has been heavily explored is that related to *moving object databases* (MODs), i.e. databases containing information about objects moving in the space. In the following some of the works that have been done so

far in these fields will be presented. By making this kind of data available in data warehouses we instead obtain *Trajectory data warehouses* (TDWs). TDWs are a relatively new research topic, and can be considered somehow an evolution of the well-studied spatial data warehouses.

2.1 Moving Object Databases

Research on MODs is a very active field and started more than a decade ago, based on previous research on spatial and temporal databases. There is a large amount of literature on this topic, starting with the works of Erwig [20] and Wolfson [78]. In the following years, the work of Güting et al. [30] defines an algebra suitable to representing and querying moving objects. In order to define such an algebra, the authors introduce a type system with some basic types as well as some type constructors. Basic types are the well known *int*, *real*, *string* and *bool*, whose domains are extended by the value $\perp$ with the meaning of undefined. The spatial types and temporal types introduced in the model are more interesting.

Basic conceptual spatial entities proposed are four different types, called *point*, *points*, *line*, and *region*. They are illustrated in Fig 2.1. Informally, these types have the following meaning. A value of type *point* represents a point in the Euclidean plane or is undefined. A *points* value is a finite set of points. A *line* value is a finite set of continuous curves in the plane. A *region* is a finite set of disjoint parts called faces, each of which may have holes. It is allowed that a face lies within a hole of another face. Each of the three set types, points, line and region, may be empty.



(a) A *point* value (b) A *points* value (c) A *line* value (d) A *region* value

Figure 2.1: Spatial data types defined in [30]

Time depending types defined by the authors are divided in two categories: a *time type*, called *instant*, representing a point in time, and a *temporal type*. The temporal types are derived from the base and spatial types by the constructor *moving*. This constructor, for each base or spatial types, yields a mapping from time to the starting type itself, generating what the authors call *moving types*, namely *mpoint*, *mpoints*, *mline*, *mregion* and so on. In other words, given for instance an object of type *line*, its moving type *mline* describes the development of that line over the time. These temporal types are functions, or infinite sets of pairs (instant,

value). In order to represent any single element of such a function, i.e., a single (instant, value)-pair, for example, to represent the result of a time-slice operation the authors propose the *intime* type constructor. This constructor converts a given type into a type that associates instants of time with values of the starting type.

On top of these abstract types, authors define the appropriate sets of operations in order to form a spatio-temporal algebra. The design of these operations adheres to three principles: they have to be designed as generic as possible; they have to achieve consistency between operations on non-temporal and temporal types; finally they have to capture the interesting phenomena. Defined operations are divided into two main classes, depending on the kind of types they refer to: non-temporal types (Figure 2.2) and temporal types (Figure 2.3) operations. In particular, among the other operators defined, the authors introduced the concept of *trajectory* of a moving object, as a sample of tuples in the form of (O_{id}, x, y, t), where O_{id} is the identifier of the moving object, (x, y) represents its position in the space and t is the timestamp of the observation, i.e. the time the object was in the given location. This sample represents a subset of all the positions visited by the moving object itself.

Class	**Operations**
Predicates	*isempty* $=, \neq$, *intersects*, *inside* $<, \leq, \geq, >$, *before* *touches*, *attached*, *overlaps*, *on_border*, *in_interior*
Set Operations	*intersection*, *union*, *minus* *crossings*, *touch_points*, *common_border*
Aggregation	*min*, *max*, *avg*, *center*, *single*
Numeric	*no_components*, *size*, *perimeter*, *duration*, *length*, *area*
Distance and Direction	*distance*, *direction*
Base Type Specific	*and*, *or*, *not*

Figure 2.2: Non-Temporal operations defined by Güting et al. in [30]

Class	**Operations**
Projection to Domain/Range	*deftime*, *rangevalues*, *locations*, *trajectory* *routes*, *traversed*, *inst*, *val*
Intersection with Domain/Range	*atinstant*, *atperiods*, *initial*, *final*, *present* *at*, *atmin*, *atmax*, *passes*
When	*when*
Rate of Change	*derivative*, *speed*, *turn*, *velocity*

Figure 2.3: Temporal operations defined by Güting et al. in [30]

In [29] Güting et al. present an example of MOD for unconstrained movements, based on the described algebra, on a platform database prototyping, called SEC-

ONDO. This prototype DBMS was developed at University of Hagen since about 1995. The main design goals were a clean extensible architecture and support for spatial and spatio-temporal applications in order to provide a "generic" database system frame that can be filled with implementations of various DBMS data models. SECONDO's architecture is shown in Figure 2.4. It consists of three major components: the kernel, the optimizer, and the GUI. The kernel does not implement a fixed data model but is open for implementation of a wide variety of DBMS data models, and it is extensible by algebra modules, describing a set of types and operators. The optimizer provides as its core capability conjunctive query optimization. Finally, the graphical user interface (GUI) is an extensible visualization tool that can be extended by viewers for new data types or models and provides a generic and rather sophisticated spatial database interface, for the visualization of spatial types and moving objects, including animation of these ones.

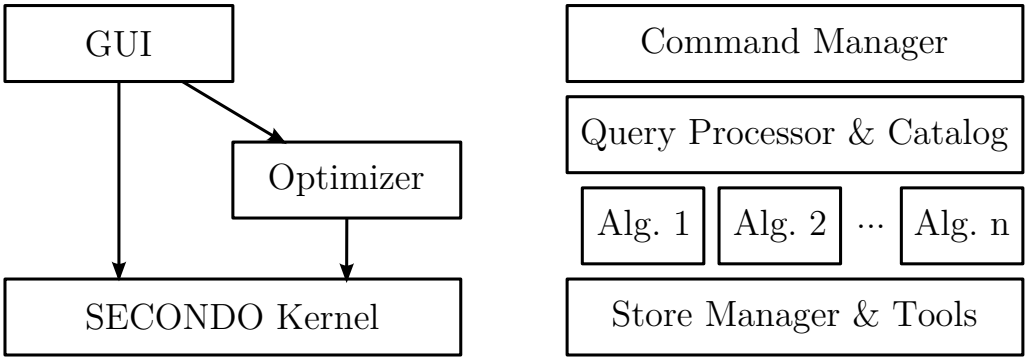


Figure 2.4: SECONDO components (left) and architecture of kernel system (right)

In [59, 60] Pelekis et al. present another example of MOD, referring to the case of unconstrained movement, based on a real world DBMS, the Hermes system. The system provides the functionalities needed for handling two-dimensional objects that change location, shape and size, through four kinds of data types: static base data types, static temporal data types, static spatial data types and moving data types. The objects belonging to the moving type are provided with a set of operations: topological and distance predicates, like *within_distance*; temporal functions, like *add_unit* (a function for adding a new unit of movement), and *at_instant* (a function that returns the union of the projection of a moving object at a time instant); distance and direction operators (for instance, the distance between two moving objects); set relationships (like intersection). Also, numeric operations on objects are supported, like area or length. Hermes supports four query types. A user can make queries on stationary objects, like point, range, distance-based, topological, and nearest neighbour queries. Or s/he can make queries on moving reference objects (distance-based and similarity queries). Finally Hermes permits join queries and queries involving unary operators (as travelled distance, speed and so on).

With a few year delay with respect to the corresponding results for unconstrained

movement, several works have presented data models to represent movement in constrained spaces, and in particular in networks. In [67] Speicys et al. present a hybrid model, combining a graph representation and a two-dimensional representation to allow the support of different kinds of queries regarding statical spatial objects. This work is focused on data modelling, and does not describe methods to compute query answers. On the other hand, in [57], Papadias et al. describe how to solve, for statical objects, nearest neighbours queries, range queries, closest-pairs queries, and e-Distance joins queries on spatial network databases. Finally, Güting et al. in [31] introduce both a data model and a comprehensive query language for network constrained moving objects, such as cars forced to move along roads.

The evolution of MOD was accompanied by an uninterrupted effort to improve efficiency and scalability through the use of indexes [62]. Spatial objects constrained to a network can be indexed more efficiently than free objects [61]. This is mainly due to the dimensional reduction entailed in restricting the 2D space to a set of line segments. Thanks to the dimensional reduction, the position of a moving object can be represented by a real number, and 3D spatio-temporal indexes can be replaced by traditional 2D indexes. A similar advantage can be observed when using symbolic positions instead of 2D ones, for example to represent the movement of the user revealed by proximity detectors in indoor spaces, as proposed by Jensen et al. in [37].

What happens in all the described approaches is that data are completely stored in the MODs. Despite the efficiency these tools can achieve by using indexes, either with [61, 37] or without [14] movement constraints, or techniques to reduce the size of data related to a trajectory while preserving error bounds [9], storing all moving object trajectories is simply unfeasible in case the amount of data to manage is unbounded. Even using aging methods combined with line simplification, as described in [9], would only postpone the exhaustion of storage. A possible solution to this kind of problems is to put off-line part of the historical data in order to make space for newer ones. However, this kind of solution makes it really difficult to use MODs in order to build up to them analytical applications for massive mobility datasets, such as the one generated by cellular phone tracks; indeed this kind of application needs to access the entire datasets, while an off-line policy forces the user to focus his/her attention only on a restricted timespan, e.g. making it really difficult to perform combined analysis between actual and historical data.

2.2 Spatial and Temporal Data Warehouses

Problems related to data storing requirements is really frequent in non-spatial contexts, and has been largely studied. In these contexts, it is common for analytical applications to make use of dedicated collection of subject-oriented, integrated, non-volatile, and time-variant data, i.e. data warehouses. In cases involving unbounded or very large amount of data, the original detail data are replaced by aggregated ones, at a granularity level that is a trade-off between analytical needs for pinpoint-

ing situations discovered at macro level and system resources. By the way, the efficient implementation of aggregate queries is a challenging task. Tao and Papadias in [71] propose a technique based on the combined use of specialised indexes and materialisation of aggregate measures. Choi et al. in [11] try to overcome the limitations of multi-tree structures by introducing a new index structure that combines the benefits of Quadtrees and Grid files. However, the above frameworks focus on calculating simple measures (e.g. count customers) and they do not cope with trajectories.

Han et al. in [32] extended the concept of data warehousing to spatial data and introduced the concept of *spatial data cube*. They propose a logical model based on a star-schema that allows the definition of different kinds of dimensions and measures. For the dimensions they identify three cases:

1. **Non-spatial** dimensions are dimensions containing only non-spatial data. For example one can have two dimensions, *temperature* and *precipitation*, which contain non-spatial data whose generalization are non-spatial, such as *hot* and *wet*.

2. **Spatial-to-spatial** dimensions are dimensions whose primitive levels and all of their high-level generalized data are spatial. For example the Italian map for administrative areas, such as municipalities, is represented by spatial data, and all of its generalized data, such as cities, regions, ..., are also spatial.

3. **Spatial-to-non-spatial** dimensions are dimensions whose primitive levels data are spatial but whose generalizations, starting at a certain high level, become non-spatial. For example, the European countries are associated with spatial data, but each country can be generalized to some non spatial value, such as 'Monarchy' in the case of a 'government type' level on the hierarchy.

On the other hand, as far as measures are concerned, authors distinguished two different cases, namely:

1. **Numerical** measures are measures containing only numerical data. For example, one measure in a spatial data warehouse could be *monthly revenue* of a region, and a roll-up may get the total revenue by year, by country, etc.

2. **Spatial** measures are measures which contain one or a collection of pointers to spatial objects. For example, during the generalization (roll-up) in a spatial data cube having *temperature* and *precipitation* mentioned before, the regions with the same range of temperature and precipitation will be grouped into the same cell, and the measure so formed contains a collection of pointers to those regions.

According to Gray et al. [27], numerical measures can be categorized into three classes, with respect to the complexity of the aggregate functions needed for computing the super-aggregates of the measures, starting from a set of already available sub-aggregates:

- *distributive*: measures using distributive functions are those for which super-aggregates can be computed by summing up the sub-aggregates at finer granularities. An example of this kind of measures could be the total revenue for a shop: given the revenue for every product, the total revenue is composed by simply summing up all the available revenues.

- *algebraic*: measures using algebraic functions are those for which super-aggregates can be computed from the sub-aggregates with a finite set of auxiliary measures. An example of this kind of measures are all those related to an average quantity. In this cases the average total could be computed by summing up the single quantities (first measure), and by dividing this value by the amount of the objects involved (second measure).

- *holistic*: measures using holistic functions are those for which super-aggregates cannot be computed from sub-aggregates, even if we employ auxiliary measures. An example of this kind of measure could be the amount of customers that have visited a shop. There is no possibilities to distinguish the same customer visiting the shop twice if only the total amount of customers is available for each time frame (i.e. you need to keep track of every customer id).

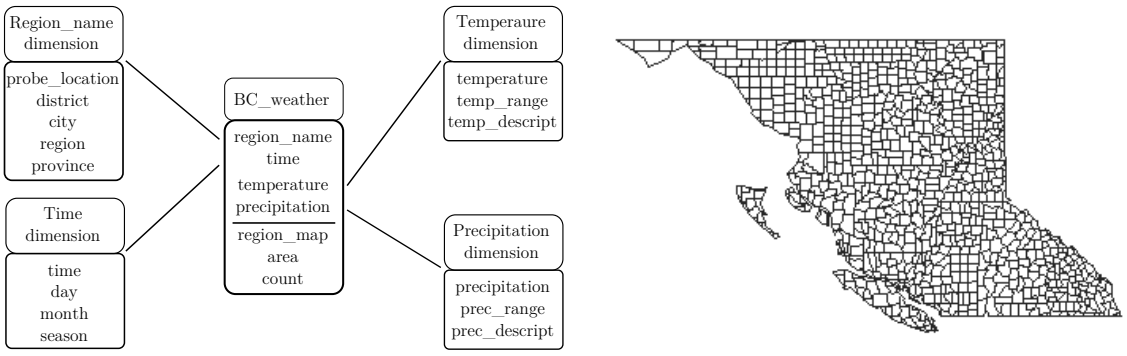


Figure 2.5: Star model of a spatial data warehouse [32]

Figure 2.5 represents an example of a star schema for a spatial data warehouse related to the analysis of the weather in the British Columbia, as well as the regions represented in the corresponding SDW. In the schema one can notice four different dimensions, *temperature*, *precipitation*, *time* and *region_name*, and three different measures. *Region_map* is a spatial measure which represents a collection of spatial pointers pointing to the corresponding regions; *area* is a numerical measure which represents the sum of the total areas of the corresponding spatial objects; *count* is a numerical measure which represents the total number of base regions accumulated in the corresponding cell. The authors present a method for the computation of the data cubes. Their goal is to balance the storage requirements and cost of aggregations at query-time by enhancing the performance of the most frequently accessed

queries, either directly by precomputing them, or indirectly, by precomputing an intermediate result shared among several queries. Pedersen et al. in [58] also focus on pre-aggregation in SDW. Both [32] and [58] present methods suitable only for aggregating facts whose measures are distributive over the aggregation operator. Malinowski et al. in [49] present a conceptual multidimensional data model for spatial data allowing for spatial dimensions, spatial hierarchies, and spatial measures.

In order to introduce the ability to cope with temporal evolution of dimension data, Mendelzon et al. in [52] introduce a model and developed a prototype and a Datalog-like query language, based on a temporal star schema. This model supports changes to the structure and/or the instances of the dimension tables by using the concept of transaction and valid time, respectively. Some structural changes also yield different fact table versions. Also, Eder et al. in [17] propose a data model for temporal OLAP supporting structural changes.

In 2001, Rivest et al. [65] introduced a paradigm aimed at exploring spatial data at different levels by using maps as normal data warehouses do with table and charts. They propose a set of features and operators such a system should have, but they do not present a formal model for it, although the concept and operators proposed in [65] have been implemented in a commercial tool called JMAP.

In [36] Jensen et al. present a conceptual model for moving objects with imprecise position allowing for dimensions with partial containment, an algebra (selection, union, aggregation), and a method to evaluate the effect of imprecision on aggregate query results. The proposed model is presented in the use case of location based services, where DW dimensions are space, time, and user, but can also be instantiated with different dimensions, hierarchies and measures. It is not possible, however, to explicitly account for constrained movement, for example due to the presence of a road network.

In [15] da Silva et al. introduced a framework able to offer an open and extensible system with the analysis capabilities available in both analytic and geographic processing tools. The idea under this framework was to classify dimensions in geographic and hybrid ones, depending on whether they represent only geographic and non-spatial data, respectively.

Timko et al. in [73] present a multidimensional model for the representation of data related to *location based services* (LBS) for vehicles moving along a road network. The proposed model is specific for LBS and allows to represent even the lower level detail of the road network, such as the parts of a crossroad that are actually crossed by cars, the lanes of a segment of road and the possible exchange of traffic among different elements (eg: if it is possible to change lane or to do a U-Turn). As the authors observe, their work is tailored to the requirement of LBS for objects moving on a network and thus it is not generic.

In [76] Wan et al. present an OLAP system for network-constrained moving objects. The proposed system is based on the efficient indexing of individual trajectories, and thus it is able to answer detail queries. On the other hand, this approach makes the system unsuitable for very large databases.

Liu et al. in [47] present a method to efficiently process large scale, real-time, traffic data and update aggregate summaries related to road segments. It should be observed, however, that the contribution of this work is mainly focused on the aggregation of raw data to feed a database, and not on the efficient aggregation of previously computed aggregates to answer complex user queries.

Finally, in [26], Gómez et al. present an extension of the Piet Spatial DW framework to deal with trajectory data. This framework, introduced by Gómez et al. in [25], makes use of overlay pre-computation for answering spatial queries (aggregate or not). Piet supports four kinds of queries: standard GIS queries, standard OLAP queries, geometric aggregation queries (like "total population in states with more than three airports"), and integrated GIS-OLAP queries ("total sales by product in cities crossed by a river", with the possibility of further navigating the results).

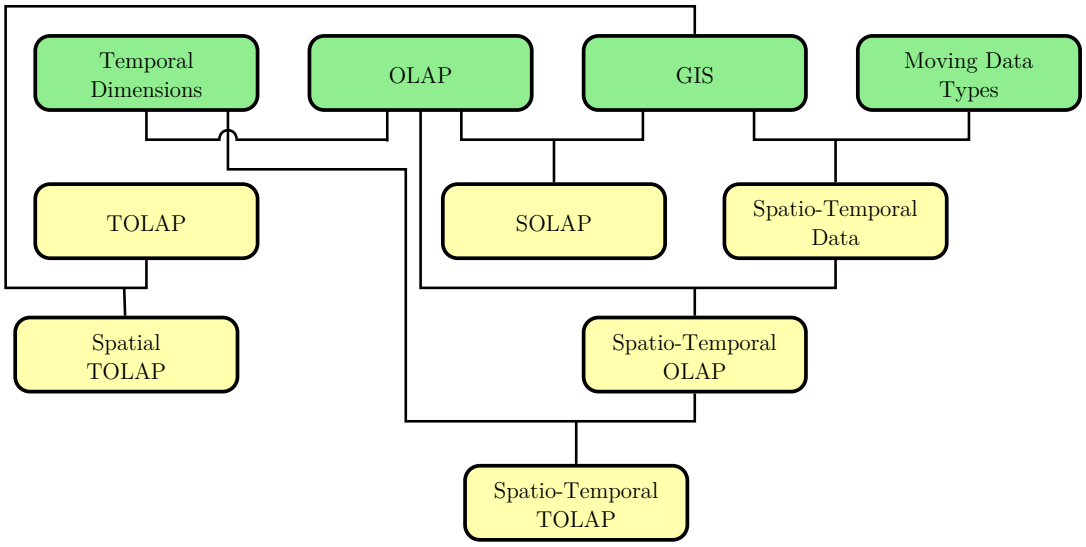


Figure 2.6: A taxonomy for spatio-temporal data warehousing [75]

Despite all the works presented so far, as for now, as stated in [75], there is still no commonly agreed definition of what a Spatio-Temporal Data Warehouse (STDW) is and what functionalities it should support. In [75] the authors proposed a conceptual framework for defining STDWs and a taxonomy, shown on Figure 2.6, for spatio-temporal OLAP queries, through which they classify the approaches in literature. The authors start by considering four basic classes: temporal dimensions, OLAP, GIS, and moving data types. As a derived basic class, adding moving data types to GIS produces *Spatio-Temporal Data*, typically allowing trajectory analysis in a geographic environment. Providing OLAP with the ability of handling temporal dimensions produces the concept of *Temporal OLAP* (TOLAP). The interaction of OLAP and GIS is denoted by *Spatial OLAP* (SOLAP). The interaction between GIS and TOLAP is called *Spatial TOLAP* (S-TOLAP). Adding OLAP capabilities to spatio-temporal data results in *Spatio-Temporal OLAP* (ST-OLAP). Finally, if

the latter supports temporal dimensions we have *Spatio-Temporal TOLAP* (ST-TOLAP).

As said, researches in this topic are still at an early stage, so there are not many works in this field. Some preliminary works can be found in [55, 50]. In [55], the authors propose a method in order to model and maintain a data warehouse for trajectories, and define a simple data cube consisting of spatial and temporal dimensions, and numeric measures concerning trajectories. In [50] the aim of the authors was to provide efficient solutions to support the whole process for warehousing, from trajectory reconstruction to trajectory-oriented OLAP analysis. However, both of these works do not deal with visualizations of the results.

In the taxonomy proposed in [75], the work by Orlando et al. [55] should be positioned in the ST-OLAP class, whose queries result to be the 2^{nd} more expressive ones, immediately after Spatio-Temporal TOLAP queries, that also allow for slowly changing dimensions [42].

2.3 Visual Analysis

In visualization and visual analytics, data aggregation is commonly used for dealing with large amounts of data. In particular, spatial, temporal, and categorical aggregations are used for spatio-temporal data, as proposed by Fredrikson et al. in [23]. In particular, the authors propose for the spatial aggregation, that the space should be divided into suitable compartments. The events that occurred in the same compartment will then be united in an aggregate. For the temporal aggregation, the time is divided into suitable intervals. The events that occurred during the same interval are put together. The attributive, or categorical, aggregation unites events characterized by the same or close values of analysis-relevant attributes. For numeric attributes, the closeness of values is defined by dividing the value ranges into intervals so that all values within an interval are considered to be close. These three basic types of aggregation can be used in various combinations.

To study the distribution of movement characteristics over space, movement data can be aggregated in different ways. For instance in [16] these kind of data are aggregated into continuous density surfaces, while in [22] and [3], spatial aggregations are done by means of discrete grids. On these techniques, temporal aggregation appears in the form of temporal histograms where the bars correspond to time intervals and their heights are proportional e.g. to the number of locations visited or the distance travelled by the objects. For spatial aggregation, the territory is divided into compartments by means of a regular grid. The results of aggregation, such as density counts, are represented by colouring or shading the grid cells on a map display. Analogously to densities, other aggregated characteristics can be computed and visualized. Spatio-temporal aggregation is done by the grid cells and consecutive time intervals. The results are shown on an animated map. Brillinger et al. in [7] aggregate movement data into a vector field using a regular grid: in each

grid cell, a vector (arrow) is built with the angle corresponding to the prevailing movement direction and length and width proportional to the average speed and the amount of movement, respectively, as shown in Figure 2.7. The picture shows movements of animals in two different moments of the day (6am and 12pm), in a certain area. The vector field aggregation allows a user to see that animals are more active early in the morning with respect to what happens at noon, as shown by longer arrows in the first image. Moreover, it can be noted that there are regions of the vector fields that converge towards some areas of attractions, eg in the area at the top left of the picture.

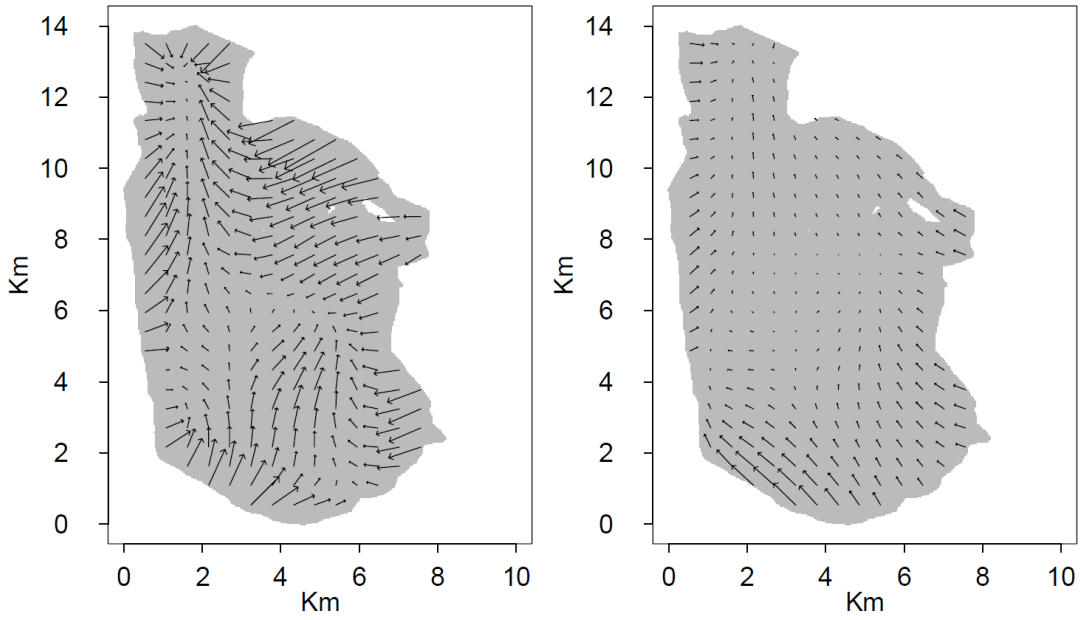


Figure 2.7: Examples of vector field aggregation showing animals movements at 6am (a) and 12pm (b)

To study links between places, movement data can be aggregated into origin-destination matrices [28] and flow maps [74, 5]. In the first case, the results are visualized as a transition matrix where the rows and columns correspond to the places and symbols in the cells or cell colouring or shading encode the derived attribute values. An example of this kind of visualization is shown in Figure 2.8. A disadvantage of such visualization is the lack of spatial context. In the second case, aggregate moves are visualized on a map by bands or arrows connecting pairs of locations. The widths of the bands or arrows are proportional to the volumes moved between these locations. An example of flow map is represented in Figure 2.9. Unfortunately, such a map may be illegible because of intersecting and overlapping symbols. In the case of spatio-temporal links, the time can again be divided into intervals. Aggregates are then built from moves having common origin, common

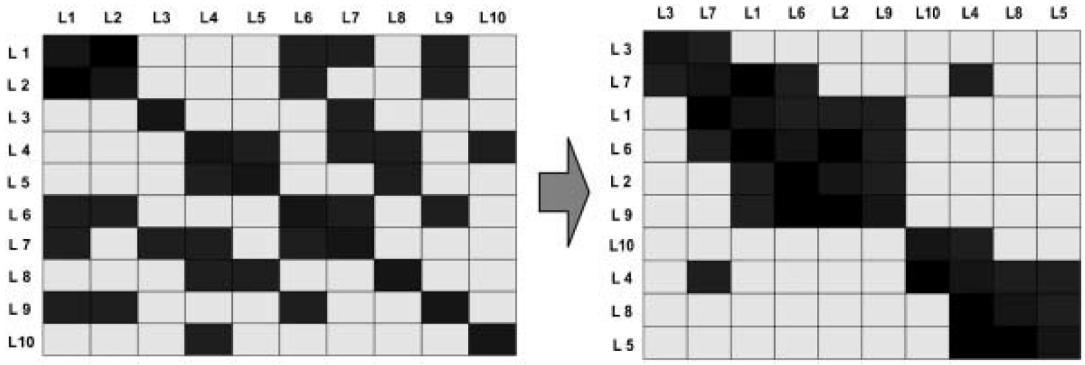


Figure 2.8: Example of an origin-destination matrix to visualize spatial interaction patterns among 10 locations. The two matrices show the same data, with dark colours representing strong interactions. The right matrix has columns and rows reordered according to interactions among locations [28]

destination, and common time interval when they occur(which means that the start and end time of each move lie within this interval). The results can then be represented by a sequence of transition matrices or flow maps, one matrix or map per time interval, or by an animation.

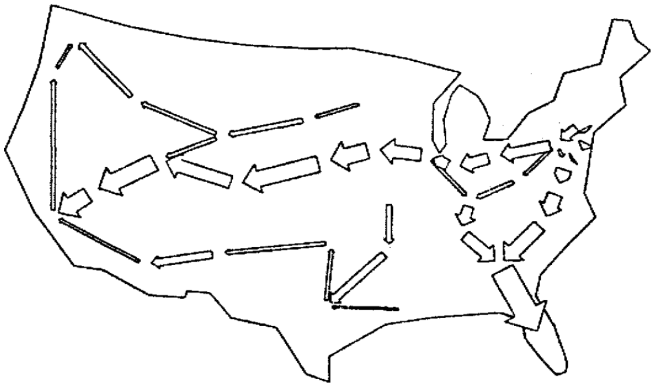


Figure 2.9: Example of a flow map for movement representation [74]

A more detailed survey of the aggregation methods used for movement data could be found in [3]. In particular, depending on the type of movement data analysed, the authors divide the aggregation methods in three different classes. Spatial, temporal, and attributive aggregations, and their combinations, could be applied to positions records treated as independent events. Links or temporal links aggregations could be applied to data that are treated as straight moves between predefined places while the actual paths are ignored. Finally route-based aggregation could be applied to trajectories with close and similar routes.

2.3.1 Visual OLAP

Visual OLAP is a clear trend in software for business visualization. The tools allow the user to explore data cubes through traditional visualisation techniques such as time series plots, scatterplots, maps, treemaps, cartograms, matrices etc., as well as more specialised visualisations. Polaris [68] and ADVIZOR are two pioneering systems in this direction. Polaris is a visual tool for multidimensional analysis developed at Stanford University. Currently, Tableau Software commercialises the pioneering Polaris work. ADVIZOR represents the commercialisation of 10 years of research in Bell Labs on interactive data visualisation and in-memory data management [19].

In [66] Shekhar et al. proposed Map Cube, an operator that inherits ideas from three different domains, namely data warehouse, visualization and GIS, as shown in Figure 2.10. Map cube is essentially a data cube with cartographic visualization of each dimension to generate an album of related maps for a dimension power-set hierarchy or a concept hierarchy or a mixed hierarchy. A map cube adds more capability to traditional GIS where maps are often independent. The data cube capability of roll-up, drill-down slicing and dicing gets combined with the map view. Hence, analysis and decision making processes based on spatial data warehouses can result easier and benefit.

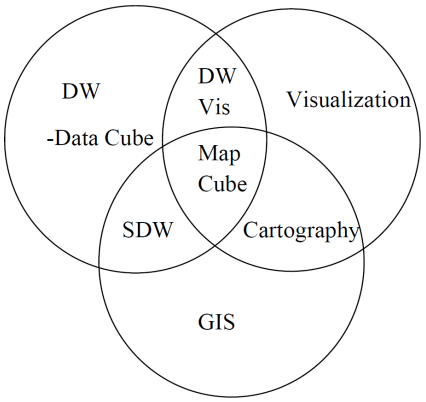


Figure 2.10: Map Cube relationship with three parent domains [66]

Visualization of streaming data is a challenging research topic in visual analytics, as pointed out in the recently published roadmap for the visual analytics research [39]. To address this problem, visual analytics requires support from the data management side: architectures for data stream management, stream-oriented query languages and operators, stream processing and efficient algorithms to keep an up-to-date online connection to the data sources. It is also necessary to design efficient algorithms for stream analysis, in particular, algorithms that are able to proceed in an incremental way and capture both trends and overall insights. Concerning the visualization of real-time data, the approach used in the existing prototypes is dynamic update of the display in response to changes of the data. Kim et al. [40] and Liu et al. [48] show only the most recent data. Krstajić et al. [43] represent data from a temporal window of a selected length. The application is news stream monitoring. It is interesting that the tool can not only represent individual news articles but also group articles by similarity and represent the groups in an aggregated way as threads. The grouping and aggregation is done on the fly as new articles come. Panopticon[1] is a commercial company suggesting data visualization tools for

[1] `http://www.panopticon.com/`

business analytics. The company claims that their tools are unique in their ability to handle true real-time streaming data feeds from message queues like SonicMQ and data services like Reuters. The displays change immediately as a change in data occurs. Real-time data may be combined with historical data. The company uses an in-memory OLAP data model for very fast data aggregation and manipulating data cubes.

3

System Architecture

The usage of location aware devices, such as mobile phones or GPS-enabled devices has exponentially increased during the last years, permitting to have access to vast volumes of spatio-temporal dataset. Since these dataset are referred to object moving in the space, they describe the development of these objects along time, i.e. they describe their trajectories, that informally are the path they follow in order to reach a certain point.

An interesting and important task is to transform this huge amount of data into valuable and meaningful new knowledge, that can be exploited in different contexts, from Location-Based Services (LBS) to traffic control management, or for advertisement purposes, etc.

In order to tackle with these challenging requirements, our proposal consists on a framework for trajectories management whose core module is a data warehouse able to handle trajectories data.

In this chapter we will present the general framework for designing a system that allows Visual OLAP analyses over trajectory data, as well as the modules of the framework needed for having a working Trajectory Data Warehouses, but that are not in the aims of this thesis. The framework can be customized according to different application scenarios. Two example scenarios will be discussed later in Section 3.1, in order to motivate the use of such a framework, and they will be used in all the thesis as explicative examples. The first scenario will be related to vessels sailing on the sea (Section 3.1.1), while the second one refer to cars moving along a road network (Section 3.1.2).

Section 3.2 will describe in details the various modules of the proposed architecture, and will underline data flow between them. In Section 3.3 we will discuss the issue of the trajectory reconstruction from a stream of spatio-temporal object observations. Such reconstructed trajectories will be used in order to load the aggregated data into the proposed TDW. This process, called ETL, i.e. extract-transformation-load phase, will be described in Section 3.4. The Trajectory Data Warehouse module, as well as the visual interface for querying it will instead be described and discussed respectively in Chapters 4 and 5.

3.1 Application scenarios

Before defining the model we propose for our TDW, we introduce two significant application scenarios, with the goal of highlighting issues and heterogeneous needs that may arise in different contexts. We also exploit the same scenarios as running examples, to make easier to understand some formal definitions, in this and following chapters. The first presented scenario is about freely moving objects, namely ships sailing on the sea. The peculiarity of the second one is the presence of network constraints on the object movements, such as cars that move along roads.

3.1.1 Vessels sailing on the sea

In this scenario we are interested in analysing the movements of vessels collected by the Vessel Monitoring System (VMS) or the Automatic Identification System (AIS). VMS and AIS are intended for specific uses, and have significant technical differences. By using both, however, each vessel periodically reports its current position.

Since 2005, the European Union legislation has required that all fishing vessels transmit vessel identification, date, time, position, course and speed hourly. VMS is mainly intended to monitor the movement of vessels with respect to restricted fishing areas. AIS transponders have been mandatory for large vessels since 2004 and an optional equipment for smaller ones and boats. Each transponder collects positions using a GPS receiver and send them to other ships, and to land base stations, using a VHF transmitter. By the way, these methods are intended as just two examples of technologies that could be used in order to track boats, and should not be considered as an exhaustive list.

Although the movement of vessels is not completely free, due to land presence, reduced water depth, or traffic constraints, ships can move freely in the open sea. We are interested in modelling and analysing such unconstrained free movements.

In order to reconstruct the trajectories of vessels starting from VMS or AIS data, i.e., to derive a global function of time to describe the whole trajectory, *local interpolation* can be used. According to this method, objects are assumed to move between the observed points following some rules. For instance, a *linear* interpolation function models a straight movement with constant speed, while other polynomial interpolations can represent smooth changes of direction.

Just to give a rough idea of a possible application, we could be interested in identifying the areas with higher risk of collision. To this end, we could divide the sea according to several regular grids with different resolutions. Then we could ask for queries of this kind *"Find the zones with the highest number of ships entering from different directions per hour"*, or *"Find the most visited zones during low visibility hours"*, or *"Find the zones with minimal distance between ships"*.

3.1.2 Cars moving along a road network

In this scenario we are interested in cars moving on a road network, which is modelled as a graph embedded in the Euclidean 2D-space. Positions are collected by on-board GPS devices at irregular rates. In this case the movement is completely constrained since cars are supposed to stay on the network. When reconstructing their trajectories, we thus take into account the topology of the road network to determine the path followed by each car between two consecutive GPS positions (see e.g., [6]). The reconstruction phase produces a sequence of lines in a $3D$ space ($\mathbb{T} \times \mathbb{R}^2$), each representing the continuous "development" of the moving object during a time interval. Notice that the spatial projection of these lines can be edges of the road network or portions of these edges.

We assume that the user in this scenario is a city manager that exploits the TDW to analyse the traffic. Some possible queries are:

- *"Which is the number of buses per hour in the morning of a given day in the neighbourhoods of a given district? Show its temporal evolution using a temporal granularity of half an hour"*
- *"From which district does a great number of cars leave in the morning? And at what hour? Is there a flow exiting/entering the town? Which are the main differences in the traffic between the working days and the week-end?"*
- *"What is the total distance covered by car vehicles per hour in the evening of a given day in the neighbourhoods of a given district?"*.

