Towards efficient database support for spatial analysis

Łukasz Gancarz

Institute of Computer Science, Faculty of Electronics and Information Technology

Warsaw University of Technology

June 2006

To my Parents and Wife

Thank you

Space is big. You just won't believe how vastly, hugely, mind-bogglingly big it is. I mean, you may think it's long way down the road to the drug store, but that's just peanuts to the space.

Douglas Adams, The Hitchhiker's Guide to the Galaxy

Abstract

The thesis is devoted to improving the effectiveness of spatial analysis through the application of the dedicated physical data organisation and data processing methods. The dedicated physical data organisation is based on the feature common for all spatial analysis methods, which is related to the locality of data references. In the thesis, the data model comprising the conceptual, logical and physical layers and reflecting the properties of spatial phenomena is proposed. On the physical level, the performance improvement is obtained with the multidimensional hierarchical clustering which preserves semantical distance relations between spatial objects and phenomena. In order to benefit from the proposed data model, the thesis refers to the issues of optimizing data access and queries specific for spatial analysis.

Keywords: Spatial analysis, data modelling, complex spatial hierarchies, performance optimization, physical schema, multidimensional hierarchical clustering.

Abstrakt

Rozprawa doktorska dotyczy poprawy efektywności analizy przestrzennej poprzez zastosowanie dedykowanej fizycznej organizacji danych oraz metod ich przetwarzania. Dedykowana fizyczna organizacja danych bazuje na właściwości charakterystycznej dla wszystkich metod analizy przestrzennej, związanej z zasadą lokalności odwołań do danych. W rozprawie zaproponowano model danych, w tym warstwę konceptualną, logiczną i fizyczną, która odzwierciedla właściwości analizowanych zjawisk o charakterze przestrzennym. Poprawa wydajności na poziomie fizycznym związana jest z wielowymiarowym hierarchicznym grupowaniem danych odwzorowującym semantyczną bliskość obiektów i zjawisk przestrzennych. W celu wykorzystania właściwości zaproponowanego modelu danych rozprawa odnosi się do kwestii związanych z optymalizacją metod dostępu i zapytań charakterystycznych dla analizy przestrzennej.

Słowa kluczowe: Analiza przestrzenna, modelowanie danych, złożone hierarchie przestrzenne, optymalizacja wydajności, model fizyczny, wielowymiarowe hierarchiczne grupowanie.

Table of Contents

Chapter I

Introduction

1.1 Background

An interest in spatial data has grown considerably in the last decade mainly as a consequence of development and popularisation of geographic information systems (GIS) [Dem99]. The technological progress in GIS – intensified by improvements in hardware performance and the abundance of digital geo-referenced data (GPS, remote sensing) – made the implementation of GIS solutions practical for various organisations. The availability of the technology and geo-information made map-driven data visualisation and navigation a standard method of interaction between users and computer systems in many spatially-related disciplines [HA03]. In recent years, however, a new idea to extend the traditional functionality of GIS with spatial analysis (SA) has attracted the attention of researchers in academia and industry. The interest is based on a presumption that analytically enhanced GIS is able to deliver new insights and improve data exploration in terms of spatiality. The potential of such integration has been the main motivation for extensive research aimed at the development of GIS-centred analytical solutions that attempt to make the most of geographic data [FSU96, BMH00, EKS01].

The following Sections of the Chapter summarise the author's literature studies relating to the aforementioned research. The studies allowed to identify the fundamental features of all SA techniques and thus provided an insight necessary for the author's doctoral work. For the sake of brevity, however, the following overview is not a detailed discussion of SA techniques but rather a synthetic survey of the features which are characteristic of SA and which are essential for the thesis's objectives.