The same queries could thus be repeated by referring road segments included in the district.

3.1.3 Common features and main differences

In general, we want to model the temporal evolution of the position of objects. To be more specific, we need to discriminate different cases, depending on their peculiarities. For example the representation of the position of an object, the set of valid positions, and the constraints on transitions between positions.

Even if we are dealing with similar problems, these distinctions strongly influence the suitability of design choices. For each object there exists a 'true', and usually unknown, trajectory. On the other hand, it is common to have some partial knowledge of trajectories, for example a set of known positions at given times (sampling of the trajectories), and those positions could be either accurate, imperfect [80] or symbolic [38]. In some cases any element of the position domain is valid (free movement), in other cases some positions are not valid for given classes of objects or cannot be consider as valid since the new position is not reachable from the previous one (constrained movement).

Observed position. The *observed position* of an object could be either *continuous* or *discrete*. In the second case the position could be *discretized* or *symbolic* (for example the user is connected to a specific WiFi hotspot and the hotspot identifier is used to indicate the position). In both cases the position could be *free* or *constrained* to parts of the space. Indeed, part of the space could be unreachable for specific users or class of users. For example, cars usually move along roads. Depending on the analytical context, the observed position may be not the same that we are interested in during the analysis. For example, in the case of vehicles constrained along a road network, we may be interested in the identifier of the segment of road containing the vehicle, even if the original position was continuous (and constrained to the road network).

Movement between positions. The transition from one position to another could be completely free or constrained. A common kinds of constraint are the adjacency constraints, that restrict the user to move from a location to other locations that are spatially or topologically adjacent.

In the case of free transitions, any sequence of positions is valid, also those containing non spatially adjacent consecutive positions. Further, each scenario has specific analysis requirements that we can classify according to characteristics of dimensions and measures.

3.2 Trajectory Data Warehouse General Framework

The abstract framework architecture does not depend on the characteristics of the scenario we are observing and we want to implement. But obviously the realization of each module composing the framework hinges on the peculiarities of this scenario.

The general framework architecture and the data flow between the modules are shown in Figure 3.1. A description of these modules can be given as follows.

A stream of object observations, each denoted as $(obj_{id}, time, geometry)$, arrives at the system from external sources (i.e. GPS enable devices). Each observation records an object position at a given time and space, where the spatial type of *geometry* depends on the scenario. In our two described scenarios, *geometry* is usually a 2D point, whereas in other cases it could be a polygon defining a closed space, as for example a room, or maybe a 3D point, and so on. The received observations are then passed to a *trajectory reconstruction module* in order to be processed. The trajectory for each moving object, modelled as a continuous function of time, is approximated by the known observations, by taking into account possible movement constraints, such as roads in the traffic scenario or possible isles or lands in the boat one. During this phase, the possible errors on the data, like misplaced points or others problems due to various reasons, as for example GPS or network problems, try to be fixed; for instance, if cars are supposed to move on the road and

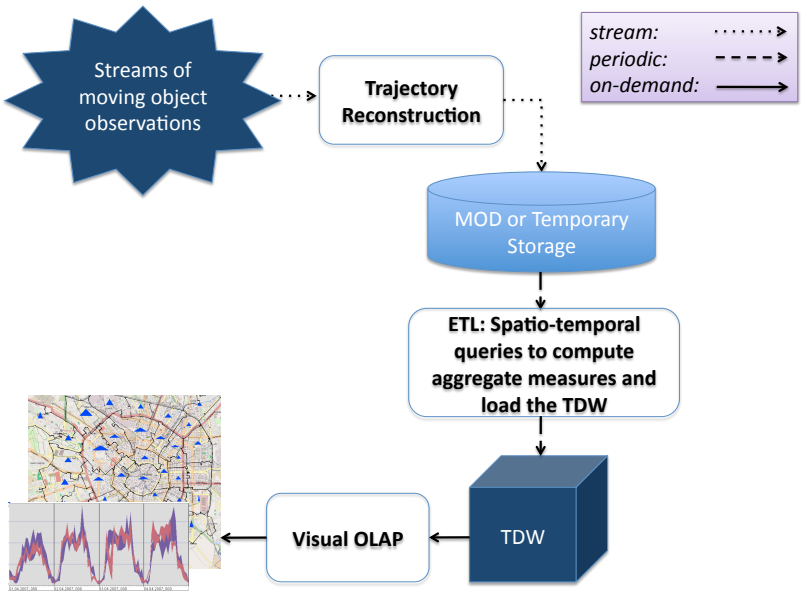


Figure 3.1: Framework architecture and data flow.

the position of a car is $20m$ away from the closest road, then its position is fixed to satisfy the constraint. Note that this procedure cannot be considered as faultless: for instance, if the car position results to be at the same distance from two roads, one of them must be selected. The output of the trajectory reconstruction module is usually a stream of positions such that for each pair of consecutive positions related to the same object, the intermediate positions can be safely computed by applying linear interpolation to the pair (i.e., there is no significant change in movement speed or direction). Section 3.3 will discuss about this topic. After the phase of trajectory reconstruction, the obtained trajectories can be used in order to extracting aggregate measures to load into the data warehouse fact table. This process is performed by the *Extract-Transformation-Load (ETL) module*, and will be discussed in Section 3.4. The natural choice in order to perform this task is the use of a Moving Object Database in order to calculate the needed aggregate measures that have to be store in the *Trajectory Data Warehouse (TDW)*. This is the main component of our framework, and its aim is to allow for OLAP operations over aggregated trajectory data. Its multidimensional model, with spatio-temporal dimensions and specific spatial hierarchies for different application scenarios, the stored measures and the associated aggregation functions, are discussed in detail in Chapter 4. Once the data are available in such a data warehouse, a sensitive task is to make them available to the users that need them. Visual representation of such a kind of data could be considered essential for enabling a human analyst to understand them, extract relevant information and derive knowledge. Hence, in order to query and access the data stored in the TDW, the framework provide the user with a *Visual OLAP*

module, based on the Visual Analytics Toolkit, which actually performs on-demand OLAP queries over the spatio-temporal dimensions. It is based on an interactive toolkit for Geographical Information Systems (GISs), and permits geo-referenced and animated views of aggregate trajectory data over a map. The functionalities of this module will be presented in Chapter 5 and will be demonstrated in Chapter 6 .

3.3 From raw data to trajectories: trajectory reconstruction

As we mentioned before, starting from a variety of location aware devices, a stream of observations related to the position of such devices is generated and arrives to the framework we presented in the previous section. Each time-stamped location is labelled with a unique identifier, associated to the moving object the observation refers to. Hence, the first step needed in order to obtain meaningful information from this stream is to reconstruct the path each object has followed, i.e. their trajectory.

From an abstract point of view, a trajectory of an object is a *continuous function* $\tau_{id} : I_{id} \rightarrow \mathbb{R}^2$, where id is a trajectory identifier and $I_{id} = [t_{id}^{first}, t_{id}^{last}]$ is the interval of definition of the trajectory. It is worth remarking that even if we are referring to objects that move in a 2D spatial domain ($\mathbb{R}^2$), we can easily generalise to an abstract spatial domain $\mathbb{S}$ without any conceptual complication.

In real-world applications, object movements are collected through a finite set of *observations*, i.e., a finite subset of points either taken from the actual continuous trajectory or reasonably close to it (the GPS or VMS data in the two presented scenarios). Such a finite set of observations is called *trajectory sample*, and is defined as follow.

Definition 3.1 *(Trajectory Sample)*
Given a set of time-stamped position in the space, in the form of (t, x, y)*, related to the movements of a moving object, then*

$$\mathsf{TS}_{id} = \{(t_{id}^0, x_{id}^0, y_{id}^0), (t_{id}^1, x_{id}^1, y_{id}^1), \ldots, (t_{id}^N, x_{id}^N, y_{id}^N)\}$$

with $t_{id}^0 < \ldots < t_{id}^N$, $t_{id}^0 \geq t_{id}^{first}$ *and* $t_{id}^N \leq t_{id}^{last}$.

Transforming the sampled trajectory TS_{id} into a continuous function T_{id} which approximates τ_{id} can be difficult due to several types of uncertainty, such as measurement errors, or unknown trajectory behaviour between consecutive observations. In some cases, e.g. in scenarios like vehicles moving along a road network, the knowledge about the movement constraints can be exploited in order to reconstruct with more accuracy the trajectory of the object.

Having in mind that raw points arrive in bulk sets, there is the need to decide whether a new point is part of an already existing trajectory, or if it is part of a new one. Depending on the application scenario, this decision can be taken in different

ways. In [50] the authors propose some generic trajectory reconstruction parameters that can be set in order to accomplish this task. In particular, the five parameters proposed are defined as follow.

- *Temporal gap* between trajectories (gap_{time}): the maximum allowed time interval between two consecutive time-stamped positions of the same trajectory for a single moving object. Any time-stamped position of object o_i, received after more than gap_{time} units from its last recorded position, will cause a new trajectory of the same object to be created. An example for this parameter is depicted in Figure 3.2, where the temporal distance between the last point of trajectory t_1 and the first point of trajectory t_2 is bigger than the selected (gap_{time}).

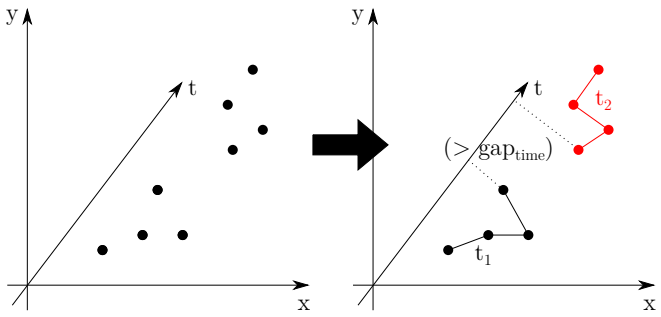


Figure 3.2: Temporal gap parameter example

- *Spatial gap* between trajectories (gap_{space}): the maximum allowed Euclidean distance in 2D plane between two consecutive time-stamped positions of the same trajectory. Any time-stamped position of object o_i, with distance from the last recorded position of this object greater than gap_{space}, will cause a new trajectory to be created for o_i (Figure 3.3).

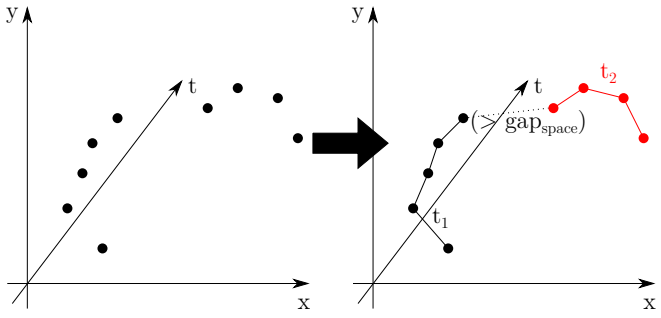


Figure 3.3: Spatial gap parameter example

- *Maximum speed* (V_{max}): the maximum allowed speed of a moving object. It is used in order to determine whether a reported time-stamped position must

be considered as noise and consequently discarded from the output trajectory. When a new time-stamped location of object o_i is received, it is checked with respect to the last known position of that object, and the corresponding speed is calculated. If it exceeds V_{max}, this location is considered as noise and (temporarily) it is not considered in the trajectory reconstruction process (however, it is kept separately as it may turn out to be useful again - see the parameter that follows). Figure 3.4 illustrates an example for the (V_{max}) parameter application.

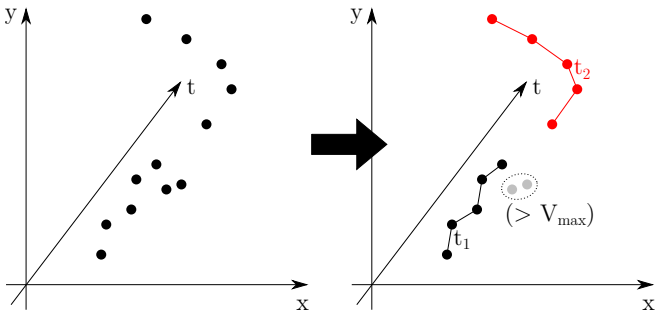


Figure 3.4: Maximum speed parameter example

- *Maximum noise duration* ($noise_{max}$): the maximum duration of a noisy part of a trajectory. Any sequence of noisy time-stamped positions of the same object will result in a new trajectory if its duration exceeds $noise_{max}$. For example, consider an application recording positions of pedestrians where the maximum speed set for a pedestrian is $V_{max} = 3\ m/sec$. When s/he picks up a means of transportation (e.g., a bus), the recorded instant speed will exceed V_{max}, flagging the positions on the bus as noise. The maximum noise length parameter stands for supporting this scenario: when the duration of this sequence of "noise" exceeds $noise_{max}$, a new trajectory containing all these positions is created, as for trajectory t_3 in Figure 3.5.

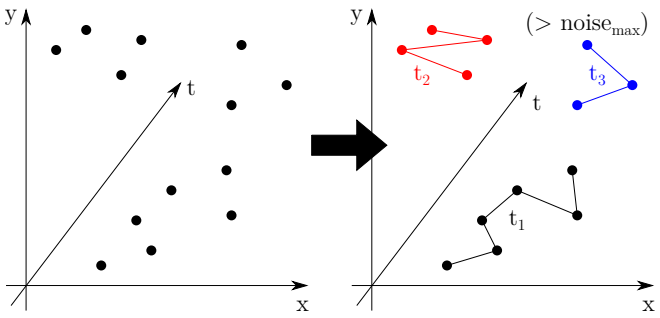


Figure 3.5: Maximum noise duration parameter example

- *Tolerance distance* (D_{tol}): the tolerance of the transmitted time-stamped positions. In other words, it is the maximum distance between two consecutive time-stamped positions of the same object in order for the object to be considered as stationary. When a new time-stamped location of object o_i is received, it is checked with respect to the last known position of that object, and if the distance of the two locations is smaller than D_{tol}, it is considered redundant and consequently discarded. This situation is depicted in Figure 3.6.

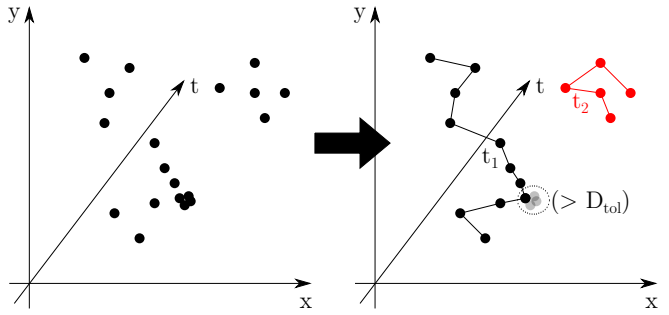


Figure 3.6: Tolerance distance parameter example

The algorithm that utilizes the aforementioned parameters is thoroughly presented and evaluated in [50]. It expects as input a set of observations, and a list containing the partial trajectories processed by the trajectory reconstruction manager; these partial trajectories are composed by several of the most recent trajectory points, depending on the values of the algorithm parameters.

As a first step, from each observation the algorithm extracts the object identifier and checks whether the object has been processed so far. If so, it retrieves its partial trajectory from the corresponding list, while, in the opposite case, creates a new trajectory and adds it to the list. Then, it compares the incoming point with the tail of the partial trajectory by applying the above mentioned trajectory reconstruction parameters. In this way, the algorithm decides if the incoming point can be considered as part of an existing trajectory or if a new one has to be created.

A second problem to solve in order to obtain reconstructed trajectories starting from raw data is related to the fact that, given two points of a same trajectory, there is no information about the movement of the object between them. In this case an interpolation method is needed in order to have a complete view of the object path. There are different possibilities depending on the amount of information available for the dataset and on the application scenario object of the analysis. In the following two of the most used method will be briefly presented.

Reconstruction in a freely movement scenario. The first presented method is the *linear (local) interpolation* reconstruction method. It is a quite standard method (see for example [62]), that can be used in almost every situation, and that does

not need of any further information about the scenario or for the dataset. Given two consecutive points of a trajectory, the linear interpolation method assumes a constant movement of the object between them, by considering a straight line joining the two points, covered with a constant speed. This assumption can be done in every scenarios where objects move in a freely context, and the approach result to be a good trade-off between flexibility and easiness of fulfilment.

In order to make more precise approximations of the movement, the linear interpolation can be substituted with others more general solutions. In particular, linear interpolation makes use of the last point of a trajectory in order to determine its path till the new arrived point, by applying a linear function between the two. If more points of the trajectory are taken into account, a *polynomial interpolation* or a *spline interpolation* can be applied. In the first case, instead of a linear function a higher degree function is used in order to estimate the object path, finding out a function going through all the available points. On the other hand, spline interpolation uses low-degree polynomials in each of the available intervals, and chooses the polynomial pieces such that they fit smoothly together, in order to construct a complete function for all the available points. However it is worth remarking that higher is the degree of the used polynomials, more expensive is their computation, and this usually makes the choice of the linear interpolation a better trade-off, in particular for the cases where the available trajectories sample is quite dense.

Reconstruction in a constrained scenario. In a constrained scenario, usually objects moves along a network, as could be the road network o a railway. In these cases a map matching algorithm (i.e. [6]) could be used. The network can be modelled as a graph $RN = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Each edge corresponds to a network segment connecting vertex pairs. The network RN is embedded in the Euclidean 2D-space $\mathbb{R}^2$ as follows: $V = \{(x_i, y_i) \in \mathbb{R}^2 \mid i \in \{1, \ldots, M\}\}$ is a set of points, while each edge $(V_i, V_j) = ((x_i, y_i), (x_j, y_j))$ is embedded as a straight line between the two vertices.

Given a time instant $t \in I_{id}$, $\tau_{id}(t)$ is the position (x, y) at time t of the object associated with trajectory id. Such position can be a vertex of RN or a point in an edge of RN. Note that, even if the observed position could be not exactly on a segment of the network due to measurement errors, the reconstructed trajectory should satisfy the network constraint. We observe that the spatial projection of a trajectory is a *path* in RN, i.e., a piecewise linear curve composed of a connected series of line segments, which are (portions of) RN edges.

Transforming the sampled trajectory TS_{id} into a continuous function T_{id} which approximates τ_{id} can be difficult due to several types of uncertainty:

1. The spatial coordinates of some observations may not belong to RN. This can be due to measurements errors which may affect trajectory observations or to the fact that in the concrete network roads are not exactly straight lines (e.g., they have a width).

2. The behaviour of a trajectory between two consecutive trajectory observations is unknown and must be induced.

   (a) There could be several paths in RN connecting two consecutive observations, especially if the sampled observations are loose and irregular.

   (b) Information like the speed of the associated moving object must be guessed.

In order to solve these problems, different methods can be applied. By assuming a simple reconstruction method, point (1) could be taken into account by a data normalization, that moves the spatial coordinates of each observation on the closest $\mathbb{R}^2$ point of RN. Note that in some cases the closest point cannot be determined, since for example a point is equally distant from two or more edges of the graph. In this case a heuristic must be applied in order to select the closest point: a probabilistic algorithm could be used in order to select the most probable network segment given the previous point or a set of previous points, or a buffer can be used in order to change the selected points as new points arrive, so that all the constraints are always verified. Point (2.a) can be solved in a somehow similar way: the Top-k fastest paths (for reasonably small values of k) can be computed, assigning to them a decreasing probability for increasing path costs. Then one of the paths can be randomly selected according to the ranking based probability. An easier solution, in this case, could be of simply connect consecutive observations along the fastest path, as it happens in standard car navigators. Finally, for point (2.b), a common solution is to assume a uniform speed between two consecutive trajectory observations, if no other information are available.

The reconstruction results in a continuous function T_{id} defined on the time interval $I_{id} = [t^0_{id}, t^N_{id}]$ where given a time instant $t \in I_{id}$, $\mathsf{T}_{id}(t) \in RN$. This corresponds to determine a sequence of lines in the $3D$ space ($\mathbb{T} \times \mathbb{R}^2$), each representing the continuous "development" of the moving object during a time interval. Notice that the spatial projection of these lines can be edges of the network or portions of these edges.

Note that when consecutive observed positions are contained into disjoint segments, the reconstruction process will also compute a route between the two points and all the segment that are traversed will contribute to the generation of inferred positions. This also means that the points dataset will be enriched with at least one point for each traversed segment, associated with the estimated time the moving object enter in it.

Example 3.1 *Figure 3.7 shows a part of a user trajectory, depicted as a solid black curved line, moving on a road network. The movement of the user is supposed to be constrained to the roads (in white), whereas grey areas are unreachable. Each point in the line represents the position $\tau_{id}(t)$ of the user at some time instant t. The diamonds along the line corresponds to observations, i.e. known positions of the users at specific time instants. The time t of each observation $(t, \tau_{id}(t))$ is reported*

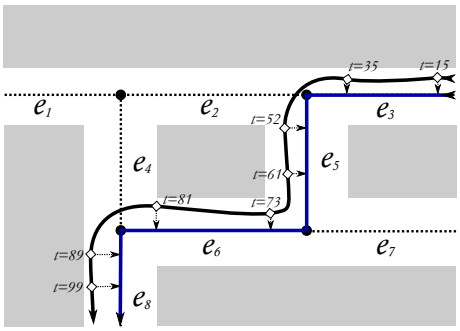


Figure 3.7: Reconstruction of a trajectory with network constraints

close to the corresponding diamond. At the centre of each road, the dashed line and the small circles represent the embedded edges (labelled $e_1, \ldots, e_8$*) and vertices of the RN graph that models the road network.*

Finally, the reconstructed *trajectory* T_{id} *of the trajectory* τ_{id} *is the solid (blue, in the coloured version) polyline in the centre of the route. The arrows from diamonds to* T_{id} *connect observations* $(t, \tau_{id}(t))$ *to the corresponding points* $(t, \mathsf{T}_{id}(t))$ *on the reconstructed trajectory.*

It is worth remarking that the reconstruction parameters proposed before could be applied independently of the interpolation algorithm applied to the data. In particular each parameter should be used according to the application scenario (i.e. gap_{space} should consider the distance between two points on the network, if the case, or the linear distance if linear interpolation is used).

3.4 ETL

Once trajectories have been constructed, they need to be temporary or permanently stored, in order to allow for the Extract-Transformation-Load (ETL) process to be computed and the Trajectory Data Warehouse to be fed-up.

The task of the ETL phase is in fact to periodically load the trajectory data warehouse dimensions and fact table with the correctly computed selected measures. In particular, having a stream of trajectories arriving constantly at the framework, this operation needs to be repeated in different times, in order to update the data in the data warehouse with the new ones. As for now, we will threat a TDW like a standard data warehouse with some predefined measures and with a spatial and a temporal dimensions that will be divided in different partitions. Informally, we will call each of these spatio-temporal partitions a *base granule*. Having to model fact related to moving objects in every base granule, some measures need to be defined.

As an example, some interesting measures that can be taken into account in a data warehouse modelling fact related to moving objects could be the total distance

covered by every object in a granule, the sum of the time spent in there, their average speed. An in depth presentation of our proposed Trajectory Data Warehouse model will be made in Chapter 4.

In order to calculate the measures to be inserted on the data warehouse data cube, the portions of the trajectories that fit into the base granules need to be extracted. Depending on the complexity of the pre-computation to be performed on the buffered trajectories, either a Moving Object Database or a dedicated small-footprint software could be used to perform this task.

ETL using a MOD. A Moving Object Database (MOD) (i.e. Hermes [59] or Secondo [29]), as described in Chapter 2, is a special database able to handle trajectories data and to perform spatio-temporal queries on them. In order to feed-up our data cube with aggregate information, reconstructed trajectories could be store in a MOD, and then its capabilities can be used. This choice is made in order to make the loading phase of the data warehouse as easier as possible. Considering trajectories related to a fleet of cars, Figure 3.8 illustrates a typical schema that can be considered as a minimum requirement for such a MOD.

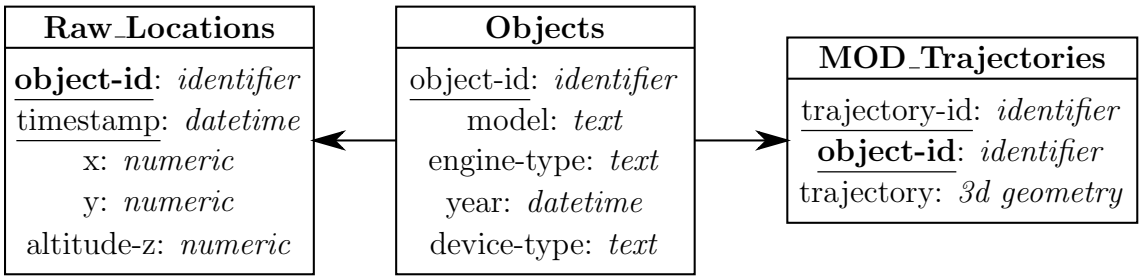


Figure 3.8: An example of a MOD schema

The MOD is composed by three tables. The first one, *Objects*, includes a unique object identifier (*id*), vehicle information (e.g. *car model, engine type, registration year*) as well as device-related technographic information (e.g. GPS type). The second table, *Raw_locations*, stores object locations at various time stamps (i.e., observations). This table has the aim to store the punctual yet not processed points received by the framework from the externa sources. Finally, the *Mod_trajectories* maintains the trajectories of the objects, after the application of the trajectory reconstruction process. Formally, let $D = \{T_1, T_2, \ldots, T_N\}$ be a collection of trajectories of a set of moving objects stored in the MOD. Assuming linear interpolation between consecutive observations the trajectory $T_i = \langle (x_{i_1}, y_{i_1}, t_{i_1}), \ldots, (x_{i_{n_i}}, y_{i_{n_i}}, t_{i_{n_i}}) \rangle$ consists of a sequence of n_i line segments in a $3D$ space, where each segment represents the continuous "development" of the corresponding moving object between consecutive locations (x_{i_j}, y_{i_j}) sampled at time t_{i_j}). Projecting T_i on the spatial $2D$ plane (temporal $1D$ line), we get the *route* r_i (the *lifespan* l_i, respectively) of the trajectory. Additional motion parameters can be derived, including the traversed length

len of route r_i, average speed, acceleration, etc. Note that the assumption of linear interpolation in the MOD trajectories could be done since, in case of reconstruction over a network, the reconstruction module will enrich the point dataset with additional observations for each segment of the network, as said before. Otherwise, a MOD with "network-aware" spatio-temporal operators should be used.

Given a MOD, in order to feed-up the Trajectory Data Warehouse, the measure associated to the base granule can be computed by finding out the trajectories portions that lie inside each base granule, using the MOD spatio-temporal capabilities, and then calculating the needed values starting from these partial data. This task can be computed in two different ways [50]. The first approach is called *cell-oriented* (COA). In this case for each available base granule, a spatio-temporal query is made to the MOD, in order to find out which trajectories cross that granule. The MOD will returns as the query result the trajectories identifier as well as the trajectories portions laying in the granule. The second approach focuses more on the reconstructed trajectories, and is called *trajectory-oriented approach* (TOA). In this case, for each trajectory the granules in which it resides in are discovered. In order to avoid checking all the granules, the minimum bounding box of the trajectory could be used to filter out the (possible) not intersecting ones. Finally the measures are computed for each granule incrementally. Further details about the two approaches, as well as a comparison study, can be found in [50].

ETL with a specialized software. Using a MOD makes it really straightforward to perform the ETL phase in order to feed-up the Trajectory Data Warehouse with the needed values. However, in some situations the MOD could be substituted by a specialized software that periodically performs the task of loading the data into the data warehouse, as proposed in [55]. In particular if there is no need to permanently store the raw data, the use of a complex tool like a MOD could be not justified. Other than this, depending on the shape of the base granules, on their number, and on the amount of trajectories that need to be processed, the use of such a specialized software can achieve better performances than those obtained using a MOD. In these cases the choice of a specialized software could be a good trade-off between the flexibility needed and the performance that can be achieved. Just to give an intuition on how this software should work, we suppose the availability of reconstructed trajectories obtained by the reconstruction module. These reconstructed trajectories should first be broken in segments completely contained in a spatio-temporal base granule of the TDW. Once all the broken trajectories occurring in a new time interval are buffered, they can be used to compute aggregate measures and to load the fact table of the TDW, by possibly updating the temporal dimension tables [55].

It is worth noting that the ETL process is periodic, and depends on the time granularities of the temporal dimension. Hence, once an entire time interval has been loaded into the data warehouse, trajectory data related to that interval can be

safely discarded. Obviously, the specific aggregations to be performed also depend on the definitions of the various measures to be loaded.

3.5 Synopsis

This chapter introduced the general framework proposed in order to define a Trajectory Data Warehouse with the aim of allowing analyses on trajectory aggregated data in a visual manner. The chapter presented two different application scenarios, namely cars moving on a road network and vessels sailing on the sea. They will be used in the rest of the thesis as working examples in order to clarify issues and solutions that will be proposed, and to highlight the different needs our framework should be able to cope with.

In this chapter we discussed the ETL phase, by showing the issues related to the reconstruction of trajectories and the storing of such data. We presented a state-of-the-art procedure to reconstruct trajectories from raw spatio-temporal observations and we illustrated the role of Moving Object Databases and ad-hoc procedures for storing and loading the Trajectory Data Warehouse.

4

TDW Conceptual Model

In the previous chapter of this thesis we have presented the general framework we proposed in order to allow for the efficiently analysis of the huge amount of trajectories data available nowadays, and coming from a variety of different sources, like mobile phone networks, GPS-equipped devices and so on.

Such an amount of data coming from many different heterogeneous fields calls for powerful tools able to adapt themselves to various situations. As an example, referring to the scenarios presented in section 3.1, if we are dealing with information related to vessels sailing on the sea, the sea itself can be partitioned in squared regular areas. On the other hand, if we are dealing with information related to road traffic, the spatial domain can be associated with segments of the road network, or again to regions with arbitrary shapes representing some administrative division of the territory. Having this in mind we develop a model for our *Trajectory Data Warehouse* (TDW), that in fact represents the central module of the proposed architecture.

The *data model* we proposed results to be very flexible and general, allowing to handle different spatio-temporal domains with associated *dimensions* and *hierarchies* as those described so far.

Some previous works on the proposal of a Trajectory Data Warehouse model can be found in [45, 63, 50, 55]. However the model that will be presented in this chapter significantly extends and generalize those proposals. In particular, the main limitations of the approaches presented in those works are the fact that they are restricted to freely moving users and spatial hierarchies induced by regular grids, and the absence on some of them of a visual user interface for querying and analysing data. On the other hand, the model presented here overcomes these limits by allowing for a general spatial hierarchy and propose a visual OLAP tool for querying needs.

The rest of this chapter is organized as follow. Section 4.1 will introduce our proposal, as well as some design choices and problems involved in the design of such a data warehouse. In Section 4.2 the conceptual model of our Trajectory Data Warehouse will be presented, as well as the measures and hierarchies such a model should contain. Section 4.3 will formally define such hierarchies and measures. On Section 4.4 we will propose the aggregate function needed in order to correctly com-

pute roll-up operations on the proposed measures starting from their sub-aggregates. On Section 4.5 we will introduce and formally define the measure *Presence*, and we will describe a method in order to approximate this measure giving an estimation of the error this approximation introduce. Finally, Section 4.6 will give some hints related to the actual implementation of the TDW model and on its translation into a logical model.

4.1 Introduction

In order to develop a Trajectory Data Warehouse model, some choices need to be made. The first of these choices is relative to the kind of data that need to be handled in it. A distinctive characteristic of our TDW model concerns the *measures* that are stored in the TDW facts associated with *base granules*, which are a collection of elements, forming the *base granularity*, obtained by partitioning both the spatial and temporal dimensions. Informally, a granule can be defined as a contiguous spatial region during a given time interval, that can be aggregated along the spatio-temporal TDW dimensions according to the associated hierarchies. The measures associated with the base granules already represent aggregate information, as they report some significant features of the sets of trajectories crossing the granules, whereas single trajectory details, like object identifiers, are not kept in the TDW at all. There are, in general, good reasons for this design choice. Often individual data can be highly volatile and require huge memory space. More importantly, in some cases they cannot be stored due to legal or privacy issues, and anonymization might not suffice to guarantee privacy of tracked people [69]. In addition, for common spatio-temporal applications aggregate measures are typically much more relevant than information about individual moving objects [71].

Together with the measures, suitable aggregate functions must be defined in order to complete the *conceptual model* of our Trajectory Data Warehouse and allow for OLAP operations over the spatio-temporal hierarchies on the proposed measures. We will show that algebraic and distributive functions [27] can be used to compute roll-up queries, by linearly combining sub-aggregates stored in the lower levels of the hierarchy.

An interesting measure in a spatio-temporal context is the so called measure *Presence*, that counts the number of *distinct* objects (i.e. trajectories, for a TDW) occurring in a spatio-temporal partition of the respectively domains. The problem with this measure is that its aggregate function is *holistic* [27], and thus causes hindrances in OLAP roll-up operations. In fact, due to the *distinct count* requirement, *Presence* cannot be computed in an exact way by means of an aggregate function, unless we use an unlimited amount of memory to store the trajectory identifiers in the base granules of the hierarchy.

By removing the "*distinct*" requirement for the measure *Presence*, we obtain a value counting the number of times a given granule has been visited by the various

moving objects. We call this measure *Visits*. With respect to the measure *Presence*, *Visits* keeps track of the fact that an object can return in a place many times. An important property of this measure, that we will show in the following, is that its algebraic aggregate function can be shown to exactly answer roll-up queries. Moreover, this new measure can be used as a very good approximation for the holistic measure *Presence*. Computing holistic functions in an *approximate* way is thus a common solution and we will demonstrate that *Visits* provides a more accurate approximation than other state of the art techniques. On this side, we will show how we can give a formal limit on the error that could be committed on its approximation.

4.2 Multi-cube conceptual model

In Figure 4.1 we present a multi-cube conceptual model for trajectories built by using the Dimensional Fact Model formalism [24]. Facts, focus of interest of the decisional process, are represented by boxes containing the fact name and a list of associated measures (quantitative aspects interesting for analysis). We recall that a granule is a couple composed by a temporal interval and a contiguous spatial region. In the model we consider two facts, namely INTRA-GRANULE FACT and INTER-GRANULE FACT. Here we introduce such facts and a set of significant measures, which will be formally defined in the next sections. It is worth remarking that the presented measures are not an exhaustive collection, but they correspond to a set of common measures which we found interesting and useful in almost any different scenarios.

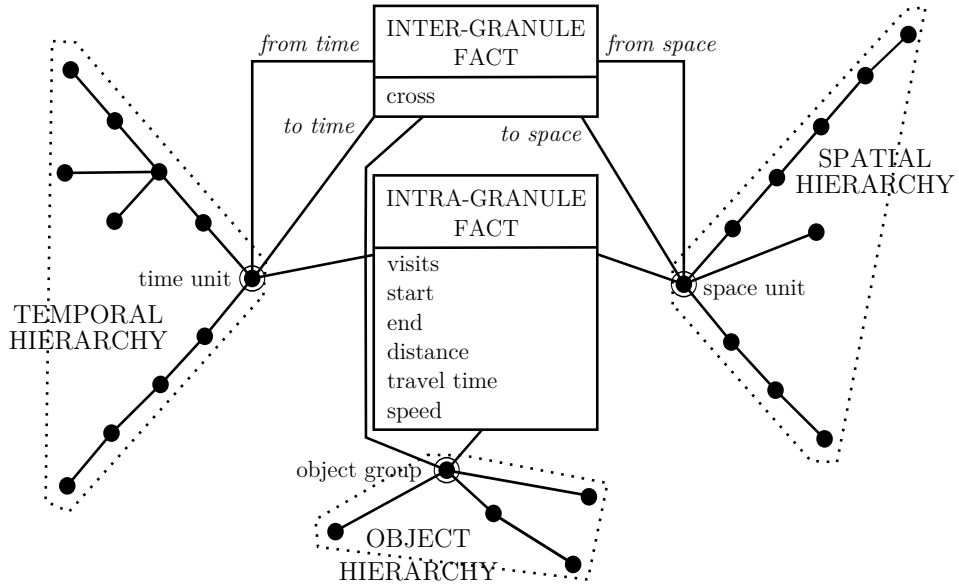


Figure 4.1: TDW Conceptual model

The INTRA-GRANULE fact models events that are related to a *single base gran-*

ule concerning a certain object group. For a given group U and a granule g, the illustrated measures are:

- *visits*: the number of trajectories belonging to U which start from or enter into g;
- *start*: the number of trajectories belonging to U starting in g;
- *end*: the number of trajectories belonging to U ending in g;
- *travel time*: the time spent by all trajectories belonging to U while moving inside g;
- *distance*: the distance travelled by all trajectories belonging to U while moving inside g;
- *speed*: the average speed of trajectories belonging to U traversing g.

The INTER-GRANULE fact models events that are related to *pairs of granules* and are concerned with a specific object group. A measure of interest that we will discuss is *cross*, i.e., the number of times the border from a granule to an adjacent one has been traversed by trajectories belonging to a given group. The measure *cross* is interesting only for adjacent granules, since for non-adjacent ones it is invariably 0. However note that in general, this fact can model events which can be meaningful for whichever pairs of granules. An example could be the *origin-destination* measure, which, for any pair of granules, represents the number of trajectories starting from the first and ending into the second granule.

Dimensions, i.e. the finest level of granularity for the analysis of facts, are represented as circles attached to the fact tables by straight lines. As already mentioned, the dimensions in our model are *spatial* and *temporal* dimensions describing geography and time, respectively, and a non spatio-temporal dimension (*object group*).

Dimensional attributes, i.e. the properties of dimensions, are represented as circles attached directly to dimensions or to other dimensional attributes. A dimensional hierarchy consists of a dimension and its dimensional attributes, hence represented as a rooted tree, having the root attached to a fact table.

For the sake of readability, hierarchies that are common to various dimensions are shared in the graphical representation. This is indicated using a double circle for the root node. Moreover, when a dimension appears more than once in a fact table, it is necessary to indicate a role to define its meaning. For example, in Figure 4.1 *from/to space* associated with the spatial dimension of the fact table INTER-GRANULE denote, respectively, the origin/destination granule of the movement.

4.2.1 Examples of possible hierarchies

An example of a complex temporal hierarchy is illustrated in Figure 4.2. We consider *minutes* as the minimal temporal interval. Every minute belongs to a *10 minutes interval* and to an *hour*, which is included in one *day*. A day is contained in both a *week* and a *month*, and it is also a *day of the week*. The *10 minutes interval* is included in intervals of larger and larger duration.

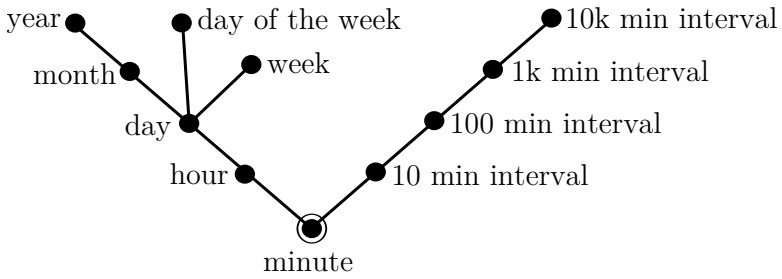


Figure 4.2: An example of a temporal hierarchy

Some different kinds of spatial hierarchies are presented in Figure 4.3 and Figure 4.4. The hierarchy represented in Figure 4.3(a) is based on a collection of regular grids of increasing size, while the one represented in Figure 4.3(b) is more articulated. In this case the hierarchy has been designed to fulfil some possible analytical needs for a more complex vessels analysis. The dataset could be composed by waypoints (i.e. sets of coordinates that identify points in physical space) and links constituting a graph. This graph represents the main traffic flows of the maritime traffic and is named the maritime traffic network. For the purpose of the analysis, the spatial extent of a link is a polygon, usually obtained as a buffer around the segment corresponding to the link. A generic position in the sea may be either inside or outside a network link.

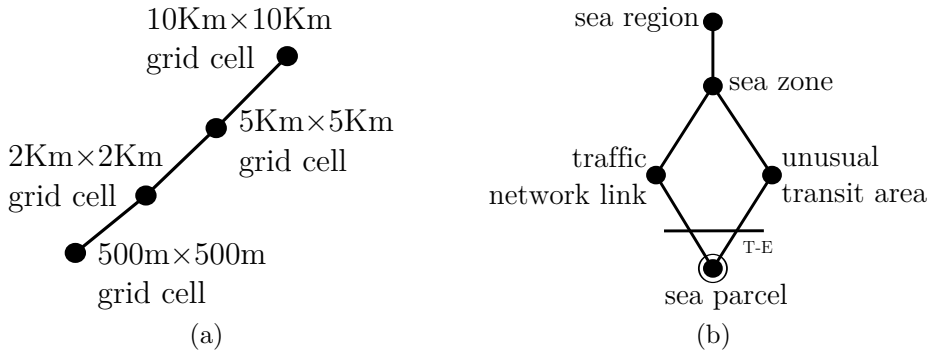


Figure 4.3: Two examples of spatial hierarchies for vessels sailing on the sea

To this end, we classified the part of the sea that is not covered by the traffic network in a generic transit area, and chose as base granules so-called sea parcels,

which are contained either in some network link or in some (unusual) transit area (T-E stands for Totally-Exclusively, and thus indicates that *all* parcels are *exclusively* contained in a transit area or in a network link). At a coarser level, those network links and transit areas are grouped in sea zones and sea regions.

Figure 4.4 represents a possible spatial hierarchy related to road traffic analysis. The hierarchy represents a tree, where the segment is the smallest spatial unit, and each segment is contained in exactly one *district*, which is included in one *zone* belonging to a *province*, and in turn contained in a *region* and finally in a *country*. At the same time, each segment is also contained into a *cell* of a $200m \times 200m$ grid, which is contained into a *cell* of a $1Km \times 1Km$ grid, and so on.

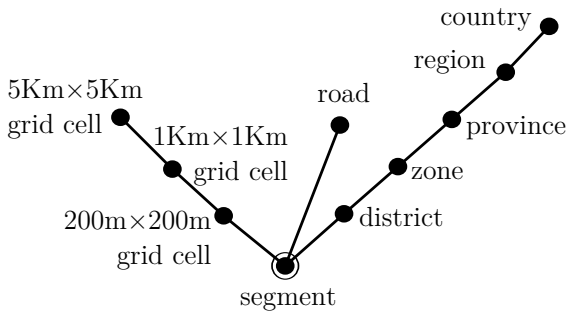


Figure 4.4: An examples of spatial hierarchies for road traffic analysis

Finally, in Figure 4.5, some possible kinds of hierarchies associated with the *object group* dimension are proposed. Usually, this kind of hierarchy is much simpler than the spatio-temporal ones. Figure 4.5(a) is related to a possible classification of ships. Each ship has a type, for instance 'LPG (Liquefied Petroleum Gas) semi pressured', and a main type, for example 'LPG', that groups several types having similar peculiarities. Figure 4.5(b) presents instead a hierarchy related to a group of people. Users, that represent the base granularity, have a gender and belong to an age groups. Furthermore, a user lives in a town that belongs to a region that lies in a country.

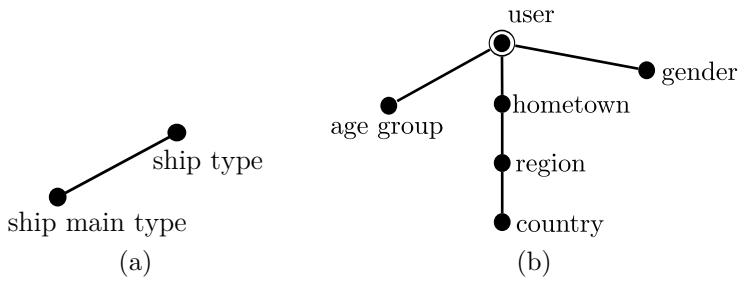


Figure 4.5: Examples of object hierarchies

4.3 TDW Hierarchies and Measures

In the following we will define a discretization of space and time domains of the proposed Trajectory Data Warehouse into so-called *spatio-temporal granules*, and then their organization in *hierarchies*. We will introduce the concept of *trajectory decomposition* according to the discretised spatio-temporal domain. This concept will be exploited in order to provide a formal definition of the TDW *measures* presented in the previous section.

4.3.1 Granules, Granularities and Hierarchies

We need to discretize the space and time dimensions in order to define the TDW base granularity. We start discussing the discretization of a generic domain.

Definition 4.1 *(Granule and Granularity)*
Let $\mathbb{D}$ be any domain (e.g. temporal or spatial). A granularity *G on $\mathbb{D}$ is any* partition *of $\mathbb{D} = \bigcup_{g \in G} g$, whose elements g are called* granules*.*

Given two granularities G and G' we say that G is *finer* than G' and write $G \preceq G'$, if for all $g \in G$, there exists $g' \in G'$ such that $g \subseteq g'$. We say also G' is *coarser* than G.

Definition 4.2 *(Hierarchy)*
A hierarchy *over a domain $\mathbb{D}$ is a set $\mathcal{H}$ of granularities on $\mathbb{D}$, ordered by $\preceq$, and with a minimum, called the* base granularity*, composed of* base granules*.*

In the following we assume that for the spatial domain $\mathbb{S}$ a finite hierarchy is fixed, denoted by $\mathcal{H}_S$, with base granularity $\mathcal{BS}$, and such that for any $G_S \in \mathcal{H}_S$ and each granule $g_S \in G_S$, g_S is topologically connected. Similarly, for the temporal domain $\mathbb{T}$ we fix a finite hierarchy, denoted by $\mathcal{H}_T$, with base granularity $\mathcal{BT}$, and each granule at any granularity is a temporal interval. Observe that these induce a finite hierarchy over the spatio-temporal domain $\mathbb{T} \times \mathbb{S}$, resulting as $\mathcal{H}_{TS} = \{G_T \times G_S \mid G_T \in \mathcal{H}_T \wedge G_S \in \mathcal{H}_S\}$. Hereinafter we denote a granule in $\mathbb{T} \times \mathbb{S}$ as a pair (g_T, g_S). In addition, when taking a granularity $G \in \mathcal{H}_{TS}$, we will denote by G_T and G_S the corresponding temporal and spatial granularities, such that $G = G_T \times G_S$. Moreover we will denote by $\mathcal{BTS}$ the base granularity $\mathcal{BT} \times \mathcal{BS}$.

4.3.2 Trajectory decomposition

In the following we will refer to sets of trajectories indexed by trajectory identifiers $\mathsf{T} = \{\mathsf{T}_{id}\}_{id \in \mathcal{I}}$ where $\mathcal{I}$ is the set of trajectory identifiers. For any trajectory identifier $id \in \mathcal{I}$, the corresponding trajectory is a function $\mathsf{T}_{id} : I_{id} \to \mathbb{R}^2$. Equivalently, T_{id} can be seen as a set of points in a 3D space $\{(t, \mathsf{T}_{id}(t)) \mid t \in I_{id}\}$, i.e., the graph of the function. In the sequel we will use these two views interchangeably.

As mentioned before, we also assume the presence of object groups, where a generic group will be denoted as U and, abusing the notation, we will write $id \in U$ to mean that the (object corresponding to trajectory) id belongs to group U.

Definition 4.3 *(Trajectory decomposition)*
Let T be a set of trajectories and let $\mathcal{BTS}$ be a base granularity. For each trajectory $\mathsf{T}_{id} : I_{id} \to \mathbb{R}^2$ in T the trajectory decomposition *is a sequence of sub-trajectories $\delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id} \rangle$, with $s^i_{id} : I^i_{id} \to \mathbb{R}^2$ and $I^i_{id} = [t_{start}{}^i_{id}, t_{end}{}^i_{id})$, satisfying the following conditions:*

- $\sup(I^i_{id}) = \inf(I^{i+1}_{id})$ *for all* $i \in \{1, \ldots, n-1\}$,
- $\bigcup_{i=1}^{n} s^i_{id} = \mathsf{T}_{id}$ *and* $s^i_{id} \cap s^j_{id} = \emptyset$, *for* $i \neq j$,
- *for any* $i \in \{1, \ldots, n\}$, *there exists a granule* $g \in \mathcal{BTS}$ *such that* $s^i_{id} \subseteq g$.

Informally, $\delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id} \rangle$ is a partition of the trajectory T_{id}, where each s^i_{id} is a fragment of the trajectory included in a granule of $\mathcal{BTS}$, such that each s^i_{id} temporally precedes s^{i+1}_{id}.

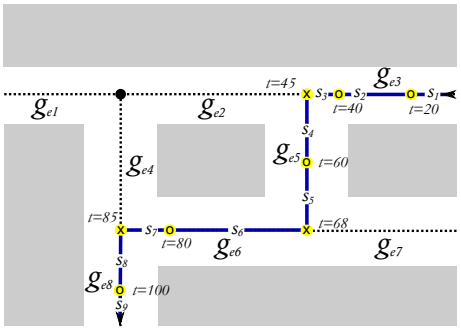


Figure 4.6: Decomposition of a trajectory.

Example 4.1 *Let us consider a road traffic scenario as presented in Section 3.1. The spatial domain is a road network RN embedded in $\mathbb{R}^2$, i.e. $RN \subseteq \mathbb{R}^2$.*

Figure 4.6 shows a part of a user trajectory T_{id}, depicted as a solid polyline in the centre of the route and the arrows indicate the direction of the movement. Notice that the movement of the user is constrained to the roads (in white), whereas grey areas are unreachable.

We assume as base spatial granules $\{g_{e1}, \ldots, g_{e8}\}$ corresponding to edges of the road network, while the base temporal granularity consists of regular intervals of 20 time units starting from 0 ($\{[0, 20), [20, 40), \ldots\}$). Hence according to this spatio-temporal granularity the decomposition of T_{id} is

$$\delta(\mathsf{T}_{id}) = \langle s_1, \ldots, s_9 \rangle$$

The symbols ○ and × indicate the points $(t, T_{id}(t))$ where the trajectory enters, respectively, a new temporal granule or a new spatial granule, and the label indicates the corresponding time t. This happens when t is in a different temporal granule, or $T_{id}(t)$ in a different spatial granule, with respect to the immediately preceding point. These points should not be confused with the trajectory observations (not represented in this figure), which are only relevant in the reconstruction phase.

4.3.3 Intra-Granule Measures

A first intra-granule measure is *visits* ($\mathcal{V}$), which represents the number of visits in g of the trajectories belonging to a certain group U. More precisely, $\mathcal{V}^{\mathsf{T}}(g, U)$ is defined as the number of times that any trajectory belonging to U enters into, or starts from the spatio-temporal granule g.

Definition 4.4 *(Visits)*
Let G be a spatio-temporal granularity and $g \in G$ be a granule, let T be a set of trajectories and let U be an object group, then

$$\mathcal{V}^{\mathsf{T}}(g,U) = |\{(id, s^i_{id}) \mid id \in U \ \wedge \ \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \ \wedge \ s^i_{id} \subseteq g \ \wedge \ (i = 1 \ \vee \ s^{i-1}_{id} \not\subseteq g)\}|$$

Informally, a trajectory visits a granule $g = (g_T, g_S)$ if there exists $t \in g_T$ such that the trajectory is in g_S at time t, but it was not in g_S immediately before t, either because the trajectory starts at time t, or the trajectory was in another spatial granule. Moreover, a trajectory can visit g even without any movement: an object, that stays in g_S for a long time period, can enter different spatio-temporal granules having the same spatial granule g_S but different temporal granules g_T.

Other intra-granule measures are *start* ($\mathcal{S}$) and *end* ($\mathcal{E}$), i.e., the number of trajectories starting and ending in a granule, *distance* (*dist*), *travel time* (*trav_t*) and *speed*, i.e., the total travelled distance by the trajectories, the total time spent in the granule and the corresponding average speed. Their definitions, which are given below, are the natural ones.

Definition 4.5 *(Other measures)*
Let G be a spatio-temporal granularity and $g \in G$ be a granule, let T be a set of trajectories and let U be an object group, then

$$\mathcal{S}^{\mathsf{T}}(g,U) \ = \ |\{id \mid id \in U \ \wedge \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \ \wedge \ s^1_{id} \subseteq g\}|$$

$$\mathcal{E}^{\mathsf{T}}(g,U) \ = \ |\{id \mid id \in U \ \wedge \ \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \ \wedge \ s^n_{id} \subseteq g\}|$$

$$dist^{\mathsf{T}}(g,U) \ = \ \Sigma_{id\in U,\, \delta(\mathsf{T}_{id})=\langle s^1_{id},\ldots,s^n_{id}\rangle} \, \Sigma_{s^j_{id}\subseteq g} len(s^j_{id})$$

$$trav\_t^{\mathsf{T}}(g,U) \ = \ \Sigma_{id\in U,\, \delta(\mathsf{T}_{id})=\langle s^1_{id},\ldots,s^n_{id}\rangle} \, \Sigma_{s^j_{id}\subseteq g} lifespan(s^j_{id})$$

$$speed^{\mathsf{T}}(g,U) = \frac{dist^{\mathsf{T}}(g,U)}{trav\_t^{\mathsf{T}}(g,U)}$$

where $len(s^j_{id})$ *is the length of the spatial projection of* s^j_{id} *whereas* $lifespan(s^j_{id})$ *is the duration of the time interval where* s^j_{id} *is defined.*

4.3.4 Inter-Granule Measures

The only inter-granule measure we discuss in this thesis is *cross* ($\mathcal{C}$), the number of times the border from a granule to another has been traversed by trajectories of a given group.

Definition 4.6 *(Cross)*
Let G *be a spatio-temporal granularity, let* $g, g' \in G$ *be two distinct granules, let* T *be a set of trajectories and let* U *be an object group, then*

$$\mathcal{C}^{\mathsf{T}}(g,g',U) = |\{(id, s^i_{id}) \mid id \in U \wedge \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \wedge i < n \ \wedge \ s^i_{id} \subseteq g \ \wedge \ s^{i+1}_{id} \subseteq g'\}|$$

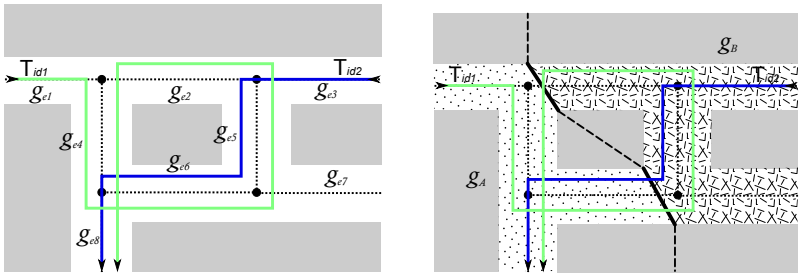
The measure $\mathcal{C}$ is not symmetric: in general $\mathcal{C}^{\mathsf{T}}(g,g',U) \neq \mathcal{C}^{\mathsf{T}}(g',g,U)$ since the measure is sensible to the direction of movements and only counts crossings from g to g'. Note that $\mathcal{C}^{\mathsf{T}}(g,g',U) = 0$ when g and g' are not *adjacent*, where *adjacency* relationship is defined in the usual way.

Definition 4.7 *(Adjacency)*
Let $g = (g_T, g_S)$ *and* $g' = (g'_T, g'_S)$ *two granules belonging to a granularity* G*; they are* adjacent *when* $g_T = g'_T \wedge Touch(g_S, g'_S)$ *or* $g_S = g'_S \wedge Meets(g_T, g'_T)$ *or* $Meets(g_T, g'_T) \wedge Touch(g_S, g'_S)$

where $Meets$ *is Allen's relation [1] and* $Touch$ *is Egenhofer's topological relation [18].*

Example 4.2 *We still refer to the running example of a TDW for road traffic analysis. Figure 4.7(a) illustrates portions of two trajectories,* T_{id1} *and* T_{id2}*, belonging to the same group* U*, during a temporal granule* g_T*. The direction of a trajectory is indicated by an arrow. The trajectory* T_{id1}*, in light green (light gray in B&W), passes through the sequence of base spatial granules* g_{e1}*,* g_{e4}*,* g_{e6}*,* g_{e5}*,* g_{e2}*,* g_{e4}*,* g_{e8}*, whereas* T_{id2}*, in blue (black in B&W), travels along* g_{e3}*,* g_{e5}*,* g_{e6}*,* g_{e8}*.*

Note that $\mathcal{V}^{\mathsf{T}}((g_T, g_{e4}), U) = 2$ *since* T_{id1} *enters twice into* g_{e4}*, whereas* $\mathcal{V}^{\mathsf{T}}((g_T, g_{e6}), U) = 2$ *since both trajectories enter only once into* g_{e6}*.*



(a) (b)

Figure 4.7: Two trajectories at base (a) and higher (b) spatial granularity

Focusing on the crosses between the granule (g_T, g_{e2}) *and the adjacent granules, we have:*

$$\begin{aligned}
\mathcal{C}^{\mathcal{T}}((g_T, g_{e2}), (g_T, g_{e1}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e2}), (g_T, g_{e3}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e2}), (g_T, g_{e4}), U) &= 1 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e2}), (g_T, g_{e5}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e1}), (g_T, g_{e2}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e3}), (g_T, g_{e2}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e4}), (g_T, g_{e2}), U) &= 0 \\
\mathcal{C}^{\mathcal{T}}((g_T, g_{e5}), (g_T, g_{e2}), U) &= 1
\end{aligned}$$

4.4 Aggregation

In order to allow for OLAP processing, our TDW has to offer aggregation capabilities over measures, i.e., operations for computing measures at some higher level of the hierarchy starting from those at lower level. Efficient OLAP roll-up operations require that measures at a coarser granularity can be determined using values at a finer granularity, by applying some defined *aggregate functions*. According to the classification given by Gray et al. [27], it is desirable that measures in a TDW use only distributive or algebraic aggregate functions. We recall that distributive functions use only sub-aggregates of a given measures to compute its super-aggregates, while algebraic functions make use of a finite number of auxiliary measures in order to perform this operation. On the other hand, holistic functions need the base data to compute the result at all levels of dimensions. Hence, measures using this kind of aggregate functions imply the need of storing the entire dataset into the data warehouse and, either if this would be possible in terms of available space, it could not be feasible with respect to other problems like privacy concerns. Moreover, either if original data could be available, given their nature, computation of holistic functions is usually more time consuming with respect to distributive or algebraic ones. In this section we will define distributive and algebraic aggregate functions for the measures we presented in the previous section and we will prove that these

definitions allow to compute in a correct way out measures at all granularities.

The first lemma we propose takes into account the aggregate functions for all the measures except $\mathcal{V}$.

Lemma 4.1 *Let $\mathcal{H}_{\mathcal{TS}}$ be a hierarchy, let T be a set of trajectories and U be an object group. For any $G \in \mathcal{H}_{\mathcal{TS}}$ with $G \neq \mathcal{BTS}$ and $g, g' \in G$ with $g \neq g'$*

$$\mathcal{S}^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g} \mathcal{S}^{\mathsf{T}}(g_p,U) \quad (4.1)$$

$$\mathcal{E}^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g} \mathcal{E}^{\mathsf{T}}(g_p,U) \quad (4.2)$$

$$dist^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g} dist^{\mathsf{T}}(g_p,U) \quad (4.3)$$

$$trav\_t^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g} trav\_t^{\mathsf{T}}(g_p,U) \quad (4.4)$$

$$\mathcal{C}^{\mathsf{T}}(g,g',U) = \Sigma_{g_p \subseteq g, g'_p \subseteq g'} \mathcal{C}^{\mathsf{T}}(g_p,g'_p,U) \quad (4.5)$$

where $g_p, g'_p \in G_p$ with $g_p \neq g'_p$ and $G_p \preceq G$ and $G_p \neq G$.

Proof.

$\mathcal{S}^{\mathsf{T}}(g,U) =$

by Definition of $\mathcal{S}$

$= |\{id \mid id \in U \wedge \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \wedge s^1_{id} \subseteq g\}|$

by Definition 4.3, and since $\mathcal{BTS} \preceq G_P$ and $G_p \preceq G$

$= |\{id \mid id \in U \wedge \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \wedge s^1_{id} \subseteq g_p$ *for some* $g_p \subseteq g\}|$

since G_p is a partition g_p is unique

$= \Sigma_{g_p \subseteq g} |\{id \mid id \in U \wedge \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}\rangle \wedge s^1_{id} \subseteq g_p\}|$

by Definition of $\mathcal{S}$

$= \Sigma_{g_p \subseteq g} \mathcal{S}^{\mathsf{T}}(g_p,U)$

The remaining statements are proved in a similar way. □

The case of measure $\mathcal{V}$ is more complex and requires the introduction of an auxiliary measure B, counting the number of times trajectories cross the border of a granule.

Definition 4.8
Let G be a spatio-temporal granularity and $g \in G$ be a granule, let T be a set of trajectories and let U be an object group. We denote by $B^{\mathsf{T}}(g,U)$ the number of intersections between the trajectories of object group U and the border of the spatio-temporal granule g.

$$B^{\mathsf{T}}(g,U) = \Sigma_{g' \in G, g' \neq g} (\mathcal{C}^{\mathsf{T}}(g,g',U) + \mathcal{C}^{\mathsf{T}}(g',g,U))$$

The following lemma states how measure B can be computed by using sub-aggregates of $\mathcal{C}$.

Lemma 4.2 *Let $\mathcal{H}_{\mathcal{TS}}$ be a hierarchy, let T be a set of trajectories and U be an object group. For any $G \in \mathcal{H}_{\mathcal{TS}}$ with $G \neq \mathcal{BTS}$ and $g \in G$*

$$B^{\mathsf{T}}(g,U) = \sum_{g_p \subseteq g,\, g'_p \not\subseteq g} (\mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) + \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U))$$

where $g_p, g'_p \in G_p$ and $g_p \neq g'_p$ with $G_p \preceq G$ and $G_p \neq G$.

Proof.

$B^{\mathsf{T}}(g,U) =$
by definition of B
$= \sum_{g' \in G,\, g' \neq g} (\mathcal{C}^{\mathsf{T}}(g, g', U) + \mathcal{C}^{\mathsf{T}}(g', g, U))$
by statement (4.5) of Lemma 4.1
$= \sum_{g' \in G,\, g' \neq g} (\sum_{g_p \subseteq g, g'_p \subseteq g'} \mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) +$
$\sum_{g_p \subseteq g, g'_p \subseteq g'} \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U))$
$= \sum_{g' \in G,\, g' \neq g} (\sum_{g_p \subseteq g, g'_p \subseteq g'} \mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) + \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U))$
G is a partition
$= \sum_{g_p \subseteq g, g'_p \not\subseteq g} (\mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) + \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U))$

□

The next lemma provides an inductive characterisation of measure B.

Lemma 4.3 *Let $\mathcal{H}_{\mathcal{TS}}$ be a hierarchy, let T be a set of trajectories and U be an object group. For any $G \in \mathcal{H}_{\mathcal{TS}}$ with $G \neq \mathcal{BTS}$ and $g \in G$*

$$B^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g} B^{\mathsf{T}}(g_p, U) - 2\Sigma_{g_p, g'_p \subseteq g, g_p \neq g'_p} (\mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) + \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U))$$

where $g_p, g'_p \in G_p$ with $G_p \preceq G$ and $G_p \neq G$.

Proof. This property easily follows by definition of B. More in detail, for the sake of simplicity assume $g = g_p \cup g'_p$. Then the border of the granule g consists of the union of the borders of g_p and g'_p minus the common border between the granules g_p and g'_p. As a consequence trajectories crossing the common border remain inside the granule g and they should not be counted in $B^{\mathsf{T}}(g,U)$.
In formulas:

$$B^{\mathsf{T}}(g,U) = B^{\mathsf{T}}(g_p,U) - \mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) - \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U) + B^{\mathsf{T}}(g'_p,U) - \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U) - \mathcal{C}^{\mathsf{T}}(g_p, g'_p, U)$$

This is exactly the desired result. □

It can then be shown how the measure $\mathcal{V}$ can be expressed analytically in terms of B, $\mathcal{S}$ and $\mathcal{E}$.

Proposition 4.1 *Let G be a spatio-temporal granularity, let T be a set of trajectories and let U be an object group. Then for each $g \in G$ the following statement holds:*

$$\mathcal{V}^{\mathsf{T}}(g,U) = \frac{B^{\mathsf{T}}(g,U) + \mathcal{S}^{\mathsf{T}}(g,U) + \mathcal{E}^{\mathsf{T}}(g,U)}{2}$$

Proof. It can be observed that, as obvious from its definition, $\mathcal{V}$ can be computed by summing up the contributions given to such a measure separately by each trajectory. More precisely, let g be a granule and $\mathsf{T} = \{\mathsf{T}_{id}\}_{id \in \mathcal{I}}$ a set of trajectories. Then $\mathcal{V}^{\mathsf{T}}(g,U) = \sum_{id \in \mathcal{I}} \mathcal{V}^{\{\mathsf{T}_{id}\}}(g,U)$.

Therefore, it can be proved the proposition for a single trajectory T_{id} and thesis can be trivially extended to a generic set of trajectories.

Let $\delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id} \rangle$, we prove the thesis by induction on n (the number of sub-trajectories of the trajectory decomposition).

[$n = 1$] In this case $\delta(\mathsf{T}_{id}) = \langle s^1_{id} \rangle$. If $s^1_{id} \subseteq g$, then $\mathcal{V}^{\mathsf{T}}(g,U) = 1$, $B^{\mathsf{T}}(g,U) = 0$, $\mathcal{S}^{\mathsf{T}}(g,U) = 1$, and $\mathcal{E}^{\mathsf{T}}(g,U) = 1$. Thus the thesis holds.

If, instead, $s^1_{id} \not\subseteq g$, then $\mathcal{V}^{\mathsf{T}}(g,U) = 0$, $B^{\mathsf{T}}(g,U) = 0$, $\mathcal{S}^{\mathsf{T}}(g,U) = 0$, and $\mathcal{E}^{\mathsf{T}}(g,U) = 0$ and the thesis holds.

Notice that in both cases the measure B is equal to 0 since at least two sub-trajectories are necessary to have $B > 0$ because each sub-trajectory belongs to exactly one granule.

[$n \Rightarrow n+1$] In this case $\delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id}, s^{n+1}_{id} \rangle$. Let $\delta(\mathsf{T}'_{id}) = \langle s^1_{id}, \ldots, s^n_{id} \rangle$. We consider four cases:

[case $s^n_{id} \subseteq g$ and $s^{n+1}_{id} \subseteq g$] By definition $\mathcal{V}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}'}(g,U)$, $\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{S}^{\mathsf{T}'}(g,U)$, $\mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}'}(g,U)$, and $B^{\mathsf{T}}(g,U) = B^{\mathsf{T}'}(g,U)$. Thus, the thesis holds by inductive hypothesis.

[case $s^n_{id} \not\subseteq g$ and $s^{n+1}_{id} \subseteq g$] By definition $\mathcal{V}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}'}(g,U)+1$, $\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{S}^{\mathsf{T}'}(g,U)$, $\mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}'}(g,U)+1$, and $B^{\mathsf{T}}(g,U) = B^{\mathsf{T}'}(g,U)+1$. Thus by using the inductive hypothesis

$$\begin{aligned}
\mathcal{V}^{\mathsf{T}}(g,U) &= \mathcal{V}^{\mathsf{T}'}(g,U) + 1 \\
&= \frac{B^{\mathsf{T}'}(g,U) + \mathcal{S}^{\mathsf{T}'}(g,U) + \mathcal{E}^{\mathsf{T}'}(g,U)}{2} + 1 \\
&= \frac{B^{\mathsf{T}'}(g,U) + 1 + \mathcal{S}^{\mathsf{T}'}(g,U) + \mathcal{E}^{\mathsf{T}'}(g,U) + 1}{2} \\
&= \frac{B^{\mathsf{T}}(g,U) + \mathcal{S}^{\mathsf{T}}(g,U) + \mathcal{E}^{\mathsf{T}}(g,U)}{2}
\end{aligned}$$

[case $s_{id}^n \subseteq g$ and $s_{id}^{n+1} \not\subseteq g$] By definition $\mathcal{V}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}'}(g,U)$, $\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{S}^{\mathsf{T}'}(g,U)$, $\mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}'}(g,U) - 1$, and $B^{\mathsf{T}}(g,U) = B^{\mathsf{T}'}(g,U) + 1$. As in the previous case we can conclude.

[case $s_{id}^n \not\subseteq g$ and $s_{id}^{n+1} \not\subseteq g$] By definition $\mathcal{V}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}'}(g,U)$, $\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{S}^{\mathsf{T}'}(g,U)$, $\mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}'}(g,U)$, and $B^{\mathsf{T}}(g,U) = B^{\mathsf{T}'}(g,,U)$. Thus the thesis holds by inductive hypothesis.

□

In order to give the aggregate function for the measure $\mathcal{V}$ it is worth trying to give an intuition on how it works, before giving the formal definition. At a coarser granule g the number of visits in g is obtained by summing up the visits in the finer granules g_p composing g, and subtracting the number of trajectories crossing the border between two distinct finer granules inside g. This is motivated by the fact that the border between two finer granules, g_p and g'_p composing g is completely inside g. Hence trajectories moving from g_p to g'_p (or vice versa) increase the number of visits of g'_p (or g_p) but they should not be counted as visits for the coarser granule g because the movement is completely inside g, i.e., they do not *enter* g. The following lemma will formalise this intuition.

Lemma 4.4 *Let $\mathcal{H}_{\mathcal{TS}}$ be a hierarchy, let T be a set of trajectories and U be an object group. For any $G \in \mathcal{H}_{\mathcal{TS}}$ with $G \neq \mathcal{BTS}$ and $g \in G$*

$$\mathcal{V}^{\mathsf{T}}(g,U) = \Sigma_{g_p \subseteq g}(\mathcal{V}^{\mathsf{T}}(g_p,U) - \Sigma_{g'_p \subseteq g}\mathcal{C}^{\mathsf{T}}(g_p,g'_p,U))$$

where $g_p \in G_p$ with $G_p \preceq G$ and $G_p \neq G$.

Proof. For the sake of simplicity, let $g = g_p \cup g'_p$ with $g_p, g'_p \in G_p$ and $g_p \neq g'_p$.

$$\begin{aligned}
&\mathcal{V}^{\mathsf{T}}(g,U) = \\
&\quad \text{by Proposition 4.1} \\
&= \frac{B^{\mathsf{T}}(g,U) + \mathcal{S}^{\mathsf{T}}(g,U) + \mathcal{E}^{\mathsf{T}}(g,U)}{2} \\
&\quad \text{by Lemma 4.3 and } g = g_p \cup g'_p \\
&= \frac{B^{\mathsf{T}}(g_p,U)+B^{\mathsf{T}}(g'_p,U)-2(\mathcal{C}^{\mathsf{T}}(g_p,g'_p,U)+\mathcal{C}^{\mathsf{T}}(g'_p,g_p,U))+\mathcal{S}^{\mathsf{T}}(g,U)+\mathcal{E}^{\mathsf{T}}(g,U)}{2}
\end{aligned}$$

By Lemma 4.1, we have

$$\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{S}^{\mathsf{T}}(g_p,U) + \mathcal{S}^{\mathsf{T}}(g'_p,U)$$

and

$$\mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}}(g_p,U) + \mathcal{E}^{\mathsf{T}}(g'_p,U)$$

By Proposition 4.1

$$\mathcal{V}^{\mathsf{T}}(g_p,U) = \frac{B^{\mathsf{T}}(g_p,U) + \mathcal{S}^{\mathsf{T}}(g_p,U) + \mathcal{E}^{\mathsf{T}}(g_p,U)}{2}$$

and

$$\mathcal{V}^{\mathsf{T}}(g'_p, U) = \frac{B^{\mathsf{T}}(g'_p, U) + \mathcal{S}^{\mathsf{T}}(g'_p, U) + \mathcal{E}^{\mathsf{T}}(g'_p, U)}{2}$$

Hence we can conclude

$$\mathcal{V}^{\mathsf{T}}(g, U) = \mathcal{V}^{\mathsf{T}}(g_p, U) + \mathcal{V}^{\mathsf{T}}(g'_p, U) - \mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) - \mathcal{C}^{\mathsf{T}}(g'_p, g_p, U)$$

This is the thesis since $\mathcal{C}^{\mathsf{T}}(g, g, U) = 0$ for any granule g. □

Finally, by joining together the Lemmas 4.1 and 4.4 we obtain the following proposition:

Proposition 4.2 *Let $\mathcal{H}_{\mathcal{TS}}$ be a hierarchy, let T be a set of trajectories and U be an object group. For any $G \in \mathcal{H}_{\mathcal{TS}}$ with $G \neq \mathcal{BTS}$, $g, g' \in G$ with $g \neq g'$*

$$\begin{aligned}
\mathcal{V}^{\mathsf{T}}(g, U) &= \Sigma_{g_p \subseteq g}(\mathcal{V}^{\mathsf{T}}(g_p, U) - \Sigma_{g'_p \subseteq g}\mathcal{C}^{\mathsf{T}}(g_p, g'_p, U)) \\
\mathcal{S}^{\mathsf{T}}(g, U) &= \Sigma_{g_p \subseteq g}\mathcal{S}^{\mathsf{T}}(g_p, U) \\
\mathcal{E}^{\mathsf{T}}(g, U) &= \Sigma_{g_p \subseteq g}\mathcal{E}^{\mathsf{T}}(g_p, U) \\
\mathcal{C}^{\mathsf{T}}(g, g', U) &= \Sigma_{g_p \subseteq g, g'_p \subseteq g'}\mathcal{C}^{\mathsf{T}}(g_p, g'_p, U) \\
\mathit{dist}^{\mathsf{T}}(g, U) &= \Sigma_{g_p \subseteq g}\mathit{dist}^{\mathsf{T}}(g_p, U) \\
\mathit{trav_t}^{\mathsf{T}}(g, U) &= \Sigma_{g_p \subseteq g}\mathit{trav_t}^{\mathsf{T}}(g_p, U)
\end{aligned}$$

where $g_p, g'_p \in G_p$ with $g_p \neq g'_p$ and G_P is a predecessor of G, i.e., $G_p \preceq G$ and $G_p \neq G$.