1.1.1 Locality in spatial analysis

In 1970 W. Tobler invoked the First Law of geography (TFL) claiming that "everything is related to everything else, but near things are more related than distant things" [Tob70]. In spite of the fact that TFL is only an empirical law, there is substantial evidence confirming the sensitivity of various phenomena to local interactions [Mil04]. The evidence – a result of research in many spatially-related disciplines – proved the validity of TFL, which led to the common acceptance of the law as the fundamental principle of spatial analysis (SA). Spatial analysis is a subfield of geography and regional science that studies properties varying with a geographic location and reveals relations among near entities at various levels of sophistication. SA derives from quantitative geography and was initially focused on the confirmatory analysis of hypotheses (CSA) by means of spatial statistics [Cre93, Han03, SU03]. The latest advances in the discipline show, however, not only the refinement of the conventional statistical approach to SA but also considerable progress in the exploratory spatial analysis (ESA). The main objective of ESA is the exploration of spatial databases for spatial patterns, associations, trends, and outliers [SHWL01, MH01]. As opposed to CSA, ESA is also better suited to cope with large volumes of spatial, incomplete or noisy data for which the conventional statistical techniques are inappropriate or cumbersome to use [Ope92, Fis93].

Although modern SA comprises many diverse techniques to deal with complex spatio-temporal processes, the spatial aspect of SA is always local. The common property results from the nature of spatial phenomena (TFL) and consequently from the character of SA, which explores relations among near entities or entities for which distance semantics exists. The following review of SA terminology also confirms the observation because terms describing spatial relations always refer to locality, e.g. nearness, proximity, spatial clusters and patterns, areas (e.g. territory, zone, district, province, region, state, country etc.), neighbourhood and topology (e.g. adjacent, overlapping etc). The subsequent Sections of the Chapter demonstrate how the locality concept underlies SA by providing a brief survey of the discipline. The survey includes an overview of both the traditional statistical approach to SA as well as recent developments in the area, i.e. spatial data warehousing, spatial OLAP, and spatial data mining. Concepts presented in the Chapter provided insightful input for the author's doctoral work, which is the development of the data model for SA described in Chapter IV.

1.1.2 Statistical spatial analysis

The traditional data analysis techniques often violate TFL by making an unrealistic assumption about the statistical independence of geo-referenced data. The inappropriateness of the assumption reveals itself for example in residual errors in regression modelling [SHWL01]. When data are spatially related, residual errors reveal systematic variation over space, which indicates that the model was unable to capture spatial relations existing in the data. Since nearby spatial objects systematically affect one another, the values of geo-referenced attributes reveal some degree of spatial autocorrelation (correlation relative to distance or topology) or heterogeneity (contradiction of global trends with distinct local trends) [CO73, SHWL01]. Initially, techniques for measuring and analysing the spatial dependencies were crude providing only the overall intensity of the spatial associations (Moran and Geary statistics) [Cre93]. Modern spatial statistics (e.g. local indicators of spatial association (LISA), the G statistics, geographically weighted regression) are able to capture all types of spatial dependencies by recognizing that every location has an inherent degree of uniqueness due to its situation relative to the rest of the spatial system [Han03, SU03].

Another SA techniques analyses the spatial patterns of entities (spatial structure) and flows that the entities generate (spatial interaction) [BH85]. The models of spatial structure concern the distributions of entities dependent upon specified assumptions about a nature of associated flows or interactions. The models of spatial interaction relate specifically to flows which are generated given the spatial distribution of geographical entities. The models of structure and interaction can be combined to deal with mutual adjustments that take place between structures and flows over varying time horizons. All the models are the special cases of general autocorrelation models but provide additional information about the dynamic nature of spatial systems using information about proximity, neighbouring relations, spatial hierarchies and distance decay profiles [Fot83].

The SA technique that also follows TFL is spatial interpolation used for the generation of missing or hidden variables in geographic space. The most sophisticated technique but producing finest interpolation estimate is kriging [Cre93]. The technique treats an interpolated variable as regionalized, i.e. the variable's values vary continuously across space according to some spatial lag or distance in a partly random and partly deterministic

manner. Together with adjustments based on qualitative information, kriging admits a wide range of distance functions and clustering patterns.

In addition to the above models, statistical SA also comprises specialised models for various application domains. In environmental sciences, SA provides models for individual (deterministic or stochastic) processes (e.g., physical dispersion, chemical reactions, biological systems) as well as comprehensive and integrated models for environmental management and economic assessment [GMZ04, WC04]. In social sciences (e.g. quantitative geography, regional economics) a wide range of spatial models has been proposed for describing, analysing and forecasting economic developments within a set of localities. Such models, however, are not only able to deal with the internal structures of regions and relations within one region but also with interregional relationships [GJ03].

1.1.3 Spatial on-line analytical processing (SOLAP)

On-line analytical processing (OLAP) is based on the multidimensional paradigm of data analysis which is a standard for many decision-support solutions. The main advantage of the approach is its ability to model processes as multidimensional space, which is consistent with the way in which practitioners perceive the processes [Cod93, Imo96, Tho97]. The multidimensional models are based on three fundamental concepts: facts, measures and dimensions [Kim96, CD97]. A measure is an attribute of a fact which represents a state of a situation with regard to some dimensions of interest. Each dimension possesses instances which are structured in a hierarchical manner, creating the different granularity levels of information. The multidimensional analysis consists in selection and aggregation of measures which may be additionally restricted by the hierarchical levels [GHQ95, GBLP96]. The intuitive data representation and flexible data manipulation resulted in the common acceptance of OLAP for confirmatory and explanatory data analysis in many application domains [Kim96, BS97, Tho97]. The confirmatory data analysis (CDA) provides an assessment of a degree of confidence that can be placed in an apparent effect (verification of hypothesis). The explanatory data analysis (EDA) is aimed at obtaining previously unknown and relevant information by the conscious exploration of associations and patterns in the data. Despite the differences in scope and objectives, both CDA and EDA are based on the same basic data manipulation operators, such as cube,

slice and dice, drill-across, drill-down, roll-up, pivoting etc. This is a consequence of the fact that the set of operators is characteristic of every OLAP activity involving the interactive and ad hoc exploration of multidimensional data.

The scope and quality of the multidimensional analysis can be considerably enhanced when spatial data are involved in the on-line analytical processing [BLPCL97, Car98, Gon99, BMH00, RBM01]. In terms of the multidimensional models, the integration entails an enhancement of the conventional OLAP model with attributes that contain spatial reference (i.e. coordinates, geometries, semantic spatial reference) either as the measures of a fact or as the attributes in dimensions [HSK98, Kim01]. In the case of dimensions, the spatial attributes usually define complex spatial hierarchies (concept hierarchies) which enable the alternative generalisations of leaf-level spatial entities to high-level concepts. The generalized concepts can be purely geometric, such as maps representing larger regions (spatial-to-spatial aggregation), or non-geometric, such as named areas or region discriminators (spatial-to-non-spatial aggregation) [LHO93, MZ04].

The inability of the conventional OLAP front-end tools to support the map-driven spatial data exploration resulted in a necessity to integrate the on-line analytical processing with GIS [MK01]. The integration created spatial OLAP (SOLAP) which is formally defined as "a visual platform built especially to support rapid and easy spatio-temporal analysis and exploration of data following a multidimensional approach comprised of aggregation levels available in cartographic displays as well as in tabular and diagram displays" [BLPCL97, RBPN03]. In addition to the cartographic visualization and standard OLAP operators, SOLAP supports navigation between the different geometric levels of details inside a geometric or mixed spatial dimension, while keeping the same level of thematic granularity. This can be used in SOLAP applications as an alternative method to navigate from fine granularity to coarse granularity (spatial generalisation) [LHO93].

1.1.4 Spatial data mining

Initially, data analysis was strictly related to statistics by focusing mainly on primary data analysis, e.g. confirmation of hypothesis [GMPS96, Han98]. Although numerous statistical exploratory data analysis (EDA) techniques have been developed in recent years, the essence of the statistical data analysis remains unchanged (e.g., central role of a model,