Proposition 4.2 suggests the aggregate functions which can be used to compute a measure m at coarser granularities by exploiting the sub-aggregates for finer granularities. In order to apply them we need to know, for each high level granule, which base level ones belong to and compose it. A possible solution is to apply a possibly complex spatio-temporal containment condition to the granules of the different levels, in order to determine whether the desired condition holds or not. By the way, the spatial and temporal hierarchies in the data warehouse are fixed a priori, hence for each base granule the temporal and spatial coarser granules it belongs to can be pre-computed. It follows that it can be constructed a map of values, in the hierarchies, telling the system, for every higher level granules which lower level ones compose it, based on the uniquely identifier associated to every granule. In this way, the containment condition does not require any spatio-temporal query but only an equality test on the granule identifiers themselves.

With respect to the classification of Gray et al. [27], presented at the beginning of this section, the measures $\mathcal{S}$, $\mathcal{E}$, $\mathcal{C}$, *dist* and *trav_t* are *distributive*, i.e. super-aggregates can be computed from the sub-aggregates. On the other hand, the aggregate function for $\mathcal{V}$ is *algebraic*. The same applies to *speed*. The first can be computed by using the auxiliary measure $\mathcal{C}$, while the latter is computed by using the auxiliary measures *dist* and *trav_t*.

Example 4.3 *Figure 4.7(b) focuses on the same temporal granule g_T of Figure 4.7(a) whereas the spatial granularity is made coarser by considering two granules $g_A = g_{e1} \cup g_{e4} \cup g_{e6} \cup g_{e8}$ and $g_B = g_{e2} \cup g_{e3} \cup g_{e5} \cup g_{e7}$*

Trajectory T_{id1} starts from the granule g_A, traverses the granule g_B, and finally ends inside the granule g_A, whereas T_{id2} begins in the granule g_B and ends in the granule g_A. Thus
$\mathcal{V}^{\mathsf{T}}((g_T, g_A), U) = 3$
$\mathcal{V}^{\mathsf{T}}((g_T, g_B), U) = 2$
$\mathcal{C}^{\mathsf{T}}((g_T, g_A), (g_T, g_B), U) = 1$
$\mathcal{C}^{\mathsf{T}}((g_T, g_B), (g_T, g_A), U) = 2$

Now we want to apply Proposition 4.2 to compute $\mathcal{V}^{\mathsf{T}}((g_T, g_A), U)$. The granules composing g_A are $g_{e1}, g_{e4}, g_{e6}, g_{e8}$ and
$\mathcal{V}^{\mathsf{T}}((g_T, g_{e1}), U) = 1$
$\mathcal{V}^{\mathsf{T}}((g_T, g_{e4}), U) = 2$
$\mathcal{V}^{\mathsf{T}}((g_T, g_{e6}), U) = 2$
$\mathcal{V}^{\mathsf{T}}((g_T, g_{e8}), U) = 2$
The only non-null crosses between granules contained in g_A are
$\mathcal{C}^{\mathsf{T}}((g_T, g_{e1}), (g_T, g_{e4}), U) = 1$
$\mathcal{C}^{\mathsf{T}}((g_T, g_{e4}), (g_T, g_{e6}), U) = 1$
$\mathcal{C}^{\mathsf{T}}((g_T, g_{e4}), (g_T, g_{e8}), U) = 1$
$\mathcal{C}^{\mathsf{T}}((g_T, g_{e6}), (g_T, g_{e8}), U) = 1$
Hence by summing the $\mathcal{V}$'s and subtracting the $\mathcal{C}$'s we obtain $7 - 4 = 3$, which is the exact value for $\mathcal{V}^{\mathsf{T}}((g_T, g_A), U)$.

4.5 Another measure: Presence

A different measure from $\mathcal{V}$ is the so called measure *Presence*, i.e. the number of *distinct* trajectories belonging to a certain object group travelling in a given spatio-temporal granule. The difficulty of handling this quantity inside a data warehouse is related to the fact that the aggregate function for *Presence* is *holistic*: the raw data are needed to compute the exact result at all granularities. This is due to a particularity of trajectory data: a trajectory might span multiple granules. Hence in the aggregation phase we have to cope with the well-known *distinct count problem* [70]: if an object remains in the query region for several timestamps during the query interval, one should avoid to count it multiple times in the result. This is problematic since, once loaded in our Trajectory Data Warehouse, the identifiers of the trajectories are lost.

In order to better understand the distinct count problem, observe Figure 4.8. The picture shows two different scenarios where a trajectory crosses four different spatial cells in two different ways, showing the two different kind of errors that can be introduced while computing the measure *Presence* during a roll-up, because

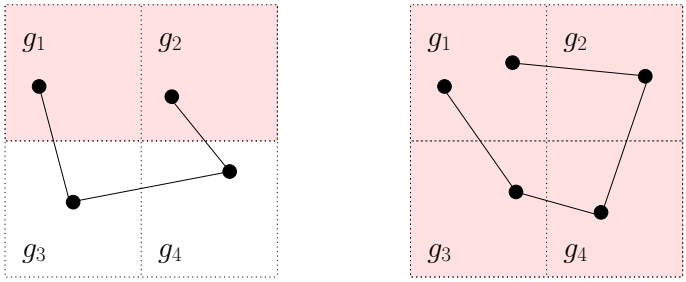


Figure 4.8: Overestimate of *Presence* (a), and underestimate of *Presence* (b) during the roll-up.

of the distinct count problem. We will refer to a given temporal granule g_T. In Figure 4.8(a), we have that

$$\mathcal{P}^{\mathsf{T}}((g_T, g_1), U) = \mathcal{P}^{\mathsf{T}}((g_T, g_2), U) = \mathcal{P}^{\mathsf{T}}((g_T, g_1), U) = \mathcal{P}^{\mathsf{T}}((g_T, g_1), U) = 1$$

If we group together the granules g_1 and g_2, we obtain, by applying the same aggregate function we proposed for $\mathcal{V}$, that the number of distinct trajectories is

$$\mathcal{P}^{\mathsf{T}}((g_T, g_1), U) + \mathcal{P}^{\mathsf{T}}((g_T, g_2), U) - \mathcal{C}^{\mathsf{T}}((g_T, g_1), (g_T, g_2), U) = 1 + 1 - 0 = 2$$

This is an overestimate of the number of distinct trajectories. On the other hand, in Figure 4.8(b), if we group together g_1 and g_2 we correctly obtain

$$\mathcal{P}^{\mathsf{T}}((g_T, g_1), U) + \mathcal{P}^{\mathsf{T}}((g_T, g_2), U) - \mathcal{C}^{\mathsf{T}}((g_T, g_1), (g_T, g_2), U) = 1 + 1 - 1 = 1$$

and similarly by aggregating g_3 and g_4. However, if we group $g_1 \cup g_2$ with $g_3 \cup g_4$ we obtain

$$\mathcal{P}^{\mathsf{T}}((g_T, g_1 \cup g_2), U) + \mathcal{P}^{\mathsf{T}}((g_T, g_3 \cup g_4), U) - \\ \mathcal{C}^{\mathsf{T}}((g_T, g_1 \cup g_2), (g_T, g_3 \cup g_4), U) = 1 + 1 - 2 = 0$$

This is an underestimate of the number of distinct trajectories.

Let us now formally define the measure *Presence* ($\mathcal{P}$).

Definition 4.9 *(Presence)*
Let G be a spatio-temporal granularity and $g \in G$ be a granule, let T be a set of trajectories and U be an object group, then

$$\mathcal{P}^{\mathsf{T}}(g, U) = |\{id \in U \mid \delta(\mathsf{T}_{id}) = \langle s^1_{id}, \ldots, s^n_{id} \rangle \wedge \\ \exists\, i \in \{1, \ldots, n\}.\, s^i_{id} \subseteq g\}|$$

This measure differs from $\mathcal{V}$ since multiple visits of the same trajectory to granule g are counted once.

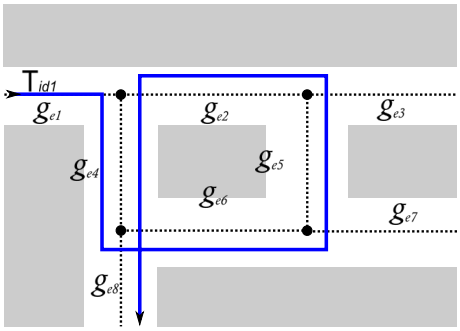


Figure 4.9: *Presence* and *Visits* for an agile trajectory

Example 4.4 *Consider the Figure 4.9, where we have a trajectory spanning different road segments in the same temporal granule g_T. By looking at the granule g_{e4}, we have that $\mathcal{P}^{\mathsf{T}}((g_T, g_{e4}), U) = 1$ whereas $\mathcal{V}^{\mathsf{T}}((g_T, g_{e4}), U) = 2$, since the trajectory comes back in the segment two times.*

4.5.1 Approximating the measure Presence

Example 4.4 highlights how the measure *Presence* is always lower or equal to the value of the measure *Visits*. Informally, given a spatio-temporal granule g, and a trajectory T, it is simple to note that if the trajectory is present on the granule g, the same trajectory should also have visited it (i.e. it has firstly appeared in the granule or it has entered in it). On the other hand, if the trajectory enters and exits from the granule g many times, the visits for that granule are highs, but since it has been visited by always the same trajectory, the amount of the objects visiting it is just one. Starting from this simple intuition, we propose the use of the measure *Visits* as an approximation for the holistic measure *Presence*. From the definitions of this two measures $\mathcal{V}$ and $\mathcal{P}$ it is immediate to see that the first one is an upper bound for the second. In particular, the following proposition holds.

Proposition 4.3 *Let G be a spatio-temporal granularity and $g \in G$ be a granule, let T be a set of trajectories and U be an object group, then $\mathcal{P}^{\mathsf{T}}(g, U) \leq \mathcal{V}^{\mathsf{T}}(g, U)$.*

It is worth noting that the errors in the computation of $\mathcal{P}$, i.e. in our method for approximating the holistic measure *Presence*, are larger when trajectories are very *agile*, i.e., they frequently change their directions. Errors are produced because the same trajectory can get back to a granule that it already visited. This phenomenon disappears when we increase the size of granules by rolling-up, since at some point all trajectories will be completely contained in a granule. This fact is formalised in the following proposition.

Proposition 4.4 *Let G be a spatio-temporal granularity and $g \in G$, let T be a set of trajectories and let U be a user group, then*

1. *if each trajectory visits g at most once then $\mathcal{P}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}}(g,U)$;*
2. *if all the trajectories are inside g then $\mathcal{P}^{\mathsf{T}}(g,U) = \mathcal{V}^{\mathsf{T}}(g,U)$.*

Proof. The first statement is a straightforward consequence of definitions $\mathcal{V}$ and $\mathcal{P}$.

In order to prove the second statement observe that if all the trajectories are inside the same granule, then there are no trajectories crossing its border. Hence $B^{\mathsf{T}}(g,U) = 0$. As a consequence of this $\mathcal{S}^{\mathsf{T}}(g,U) = \mathcal{E}^{\mathsf{T}}(g,U) = \mathcal{P}^{\mathsf{T}}(g,U)$. So, by Proposition 4.1, we have that

$$\mathcal{V}^{\mathsf{T}}(g,U) = \frac{\mathcal{S}^{\mathsf{T}}(g,U) + \mathcal{E}^{\mathsf{T}}(g,U)}{2} = \frac{2\mathcal{S}^{\mathsf{T}}(g,U)}{2} = \mathcal{P}^{\mathsf{T}}(g,U)$$

□

According to the first statement if any trajectory visits a granule g at most once then *Presence* is computed correctly by computing $\mathcal{V}$. In particular, if $\mathcal{V}^{\mathsf{T}}(g,U) = 0$ then $\mathcal{P}^{\mathsf{T}}(g,U) = 0$. The second statement suggests that for coarse granularities *Presence* could be computed correctly.

Approximation errors

In order to demonstrate the level of accuracy of the proposed approximation method, we have computed some tests comparing our proposal with other approaches proposed in literature, namely FM sketches [70] and the distributive function that simply sums the *Presence* at the finer levels to compute the super-aggregate at a coarser granularity (e.g., [56]). The results have highlighted that *Visits* provides a very accurate estimate of *Presence*. In the following paragraph it will be first briefly presented the Flajolet-Martin algorithm for the estimation of the number of distinct objects in a dataset, and then our experiments will be explained.

FM sketches The FM algorithm is a bitmap-based algorithm devised by Flajolet and Martin [21] that can be used to estimate the number of distinct items in a set using a limited amount of memory. Many methods in the literature are based on this algorithm. The algorithm makes use of a hash function h which take in input an object identifier i (trajectory identifiers in our case), and gives in output a pseudo-random integer number $h(i)$ with a geometric distribution, that is, $p[h(i) = v] = 2^{-v}$ for $v \geq 1$. A *sketch* consists of a bitmap of length r, whose bits are initially set to 0. For every object, the algorithm set the $h(i)$-th bit of the sketch to 1.

After processing all objects, the most simple version of FM algorithm finds the first bit of the sketch that is still unset. Hence it approximate the number of distinct objects in the dataset with the value $n = 1.29 \times 2^k$, where k is the position of the aforementioned leftmost unset bit. Unfortunately, this approach may entail large errors in the count approximation, frequently off by a factor of two or more [21].

For this reason, the authors of [21] propose the adoption of m sketches that use different and independent hash functions, and averaging the resulting values. Let $k_1, k_2, \ldots, k_m$ be the positions of the first unset bit in the m sketches, then the new estimate of n is 1.29×2^{k_a}, where $k_a = (1/m) \sum_{i=1}^{m} k_i$. In this case, the expected processing cost of each object increase from $O(1)$ to $O(m)$. To solve this problem and to have again a constant cost, the authors of FM algorithm propose to use another algorithm, the *Probabilistic Counting with Stochastic Averaging (PCSA)*. This new algorithm applies a second hash function to choose one of the m sketches and only insert the object into that sketch, reducing the expected insertion cost. Thus each sketch becomes responsible for approximately n/m (distinct) objects, resulting in the new formula for estimation $n = 1.29m \times 2^{k_a}$, with an expected standard error of $O(m^{-1/2})$. Algorithm 1 shows the pseudo-code of the FM algorithm with PCSA.

Algorithm 1 Probabilistic Counting with Stochastic Averaging Algorithm for FM Sketches

INPUT: DS is a dataset; h is a random function such that, given an object $i \in DS$, $p[h(i) = v]] = 2^{-v}$; m is the number of sketches used; r is the number of bits in each sketch.

OUTPUT: the estimated number of distinct objects in DS

```
1: initialize m sketches s_1, s_2, ... s_m, each with r bits set to 0
2: for all do i ∈ DS
3:     randomly pick a sketch s_j (1 ≤ j ≤ m) s_j[h(i)] ← 1
4: end for
       k ← 0
5: for j = 1 → m do
6:     for q = 1 → r do
7:         if s_j[q] = 0 then
8:             k ← k + q
9:             break                          ▷ go to the next sketch
10:        end if
11:    end for
12: end for
13: return 1.29 * m * 2^(k/m)
```

As shown in [21], a proper value for the number r of bits of each sketch is $\log UB$, where UB is an upper bound on the number n of distinct objects.

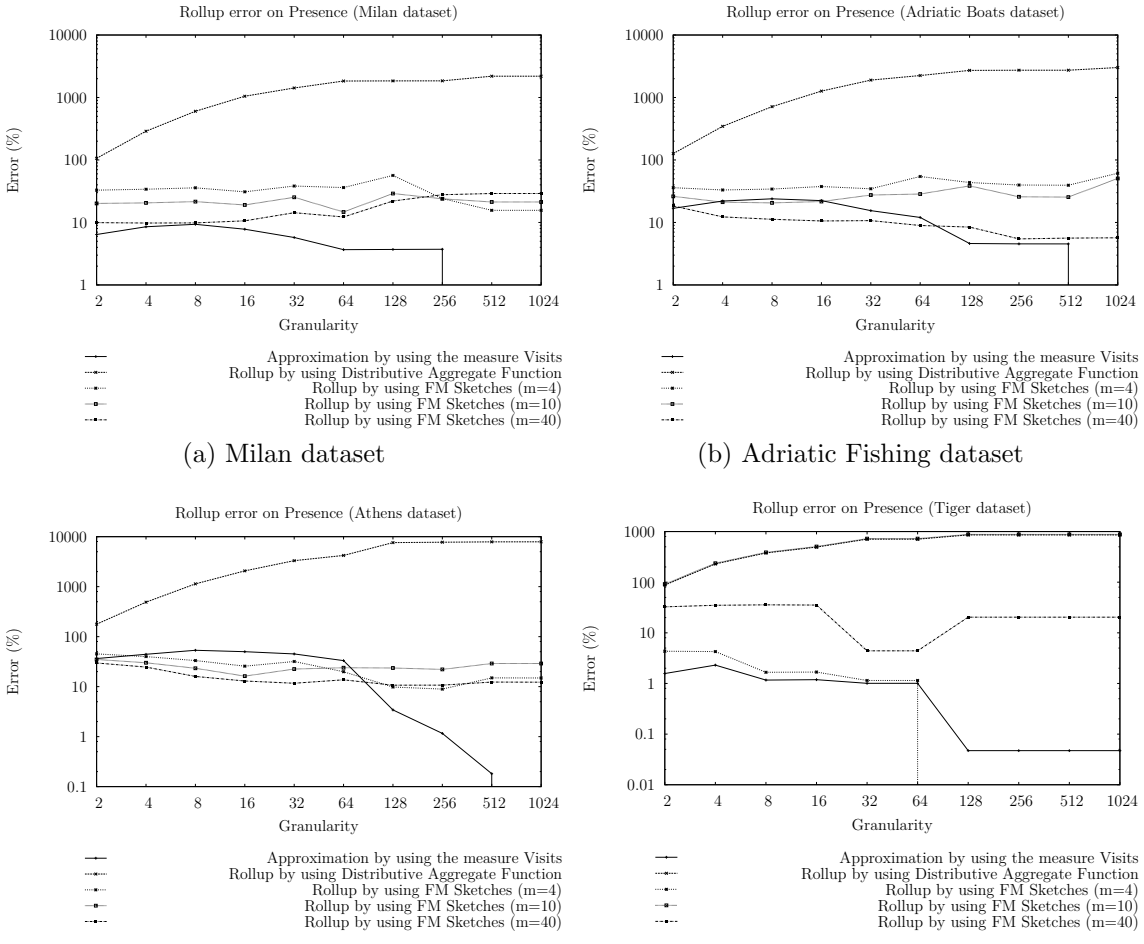
Interestingly, FM sketches can be merged in a *distributive* way. Suppose that a pair of sketches are updated according to the IDs of the objects contained in a different set, and that the intersection of those sets is possibly not empty. The sketch obtained as the *bitwise-OR* of the corresponding bitmaps in the original sketches will be identical to the one directly updated using the union of the sets of items.

Quantitative evaluation An experimental comparison of the various approximations, reported in Figure 4.10, has been obtained by using some different dataset.

In order to compare the errors we chose to adopt as an aggregation accuracy metric the normalised absolute error defined as follows:

$$Error = \frac{\sum_g Error(g)}{\sum_g g.Pres} = \frac{\sum_g |\widetilde{g.Pres} - g.Pres|}{\sum_g g.Pres}$$

where g are granules at a coarser granularity than the base one, $g.Pres$ is the exact value of *Presence* in the granule g whereas $\widetilde{g.Pres}$ is the approximated value obtained using one of the discussed methods, i.e. FM sketches, distributive function and the measure $\mathcal{V}$.



(a) Milan dataset

(b) Adriatic Fishing dataset

(c) Athens dataset

(d) Tiger dataset

Figure 4.10: Cumulative error of roll-up phase

The first dataset, whose results are presented in Figure 4.10(a), contains information about cars moving in the city of Milan. An in-depth description of the

dataset will be provided in Section 6.1. The dataset contains about 200,000 trajectories moving along the city during a week. The spatial extent of the trajectories has been divided by a regular grid having as base granularity some rectangles, of size $330m \times 440m$, while the time extent has been divided into intervals of 1 hour.

The second dataset, used for generate the graph in Figure 4.10(b), contains information by some fishing boats moving on the Adriatic Sea between January and September 2007. The dataset contains about 33,000 trajectories, while a more detailed description of it will be found in Section 6.2. In this case, the spatial extent of the dataset has been divided by a regular grid of about 430×315 cells, having each a dimension of about $2.5km \times 1.5km$, while the temporal extent has been split into 1 days intervals.

Figure 4.10(c) refers to a third dataset[1]. This dataset is composed by 1100 trajectories obtained by 50 trucks transporting concrete in the area of Athens between August and September 2002. The dataset is composed by 112,300 position records consisting on the trucks identifiers, dates and times, and geographical coordinates. Trajectories have been obtained by splitting the recordings of a truck in subsets if there was a temporal gap between two consecutive recordings larger than 15 minutes, a gap that can indicate a stop of the vehicle not due to traffic or traffic lights. The spatial extent of the dataset is of about $45km \times 55km$ that has been divided by a regular grid of about 150×180 cells. The time intervals have been divided into 3-hours gaps.

Finally, graph in Figure 4.10(d) has been obtained by using a synthetic dataset. This dataset has been generated by using the San Joaquin country road map (US Census TIGER/LINE, country code 06077), through the traffic simulator described in [8]. It is the largest dataset among the four we use for our tests, and it is composed by about 151,000 trajectories and more than 10,000,000 raw points. The spatial extent of the dataset is of about $660,000 \times 720,000$ spatial units, while the moving objects have been monitored for a time of 300 temporal units. The spatial extent has been divided into a grid of 110×120 cells, while time extent has been divided into 20 intervals.

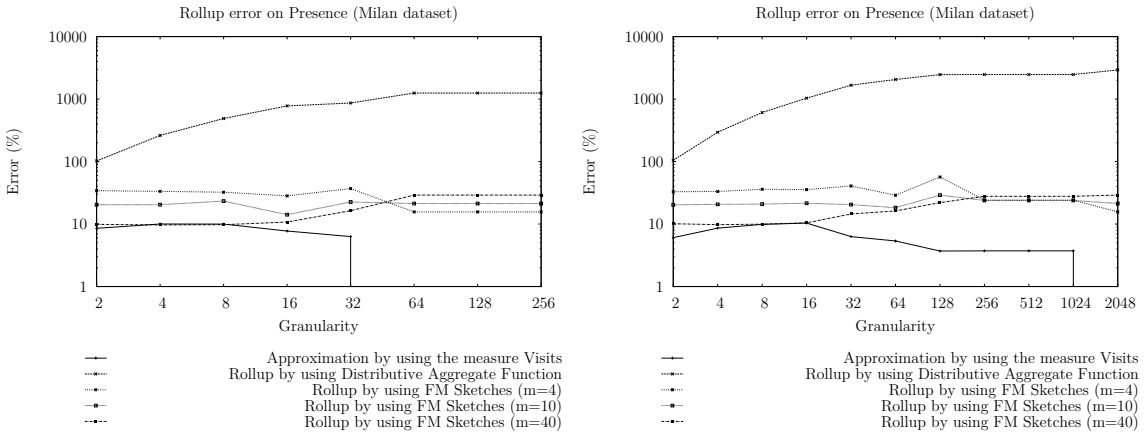
The graphs in Figure 4.10 show the normalised absolute errors as functions of the granularities. The values indicated for the granularities are relative to the base one. For example, a value 2 for granularity states that we are considering granules having double size with regard to the base granules along all dimensions. The different curves for FM sketches correspond to a different number, m, of bit vectors (i.e. four, ten and forty).

As shown by the corresponding curves, the *distributive* aggregate function (the top curve) quickly reaches very large errors for all the dataset, as the roll-up granularity increases. This is due to the fact that we simply sum the sub-aggregates and as a consequence trajectories crossing different granules are counted many times: the number of duplicates becomes higher and higher at coarser granularities.

[1] `http://www.chorochronos.org/Default.aspx?tabid=71&iditem=44`

Looking at the graph in Figure 4.10(a), we can note that, for all granularities, $\mathcal{V}$ also outperforms FM sketches and, for coarser ones, $\mathcal{V}$ is no longer an approximation but it coincides with the measure *Presence*. On the other hand, on the graphs for the dataset of Adriatic fishing boats (Figure 4.10(b)) and Athens trucks (Figure 4.10(c)), this no longer holds for every granularity. In particular, the errors obtained by the approximation of the value of the *Pres* with the measure $\mathcal{V}$ and with the FM algorithm are quite similar for low granularities aggregations, with the FM algorithm having lower errors for one (Figure (b)) or all (Figure (c)) the curves. However, increasing the number of base cells aggregated together, the error made by sketches tend to be constant, while the one due to the approximation with the measure $\mathcal{V}$ decrease constantly till reaching zero. The only situation in which this does not happen is the one reported in Figure 4.10(d). In this case it is interesting to note that the relative errors for all the approximation methods reported in the graph are lower than those in the other three dataset. Other than this, we can also note that the approximation obtained by using the FM algorithm with only four sketches, outperform the same methods using higher number of bit vectors.

In order to better understand the differences on the errors for the last two dataset, Athens trucks and Tiger, we try to analyse the characteristics of them. The Athens dataset contains long cyclic trajectories. This means that trajectories goes around and often visit the same places, so there are a lot of duplicates identifier per cells. On the other hand, Tiger dataset contains very long trajectories that rarely visit the same places, hence duplicates are very few.



(a) $580m \times 770m$ cells grid (b) $250m \times 330m$ cells grid

Figure 4.11: Cumulative error of roll-up phase for different granularities

Figure 4.11 presents two graphs obtained by varying the granularities for the Milan dataset. The first picture has been obtained by increasing the base grid of the experiments. In particular the minimum temporal interval has been increased to 3 hours, while the spatial cells has been defined to be of about $580m \times 770m$. The second figure has instead been obtained by reducing the cells size to $250m \times 330m$

(temporal intervals have been left to one hour). By looking at the graphs it is visible how the granules dimension is proportional to the error. In particular, by increasing the granules, the error decrease faster, and vice-versa. This is due to the fact that larger are the granules, lower is the number of crosses for each trajectory, hence the second condition of proposition 4.4 is rapidly verified. On the other hand, in order to reach the stability when the granules are smaller we need to aggregate together more base cells, so the error decrease slowly. Another thing worth noting is that either by changing granularities, the overall distribution of the errors for all the methods compared does not change. In particular each one of the analysed methods reaches a stability condition in more or less time depending on the granules dimension while the values of the errors could be considered equals to each other in every graph. Hence, the quality of the approximate computation of *Presence* depends mainly on the features of the dataset and the level of spatio-temporal hierarchy we refer to.

4.6 Trajectory Data Warehouse Implementation Hints

We would like to briefly describe how the proposed conceptual model has been translated into a logical model.

The actual prototype has been implemented using Oracle™ 11 DBMS suite with the Oracle Spatial extension, needed in order to allow for spatial queries using the visual interface described later in Chapter 5.

The main difference between the proposed conceptual model and the adopted logical model is represented by the fact that in the latter we decided to unify the two fact tables into a single one, introducing some sort of redundancy but making it easier the management of the data and the query process. Figure 4.12 presents a simplified version of the actual implementation of our TDW, where only spatial and temporal dimensions are taken into account. Other dimensions could be added by defining a new table having a similar schema to the proposed one.

The *SPATIAL_GEOM* table contains the Oracle *SDO_GEOM*[2] objects representing the various spatial partitions available in the spatial hierarchy. In particular, we decided to store all the spatial areas instead of just the base ones, in order to avoid difficult spatial join queries to determine the geometries of higher spatial granules. This is possible since each hierarchy must be defined when instantiating the TDW conceptual model for a certain scenario. With respect to Figure 4.13(a), the geometries of all the spatial objects represented in the picture are stored in the table.

The two dimensional tables *SPATIAL_DIM* and *TEMPORAL_DIM* define the hierarchies of the spatial and temporal dimensions respectively. Each column of the tables represents one level of the hierarchies, starting from the base level, which

[2] `http://docs.oracle.com/cd/E11882_01/appdev.112/e11830/sdo_objgeom.htm`

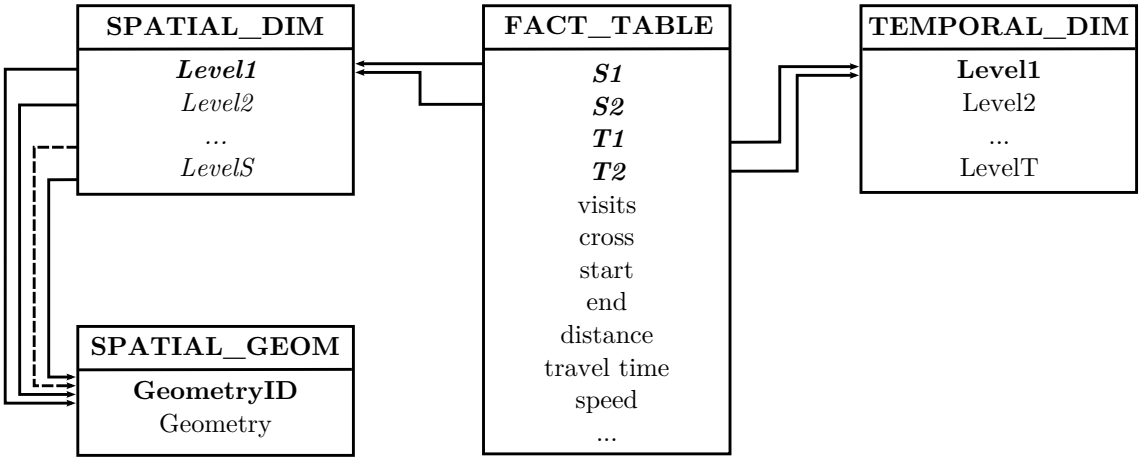


Figure 4.12: TDW Logical Model

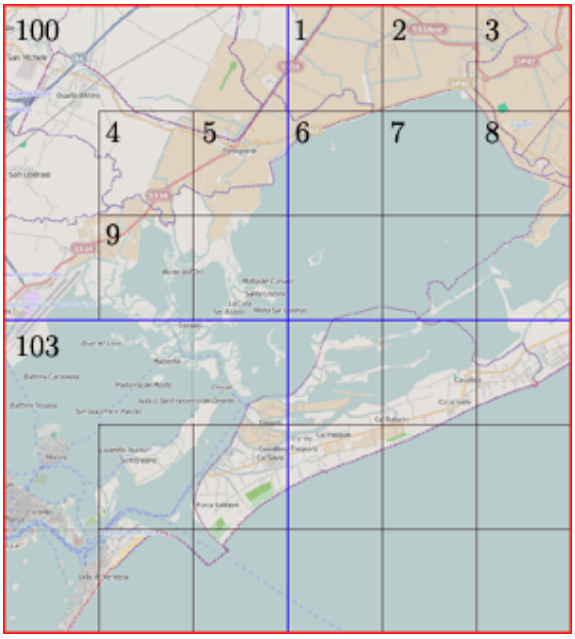
is the primary key of the tables. For example consider the first tuple of the *SPATIAL_DIM* table shown in Figure 4.13(b), $(1, 101, 1000)$. The first value of the tuple represents the identifier of one of the geometries available in the *SPATIAL_GEOM* table. This geometry, g_1 (i.e. the geometry having identifier 1 in Figure 4.13(a)), is a base granule of the defined spatial hierarchy. The second value of the tuple, 101, represents the identifier of another geometry contained in the *SPATIAL_GEOM* table, g_{101} (i.e., geometry with id 101 in Figure 4.13(a)). This geometry represents the father of the base granule g_1 in the given hierarchy. This means that g_{101} is a geometry that belongs to the second level of the defined hierarchy and completely contains g_1. The same holds for the third value, 1000, representing the geometry g_{1000}, that belong to the third level of the defined hierarchy and completely contains g_{101} (note that in the picture g_{1000} is the geometry depicted in red and containing all other geometries, and its identifier is not written). It is worth noting that only geometries containing information are stored in the table. The same schema is applied to each other tuple, and to all the available dimensions.

The fact table contains the values of the various measures available for couples of base granules. We distinguish two kinds of tuples in the fact table:

1. if the granules $(S1, T1)$, $(S2, T2)$ are equals, then the tuple contains intra-granule measures for the granule $(S1, T1)$ and the *Cross* value (the only inter-granule measure), is set to 0;

2. otherwise, i.e. $(S1, T1) \neq (S2, T2)$, the tuple stores the inter-granule measure *Cross*, and the intra-granule measures are set to 0.

In the second case we can consider three possibilities. If $S1 = S2$ and $(T1 \neq T2)$, the information associated to the tuple refers to a certain spatial geometry $(S1)$, but to two temporal intervals, $T1$ and $T2$. If $S1 \neq S2$ and $(T1 = T2)$, the information associated to the tuple refers to two different spatial geometries ($S1$ and $S2$), but



(a) Example of simple spatial granules

SPATIAL_DIM		
Level1	**Level2**	**Level3**
1	101	1000
2	101	1000
3	101	1000
4	100	1000
5	100	1000
6	101	1000
7	101	1000
8	101	1000
9	100	1000
...	...	...

(b) Example of spatial dimension table

Figure 4.13: A spatial hierarchy (a) and its representation into our TDW implementation (b)

during the same temporal interval $T1$. Finally, if both $S1 \neq S2$ and $(T1 \neq T2)$, the information associated to the tuple refers to two different spatial geometries ($S1$ and $S2$), during two temporal intervals ($T1$ and $T2$). It is worth noting that this representation is general enough to allow for different kinds of inter-granule measures. For instance we can store measures between granules which are not adjacent as required by our *Cross* measure. In order to clarify the aforementioned concepts, Figure 4.14 presents an example of a possible fact table, referring to the spatial hierarchy proposed in Figure 4.13(a). Entries in the table refer to both intra-granule measures, as the first three rows, or inter-granule measures, as the last two. It is worth noting that rows related to the first kind of measures contain a 0 value in the *Cross* columns, while rows modelling to inter-granule measures contain a 0 value for columns *Visits* and *Distance*. Taking the first tuple of the table, we are able to know that the spatial granule g_1 has been visited 3 times during the time interval $t1$, and trajectories visiting it travel for about 2.5Km during this time. On the other hand, looking at the last tuple of the table we can know that one trajectory remained travelling in g_1 during the first and second time intervals.

Aggregation The proposed logical model allows also for a somehow easy way to construct aggregate values during the roll up operation. The task is made by a standard SQL query, containing a SQL UNION operation. The query in the case

FACT_TABLE							
S1	**S2**	**T1**	**T2**	**Visits**	**Cross**	**Distance (Km)**	**...**
1	1	1	1	3	0	2.5	...
1	1	2	2	4	0	3.7	...
1	1	3	3	1	0	6	...
1	9	3	4	0	2	0	...
9	9	2	2	6	0	4.1	...
4	5	3	3	0	4	0	...
3	3	2	2	5	0	1.6	...
2	2	8	8	2	0	1.1	...
1	2	1	1	0	2	0	...
1	1	1	2	0	1	0	...
...	...	...	...	...	...	...	...

Figure 4.14: Example of fact table

of the model proposed in Figure 4.12 is the following, where s_gran_0 represents the base granularity for spatial hierarchy, t_gran_0 represents the base granularity for temporal hierarchy, and s_gran_S and t_gran_T represent the granularity level of the desired aggregation, respectively over space and time.

```
SELECT s1.s_gran_S AS S1,
    s1.s_gran_S AS S2,
    t1.t_gran_T AS T1,
    t1.t_gran_T AS T2,
    SUM(f.Visits) - SUM(f.Cross) AS Visits,
    0 AS Cross,
    SUM(f.Distance) AS Distance
FROM SPATIAL_DIM s1, SPATIAL_DIM s2,
    TEMPORAL_DIM t1, TEMPORAL_DIM t2,
    FACT_TABLE f
WHERE f.S1 = s1.s_gran_0
    AND f.S2 = s2.s_gran_0
    AND f.T1 = t1.t_gran_0
    AND f.T2 = t2.t_gran_0
    AND s1.s_gran_S = s2.s_gran_S
    AND t1.t_gran_T = t2.t_gran_T
GROUP BY t1.t_gran_T,
    s1.s_gran_S
```

```
UNION ALL
SELECT s1.s_gran_S AS S1,
    s2.s_gran_S AS S2,
    t1.t_gran_T AS T1,
    t2.t_gran_T AS T2,
    0 AS Visits,
    SUM(f.Cross) AS Cross,
    0 AS Distance
FROM SPATIAL_DIM s1, SPATIAL_DIM s2,
    TEMPORAL_DIM t1, TEMPORAL_DIM t2,
    FACT_TABLE f
WHERE f.S1 = s1.s_gran_0
    AND f.S2 = s2.s_gran_0
    AND f.T1 = t1.t_gran_0
    AND f.T2 = t2.t_gran_0
    AND (s1.s_gran_S <> s2.s_gran_S
    OR t1.t_gran_T <> t2.t_gran_T)
GROUP BY t1.t_gran_T,
    t2.t_gran_T,
    s1.s_gran_S,
    s2.s_gran_S
```

The aggregation query could be divided into two parts corresponding to the two multisets that the union statement puts together. The first part computes the aggregate values for intra-granules measures, while the second one deals with inter-granule measures. In the two queries, we do not distinguish between tuples for intra-granule or inter-granules measures, by summing up all the available data. This is because every tuple in the fact table sets to 0 measures of one or the other kind, as we already stated. Hence, summing up their values does not introduce any issue during the computations. In order to obtain the willing results, we need to duplicate each dimensional table in the query, and since we pre-compute all the spatial geometries, we do not need to perform spatial queries in order to obtain aggregations for the spatial geometry. It is worth noting that the implementation of the aggregate function is straightforward, starting from their definitions given in this chapter.

4.7 Synopsis

In this chapter we presented the core module of our framework, i.e. the Trajectory Data Warehouse. We proposed an adequate conceptual model that results to be flexible and allows for representing different kinds of scenarios.

As far as data warehouse dimensions are concerned, the model includes two main dimensions, spatial and temporal, and a number of other auxiliary dimensions

related to the specificity of the observed objects.

Facts are modelled by two fact tables, intra-granule fact table and inter-granule fact table, respectively containing information related to a single granule or a couple of granules. For each of them we proposed some interesting measures that should be taken into account.

In order to give OLAP capabilities to the model, we defined appropriate aggregate functions for the measures, distinguishing between algebraic, distributive and holistic ones. We formally proved that the given functions correctly compute measures at coarser granularities starting from those at finer levels in the hierarchies. Moreover we gave a detailed discussion of the well-known distinct count problem. We coped with it by approximating the measure *Presence* by using the algebraic measure *Visits*. We provided analytical and experimental estimations of the error introduced by this approximation and we obtained good results.

Finally, we gave some hints about the implementation of the model, providing some information on how the conceptual schema and theoretical aggregate functions have been developed in our system.

5

Visual OLAP with the Visual Analytics Toolkit

A fundamental feature of spatio-temporal data is the fact that this kind of data acquires a meaning in relation to the spatial area they are referred to. This characteristic make it really difficult to grasp useful knowledge by using standard data visualization methods available in normal data warehouses, that have been developed with the aim to show information related to, for example, sales of a shop or production of a factory. Even if standard, table based OLAP operations could be used to investigate also this kind of data, the interpretation of results, and the consequent refinement of queries and exploration of results, is not easy. In particular, representations based on relational tables make it very difficult for a user to grasp the relationships between areas in the same neighbourhood, the evolution in time of spatial areas, or the correlations of different measures. It is really more useful to look at these data in the context they refer to, that means to look at them over a map. Integrating OLAP tools with Geographical Information Systems (GISs) provides advanced analysis capabilities. For instance, trajectory data can be geo-referenced on a map or combined with several layers (such as topographic, demographic, thematic). Performing OLAP operations on TDW specialised measures in a visual way and analyse them with a visual tools make the exploration of the data cube more rapid and intuitive. For these reasons, we have provided the TDW with an interface that allows for OLAP visual operations, based on the Visual Analytics Toolkit [4], an interactive Java™ based geographical information system. This toolkit permits a user to view geo-referenced data over a map and to run analyses on them, for example to find clusters or to tessellate the space. It also offers functionalities to handle temporal data, by using graphs or animations, according to the type of data to analyse.

As stated before, the tool is written in Java™, and so can work on different operating systems. Other than this, the tool supports different kind of database servers and, among others, it can be connected to Oracle™ Server and PostgreSQL server.

In the following sections of this chapter the tool and its functionalities will be presented. In particular, Section 5.1 gives some hints about the software architecture

of the Visual Analytics Toolkit and its implementation. Section 5.2 describes how to connect the tool to a Trajectory Data Warehouse and how to load the data. Section 5.3 discusses about the possible data warehouse operations VA-Toolkit allows to do. Finally, Section 5.4 focuses on different kinds of analysis and visualization methods the tool can perform to TDW data.

5.1 Visual Analytics Tool implementation hints

Visual Analytics Toolkit can be considered as a framework containing many different tools for the manipulation of spatio-temporal data and their visualization. It is an evolution of various software, namely Iris, Descartes and CommonGIS, and it is currently developed by the Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS)[1]. The framework could be customized and extended by programmers in order to add new functionalities to the software, both for visualization methods or tools for data management.

Visual Analytics Toolkit framework is composed by four main modules, each cooperating with the others, as shown in Figure 5.1.

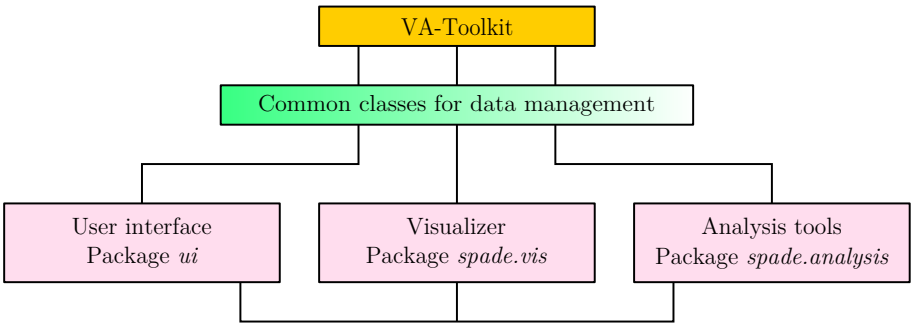


Figure 5.1: VA-Toolkit framework architecture

The *common classes for data management* module is composed by a set of packages and classes useful in order to manage different situations common to other modules, as database connections, use of external tools in order to perform certain operations, and so on.

The *user interface* module is the responsible of the user interaction part of the program. This module defines the windows of the software and manages the events produced by a user, dispatching them to the classes in charge for their processing.

The *visualizer* module deals with the visualization methods available for the data. VA-Toolkit works in an in-memory base, so it needs to have all the data previously loaded into the tool in order for the user to use its visualizations methods and capabilities. Spatial data are associated to a number of different layers, each one containing spatial information of some objects of interest. Each layer is associated

[1] `http://www.iais.fraunhofer.de`

to a table containing information related to the spatial objects in the layer. The visualized objects could be geo-localized figures, signs, or geometries like points, lines, poly-lines, polygons, defined by the classes in the package *spade.vis.geometry*. Different methods of data representation on a map are realized by writing Java™ classes that implement a given *Visualizer* interface in the package *spade.vis.mapvis*. The classes have the task of defining how to visualize the data they are associated with, on a map. The basic idea is that a visualizer can be attached to a map layer, which includes a collection of geographical objects, and that each object on the layer is responsible to draw itself on the map. Hence, when the system detect that a layer need to be drawn or refreshed, the visualizer associated with the layer is given as parameter to each object on it, and each object draws itself through the appropriate methods of the visualizer. Some examples of visualizers defined in the tool will be presented in Section 5.4.

As said before, in order to produce visualizations for some spatial objects, the layer containing the objects needs to be associated with a data table containing the information that has to be shown. These data are called *thematic data*. For example, if a layer contains the geometry representation of the regions of a state, and the user wants to visualize the average speed of vehicles in each region, the data table associated with the objects layer needs to contain such information. Thematic data are organized into tables, implemented by the class *spade.vis.database.DataTable*. Each table row corresponds to an object of the layer, while each column represents a different information, called attribute, related to it. Figure 5.2 illustrates an example of a VA-Toolkit table containing data related to various Italian regions, as their population, their area, their average height above mean sea level, and so on.

Country	**Total population**	**Municipalities #**	**Height (amsl)**	**Area (m^2)**	**Density ($/km^2$)**	**GDP (billion €)**
Abruzzo	1342975	305	563	10753	124.89	28.7
Basilicata	586853	131	633	9992	58.73	11.4
...	...	...	...	...	...	...

Figure 5.2: VA-Toolkit thematic data table example

A special case of thematic data is time-dependent thematic data also called time-series. In this case, each attribute associated with a spatial object is defined by several values, one for each time moment. For example, suppose to have the population data of different countries in the world, referred to different years, 2000, 2001, 2002. In this case, the internal representation of the data table becomes slightly different.

In particular the attribute associated with the spatial objects, i.e. the population, becomes what is called a *super-attribute* (or top-level attributes), while each table column represents the values of this attribute for a given time moment (low-level attributes). The same structure is used for any parameter-dependent attributes or multi-parameters attributes, whether a *parameter* could be considered as something that specialize the information contained into an attribute. For instance, in the case of population, a parameter could be the year (temporal), or the gender and age this information is related to (i.e. population of males between 14 and 16 years old). This is a common situation for data warehouses, where a measure depends on different dimensions. Attributes are implemented by the class *spade.vis.database.Attribute*, and could be of type *integer*, *real*, *character*, *logical* or *temporal*. Apart the last one, all the attributes value are stored in the system as strings, while numerical values are also stored as floats. On the other hand, temporal values are instances of classes implementing the interface *spade.time.TimeMoment*, and currently could be both *spade.time.TimeCount* for simple values stored as integers, or *spade.time.Date* for more complex values representing full dates. Parameters are implemented by the class *spade.vis.database.Parameter*, and could be of any type. Figure 5.3 describes a possible internal table representation of some data related to the population of several states, by gender, age and year. As one can notice, the top-level attribute is the *population*, while the low-level ones describe the possible combination between the three parameters.

	Population (*top-level attribute*)							
Country	**Gender: *M* Age: *<10* Year: *2000***	**Gender: *M* Age: *<10* Year: *2001***	**Gender: *M* Age: *<10* Year: *2002***	**Gender: *F* Age: *<10* Year: *2000***	**Gender: *F* Age: *<10* Year: *2001***	**Gender: *F* Age: *<10* Year: *2002***	**Gender: *M* Age: *10–20* Year: *2000***	**...**
Albania	12630	36142	17536	11536	21363	17856	14536	...
Austria	10256	21536	16321	15236	11236	14536	21236	...
...	...	...	...	...	...	...	...	...

Figure 5.3: Representation of thematic data table with parameters in VA-Toolkit

Finally, a table may contain values of several parameter-dependent (and, in particular, time-dependent) attributes, for example, “Property crime rate”, “Burglary rate”, “Motor vehicle theft rate”, etc. In this case, the table will have a list of super-attributes corresponding with those that have to be stored, each of them associated with a collection of low-level attributes, one for every value the attribute can take. Obviously, it is also possible that a table has a mixture of simple and parameter-dependent attributes.

While the visualizer module is in charge of the visualization of the data stored in the system, the *analysis tools* module gives the system the abilities to manipulate

the data stored in it. In particular a programmer can add a tool to the framework in order to extend its capabilities of data analysis. Currently available tools permit to load data from a database, to perform clustering operations to the data, to explore and load data referred to moving objects (i.e. trajectories), or more simpler, they allow one to export spatial data to be seen in Google Earth, or to save different frames of an animation as PNG pictures. In this module we inserted a new tool in order to add to the system the functionalities for manage the trajectory data warehouse presented in Chapter 4. This tool is designed in order to permit to load the data from the TDW into the VA-Toolkit framework to permit their visualization using its capabilities, and to permit a user to easily navigate into the data cube by performing visual OLAP operations.

5.1.1 Trajectory data warehouse integration

We extended the VA-Toolkit framework by adding a new data management tool that allows a user to load a Trajectory Data Warehouse schema, load the data from the data warehouse into the software and supporting in a visual way data warehouse operations like roll-up and drill down, showing the resulting data to a user.

One of the aspect that characterize the TDW model presented in Chapter 4 is that it is very flexible and allows to analyse data related to different scenarios. In particular the model has no limits on the number of measures that can be taken into account or on the number of the dimensions that can be analysed. These characteristics introduce the need to have the same flexibility on the visual tool developed to manage such a TDW. To obtain this, the tool has been divided in different classes. The core class is the *TDWManager* one. This class maintains the information related to the data warehouse schema currently analysed. Given that each scenario can have a different number of tables (i.e. dimension tables), and each table can have a different number of columns (i.e. measures for the fact table and different granularities for each dimension), this class is needed in order to reproduce this flexibility into the tool. Figure 5.4 illustrates the complete classes schema. The manager, shown in the middle of the picture, contains a reference to the main table of the data warehouse, the fact table, two references to the spatial and temporal tables, that are common to each TDW schema, and then a reference to a list of other dimension tables that can be empty or contain a number of table references to other dimensions of the TDW. The fact table includes the list of the measures that are available in the loaded schema, and each measure contains a flag indicating whether it is an intra-granule or an inter-granule measure. Finally each dimension table contains the list of the different granularity levels associated with it.

From the point of view of the implementation, there are some base classes used to implement the whole schema. In particular, the *Dimension* class contains the information related to the dimensions and the pointers to the granularities, implemented by the *Granularity* base class. The *FactTable* and *Measure* classes are two separated classes providing functionalities for the fact table and the measures associated

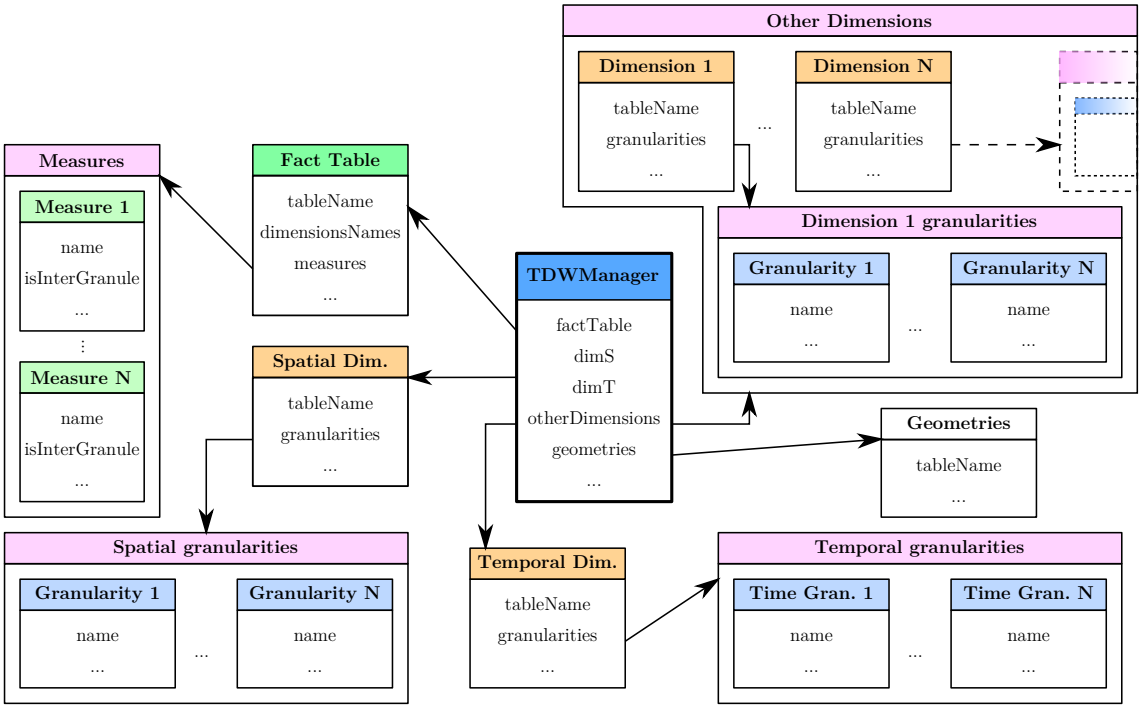
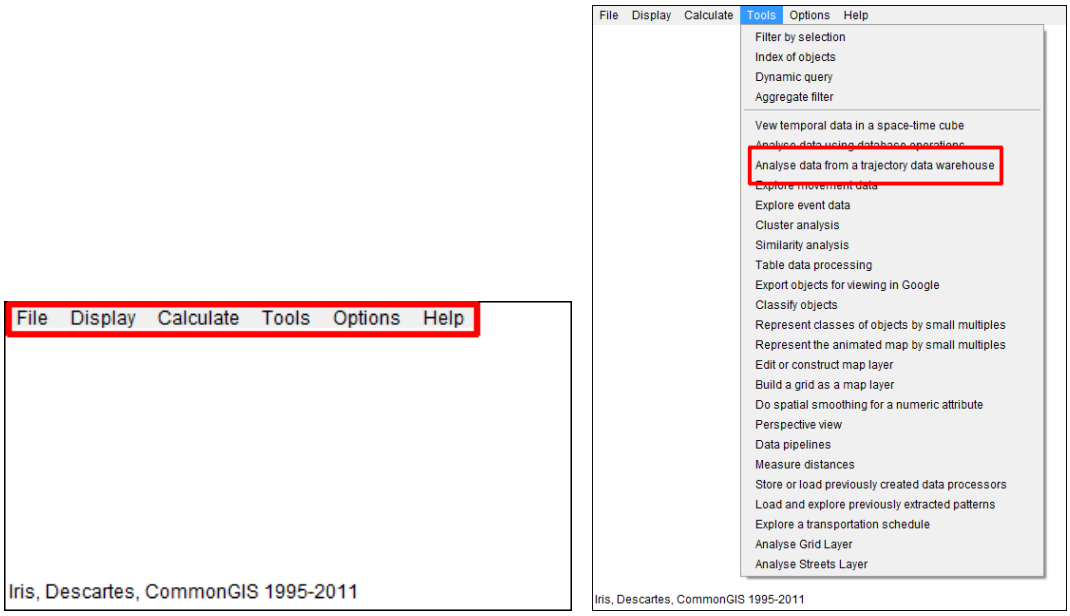


Figure 5.4: VA-Toolkit TDW internal representation

with, such as information on the primary keys of the table, associations between table columns and dimensions, and so on. Two particular cases are represented by the *TemporalDimension* and the *TemporalGranularity* classes, both extending their respective base classes *Dimension* and *Granularity*. These two classes add some features to the base classes in order to make easier the manipulation of temporal data. The motivation behind this is essentially the fact that temporal data could be of different types. In particular we can have a single instance of time, as for example a timestamp, that is a quantitative value, but we can have also time intervals, that need to be treated in a different way, or we can have some quantitative values (as for example month or season names) or quantitative interval values that need to be treated again in a different way. In particular, one can make operations between quantitative values, and it is easy to understand which one comes before another, while it is not so easy if the values are quantitative.

Another important class for the implementation of the new tool is the *TDWAnalyser* one. This class is the bridge between the VA-Toolkit framework and the TDW schema. It is responsible for the management of the physical connection between the system and the TDW, and to execute the needed queries for the data loading or OLAP operations. Moreover, the class manages the user interaction giving him/her the ability to load the data or to make common data warehouse operations such as roll-up and drill down. Since a user can be interested in managing several trajectory data warehouses at the same time, maybe to compare data, the class is able



(a) VA-Toolkit main window (b) VA-Toolkit tools menu

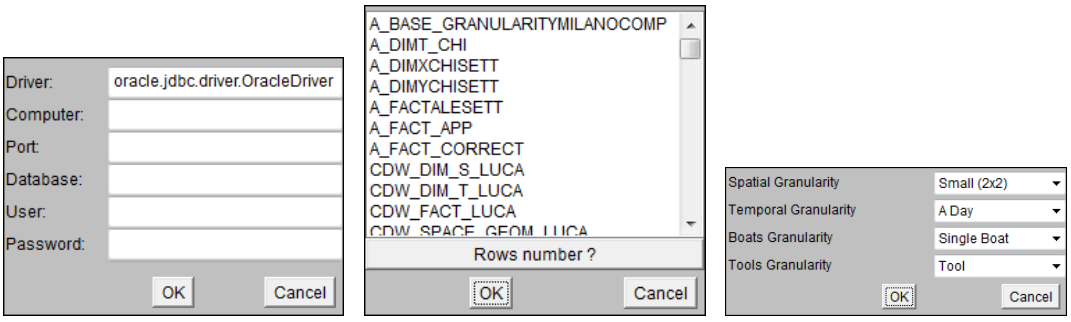
Figure 5.5: VA-Toolkit interface

to handle one or more instances of *TDWManager*, one for each TDW that must be loaded into the system. Finally, the class is in charge to visualize the tool summary window to the user, called *TDW Operations Window*, shown in Figure 5.7(b) and presented in the following Section 5.3.

5.2 Data loading

Once launched the tool, the user is presented with the main window of the program, where the menu with the available operations is shown on top, as illustrated in Figure 5.5(a). Visual Analitycs Toolkit is very modular and contains many different functions. The first step a user needs to do in order to analyse data from a Trajectory Data Warehouse consists of selecting the right tool by using the *Tools* menu (Figure 5.5(b)). This starts the wizard for the data loading process. The user is requested to insert some information related to the TDW s/he wants to connect to. In particular, as shown in Figure 5.6(a), the user is asked to insert the parameters needed for the connection to the data warehouse, such as its address, port, user name and password.

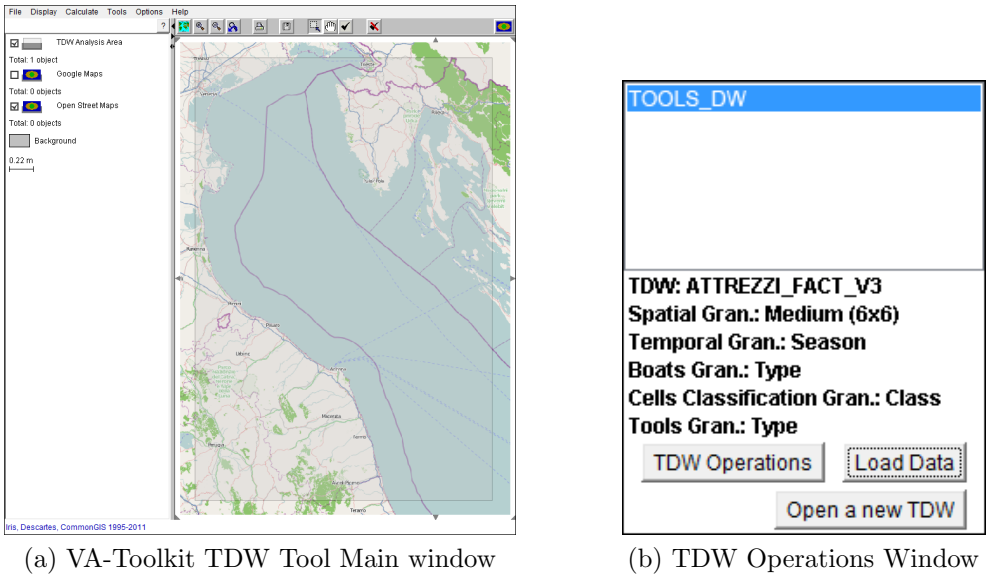
In the next step the user has to select the name of the fact table s/he wants to analyse, from a list (Figure 5.6(b)). At this point the system connects to the selected TDW in order to analyse the model and find out the various measures, dimensions and granularities. After this, the user needs to select, for each dimension, the



(a) TDW Connection Panel (b) TDW Fact Table Selector (c) TDW Granularities Panel

Figure 5.6: VA-Toolkit TDW connection wizard windows

desired granularity that should be loaded (Fig 5.6(c)). The aggregations are then computed, the tool shows a new map of the whole spatial extent of the loaded TDW in its main window (Figure 5.7(a)) and the *TDW operations window* (Figure 5.7(b)) with information about the chosen granularities.



(a) VA-Toolkit TDW Tool Main window (b) TDW Operations Window

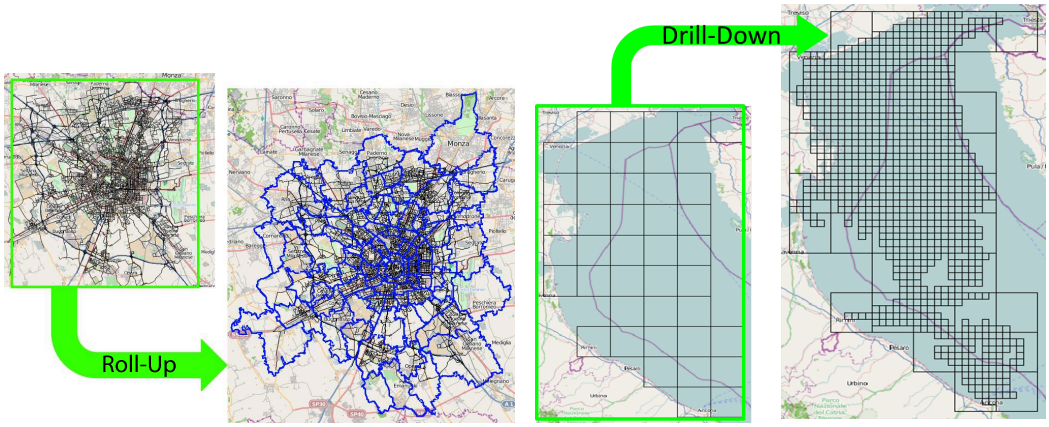
Figure 5.7: VA-Toolkit windows with a loaded TDW

5.3 TDW Operations

One of the key features for a visual OLAP tool is the ability to handle and visualize the different granularities composing the spatio-temporal hierarchy of the trajectory data warehouse, and to give the user the ability to navigate through them. The two main operations in order to explore the different levels of the granularities are

roll-up and *drill-down*. In order to perform these operations, the user should select the *TDW Operations* button in the *TDW Operations Window* (Figure 5.7(b)). The system prompts for the kind of operation, roll-up or drill-down, the user wants to perform. Now the user is prompted with the *Granularities Panel* (Figure 5.6(c)) in order to choose the new granularities. If the operation selected is a roll-up, then the user will be able to choose only granularities bigger than the actual one, otherwise only lower granularities will be available.

If the roll-up or the drill-down operations involve the spatial dimension, visually these affect the partition of the spatial domain. Figure 5.8(a) illustrates a roll-up operation where the base granularity consists of segments of the road network, while the coarser granularity represents a partition of the territory based on administrative areas. On the other hand, in Figure 5.8(b) is illustrated the effect of a drill-down operation on a grid partition of the space. In this case, from a coarser grid a more detailed one can be obtained.



(a) Roll-up (b) Drill-down

Figure 5.8: Effect of data warehouse operations on the spatial dimension

It is important to remark that the user can choose the spatio-temporal hierarchy which is more adequate for his/her needs. Further, there is no constraint on the spatial type of dimensional attributes of the spatial hierarchy. For example the spatial dimension could be a road segment, i.e a line, which is contained in a city district, a polygon. The visual tool reproduces faithfully this choice.

5.4 Data Analysis

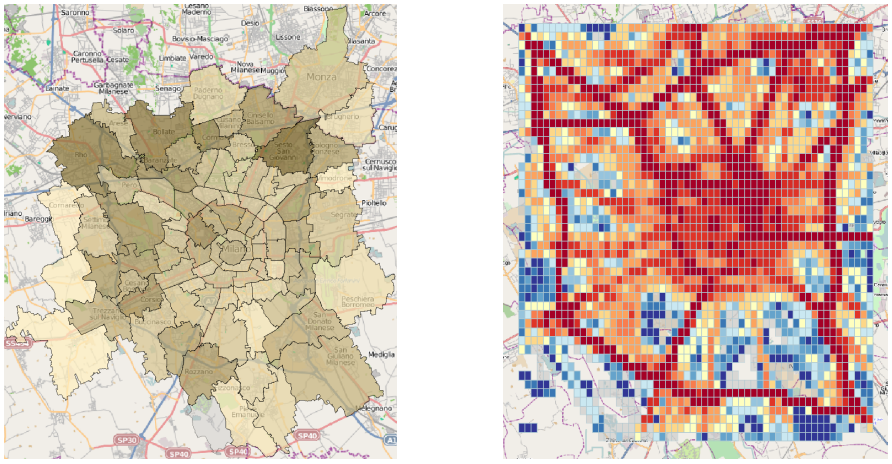
After the user has selected the granularities s/he wants to analyse, the data loaded in the TDW need to be analysed. In a TDW, each measure contained in the fact table is referred, at least, if no other dimensions are available, to a spatio-temporal granule, that visually is translated into a geometry with a value associated to it

that changes for every available time interval. So, different methods can and should be used in order to show these changes to the user, depending on the kind of the measure or the kind of geometry has to be represented. The Visual Analytics Toolkit offers several visualization styles and graph representations in order to see how values change along the time. Some of them will be presented in the following, divided into three classes: methods for intra-granule measures will be presented in Section 5.4.1; multi-dimensional measures will be treated in Section 5.4.2; finally inter-granule measures will be illustrated in Section 5.4.3.

5.4.1 Intra-granule measures

In this section we are going to discuss methods to visualize intra-granule measures, i.e. measures related to a single spatial granule.

Choropleth Maps



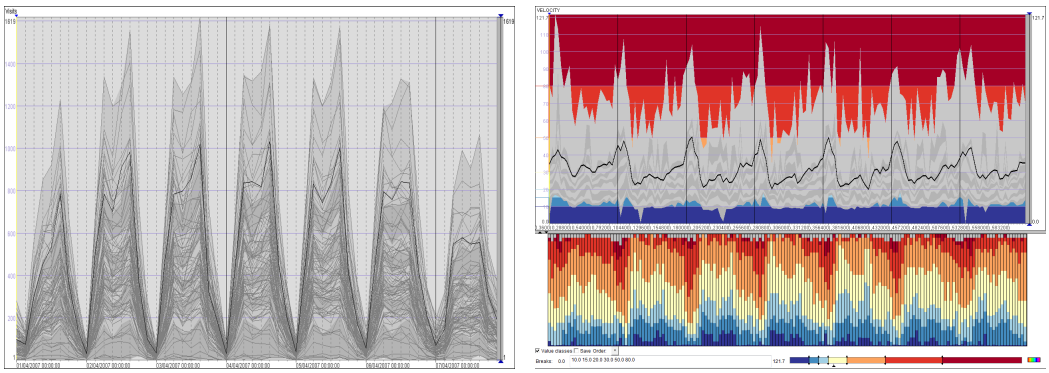
(a) Unclassified Choropleth Map (b) Classified Choropleth Map

Figure 5.9: Different kinds of Choropleth Maps

The first two visualization styles have the name of choropleth maps (Figure 5.9). Given a measure having a value associated with each spatial granule of our TDW, this visualization produces a map animation. Each granule on the map is filled by a colour that represents the value associated with it. Figure 5.9(a) shows a single frame of the animation produced by the so called *unclassified choropleth map* visualization. In this version of the choropleth map, a colour shade is used for the visualization. Darker is the colour in a granule, higher is the value associated with it. On the other hand, Figure 5.9(b) shows a frame obtained by the *classified choropleth map*. Unlike the unclassified version of the map, in this case data are divided into

a finite set of classes, and each class is associated with a colour. Then each granule is filled in with respect to the colour of the class the value belongs to.

Time series



(a) Time Graph (b) Classified Time Graph

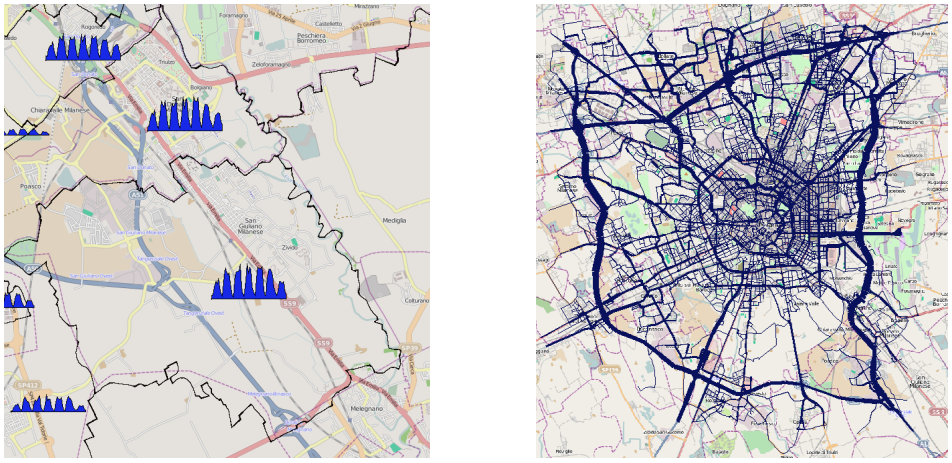
Figure 5.10: Examples of time graphs

Time series represents a useful tool in order to have a comprehensive view on what happens along the entire period of time the user is taking into account. Unlike the animations, where the user should look at different frames in order to see differences during the time, time series give an overall view of the phenomena along it, but are less focused on the space, so the ideal thing is to use them in combination with other visualization methods. Visual Analytics Toolkit has a really powerful time series tool, that allows the user to have various views of the data.

Figure 5.10(a) shows the simplest time graph available in the tool. Time is represented on the y-axis of the graph, while on the x-axis one can find the value of the measure the user is analysing. Each curve on the graph identifies the time series of the value associated to a spatial geometry. By moving the mouse over one of these curves, the corresponding geometry in the main window is highlighted. The user can decide to show in the graph also other useful information as the average time series or some quantiles, calculated on the fly starting from all the available values.

As for the choropleth maps, it is possible to have a classified version of the time graph, as shown in Figure 5.10(b). In this case the window is divided into 2 different sections. The graph on the top shows the progress of the values as in the previous graph, enriched with some coloured band representing the classes obtained from the classification of the time series. Note that in the picture, instead of all the time series available, only the average series is shown (in black) together with the value flows of the series. On the other hand, the graph at the bottom visually shows the number of spatial granules belonging to a given class, for each time intervals the period has been divided into.

A similar kind of visualization is the one shown in Figure 5.11(a). Each time graph related to a spatial granule is displayed directly inside the corresponding geometry. In this case the user can see immediately what is the progress of the selected value for each spatial granule, and directly compare it with others.



(a) Time graphs in spatial granules

(b) Line thickness

Figure 5.11: Examples of visualizations techniques

Line Thickness

If the geometries associated with a given spatial dimension correspond to simple lines, the use of the choropleth maps could not be feasible. In this case VA-Toolkit permits to associate the values the user wants to visualize with the thickness of the corresponding line, as shown in Figure 5.11(b).

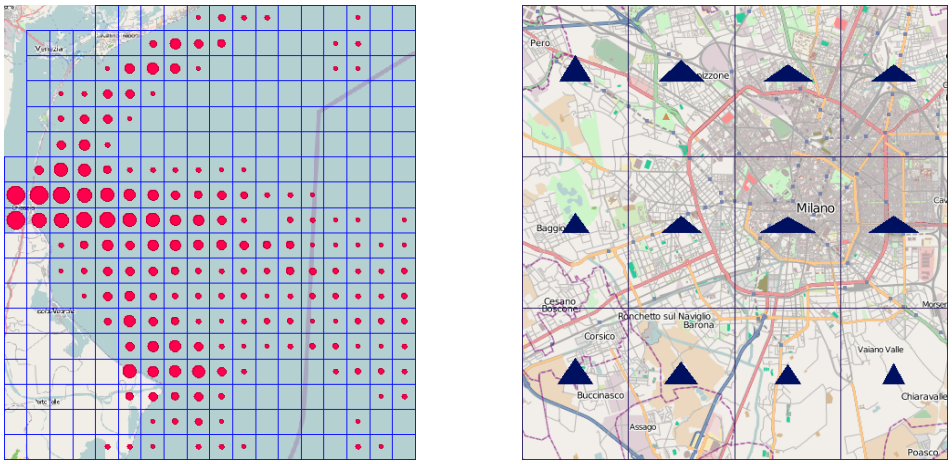
In this case, the thickness of each line is depicted proportionally to the value associated with that geometry. Also in this case the tool permits to obtain an animation along the time, if needed.

Symbol Maps

Another way to visualize the measures contained in our TDW is illustrated in Figure 5.12(a). In this case the values are associated to the dimensions of circles that are drawn inside each spatial granule. The bigger are the circles, the higher are the values associated with them. The user can select different shapes: circles, as in the picture, bars, images.

Triangles

A useful visualization technique in order to compare different measures to see if there is, for instance, any correlation between them, is the *triangle visualization*



(a) (b)

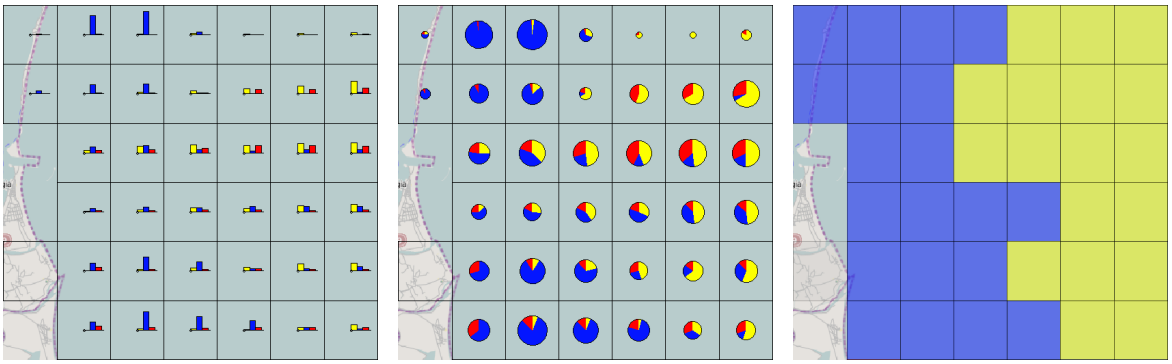
Figure 5.12: Symbols map (a) and triangles visualization (b)

(Figure 5.12(b)). In this case a triangular symbol is shown inside each spatial geometry. The base and the height of the triangles are proportional respectively to one of the measures the user wants to visualize. Again, the map could be animated and the variations of the values are reflected on the shapes of the triangles, in order to let the user see any possible correlation between the values along the time.