inability to cope with noisy and unstructured data, poor performance on large data volumes) [EP96, Han98]. Latest technological advances in data acquisition and computing (database systems, machine learning, artificial intelligence, pattern recognition and visualisation) led to an invention of a different approach to data analysis. The approach known as Knowledge Discovery in Databases (KDD) is a cyclic, multi-stage EDA process including a phase called data mining (DM). DM is an activity aimed at finding "implicit, potentially useful, previously unknown and relevant information in data" [PF91]. The goal is achieved by the application of techniques such as clustering, classification, detection of associations or outliers, prediction, forecasting, concept generalisation and visualisation [PF91, HK01]. Although the general-purpose DM techniques have been successfully applied in many application domains, the techniques may produce biased or inconsistent results in the presence of spatial data. This is a consequence of an inability of traditional DM to correctly handle spatial data and to capture spatial autocorrelation. The inability confronted with a necessity existing in many disciplines to analyse spatial phenomena entailed some modifications of conventional DM algorithms to spatially enhance the traditional KDD process [KHA96, EKS99, MH01]. The following part of the Section surveys several spatial DM techniques with an intention to demonstrate that the DM approach to SA is consistent with TFL. The central role of locality and topological relations existing in every spatial DM technique provides crucial insight for the author's doctoral work, which is the development of the data model efficiently supporting SA.

Spatial associations enhance the traditional association rules with spatial predicates in a precedent or an antecedent. The pioneer technique for the discovery of spatial associations [KH95] explores relations among objects whose minimal bounding rectangles are located in a distance no greater than a threshold. As a result, the technique supports topological predicates such as near, adjacent, intersect, and distance, organised in the form of a concept hierarchy of the topology relations (feature-centric model). Co-location rules are another specific generalization of the association rules to spatial data, discovering the subsets of spatial objects which are frequently located together (event-centric model) [SH01, SHWL01]. Both the spatial association and co-location rules are the examples of spatial characterisation which describe relations existing among spatial objects based on the common properties of the spatial objects and their neighbours [EGKS99].

Spatial classification is a data analysis technique that organises spatial objects into categories that recognize the spatial properties (e.g. distance, direction, topology, geometry) of the objects. However, the complexity of the spatial properties in terms of structure and semantics requires some modifications of traditional classification techniques in order to generate correct results. Consequently, all the classification techniques for spatial data are sensitive to distance-based proximity among objects [KHS98] or more sophisticated relationships in a defined neighbourhood [EKS97, EKS01]. Modelling spatial discrepancy can also improve classification accuracy by the enhancement of the traditional models with congruity matrix that captures spatial autocorrelation [SZHV03].

Clustering is an automatic identification of groups (clusters) of similar objects with respect to a similarity measure. Spatial clustering consists in the grouping of spatial objects using a distance or topology relation as a similarity measure [HKT00, HK01]. Although the algorithms for spatial data clustering can be divided into four apparently different categories (i.e. partitioning, hierarchical, density-based and grid-based), the heuristics underlying all the categories pursue the same objective, i.e. clustering near objects into groups which are separable from other groups of locally near objects. Another SA technique related to clustering is outlier analysis which consists in the identification of objects not belonging to any cluster. Outlier analysis in spatial data also focuses on the identification of objects inconsistent with their neighbours [SHWL01].

Spatial trend analysis consists in the exploration of regular changes in the values of non-spatial attributes observed during change in location. The existence of a positive/negative trend is confirmed when the values of the non-spatial attributes increase/decrease along all movement paths. The paths are usually modelled as neighbour paths and regression analysis is performed to describe the trend's regularity [EKS01]. An alternative spatial trend detection technique infers direction relations among objects based on the relations of their ancestor regions or through some chains of the objects using path consistency [PE97]. The approach is an example of the hierarchical spatial reasoning which is a data analysis theory based on the hierarchical organisation of geographic space [Car98a].

In addition to the techniques derived from the traditional DM, some dedicated algorithms have been proposed to capture the specifics of the SA. Three such techniques are described below, namely knowledge discovery based on concept hierarchies, spatial aggregation, and extraction of proximity patterns by concept generalisation. The techniques are

representatives of the spatial DM that focus on the exploration of proximity relations at the different levels of aggregation and conducted along spatial hierarchies.

Generalisation-based knowledge discovery requires background knowledge in the form of concept hierarchies. In spatial databases, there are two kinds of concept hierarchies: non-spatial and spatial, that can be explicitly provided by experts or generated automatically [HF94]. The attribute oriented induction is performed by climbing concept hierarchies and summarizing relationships between spatial and non-spatial data at higher concept levels. The process is continued until a number of different values for an attribute is not greater than a generalisation threshold for the attribute. Depending on an order of the analysis two opposing approaches, favouring either spatial (spatial-data-dominant) on non-spatial (non-spatial-data-dominant) data, are feasible [LHO93].