5.4.2 Multi-dimensional measures

All the visualization methods presented so far permits the visualization of a measure that is directly related to a spatial granule. This is not always the case. Our TDW model can be composed of many different dimensions other than the spatial and the temporal one. Suppose, for instance, to have a dimension containing information about the users, and one of its dimensional attribute is the gender. Suppose now that an analyst is interested to visualize a measure, observing the differences between genders. For example s/he could be interested in observing the average speed of the males and females in a certain zone. In this case VA-Toolkit allows to depict some small graphs, like those seen before for the time series, inside each spatial granule, in order to show values related to different characteristics at the same time.

Figure 5.13 illustrates three kinds of visualization that can be used in these situations, related to the same data. In Figure 5.13(a), *parallel bars* are used. Each bar corresponds to a characteristic of the measure we are observing (three in the figure). In Figure 5.13(b) *pie charts* are used. In this case the area of the circle is proportional to the sum of all the available values. Then the circle is divided into pies, whose sectors are proportional to the corresponding values of the measure for each feature. Finally Figure 5.13(c) shows the *dominance classification* visualization. In this case, a colour is assigned to each available feature. Then each spatial granule



(a) Parallel bars (b) Pie charts (c) Dominance classification

Figure 5.13: Multi-dimensional measures visualizations techniques

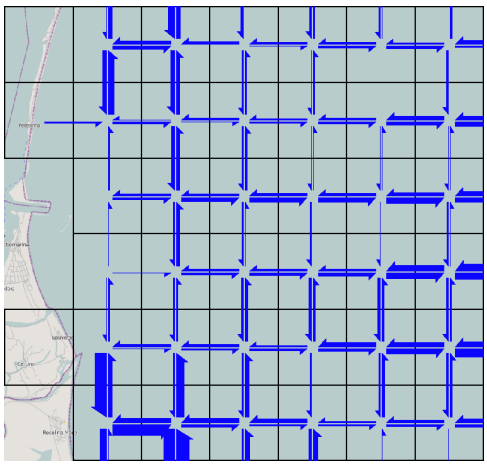
is filled in with the colour corresponding to the characteristic with the higher value.

5.4.3 Inter-granule measures

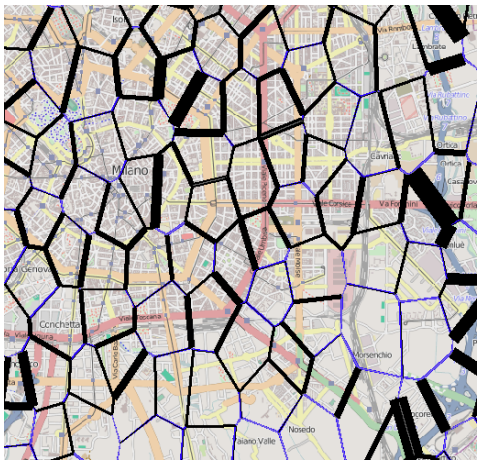
The main difference between intra-granule measures and inter-granule ones is that while the former are measures related to single granules, the latter are related to pairs of granules. Hence, in order to visualize them, pairs of geometries need to be shown or connected together in some ways. The only inter-granule measure that has been taken into account is the cross measure.

The user can choose between two methods to load the measure cross from the TDW by VA-Toolkit, depending on the situation and needs. Directional crosses take into account the direction of the cross between two granules, so having two different granules, a value will be present for the number of crosses from the first to the second geometry, and another value for the opposite direction (from the second to the first one). On the other hand, aggregated crosses are useful when the direction is not important. In this case each directional cross value is summed up in order to obtain a single cross value for all the geometries involved. For example, consider a crossroad where several road segments meet each other. In this case, the visualization of directional crosses should be difficult, since in a small space, i.e. the point of conjunction of all the segments, many different values should be visualized, while for the aggregated cross a single value representing all the crosses on that point is required.

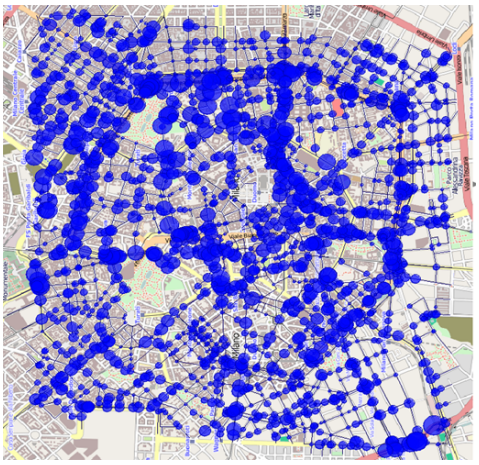
Figure 5.14 shows different kinds of visualizations for the measure cross. Figure 5.14(a) illustrates how directional crosses are displayed. Each cross value between two different geometries is associated with an arrow representing the direction of the cross itself. The thickness of the arrow is proportional to the value of the cross. The user could also decide to draw each arrow with a different colour, depending on its direction. In this case arrows compass directions are calculated on the fly and aggregated together in eight different classes, i.e. N-NE-E-SE-S-SW-W-NW.



(a) Directional crosses between regions



(b) Aggregated crosses between regions



(c) Aggregated crosses between lines

Figure 5.14: Cross visualizations

On the other hand, Figure 5.14(b) and Figure 5.14(c) show how aggregated crosses could be represented. The first picture models the case of crosses between different polygons. In this case the thickness of the borders between the polygons is used in order to represent the value of the measure. In the second case, aggregated crosses among several lines are visualized. Here the dimension of the point where all the segments intersect is proportional to the value of the measure.

5.5 Synopsis

The chapter described the tool to provide users a visual interface for performing visual OLAP operations. We discussed the functionalities and the architecture of VA-Toolkit, and presented the module we developed in order to connect VA-Toolkit

to a Trajectory Data Warehouse and to support the users with visual OLAP. The chapter then illustrated various kinds of analyses we can perform interacting with the TDW, given some hints on which visualization techniques can be used for the different kinds of measures stored in the TDW.

6

Case Studies

In this chapter we will applied the proposed framework to two different real world datasets. The first one, discussed in Section 6.1, is a dataset containing location information related to a fleet of cars, hence the movements of the objects are constrained in a road network. Unfortunately this dataset has no information related to moving objects, other than their positions. On the other hand, the second dataset, presented in Section 6.2, contains information about boats sailing on the sea, hence their movement could be consider free and not constrained in any way. The dataset contains the positions of the moving objects, as well as many descriptive information for any observed boat.

6.1 Analysis of cars in Milan

In this section the Trajectory Data Warehouse presented in the past chapters will be used in order to analyse some traffic data related to the city of Milan in Italy.

6.1.1 Dataset acquisition and description

Data contained in the Milan dataset have been collected by an insurance company, that gives discounts to clients who voluntarily install a GPS device in their cars, and submit all GPS data to the insurance company (to decease the burden of proof in case of an accident). The data have been made available, after an anonymization process, in the context of GeoPKDD European project, to which we have taken part, and are not publicly disclosable.

The dataset consists of two millions of time-stamped location records, sampled at irregular time intervals, representing the movement of seventeen thousand objects moving during a week period from Sunday, April 1st to Saturday, April 7th 2007. The structure of each record is in the form $(V_{id}, t, lat, long)$, where T_{id} represents the unique identifier of each moving object, while $(t, lat, long)$ represents its time-stamped location.

Due to the anonymization process applied to the data, this dataset has unfortunately no information about the identity of the moving objects. Only data about the movement are kept.

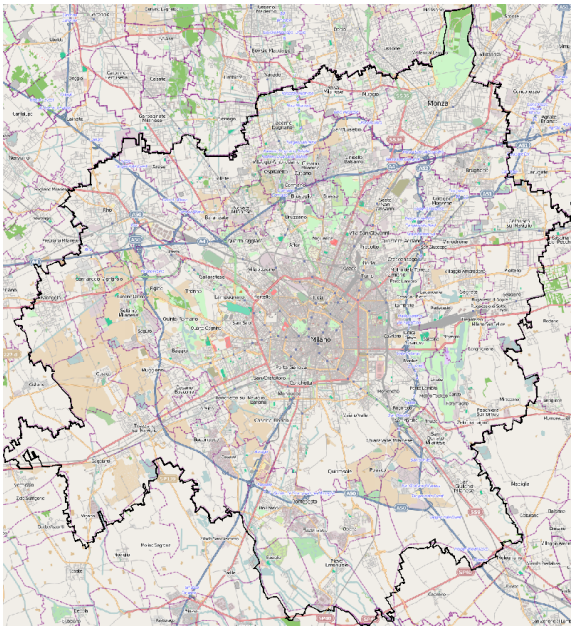


Figure 6.1: Milan dataset extension

The overall area covered by the data can be approximated as a rectangle of size $32km \times 35km$, and is shown in Figure 6.1

6.1.2 Data Warehouse description

In order to load the dataset described in Section 6.1.1 in a TDW a correct instantiation of our proposed model has to be done, defining adequate dimensions and associated hierarchies. Since, as already pointed out, the dataset does not contain any demographical information concerning moving objects, the dimensions of the data warehouse will be only the spatial and the temporal ones as shown in Figure 6.2. However, even in this case where the schema results to be really simple, our framework does not loose in expressive power, as it will be demonstrated in the following sections.

Since the time span of the data is one week the **temporal dimension** is based on a simple collection of regular intervals of increasing size as shown in Figure 6.3. The base granularity consists of one hour intervals. Coarser granularities sum up these base intervals in order to obtain 3-hours intervals, 24-hours intervals (i.e. an entire day) and 168-hours intervals (i.e. a week), covering the whole time period of interest.

On the other hand, the **spatial dimension** is composed by a more complex hierarchy. Figure 6.4 describes the tree representing this dimension. In particular, the figure shows three different kinds of aggregations available, all having the same shared base granularity. A natural choice for the base granularity, in the context

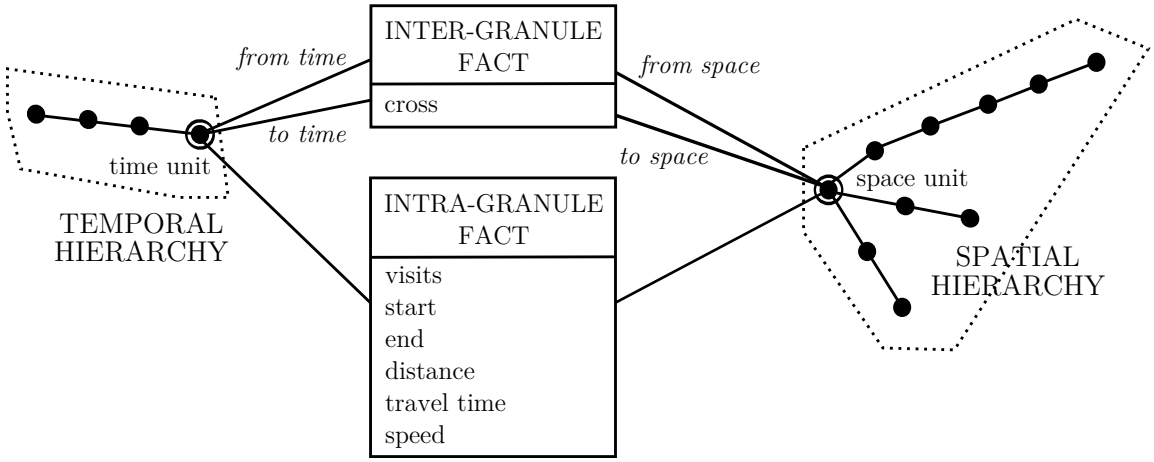


Figure 6.2: Actual TDW schema for Milan traffic scenario

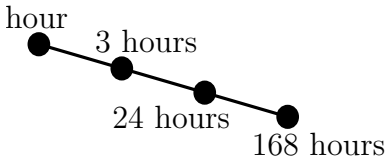


Figure 6.3: Actual temporal dimension hierarchy for Milan traffic scenario

of traffic analysis, is road segments as the smallest spatial unit to observe. These segments are shown in Figure 6.5(a). The first branch of the hierarchy proposed in order to aggregate together the road segments is based on a regular grid that becomes coarser along the hierarchy. The smallest grid is composed by $330m \times 440m$ rectangles, and it is shown in Figure 6.5(b). These smallest cells are then aggregated in groups of 10, 20, 40 and 80 spatially adjacent rectangles, by forming respectively grids composed of $3.3Km \times 4.4Km$, $6.6Km \times 8.8Km$, $13.2Km \times 17.6Km$ and $24.4Km \times 35.2Km$ cells.

The second branch takes into account the administrative areas the territory of interest is divided in. The first level represents the districts the city is partitioned.

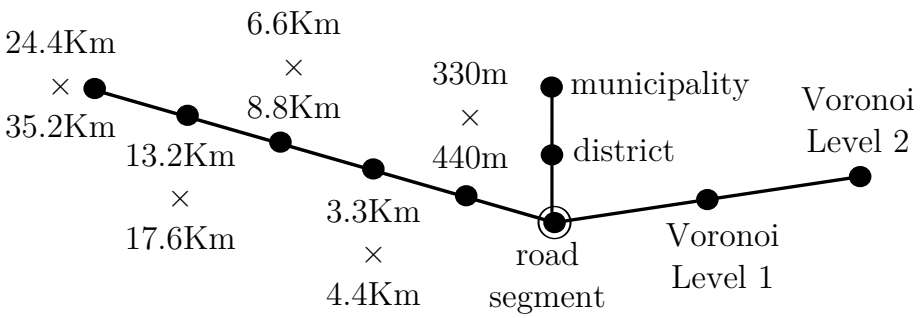
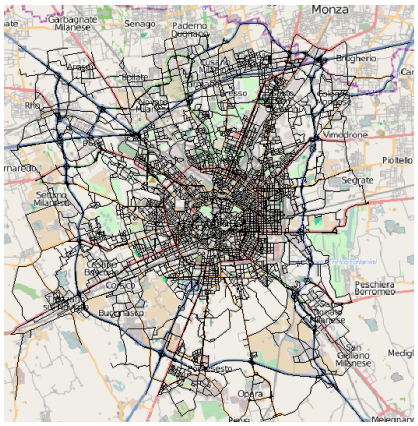
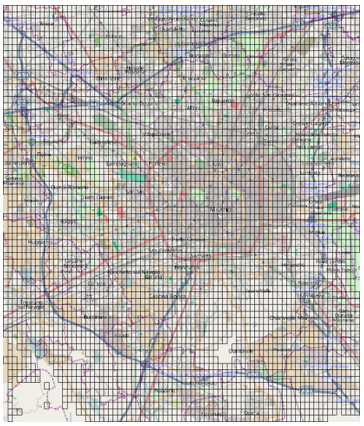


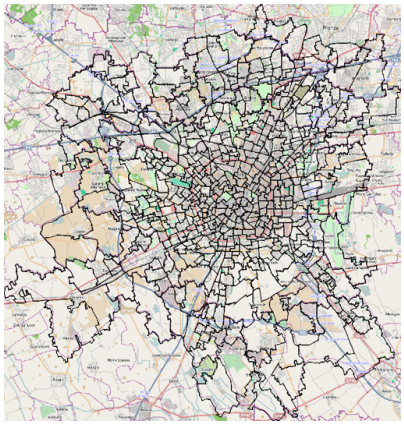
Figure 6.4: Actual spatial dimension hierarchy for Milan traffic scenario



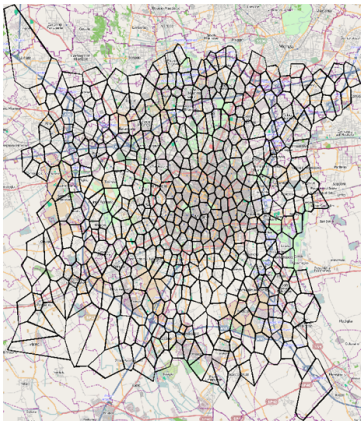
(a) Spatial base granularity for Milan dataset



(b) Smallest cells for the grid hierarchy



(c) District level



(d) Smallest Voronoi polygons

Figure 6.5: Different spatial partitions for the Milan dataset

This division is visible in Figure 6.5(c). Districts can then be further aggregated and they form municipalities.

One of the problems of having regular grids or user defined regions partitioning the space is that the obtained areas do not bear any semantics and do not correspond well to the real geographic and topographic properties of the data, and this can led to misleading results. The third branch of the hierarchy has been defined to cope with this problem. In particular, the space has been divided by using the Voronoi tessellation technique [2]. Given a set of points $\mathcal{P}$ in the plane, the Voronoi tessellation is a partition of the space into regions such that for each point $p \in \mathcal{P}$ there is a region $R(p)$ which consists of all points of the plane that are closer to p than to any other point in $\mathcal{P}$. Hence, the region $R(p)$ describes some kind of "neighbourhood" of p. Also in this case the hierarchy is composed by two levels (further the base one), and each polygon of the lower level is completely contained

into a polygon of the higher one. The lower level polygons have been constructed starting from the raw data of the trajectories observations. The points have been clustered together in clusters of spatially near points, and clusters centroids have been used in order to generate the polygons. On the other hand, since there is no defined ways to generate a hierarchy of Voronoi polygons, a heuristic has been applied in order to generate the second level of this branch of the hierarchy. We have generated a Voronoi tessellations with a radius bigger than the one used for the lower level. Then, using this second partitioning, we have aggregated together all the Voronoi polygons of the first level, contained for at least 50% of their area in a bigger polygon. It is worth noting that applying this heuristic we do not obtain any longer genuine Voronoi polygons. The result of the low level tessellation is shown in Figure 6.5(d).

In order to complete the definition of the TDW we have to specify the measures we want to load. Measures are needed in order to grasp information about the trajectories crossing a given spatial cell in a given time interval. As shown in Figure 6.2, the measures we are interested in for this dataset are those already presented and discussed in Chapter 4, i.e. *visits*, *start*, *end*, *travel time*, *distance* and *speed* for the intra-granule fact table, and *crosses* for the inter-granule fact table.

6.1.3 Trajectories reconstruction and data warehouse loading

In order to obtain the trajectories needed to feed-up our TDW from raw data, a process of trajectory reconstruction has been applied. As the base granularity of the data warehouse is composed by single road segments, the reconstruction process needs to cope with this constraint, as described in Chapter 3.3.

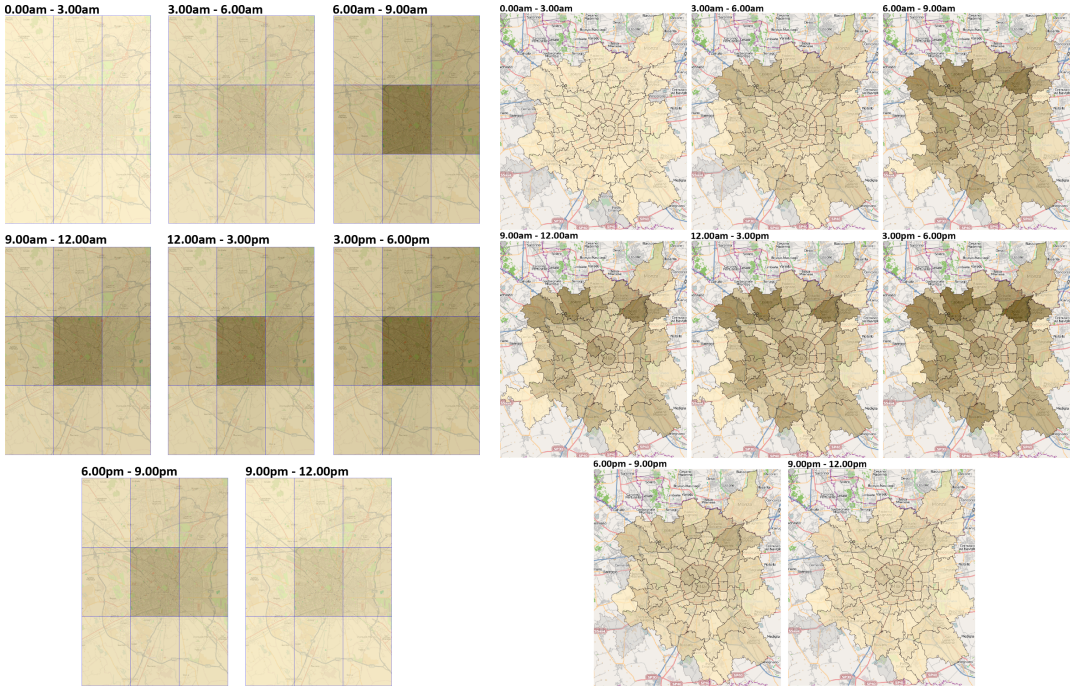
The first task to perform consists of the so called map matching process: during this phase, each point of the dataset has been associated with one of the road segments composing our base granularity, i.e. the segment nearest to the point. After this step, point coordinates correspond to those of a point in the road network, so the second step could be performed. Since there is no information about the route each object follows, in order to concatenate different road segment to create a complete trajectory, a shortest path algorithm coded in Java has been used. It has been assume that between two different points in the dataset, the speed was constant. In order to determine trajectories, from raw data, we have use some of the parameter discussed in Section 3.3. In particular, we have imposed a maximum speed between two consecutive observations of about $200km/h$. Moreover, we have fixed a maximum temporal gap of 15 minutes between one to its consecutive points. Higher delays between consecutive observations led to the creation of a new trajectory, supposing that in this case the GPS device was simply switched off.

At the end of the task, the total amount of trajectories obtained has been of about 200,000, i.e. 11 trajectories per vehicle composed by eight points each.

Then, the TDW has been built and loaded with the described measures, according to the spatial and temporal dimensions described.

6.1.4 Data analysis

In the context of traffic data analysis, there are some interesting questions that traffic managers could be interested in answering. Some example could be *"From which district does a great number of cars leave in the morning? And at what hour? Is there a flow exiting/entering the town? Which are the main differences in the traffic between the working days and the week-end?"*. In this section it will be shown how a traffic manager could use the proposed TDW in order to find an answer to these questions.



(a) $\mathcal{V}$ on $6.6km \times 8.8km$ cells (b) $\mathcal{V}$ on municipalities

Figure 6.6: *Visits* at different granularities and hierarchies on Tuesday (3-hours intervals)

Figure 6.6 represents the measure *visits* ($\mathcal{V}$) on Tuesday. Temporal dimension has been aggregated in 3-hours intervals, while the picture shows two different kinds of aggregation for the spatial dimension, i.e. $6.6km \times 8.8km$ cells (Figure 6.6(a)) and municipalities (Figure 6.6(b)). Despite the two figures represent the same measure, there are some differences between them. Both the pictures show how the traffic starts increasing from early morning reaching the maximum values between 6am-9am and 3pm-6pm, and then decreases on the nightly hours. On the other hand,

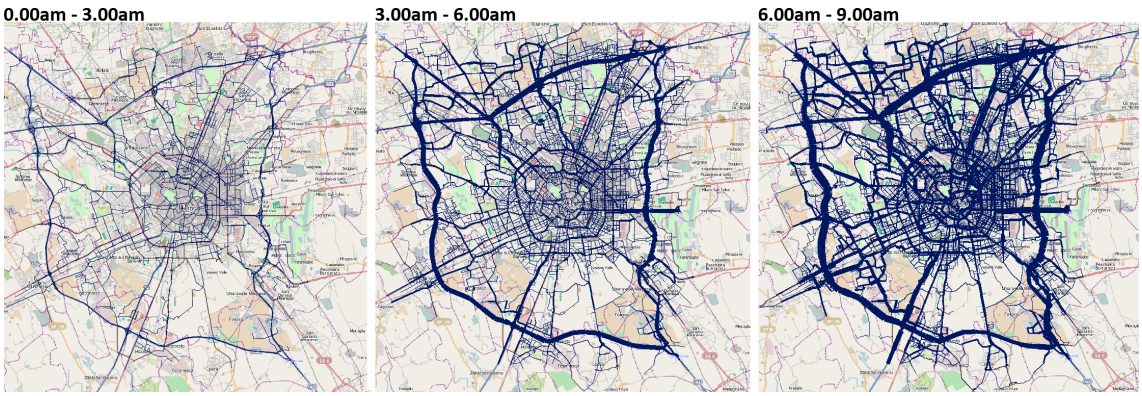


Figure 6.7: *Visits* at road segment granularity on Tuesday

by looking for example at the interval 6am-9am, in the two pictures, one can note how in the cells partition it results that the mostly trafficated area is the one in the centre of the city; on the contrary, in the municipalities partition the situation is slightly different, showing that the traffic is concentrated on both the Milan down town and in the external suburbs on the top left of the city. This is because the second spatial partition is tighter with the territory while the grid groups areas with a small number of streets together with those having a lot of streets.

By applying a drill-down operation, we obtain a view of the same data at the base granularity, i.e., street segments. In this case in order to visualise the measure visits the line thickness technique is used. The corresponding frames, shown in Figure 6.7, allow the user to perceive a very different image of the traffic distribution of the town: the information about the visits is connected to the roads, i.e., thicker lines indicate a higher value for the measure visits. We can distinguish several rings around the centre and some radial streets that are used to enter to the centre, or to exit from it. This allows us to answer queries about the situation of traffic at the road network level. For example, we can discover that the high number of visits in the suburbs during rush hours, visible in Figure 6.6(b), is mainly due to the traffic in the external ring highway. It is interesting to note also that from 0am to 3am there are few cars moving around, and there is no dense area. Then, as time goes by, the outer ring of the town becomes denser and after the inner rings and the radial roads too.

In order to have a complete view of what happens during the whole week, a graph could be a very helpful tool. The time series display in Figure 6.8 summarizes the temporal variation of the measure *Visits* over the whole territory, divided in $3.3km \times 4.4km$ cells. The temporal granularity chosen is of 1-hour intervals. The appearance of the display shows a clear subdivision of the whole time period into days. We can observe that the visits are much higher in the day hours than in the night and noticeably higher on the working days than on Sunday and Saturday (respectively last and first day of the period). On each of the working days, there

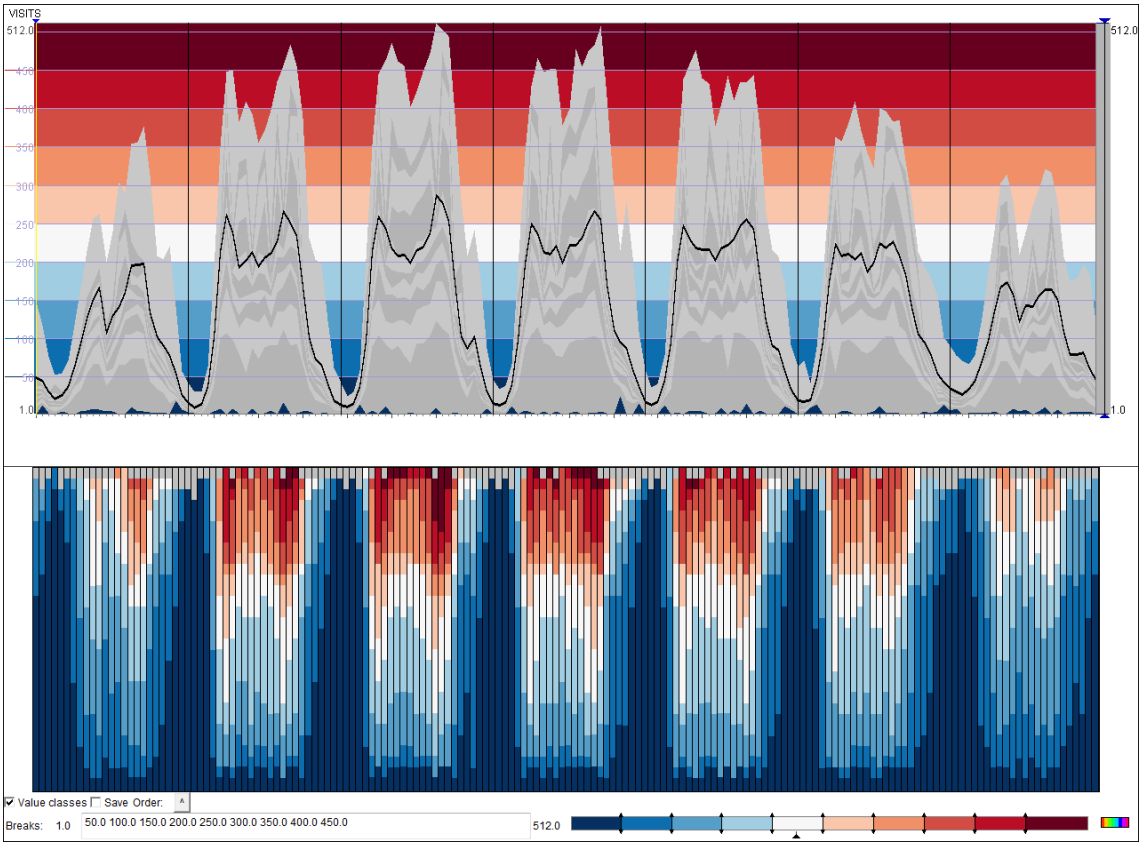


Figure 6.8: The evolution of *Visits* during the week

are two peaks of the number of cells with high visits, signified by the shades of red. These peaks correspond to the morning and afternoon rush hours, which occur in the intervals $6-9am$ and $3-6pm$. Interesting is the increase of traffic intensity on Sunday afternoon. It is also visible that the traffic on Friday is less intense than on the previous working days: there are no cells with the values lying in the upper two classes of the values of visits.

Comparing the display of the visits with the display of the speed of objects at the same granularity, shown in Figure 6.9, one can immediately realise that visits and average speed are inversely proportional. During the early and late hours of the day the speed is high whereas from 6am up to 6pm the speed decreases significantly, exhibiting a dual behaviour with respect to the visits.

The composite time series displays representing the temporal evolution of the measures need to be combined with cartographic visualisations showing the data in the spatial context. For example, Figure 6.10 is a screen-shot of the animation representing the values of the speed and the visits by triangular symbols. The height of the triangle is proportional to the speed and the base to the visits. One animation frame corresponds to one hourly interval and the whole animation shows the variation of the visits and speed over the week. This reveals additional information

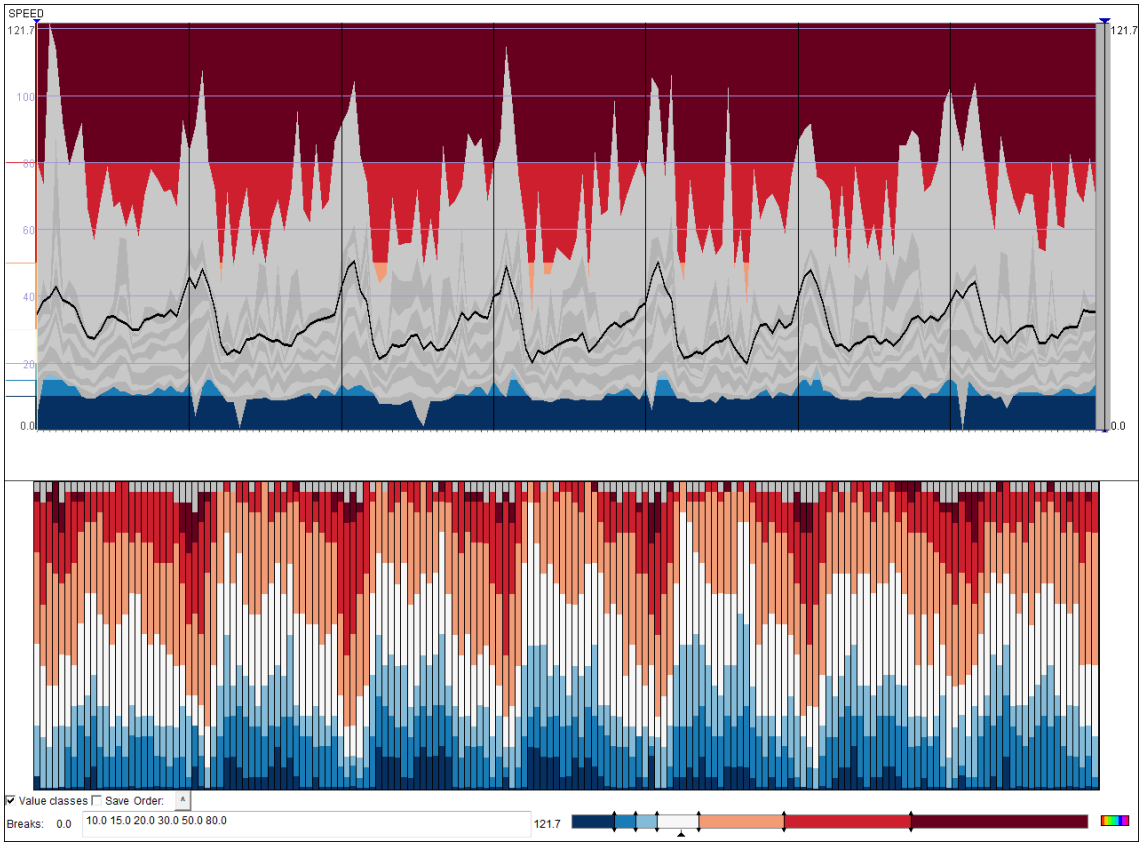


Figure 6.9: The evolution of *Speed* during the week

with respect to the time series displays. Thus, the image shows that the visits are higher in the centre and this has a strong impact on the speed of cars, which is very low. On the other hand, it highlights that along the ring roads, the speed is higher except in the north-east zone where the larger number of cars slows down the traffic.

As shown on Figure 6.10, the combination of different measures in order to visualize them together could bring new information available to a traffic manager. Another example of this fact is represented in Figure 6.11. The picture shows three different screenshots of the flow balance at municipality granularity using a thematic map. The time granularity is of 1-hour intervals, and the screenshots are related to three different intervals, two in the early morning and one on late afternoon, on Monday. Each municipality is coloured depending on the difference between the overall number of starting ($\mathcal{S}$) and ending ($\mathcal{E}$) trajectories in it. Zones painted in blue has a positive balance, that is the number of trajectories starting from the zone is bigger than the number of those ending in it. On the other hand, green represents a negative balance, that is the number of trajectories ending in the zone is bigger than the number of those starting in it. Finally, white represents zones where the number of starting and ending trajectories is almost the same. One can clearly observe the duality of patterns in early morning hours with respect to those in the

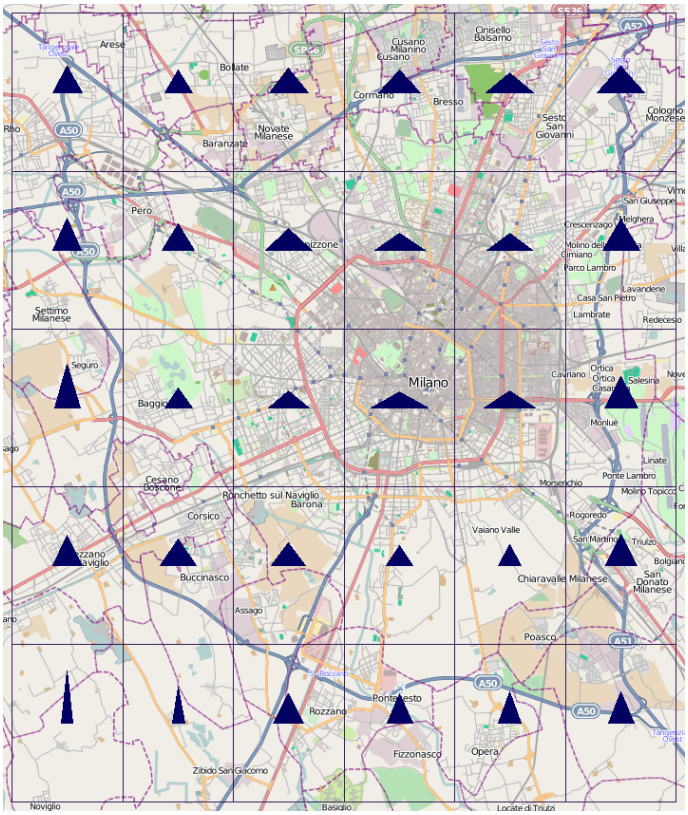


Figure 6.10: Relationship between *Visits* (widths of the triangles) and *Speed* (heights of the triangles)

evening and it can be discovered at what time usually people start moving or they turn back and from which zones. It is interesting to notice also that between 7am to 8am there are many zones where the balance is almost zero and this is due to the fact that at that hour people are usually travelling, hence their trajectories do not appear in the $\mathcal{S}$ or $\mathcal{E}$ measures (i.e. they are not starting their trip and neither ending it).

Figure 6.11 represents in some ways an overview of what can be the movements of the cars in the city, but does not give any information about the directions these movements have. In order to cope with this task, and better understanding this kind of phenomena, the Visual Analytics Toolkit can be used in order to represent the measure *Crosses* ($\mathcal{C}$), that in fact represents how the trajectories in the original dataset move between the granules of our TDW.

Figure 6.12 shows two different visualizations of this measure, for two different branches of our spatial hierarchy. Both the pictures refer to the city centre of Milan, and not to the entire spatial extension. Figure 6.12(a) represents the number of trajectories crossing the borders of two spatially adjacent granules (in this case two cells of our $330m \times 440m$ grid). Crosses are summed up together regardless of their direction, and are visualized by the thickness of the border between two granules.

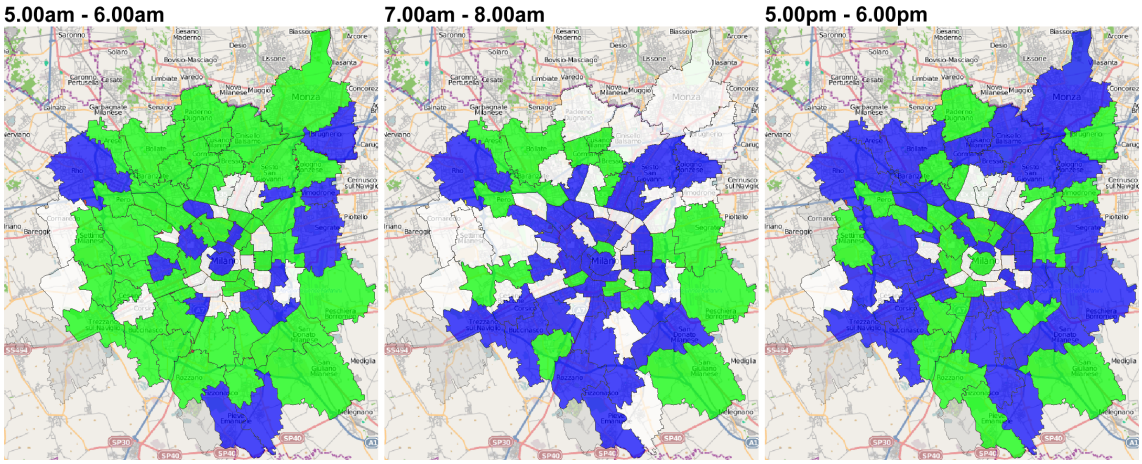


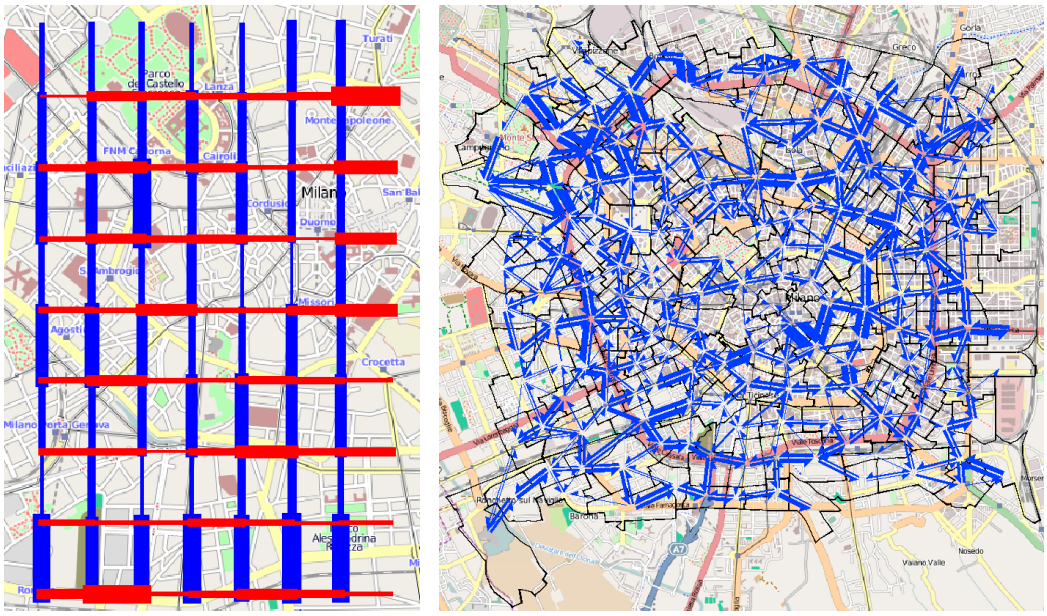
Figure 6.11: Zones with positive, negative, and neutral (blue/green/white) flow balance on Monday

On the other hand, Figure 6.12(b) shows the measure $\mathcal{C}$ between different districts, with respect to their direction. In this case the flow of the traffic is more visible, and one can note also some different dominant directions between some districts, identified by huger arrows.

Using user defined spatial hierarchies could on identify phenomena related to features that are interesting for an analyst. By the way, this kind of hierarchies is not always the best choice for some kind of situations. For example, these hierarchies could produce hardly understandable or even misleading results. Looking at the crosses showed on both the images in Figure 6.12, in the case of the regular grid, the major directions of the movements are greatly distorts, since all the crosses are divided in only 4 different directions, either by choosing directional crosses instead of aggregated ones. In the case of user defined granularities, as can be the districts, the problem is related to the fact that a single road can be situated between two granules, crossing between them many times. This could then generate a high number of crosses between the two areas, producing a misleading information.

In order to solve this problem, one can try to partition the space according to the density of the points in the dataset. Figure 6.13(a) shows the measure crosses by using Voronoi polygons. In this case the arrows representing the crosses are coloured depending on the directions they have (8 directions have been considered, namely S, SW, W, NW, N, NE, E, SE). One can easily comprehend the overall flow of moving objects. For example, it is easy to see two flows from South to North and vice-versa on both left and right part of the figure, corresponding to two ring roads, and another flow with direction SW/NE starting from the bottom of the picture and reaching almost the city centre. Other flows are clearly visible on the top of the picture.

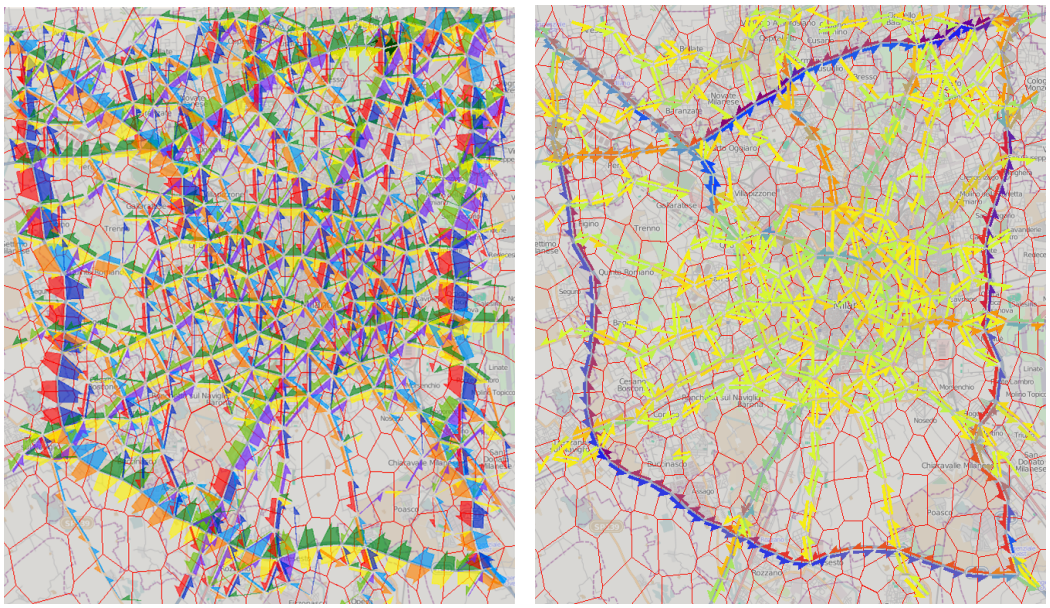
The aggregates obtained by OLAP operations should not be considered as final



(a) Aggregated crosses between cells (b) Crosses between districts

Figure 6.12: Crosses visualizations at two different spatial hierarchies

products that only need to be nicely represented for reporting but rather as data that need to be further explored. The analysis could require the application of additional computational techniques such as clustering and statistical analysis of time series. Using the Visual Analytics Toolkit one can analyse measures with the help of clustering on the basis of the time series of the measure values. Informally, a comparison between the time graphs of the measure is made, then they are clustered together in classes containing similar graphs, and a colour is assigned each of them. Applying the technique to the cross measure, we can analyse, in particular, the differences in the temporal variation of the traffic between two regions (or along a road) in opposite directions. In Figure 6.13(b) the crosses between adjacent Voronoi regions are shown. Arrows between areas are coloured according to the similarity of the time series of the cross measure. Distinct colours of the opposite arrows between two areas mean that their temporal variation patterns are rather different. Same or similar colours of successive links going along a road mean that the temporal variation patterns are similar on rather long parts of the road. In this case the behaviour of trajectories could clearly be seen. Movements in the outer ring road are really similar for both the directions of the cars, as those in the centre of the cities. These two kinds of movements are anyway really different between them. In order to better understand how these movements proceeds along the week, a graph as the one presented in Figure 6.14 could be used. It shows the temporal variation of crosses for two of the clusters: each coloured band indicates the evolution of the maximum and minimum values of the cross measure for all the region borders



(a) Crosses between Voronoi polygons (b) Clustering of crosses based on their time series

Figure 6.13: Different visualizations of *Crosses* between Voronoi polygons

belonging to the corresponding cluster.

6.2 Analysing Boats Sailing on the Adriatic Sea

During a collaboration with the environmental science department of our university, we have been asked from environmental scientists to use our TDW and analysis tool in order to analyse a dataset containing information about fishing boats sailing in the Northern Adriatic Sea. This dataset is not available for public disclosure, and we have been allowed to publish obtained results only in this thesis. In this section we will present these results.

6.2.1 Dataset acquisition and description

Data contained in this dataset have been collected during a joint work with the Venice Coast Guard Operations Centre in the period between January and September 2007. In order to track the boats to obtain their location over the sea during their fishing activity, information provided by the *Vessel Monitoring System* has been adopted. This system is used for security purposes and permits to track every boat in real time, while data are successively stored in a database and kept for one month. Each boat is equipped with a device called *blue box*, composed by a GPS receiver and a satellite transmitter. Each *blue box* transmits its GPS position to a

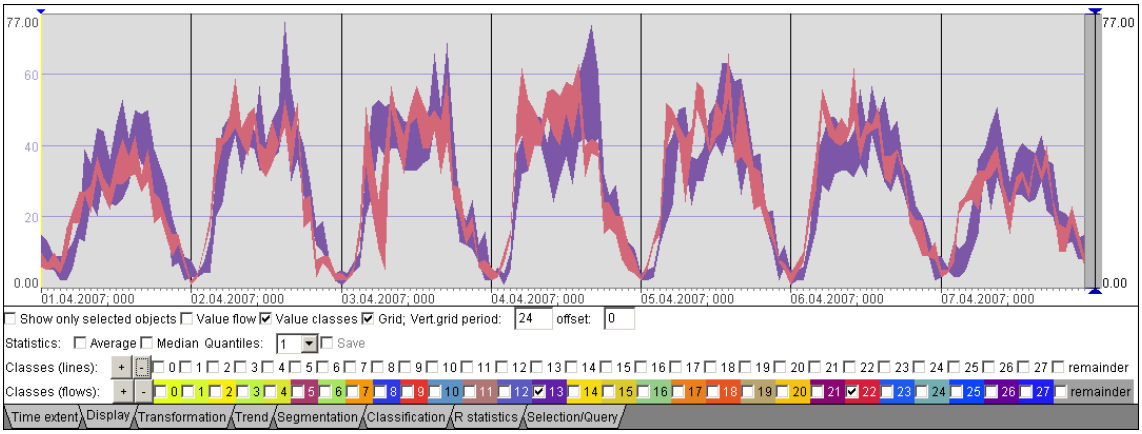


Figure 6.14: Variation intervals for the time series belonging to two selected clusters

Land Earth Station (LES) via the INMARSAT-C satellite at regular time intervals. The LES then sends the information to the Fishing Control Centre, where they are stored. Figure 6.15 describes this process.

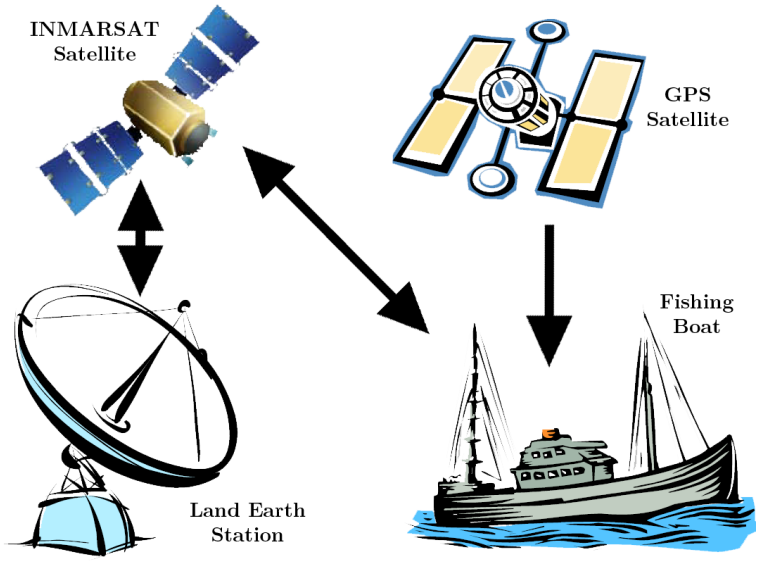


Figure 6.15: VMS operation

The total number of boats that have been tracked is 270 units sailing on the Adriatic Sea. These boats produced a dataset of about 326800 records. Each message sent from a boat to the LES contains the following information:

- Protocol number of the message
- Date and time of the message
- INMARSAT satellite identification number

- Boat identification number in the fleet register
- *Plate of the boat*
- Speed of the boat
- Direction of the boat
- *Latitude of the observation*
- *Longitude of the observation*
- *Date and time of the observation*
- Report type, i.e. information about anomalies on the boat

Only the field in italic has been taken into account. Some other fields could have been used too, as speed or direction, but during the phase of data clean-up and reconstruction of the data, we noticed that there was some inconsistency with other data, so we decided to calculate such values starting from the GPS positions.

To obtain information related to each boat as the name of the boat, the navy it belongs to, the fishery tools it is equipped with, the Community Fishing Fleet Register on-line site [1] has been consulted. Inside this database further information is available, as the boat power, dimensions, weight, name of the owner; since those pieces of information do not give any additional knowledge with respect to our study, we decided not to take them into account. Unfortunately, the register does not give any information related to the tool each boat has aboard, other than the class it belongs to. Hence, in order to have more accurate information related to it, it has been needed a visual check at the docks for each boat.

Finally a third dataset has been used, containing information about the amount of fish caught by each boat every day. This dataset has been obtained by taking information from the fish market, together with direct observations at the docks for the real caught of each boat. This database contains around 141000 records. Each record includes the date of the sale, the fish species, the boat in charge for the captures, and the amount of fish sold (in kilos). This information has then been used in order to estimate and distribute the catches along each boat path, as described in Section 6.2.3.

Figure 6.16 represents the spatial extent of the available data obtained from the datasets described so far.

6.2.2 Data Warehouse description

In order to cope with the task of analysing this kind of data, our TDW has to model facts related to multitudes of ship trajectories moving in the Adriatic Sea. The first

[1] `http://ec.europa.eu/fisheries/fleet/`

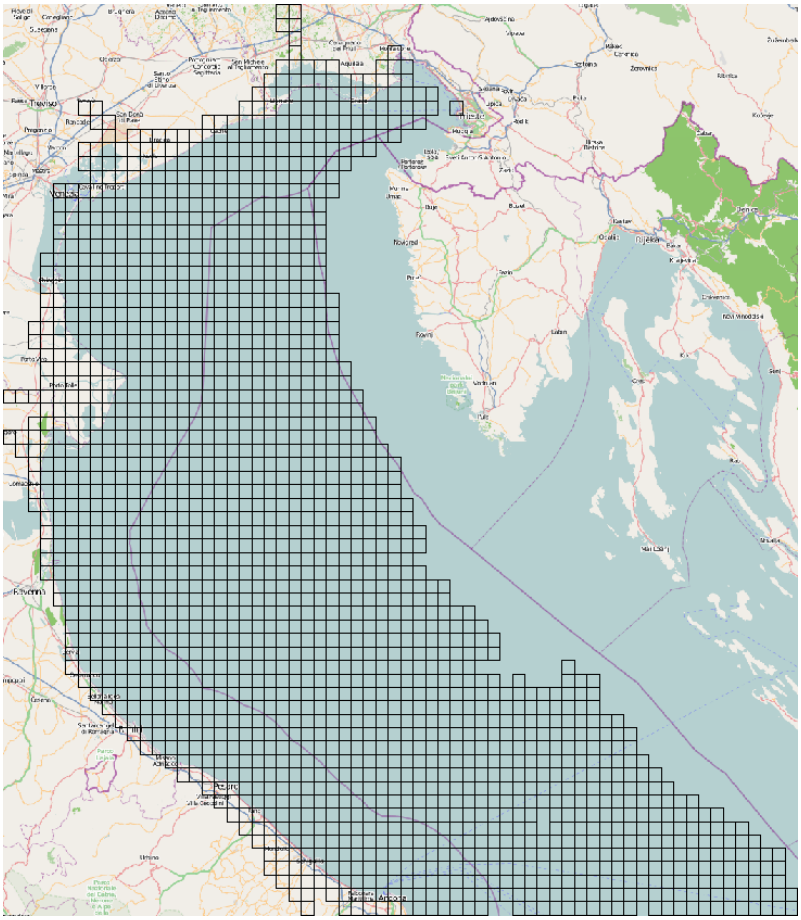


Figure 6.16: Northern Adriatic Sea Map with the base grid.

task that has to be accomplished is to adequately instantiate our framework, i.e. to choose right dimensions and hierarchies, select measures, to solve the given problem. Figure 6.17 presents the resulting schema for the TDW. Six different dimensions have been defined, each with a given hierarchy. For the **spatial dimension** the proposed hierarchy divides the space into a collection of regular grids of increasing size. The base grid is composed by 2x2 nautical miles (nm) squares. The intermediate grid groups together three base cells, in order to compose a grid of 6x6 nm cells. Finally the largest grid is composed of collection of 6x6 base cells composing a grid of 12x12 nm cells. The **temporal dimension** has been defined with a hierarchy having as base granularity a one day interval. Coarser granularities divide the time into 7 days intervals, 14 days intervals, months, trimesters and seasons. In this case the hierarchy, as shown in Figure 6.18, is a tree, and contains both numerical and descriptive levels. This second types of levels are more difficult to handle since containment operations must be manually defined, while on numerical levels they can be computed with some operation. The **boat dimension** contains information about boats that generate the data. The proposed hierarchy identifies at the base

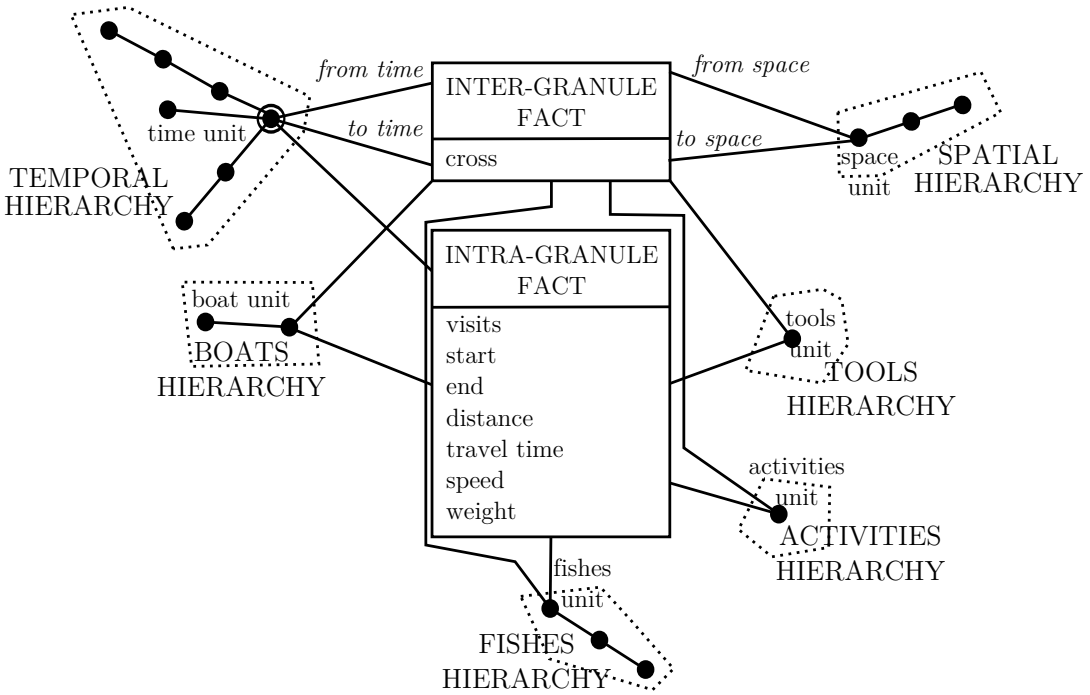


Figure 6.17: Actual TDW schema for ships scenario

level each single boat, that can then be aggregated accordingly to the navy the boat belongs to. The **fishes dimension** gives information about the caught fishes. The lower level of the hierarchy defined for this dimension identifies each kind of fish that lives in the Adriatic Sea. Fishes can be grouped by their species and again by their taxonomy. The **tools dimension** divides the data according to the fishing gear used by a boat (i.e. *coccia*, *volante* and *rapido*). Since each boat can use just a single tool a time, and each tool can catch only some kinds of fishes, these three dimensions, boat, tool and fishes, are tied each other. The associated hierarchy consists of the single root (base level). Finally, the **activities dimension** determines which activity a boat performs when it crosses a cell in our spatio-temporal grid, namely if

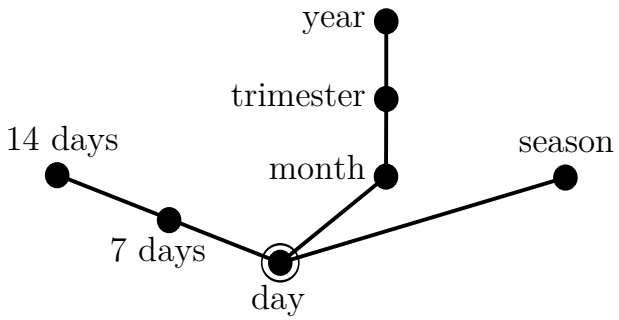


Figure 6.18: Actual temporal dimension hierarchy for ships scenario

the boat is fishing or moving to reach a certain place or to go back to the harbour. Even in this case the hierarchy is formed only by the base granularity.

Let us now describe the measures. As we have already stated, measures give information about trajectories crossing a given spatial cell, in a chosen time interval, having the characteristics selected by defining a single granularity for each of the available dimensions. Beside the standard measures already presented in Chapter 4, that are visits ($\mathcal{V}$), crosses ($\mathcal{C}$), total distance covered into a cell (*dist*), total time spent in a cell (*trav_t*) and average speed of boats in a cell (*speed*), we add the measure *weight*. This measure indicates the amount (in terms of kilograms) of fish caught in a cell.

In order to clarify the selected dimensions, hierarchies and measures, informally we can say that at the base level, given a measure, for example $\mathcal{V}$, the value associated with that measure defines the number of times a certain cell has been visited by a given boat (identified by a plate) during a single day of the year. Moreover, we also know that the boat visited that cells while making a certain activity (i.e. if it is for fishing, or if it is just moving somewhere), equipped with a certain tool and, in the case it is fishing, the kind of fish caught.

6.2.3 Trajectories reconstruction and data warehouse loading

As already stated, data about the movements of the vessels in the Adriatic Sea have been collected using the *Vessel Monitoring System*, a security system which uses a GPS transmitter in order to know the exact GPS location of a boat in the sea. Points collected using these transmitters must be processed in order to reconstruct the trajectories followed by each boats. It has been assumed that between two points, a boat travels at a constant speed, so that a linear interpolation model has been applied in order to obtain the complete path. Moreover, the sampling concerning a single boat has been split into several trajectories after a 16 hours time interval. This is due to the fact that it is really difficult that a fishing boat stays away for a time longer or lesser than this time. In this case using limiting the temporal interval between two consecutive observations would not be a good choice, since sometimes VMS observations are lost or the transmitter could be temporally turned off. Moreover, during this trajectory reconstruction phase a clean-up process of the data have been applied, by removing observations landing on earth, duplicated, or implying too big speed values.

Finally, in order to distribute the catches along each trajectory, each of their portions has been classified labelling whether the boat was fishing or simply sailing on the sea. This has been done by means of speed values [44]. In particular, depending on the tool used by each boat, it has been considered fishing if its speed was lower than 2.5 nodes for *volante* and *coccia*, and lower than 5 nodes for *rapido*. We have also considered as non-fishing trajectories portions within three miles from

each harbour. This choice has been taken since usually in those areas boats speed is low either if they are not fishing. After this classification, the total amount of daily catches related to a boat has been uniformly distributed along all its trajectories portions classified as fishing.

After this cleaning and transformation step the loading phase is performed for the about 33,000 trajectories obtained, and the TDW is built and loaded according to the described dimensions.

6.2.4 Data analysis

Let us now present some meaningful analyses we accomplish by using the TDW. The environmental scientists were mainly interested on analyse two aspect of the fishing activity, namely the *fishing effort index* and the distribution of the species on the sea. In order to reply to their question, we have produced, using our tool, two different kinds of maps that will now be described.

Fishing Effort

The first set of maps is related to the so called *fishing effort index*. This index consists of a value that indicates how much a given area has been exploited by the boats fishing in it. It is calculated as the ratio between the sum of the *swept area* by each boat in the cell and the area of the cell itself. The swept area is the total area that a boat has used for its fishing activity. It is calculated as the product between the area occupied on the sea by the tool used for fishing (one can think at the tool as a net that occupies a certain area) and the total distance that the boat has covered inside the cell. By formulas, we have:

$$E_i = \frac{\sum_{j=1}^{n} SA_j}{CA} = \frac{\sum_{j=1}^{n} TS_j \times d_j}{CA}$$

where E_i is the fishing effort index, SA is the swept area, CA is the cell area, TS is the tool surface and d is the total distance spent by each boat inside a cell. Since the surface occupied by a tool is always the same, it could be consider as a constant in our index creation, so we simplify the formula using the following to obtain the index:

$$\frac{\sum_{j=1}^{n} d_j}{CA}$$

In the context of analysing the fishing effort for the Adriatic Sea, some questions of interest are "What is the fishing effort the area has been subjected to during 2007?", "How is this effort divided during the year?", "What effort is due to a boat of a given navy?", "How is the effort affected by the kind of tool used by the boats?".

Figure 6.19 shows the fishing effort the Northern Adriatic Sea has been subjected to during 2007, at the lower level of our spatial granularity. Cells in darker colours are the most exploited, by the means that they have been completely explored more

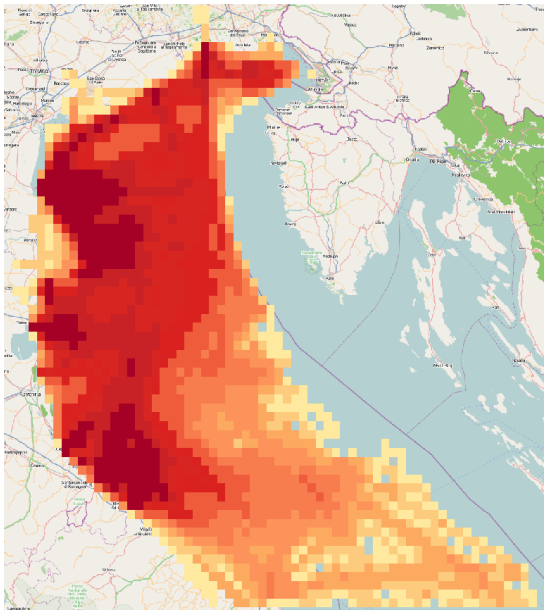
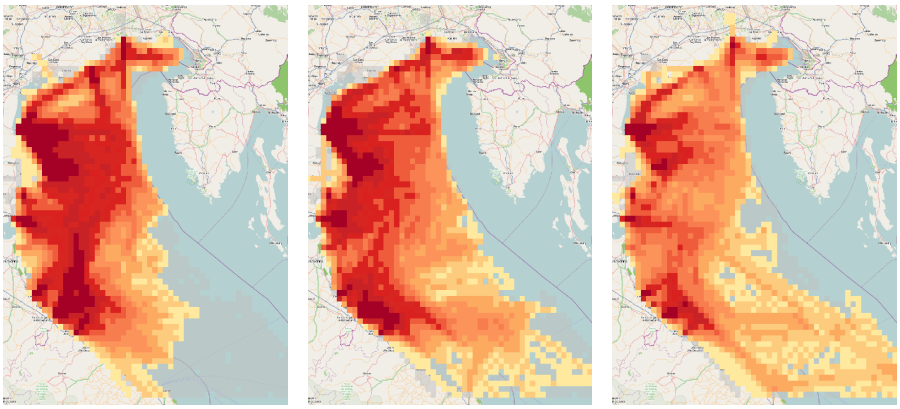


Figure 6.19: Fishing effort distribution in the period January-September 2007

than 1.5 times during the entire period. Since the fishing effort index is obtained by a ratio between areas, this means that each of these cells have been completely swept by boats sailing in it about two times in the considered period. While it is visible that the whole area has been explored during the period, from the Italian coast to the Croatian territorial waters, we can observe that the most exploited one is the area that goes from the Italian coast line to the first half of the entire distance between the two countries.

One can now wonder if the aforementioned scenario is the same during the year or change during the months. By using a drill-down operation on the temporal dimension of our TDW, we can inspect the situation to a higher level of detail. Figure 6.20 represents the fishing effort in the Adriatic Sea, divided into trimesters. What can be noticed is that during Winter (Figure 6.20(a)) the explored area is somehow reduced compared with the other months, while one can see higher values for the effort in the areas nearest to the harbours. This is due to the fact that boats tend to avoid going on open seas during the cold season, when days are shorter. During Spring (Figure 6.20(b)) boats reach also southern areas, and the effort results to be more uniformly distributed over the space. Finally, it is worth noting that during Summer (Figure 6.20(c)) fishing activities are less intense.

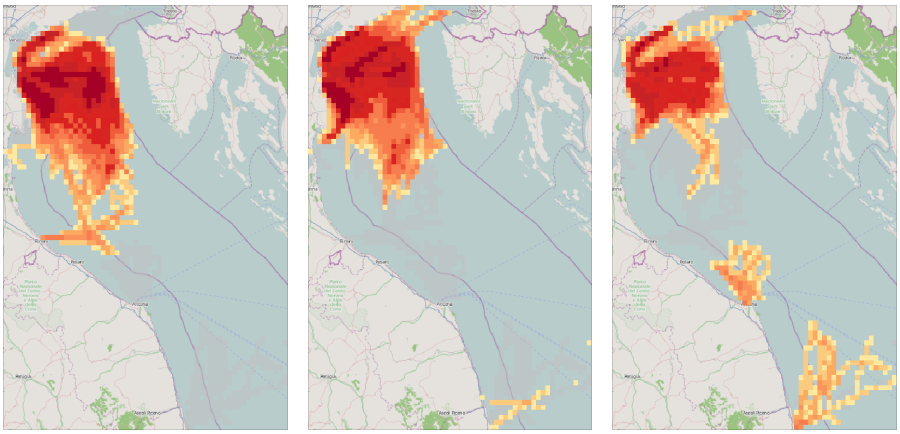
A different analysis in order to see how boats belonging to a particular navy contribute to the total fishing effort we have just observed, we can make a slice on our data cube selecting the navy we are interested in. In Figure 6.21 we can see the effort due to Chioggia navy's boats. Looking at this picture we can notice that along the year some Chioggia boats change their base harbour in order to reach far places for fishing. In particular we can see that in Winter the fishing area is more



(a) January-March (b) April-June (c) July-September

Figure 6.20: Fishing effort distribution by trimesters

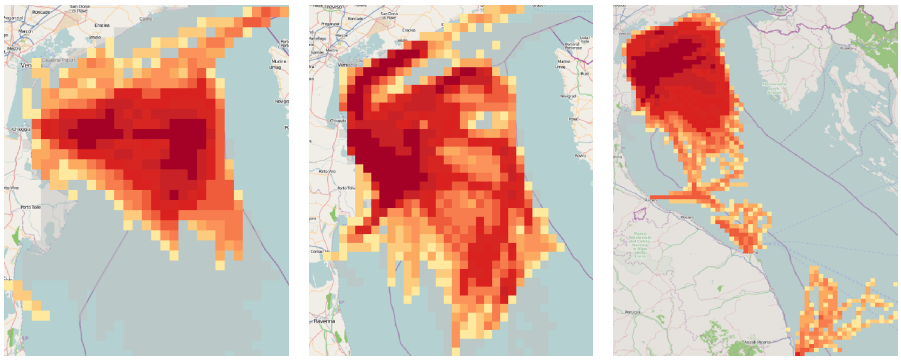
extended on South thanks to the use of Rimini harbour. During Spring boats reach Northern areas and leave southern ones; finally during Summer when the explored area is reduced with respect to the other seasons, due to laws that block fishing activities on these months, many boats use some more Southern harbours for their activities. During all the given periods, the areas nearest the coasts are the most exploited, while during Winter also farthest areas are reached.



(a) January-March (b) April-June (c) July-September

Figure 6.21: Fishing effort distribution of Chioggia boats by trimesters

By making a drill-down operation on the tools dimension we can observe how the different tools are used for fishing activity. Figure 6.22 represents the fishing effort obtained by using the different fishing tools, namely *coccia* (Figure 6.22(a)), *rapido* (Figure 6.22(b)) and *volante* (Figure 6.22(c)). By looking at this picture we can remark how the distribution of the effort changes depending on the used tool.



(a) *Coccia* (b) *Rapido* (c) *Volante*

Figure 6.22: Fishing effort distribution related to the used tools (January-September)

Figure 6.22(a) shows that *coccia* is used for fishing in an area that recalls a triangle whose head point is on the Chioggia harbour. One can also identify two different areas, one close to the coast, and another a bit far from it, where the fishing effort results higher.

Referring to Figure 6.22(b) the distribution of the fishing effort related to *rapido* appears to be different than the previous one: most exploited areas appear to be two belts close to the coast, one on the North and the other on the South of Chioggia harbour; the rest of the exploited area appears to be uniformly visited, and this area results to be wider than the one observed for *coccia*.

Finally Figure 6.22(c) describes the distribution of the effort related to *volante*. Recalling that the picture refers only to Chioggia boats, one can observe that some of the boats using this tool are visible also on different harbours than the Chioggia one. The same behaviour could be observed also on Figure 6.21(b) and Figure 6.21(c). So we can say that boats using Volante are also those using different harbours during the hot season. Finally, observing the figure we can see that all the area exploited with this tool is somehow uniformly explored, with the exception of an area between the Italian coast and the Croatian territorial waters where the values for the fishing effort is somehow high.

Species distribution

The second set of maps we created is intended to give environmental scientists an overall view on how each fished species in the North Adriatic Sea is distributed. In this case the data related to the sale at the fishing market have been distributed over each boat path, and then data have been aggregated in order to obtain information regarding each species. Our TDW, and the Visual Analytics Toolkit have been revealed to be useful tools in order to obtain this information. By aggregating data over the fishes dimension at the species granularity and by defining the desired

spatial and temporal levels for the corresponding hierarchies, we can obtain different snapshots about the distribution of the fishes.

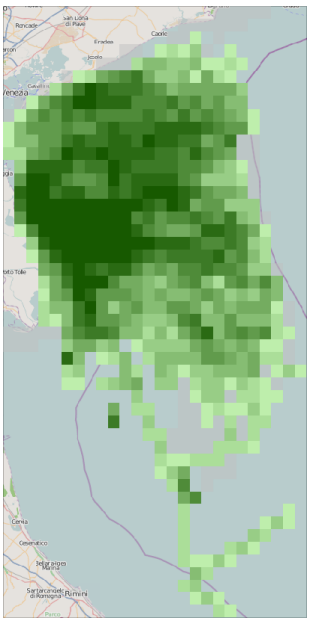
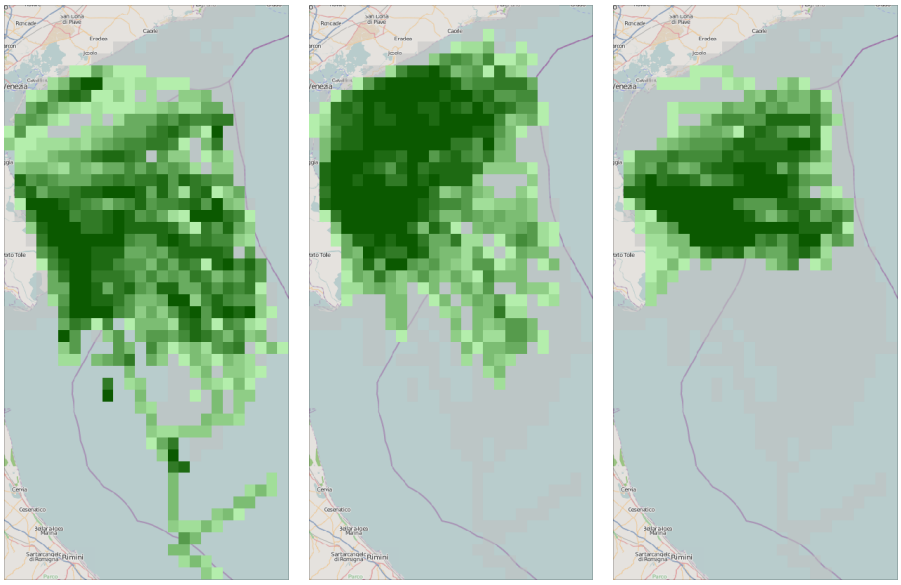


Figure 6.23: Spatial distribution of *anchovies* in the period January-September

Figure 6.23 represents the spatial distribution of *anchovies*, by means of the amount of fishes caught in a given spatio-temporal cell, during the whole period of interest(January-September 2007). Analysing the picture we can see how the density of this kind of fishes is considerable in all the areas of the Northern Adriatic, and reaches its maximum values in an area at South of Chioggia. In that area many important rivers, as Po, Brenta and Adige, have their mouth and reach the sea. By making a drill-down on the temporal dimension we can have a more precise view concerning seasonal changes of fishes. Figure 6.24 shows how the distribution of the *anchovies* varies during the first three trimesters of the 2007. This figure depicts a different scenario than the one given by Figure 6.23. During Winter (Figure 6.24(a)) the *anchovies* are mostly concentrated near the coasts, while the density results lower, either if relevant, on open waters. During Spring (Figure 6.24(b)) the area occupied by the *anchovies* is reduced, while the density of this area is generally very high. This could suggest that shoals of *anchovies* usually compact themselves in this season. Finally, during Summer we can notice further reduction of the area occupied by the species, while the density in the occupied territory remains high.