Spatial aggregation consists in the computation of multi-layer spatial aggregates that form equivalence classes and adjacency relations [BZY96, HZ98]. The approach enables the automatic identification and extraction of features by organizing similar objects into coherent geometric structures. The structures can be further transformed into higher-level spatial representations and produce a multi-layer ad hoc hierarchy of iso-geometries [YZ96, HZ98]. The hierarchical levels are typically modelled as neighbourhood graphs and may also serve as an interface to modularise computations.

The extraction of proximity patterns by concept generalisation is a technique for the discovery of properties located in proximity to clusters [KN95]. The technique enhances spatial clustering by the identification of distinguishing features or concepts that serve as discriminators among clusters. The discriminating features are usually organized with respect to concept hierarchies with intention to identify the maximal discriminators, i.e. the features that most adequately fit spatial patterns.

1.2 Motivation

Accessing large data volumes which are stored on magnetic disks is orders of magnitude slower than accessing the data at higher memory levels. The architectural property of modern computer systems makes the massive data processing, which results in a large number of input/output (I/O) operations, the major performance bottleneck [Vit01]. The

effective I/O communication is also challenging for analytical applications since most data analysis algorithms are I/O bound [AS96, IM96, Vit01]. The unsatisfactory performance is even more evident when spatial data are involved in the data exploration [EKS97, EFKS00, Kub01]. The problem results from the character and scope of SA as well as the complexity of data models and data analysis techniques.

The data models for SA usually have a complicated structure. For instance, the multidimensional character of SA manifests itself in a number of features (attributes) used to model a spatial phenomenon as multidimensional data space. In order to improve the expressiveness of such models, various data types are applied for the attributes, i.e. spatial (e.g., coordinates, geometries) and non-spatial (e.g., numerical, descriptive). The coexistence of the various data types entails integrating specialised data structures with metadata to ensure the clarity and consistency of the models. The metadata may also store additional information about certain properties of spatial phenomena which are essential for SA techniques, such as concept hierarchies, topologies etc.

SA characterises the unique data processing profile which is a combination of the traditional data exploration and the typical spatial queries. The ad hoc complex operations (e.g. range queries, nearest-neighbour queries, spatial joins) spanning both spatial and non-spatial data are time-consuming especially in the presence of complex data models and large data volumes. The crucial but most difficult operation to manage is the retrieval of measures (e.g. cube operator) in the scope determined by restricted dimensional attributes, including predicates on spatial attributes (e.g. spatial join). Such query pattern is predominant in most SA techniques and thus requires special consideration.

The author's study of SA confirms that the requirements of the discipline in the area of data organisation and data processing are complex. The study also confirms the author's presumption that embedding data management in the core of SA algorithms is a challenging task and rarely yields satisfactory results. A solution that appears to adequately address the problem is a data abstraction layer separating the logic of SA algorithms from the data storage. The data layer would consist of a data model capable of handling both spatial and non-spatial data as well as data access primitives improving the effectiveness of I/O communication. Such architecture could provide a standardised data interface for SA algorithms and because of its efficiency deliver considerable performance gains. Given the

above assumptions, the development of the data model efficiently supporting SA (mainly in terms of I/O operations) is the primary objective of the thesis.

1.3 Edith and data mining projects for PTC

In addition to the motivation given in Section 1.2, the author's research concerning the performance improvement of analytical processing in the presence of spatial data was also motivated by the author's participation in the following R&D projects: "Development Data Mining Platform for Technical Department of Polska Telefonia Cyfrowa (PTC)" and EDITH. The scope of the data mining projects for PTC involved the analysis of various challenges of the mobile telecom provider in the area of network operations and performance [DGGKMMORTW00]. Data for the projects were collected at network cell stations which allowed to position the phenomena geographically which were subsequently analysed with data mining algorithms. The availability of the geo-locations enhanced the analysis and improved the quality of obtained results because much of the inferred knowledge involved spatial properties.