Similar results have been obtained for every kind of fishes in the Adriatic Sea, and this can be done by simply changing the fish species on which we slice our data cube.



(a) January-March (b) April-June (c) July-September

Figure 6.24: Spatial distribution of *anchovies* by trimesters

A comparison with environmental science techniques

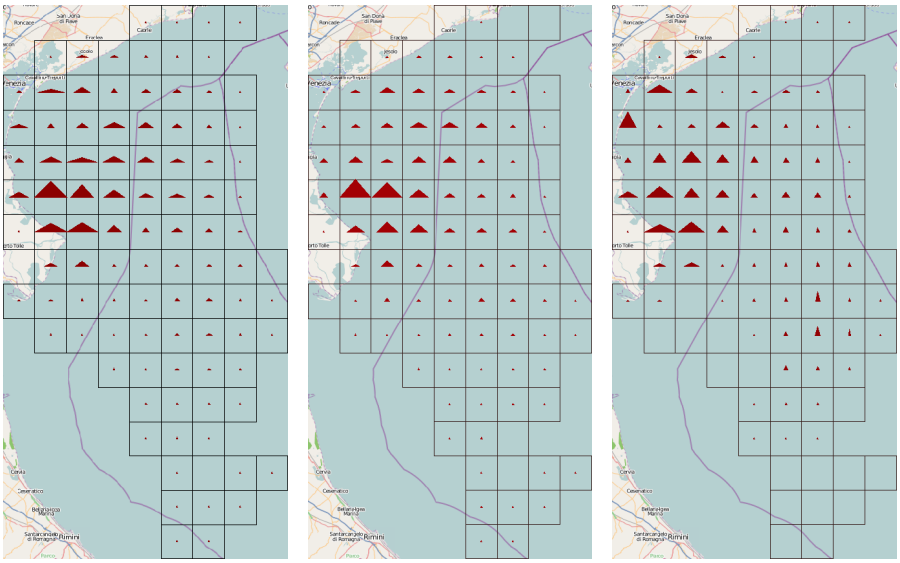
Beyond the results obtained using the analysis proposed so far, it is important to strength which are the benefits environmental sciences could have by using and adopting a tool like the proposed TDW.

Fishing Effort Indexes One of the most popular indexes for fishing effort currently used is the *CPUE* (*Catch Per Unit Effort*). This index has the advantage of giving information about the efficiency of a given fishing technique and the exploitation of a certain environment. On the other hand, the disadvantage of this kind of indexes is that they are based on the catches, and so they are not always suitable in order to give information for the management of the fishing effort. The effort index proposed in Section 6.2.4 presents two main advantages over this kind of indexes:

1. it is a measure of the effort based on *swept area* and not on catches

2. since it is based on boats trajectories, the information are related to a specific location and time, giving high resolution data that permit both to monitor the fishing effects on the species and to elaborate more precisely strategies for fishing management based on effort limitations [53].

By using our TDW and visual analysis tool it is anyway possible to obtain information as those given by the CPUE index, by bringing together catches and effort data as shown in Figure 6.25. These pictures show, with the help of triangles, the



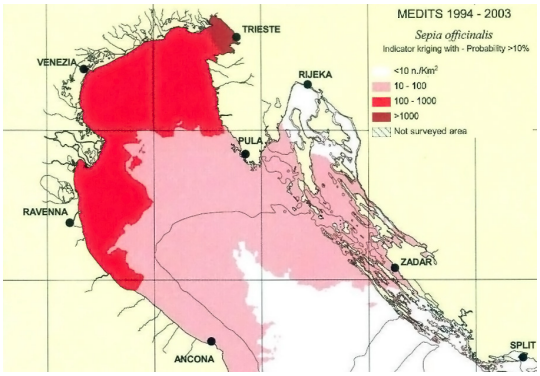
(a) Any fish (b) *Anchovies* (c) *Cuttlefishes*

Figure 6.25: Correlation between catches (height) and fishing effort (base) in the period January-September

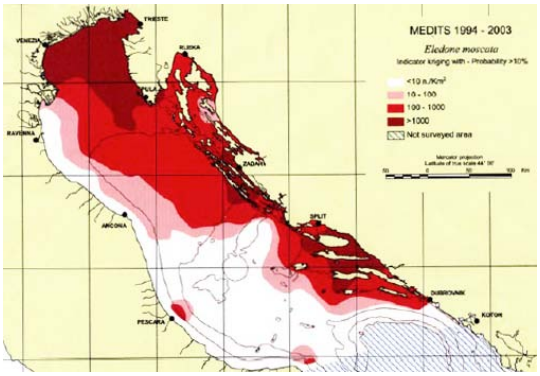
relation between the amount of catches in a given cell (height of the triangle) and the effort computed for that cell (base of the triangle). In particular, Figure 6.25(a) illustrates the correlation between the two values related to the total of the catches independently from the fish species, while Figure 6.25(b) and Figure 6.25(c) represent the situation related to *anchovies* and *cuttlefishes*. Figures refer to data for the whole period of interest, and spatial cells have been aggregated to form 6x6 nautical miles squares. By observing the three figures we can note that in general to a higher effort corresponds more catches. By the way, in some cells we can see how either with a high value of the effort, the number of catches is not commensurate. In this case we have triangles with a large base but a small height. On some other cells, mostly those farthest from the coast, in the picture related to *cuttlefishes*, we have the opposite situation, where lots of catches can be made either with little effort.

Fishes Distribution Currently, the most used techniques in order to estimate the distribution of the species on the sea are based on annual samplings made with trawl nets and respecting defined protocols. An example of results obtained with one of this campaign is shown in Figure 6.26, that presents some maps obtained during a MEDITS (*Mediterranean Trawl Survey*) campaign, that is done exactly in this way. Maps are related to four different kinds of fishes, *cuttlefishes* (Figure 6.26(a)), *eledones* (Figure 6.26(b)), *cods* (Figure 6.26(c)) and *squids* (Figure 6.26(d)).

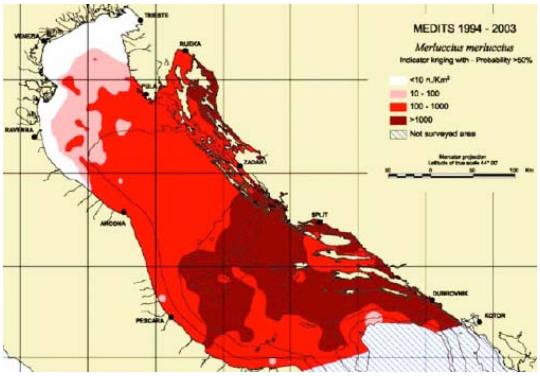
On the other hand, our results are based on *fishery dependent* data, i.e. the data are affected to the fishing strategies each boat adopts: areas with a high presence



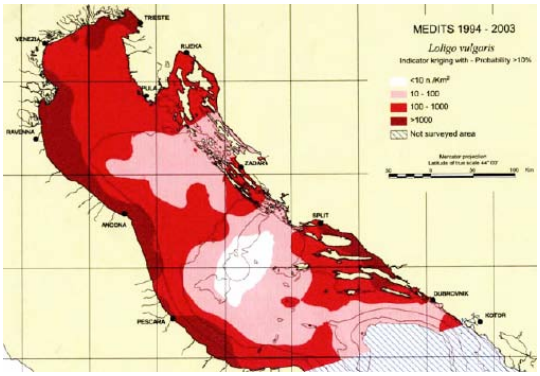
(a) *Cuttlefishes*



(b) *Eledones*

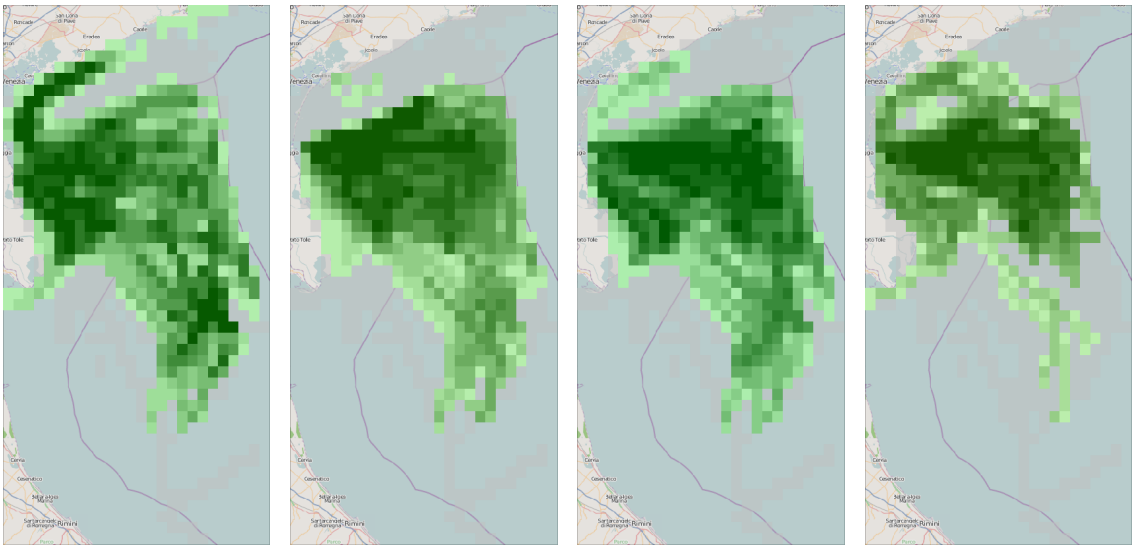


(c) *Cods*



(d) *Squids*

Figure 6.26: MEDITS species distribution maps



(a) *Cuttlefishes* (b) *Eledones* (c) *Cods* (d) *Squids*

Figure 6.27: TDW species distribution maps

of fishes will be more exploited than those where the presence is low. Despite this, the high availability of such data, combined with the high spatial and temporal resolution they have, really compensate this limit.

Figure 6.27 represents the map obtained by our TDW for the same species presented in Figure 6.26. A first thing to remark is how the detail level one can achieve by our tool is deeper than the one obtained by the MEDITS campaigns. This bigger detail level gives environmental scientists the ability to find out also small variations on the density of a given species. For example considering the map of *Cuttlefishes* (Figure 6.27(a)), one can observe a high density area around the middle of the figure, that is not present in Figure 6.26(a).

Comparing Figure 6.26(c) and Fig 6.27(c) can be noted how the MEDITS data and the data exploited by our TDW are contradictories: TDW data show how the bigger part of caught of *cods* are in areas where the density of this species is minimum with respect to MEDITS data. This contradiction is either more evident comparing Figure 6.26(d) with Figure 6.27(d). Here one can see how the catches of *squids* result to be higher where MEDITS map reports a lower presence of the species, and vice-versa. Recalling that maps related to MEDITS campaign are based on single annual samplings, these differences highlight the higher accuracy we can obtain with fishery dependent data, that results to be also more precise, environmental scientists said.

Finally, while MEDITS maps can give information only in an annual base, data given by the TDW approach can be analysed also on different time levels, so that also small variations in time can be achieved. This can be helpful in order to propose new rules for the preservation of fish species, for instance by introducing limitation on the amount of fish that can be caught in a given area during a given period to help those species to reproduce.

6.3 Synopsis

In this chapter we presented two different case studies in order to highlight the usefulness of the proposed framework and various kinds of analysis. The first case study shows a constrained scenario in which objects of interest are cars moving along the road network of the city of Milan, in Italy. The second case study concerns fishing boats sailing on the Adriatic Sea without any constrained movement. For both use cases we illustrated the preprocessing step for obtaining trajectories and how to instantiate our framework to build an adequate TDW. Finally, we discussed some interesting analyses taking into account requirements of the two scenarios, and drew some conclusions about what can be obtained by using our framework.

7

Conclusions

In this thesis we have proposed a theoretical and general framework for the management and analysis of trajectories data. The framework can be exploited in different application scenarios and is able to handle time-stamped locations that arrive in streams from variety of moving objects, from cars to boats.

The core module of the framework is the Trajectory Data Warehouse, a data warehouse aimed at storing aggregate measures computed over trajectories of moving objects. The model supports a flexible discretization of the spatial and temporal domains, along with the associated hierarchies, and allows for any number of other dimensions to be created. In this way users can obtain a more suitable model and representation of the reality where moving objects are embedded.

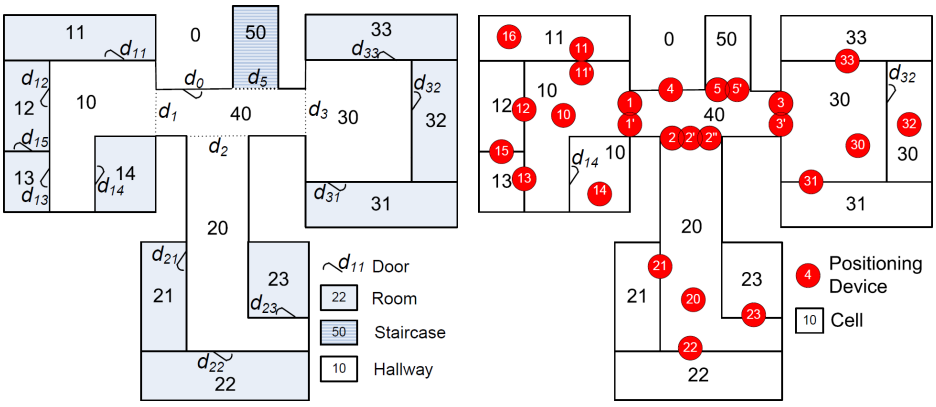
Concerning the model, we have formally defined abstract spatial and temporal dimensions, as well as some significant measures to be stored in the facts of the data warehouse. Moreover, we have provided adequate aggregate functions for the proposed measures, in order to support OLAP capabilities. Among the measures, we have introduced and analysed in depth the aggregation function of the measure *Visits* ($\mathcal{V}$), which represents the number of times a given granule has been visited by all the various trajectories. With respect to the measure *Presence* ($\mathcal{P}$), which counts the number of *distinct* trajectories inside a certain granule, *Visits* keeps track of the fact that an object can return in a place different times. We have formally proved that *Visits* can be computed in an exact way by our TDW, and it provides a very good approximation of *Presence*. This result is independent of the discretization of the spatio-temporal domains and the specific hierarchies adopted in the TDW according to the application scenario. This fact is definitively of interest, since the aggregate function for *Presence* is *holistic* [27] and such a kind of functions represents a big issue for data warehouse technology (i.e., there is the need to maintain original data in order to correctly compute aggregate values for this kind of measures).

In order to feed-up the data warehouse with trajectory data, the framework has been designed with a trajectory reconstruction module, that is in charge of processing the raw location data received as a stream of points from external devices. Trajectories are the output of this module, modelled by taking into account possible movement constraint or other parameters trajectory related. These trajectories are then used to compute aggregate measures to be stored in the data warehouse facts,

during the ETL phase.

In order to query the TDW and perform OLAP operations, the framework provides users with a visual interface. This module can be exploited in order to perform OLAP operations over the data, as roll-up queries, and permits to analyse the information contained into the data warehouse with the help of visualization techniques like graphs or animated maps.

Finally, in order to demonstrate the functionalities of our framework we have experimented it with two large real world datasets. The first dataset has been used in order to perform some traffic analysis. It contains in fact data related to cars moving in the city of Milan, collected during a week. The second dataset has been used to perform analysis on fishing activity in the North Adriatic sea. This second data collection contains more descriptive information related to the observed moving objects (i.e. boats), so it has been helpful in order to test both the trajectory data warehouse and the visual interface on a model with different dimensions other than the spatial and the temporal one.



(a) Indoor setting (b) Locating devices deployment

Figure 7.1: Example of indoor space

An interesting topic for future research consists of applying our framework in a different kind of scenario. In fact, in the last few years researcher started putting more attention to indoor movements, as those made by users inside buildings, such as airports, malls, museums. These scenarios concern objects/people moving in indoor spaces [38]. This indoor setting differs from the outdoor one primarily for two reasons. First of all in indoor spaces movements are strongly constrained, as people can move between rooms only by passing through doors. An example of an indoor scenario is depicted in Figure 7.1(a). Beside this, GPS-like positioning is usually not available indoor. Object positions are usually revealed, using Bluetooth, RFID, WiFi hotspots, by a set of short-range proximity detectors positioned inside rooms and at doors. These sensors have limited covering area, and often rely on proximity analysis and are unable to report velocities or accurate locations as GPS-devices.

Indoor spaces exhibit complex topologies, and the positioning is usually much less reliable and accurate than that achieved in outdoor scenarios. In particular, an indoor object is detected only when it enters the activation range of a positioning device, e.g., an RFID reader or a Bluetooth base station. Depending on the deployment of devices, such detections occur more or less frequently. As a result, the indoor positioning technologies create much more uncertain tracking data in indoor spaces when compared to outdoor settings. Figure 7.1(b) shows a possible positioning device deployment, where the numbered circles indicate the positioning devices and their activation ranges.

Some examples of TDW queries that could be interesting to answer in this scenario are *"At which floor of the building is there the largest number of people leaving at the same time?"*, or *"Which are the most crossed doors of the building during a fire evacuation drill? Is there any bottleneck in evacuation paths?"*.

Two examples of this application scenario are the study of movement of people with a Bluetooth enabled mobile device moving inside an airport [38] and the study of the behaviour of the customers of a city mall. But also an apparently different scenario, such as the analysis of the movement of luggages with RFID tags inside an airport logistic infrastructure, is likely to have similar requirements.

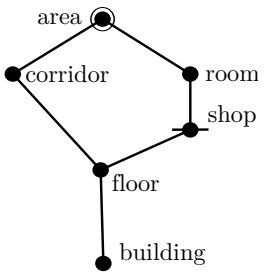


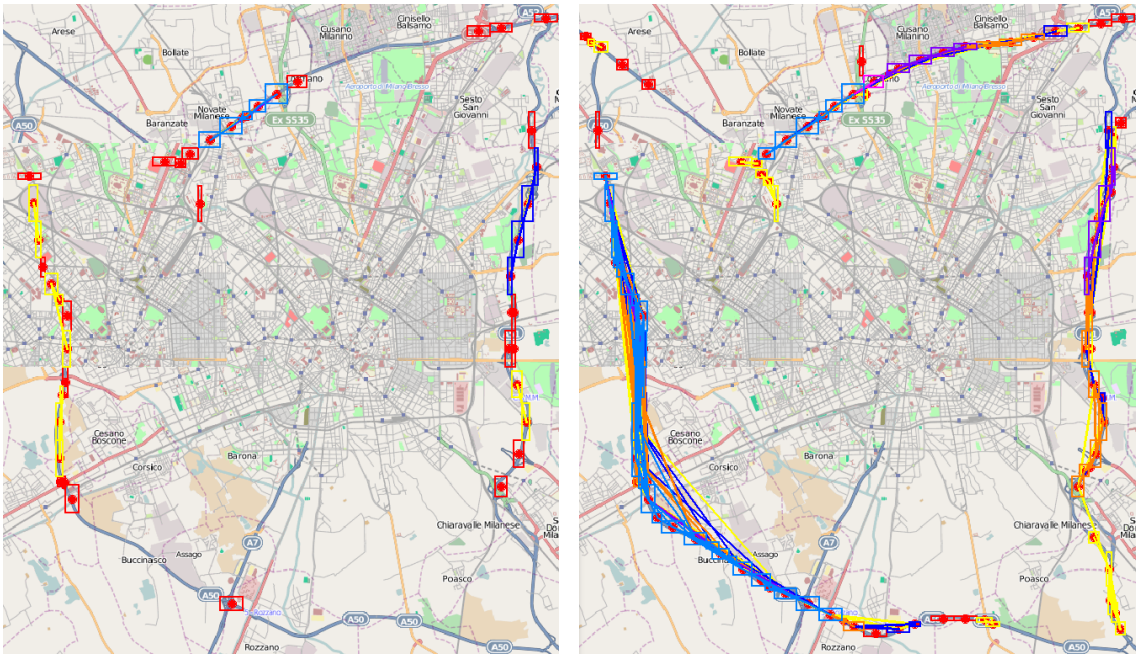
Figure 7.2: An examples of spatial hierarchies for indoor setting

We believe that the TDW model we presented in this thesis could be easily instantiated in order to model also this kind of scenario. In particular, Figure 7.2 shows an example of a hierarchy related to an indoor environment. The example considers the analysis of the behaviour of customers in a shopping mall. We suppose that after some preprocessing of the list of tracking devices (Bluetooth base stations, WiFi hotspots, ...) spotting a specific phone we are able to decide that the corresponding user is inside a specific area. These areas are the base granules of the spatial hierarchy. Each area could be contained either into a corridor or a room. Rooms may be part of a shop (or service rooms otherwise). Finally, corridors, rooms and shops are located into a building floor. Moreover, the measures crosses and visits can be used to answer queries like the one used in the example above.

A second promising line of research concerns the definition of complex measures obtained by a data mining process. A preliminary work on this can be found in [46]. In this work we are interested in extracting frequent patterns consisting of sets of

spatial regions, frequently visited by a large number of trajectories. To define the relevant spatial regions, the concept of *region of interest* (ROI) is introduced. A ROI is an area to be considered as important for some reasons related to the observed phenomena. ROIs can be both calculated on the data by some defined algorithm, or given by the user as areas considered of any interest to him/her, or for the observed scenario.

Now we can define a *frequent set of ROIs* (FSROIs) as a collection of ROI which are visited together, even if in different order, by a large number of trajectories. These patterns are obtained by a data mining process on trajectories and adequate aggregate functions are designed to roll-up the frequent patterns stored in the base cells of the data cube.



(a) Sunday (b) Tuesday

Figure 7.3: FSROIs extracted from Milan dataset

In Figure 7.3 is reported an example of FSROIs extracted on the spatial domain of the Milan dataset (Section 6.1) for Sunday and Tuesday. The patterns have different colours depending on the number of ROIs they contain. First, one can observe that during the Sunday only short pieces of the external ring roads are used whereas in the Tuesday entire ring is intensively crossed. However, there are not patterns which cover long segments of the ring.

It is worth noting that the TDW model used in [46] could handle only regular grids as partitions for the spatial dimensions. This allows for an easier definition of ROIs. Using the model proposed in this thesis, base granules can be of any shape and dimension. Figure 7.4 shows an example patterns built by using as ROIs all the

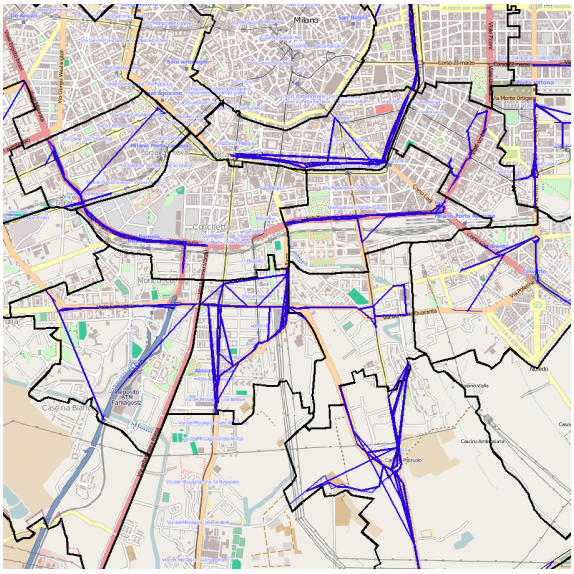


Figure 7.4: Patterns calculated starting from the road segments for each base cell

segments of the base granularity. As one can note, the visualization of such a measure is not straightforward, since the amount of information to visualize is usually huge, with lots of overlapping part. Hence, this research needs further developments, both in the definition of the ROIs and in the visualization of the results.

Bibliography

[1] J. F. Allen. A general model of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] G. Andrienko and N. Andrienko. Spatio-temporal aggregation for visual analysis of movements. In *Proceedings of VAST*, pages 51–58. IEEE, 2008.

[3] G. Andrienko and N. Andrienko. A general framework for using aggregation in visual exploration of movement data. *The Cartographic Journal*, 47(1):22–40, 2010.

[4] G. Andrienko, N. Andrienko, and S. Wrobel. Visual Analytics Tools for Analysis of Movement Data. *ACM SIGKDD Explorations*, 9(2):28–46, 2007.

[5] N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 2011.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *Proceedings of VLDB*, pages 853–864, 2005.

[7] D.R. Brillinger, H.K. Preisler, A.A. Ager, and K.G. Kie. An exploratory data analysis (eda) of the paths of moving animals. *Journal of Statistical Planning and Inference*, 122(2):43–63, 2004.

[8] T. Brinkhoff. A Framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[9] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB J.*, 15(3):211–228, 2006.

[10] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. 26:65–74, March 1997.

[11] W. Choi, D. Kwon, and S. Lee. Spatio-temporal data warehouses using an adaptive cell-based approach. *DKE*, 59(1):189–207, 2006.

[12] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line analytical processing) to User-Analysis: An IT mandate, 1993.

[13] V. Coors, C. Elting, C. Kray, and K. Laakso. Presenting route instructions on mobile devices: From textual directions to 3d visualization. In J. Dykes, A.M. MacEachren, and M.-J. Kraak, editors, *Exploring Geovisualization*, pages 529–550. Elsevier, 2005.

[14] P. Cudré-Mauroux, E. Wu, and S. Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *Proceedings of ICDE*, pages 109–120, 2010.

[15] J. da Silva, V. C. Times, and A. C. Salgado. An open source and web based framework for geographic and multidimensional processing. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 63–67. ACM, 2006.

[16] J. A. Dykes and D. M. Mountain. Seeking structure in records of spatio-temporal behaviour: visualization issues, efforts and applications. *Computational Statistics & Data Analysis*, 43(4):581–603, August 2003.

[17] J. Eder, C. Koncilia, and T. Morzy. The COMET metamodel for temporal data warehouses. In *Proceedings of the 14th Int. Conference on Advanced Information Systems Engineering (CAISE'02*, pages 83–99. Springer Verlag, 2002.

[18] M. J. Egenhofer. Topological relations between regions with holes. *International Journal of GIS*, 8:129–142, 1994.

[19] S. G. Eick. Visualizing multi-dimensional data. *SIGGRAPH Computer Graphics*, 34:61–67, February 2000.

[20] M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.

[21] P. Flajolet and G. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[22] P. Forer and O. Huisman. Space, time and sequencing: Substitution at the physical/virtual interface. *D.G. Janelle and D.C. Hodge (editors), Information, Place and Cyberspace: Issues in Accessibility*, pages 73–90, 2000.

[23] A. Fredrikson, C. North, C. Plaisant, and B. Shneiderman. Temporal, geographical and categorical aggregations viewed through coordinated displays: a case study with highway incident data. In *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM internation conference on Information and knowledge management*, NPIVM '99, pages 26–34, New York, NY, USA, 1999. ACM.

[24] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *International Journal of Cooperative Information Systems*, 7(2&3), 1998.

[25] L. I. Gómez, S. Haesevoets, B. Kuijpers, and A. A. Vaisman. Spatial aggregation: Data model and implementation. *CoRR*, 2007.

[26] L. I. Gómez, B. Kuijpers, B. Moelans, and A. A. Vaisman. A survey on spatio-temporal data warehousing. *International Journal of Data Warehousing and Mining*, 5(3):28–55, 2009.

[27] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. *DMKD Journal*, 1(1):29–54, 1997.

[28] D. Guo. Visual Analytics of Spatial Interaction Patterns for Pandemic Decision Support. *International Journal of Geographical Information Science*, 21(8):859–877, 2007.

[29] R. H. Güting, T. Behr, and C. Düntgen. Secondo: A platform for moving objects database research and for publishing and integrating research implementations. *IEEE Data Engineering Bulletin*, 33(2):56–63, 2010.

[30] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000.

[31] R. H. Güting, V. T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *VLDB Journal*, 15(2):165–190, 2006.

[32] J. Han, N. Stefanovic, and K. Kopersky. Selective materialization: An efficient method for spatial data cube construction. In *Proceedings of PAKDD*, pages 144–158, 1998.

[33] W. H. Inmon. *Building the data warehouse*. QED Information Sciences, Inc., Wellesley, MA, USA, 1992.

[34] W. H. Inmon and Richard D. Hackathorn. *Using the data warehouse*. Wiley-QED Publishing, Somerset, NJ, USA, 1994.

[35] W. H. Inmon, J. D. Welch, and K. L. Glassey. *Managing the data warehouse*. Wiley-QED Publishing, Somerset, NJ, USA, 1997.

[36] C. S. Jensen, A. Kligys, T. B. Pedersen, and I. Timko. Multidimensional data modeling for location-based services. *VLDB J.*, 13(1):1–21, 2004.

[37] C. S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *Proceedings of SSTD*, pages 208–227, 2009.

[38] C. S. Jensen, H. Lu, and B. Yang. Indoor - a new data management frontier. *IEEE Data Engineering Bulletin*, 33(2):12–17, 2010.

[39] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, editors. *Mastering the Information Age Solving Problems with Visual Analytics*, chapter Data Mining, pages 39–56. Eurographics Association, 2010.

[40] S. Y. Kim, Y. Jang, A. Mellema, D. S. Ebert, and T. Collinss. Visual analytics on mobile devices for emergency response. In *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 35–42, Washington, DC, USA, 2007. IEEE Computer Society.

[41] R. Kimball, L. Reeves, W. Thornthwaite, M. Ross, and W. Thornwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998.

[42] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker. *The Data Warehouse Lifecycle Toolkit.* John Wiley & Sons, 2008.

[43] M. Krstajić, E. Bertini, F. Mansmann, and D. A. Keim. Visual analysis of news streams with article threads. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, StreamKDD '10, pages 39–46, New York, NY, USA, 2010. ACM.

[44] J. Lee, A.B. South, and S. Jennings. Developing reliable, repeatable and accessible methods to provide high-resolution estimates of fishing-effort distributions from vessel monitoring system (VMS) data. *ICES Journal of Marine Science*, 67:1260–1271, 2010.

[45] L. Leonardi, G. Marketos, E. Frentzos, N. Giatrakos, S. Orlando, N. Pelekis, A. Raffaetà, A. Roncato, C. Silvestri, and Y. Theodoridis. T-Warehouse: Visual OLAP Analysis on Trajectory Data. In *Proceedings of ICDE*, pages 1141–1144, 2010. demo paper.

[46] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, and C. Silvestri. Frequent spatio-temporal patterns in trajectory data warehouses. In *Proceedings of of SAC*, pages 1433–1440. ACM, 2009.

[47] K. Liu, K. Deng, Z. Ding, M. Li, and X. Zhou. Moir/mt: Monitoring large-scale road network traffic in real-time. *Proceedings of VLDB Endowment*, 2(2):1538–1541, 2009.

[48] Y. Liu, D. Hill, L. Marini, R. Kooper, A. Rodriguez, and J. Myers. Web 2.0 geospatial visual analytics for improved urban flooding situational awareness and assessment. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 554–555, New York, NY, USA, 2009. ACM.

[49] E. Malinowski and E. Zimányi. Representing spatiality in a conceptual multidimensional model. In *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems, ACM GIS*, pages 12–22, 2004.

[50] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis. Building real world trajectory warehouses. In *Proceedings of the 7th International ACM MobiDE Workshop*, pages 8–15, 2008.

[51] B. McCormick, T. DeFanti, and M. Brown. Definition of visualization. *SIGGRAPH Computer Graphics*, 21:3–3, November 1987.

[52] A. O. Mendelzon and A. A. Vaisman. Temporal Queries in OLAP. In *Proceedings of the 26th International Conference on Very Large Data Bases*, Proceedings of VLDB, pages 242–253. Morgan Kaufmann Publishers Inc., 2000.

[53] C. M. Mills, S. E. Townsend, S. Jennings, P. D. Eastwood, and C. A. Houghton. Estimating high resolution trawl fishing effort from satellite-based vessel monitoring system data. *ICES Journal of Marine Science*, 64(2):248–255, 2007.

[54] I. S. Mumick, D. Quass, and B. S. Mumick. Maintenance of data cubes and summary tables in a warehouse. In *Proceedings of SIGMOD*, pages 100–111, 1997.

[55] S. Orlando, R. Orsini, A. Raffaetà, A. Roncato, and C. Silvestri. Trajectory Data Warehouses: Design and Implementation Issues. *Journal of Computing Science and Engineering*, 1(2):240–261, 2007.

[56] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing Spatio-Temporal Data Warehouses. In *Proceedings of ICDE*, pages 166–175, 2002.

[57] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of VLDB*, pages 802–813, 2003.

[58] T. Pedersen and N. Tryfona. Pre-aggregation in Spatial Data Warehouses. In *Proceedings of SSTD*, volume 2121 of *LNCS*, pages 460–480, 2001.

[59] N. Pelekis and Y. Theodoridis. Boosting location-based services with a moving object database engine. In *Proceedings of MobiDE*, pages 3–10, 2006.

[60] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes - A Framework for Location-Based Data Management. In *Proceedings of EDBT*, pages 1130–1134, 2006.

[61] D. Pfoser and C. S. Jensen. Trajectory indexing using movement constraints. *GeoInformatica*, 9(2):93–115, 2005.

[62] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel Approaches in Query Processing for Moving Object Trajectories. In *Proceedings of VLDB*, pages 395–406, 2000.

[63] A. Raffaeta, L. Leonardi, G. Marketos, G. Andrienko, N. Andrienko, E. Frentzos, N. Giatrakos, S. Orlando, N. Pelekis, A. Roncato, and C. Silvestri. Visual mobility analysis using t-warehouse. *International Journal of Data Warehousing and Mining*, 7(1):1–23, 2011.

[64] P. Rigaux and A. Voisard M. Scholl. *Spatial Database: With Application to Gis*. Morgan Kaufmann, 2001.

[65] S. Rivest, Y. Bédard, and P. Marchand. Towards better support for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). *Geomatica*, 55(4):539–555, 2001.

[66] S. Shekhar, C.T. Lu, X. Tan, S. Chawla, and R. Vatsavai. *Map Cube: a Visualization Tool for Spatial Data Warehouse*, chapter Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001.

[67] L. Speicys, C. S. Jensen, and A. Kligys. Computational data modeling for network-constrained moving objects. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems, ACM GIS*, GIS '03, pages 118–125, 2003.

[68] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8:52–65, 2002.

[69] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 10:571–588, 2002.

[70] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio-temporal aggregation using sketches. In *Proceedings of ICDE*, pages 214–225, 2004.

[71] Y. Tao and D. Papadias. Historical spatio-temporal aggregation. *ACM TOIS*, 23:61–102, 2005.

[72] J. J. Thomas and K. A. Cook, editors. *Illuminating the Path: The Research and development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[73] I. Timko and T. B. Pedersen. Capturing complex multidimensional data in location-based data warehouses. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 147–156, 2004.

[74] W. Tobler. Experiments in migration mapping by computer. *The American Cartographer*, 14(2):155–163, 1987.

[75] A. A. Vaisman and E. Zimányi. What is spatio-temporal data warehousing? In *Proceedings of the 11th DaWaK*, pages 9–23, Berlin, Heidelberg, 2009. Springer-Verlag.

[76] T. Wan, K. Zeitouni, and X. Meng. An OLAP system for network-constrained moving objects. In *Proceedings of SAC*, pages 13–18, 2007.

[77] J. Widom. Research problems in data warehousing. In *Proceedings of the 4th international conference on Information and knowledge management*, CIKM '95, pages 25–30, New York, NY, USA, 1995. ACM.

[78] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proceedings of SSDBM*, pages 111–122, 1998.

[79] M. Worboys. *GIS, a computing perspective*. Taylor & Francis, London; Bristol, PA, 1995.

[80] M. F. Worboys and E. Clementini. Integration of imperfect spatial information. *Journal of Visual Languages and Computing*, 12(1):61–80, 2001.