The objective of the EDITH project (http://edith.in.tum.de) was to introduce multidimensional hierarchical clustering (MHC/HI) into data warehousing and to integrate MHC/HI into a database engine Trasbase. The project scope also included the development of a modelling methodology for hierarchically-organised multidimensional databases as well as the its verification in real-life application domains. Additionally, the applicability of MHC/HI to spatial analytical processing was performed using an application (GIS-Warehouse) combining the functionality of a data warehouse with that of a geographic information system Smallworld [GW03].

1.4 Related work

The academic and industrial research related to data analysis can be divided into two categories. The first category concerns the design of new algorithms enabling progressively sophisticated and accurate data exploration. The second category focuses on performance improvements necessary to make analytical applications efficient in data rich

environments. Although the first category dominates the scientific literature, numerous papers on the performance-related aspects of the analytical processing have been published in recent years. The papers acknowledge that the main-memory data structures are inefficient when data volumes exceed the amount of available memory (memory swapping effect). To overcome the problem, the papers propose the rationalisation of I/O transfers (the major bottleneck of the analytical processing of large data sets) through the integration of data analysis algorithms with specialised data management systems [CHY96, EFKS00, Kub01] For example, a methodology for the tightly-coupled integration of data mining applications with relational database systems was presented by Agrawal et al. [AS96]. The methodology recommends the implementation of data analysis algorithms using a database programming language. As a result, most computations are performed within a DBMS as opposed to an approach where data are retrieved from a database to an application written in a general-purpose programming language. Imielinski et al. [IM96] postulate even further integration by treating data exploration as a querying process which sets new challenges to database technology. The challenges include new data models, query languages, basic operators, and query processing strategies which efficiently support the advanced concepts and techniques of data analysis.

The recognition of the distinct properties of the analytical data processing has reinforced the motivation to integrate many performance-oriented enhancements into DBMS engines. The following Sections of the Chapter overview some of the enhancements, focusing mainly on those which apply to SA. The overview is also a description of work related to the author's doctoral research. The identification of the major deficiencies of the aforementioned enhancements allows to position the outcome of the thesis with the state-of-the-art DBMS techniques supporting SA.

1.4.1 Pre-computation of aggregates

One of the heuristics aimed at the performance improvement of data analysis consists in pre-computing aggregates containing summarised data which are likely to be accessed frequently [Raf03] or are relevant for the analysis [MMK04, MMWZ05]. In the context of SA, the pre-computation has been mainly discussed for the multidimensional analytical processing, i.e. spatial data warehousing and spatial OLAP, in order to improve query

response time. Since the conventional spatial databases are inefficient in handling ad hoc queries of summarized data, the spatial extension of data cubes seems reasonable. Han et al. [HSK98, SHK00] were first to introduce the concept of the spatial data cube by describing multidimensional data models with spatial measures and dimensions. The paper postulates that data models containing only spatial dimensions could be implemented in a way similar to the non-spatial data cubes. For models containing spatial measures, the materialized view approach (pre-computation) for the merge operations of spatial regions is proposed. The goal of the materialization is to select and merge groups of adjacent spatial objects that, given storage space constraints, provide the shortest time to evaluate query results. However, when a priori knowledge about a grouping hierarchy is not available, the conventional pre-aggregation techniques used to improve OLAP performance cannot be applied. One of the solutions which attempts to address the problem was proposed by Papadis et al. [PKZT01]. The solution consists in the adaptation of the modified R-tree to derive a definition of a grouping hierarchy from the index structure. Such implicit hierarchy can be subsequently incorporated into a conventional lattice model (Harinarayan et al. [HRU96]) and appropriate aggregations for materialization can be selected. An alternative approach proposed for the implementation of a spatial data cube with no apparent spatial hierarchy required dynamic calculations. In order to improve the performance of the calculations, Prasher et al. [PZ04] presented an algorithm for the dynamic aggregation of spatial data at variable resolutions. The multi-resolution algorithm allows for the fast generation of intermediate results before committing additional time and resources to the typically time-consuming amalgamation of spatial objects. This algorithm is an example of the progressive refinement approach which is typical of many SA techniques [BKSS94, KH95, KHA96]. The performance improvement strategy assumes that the exploration of patterns at high conceptual levels may apply efficient spatial algorithms at relatively coarse resolution. Only candidate spatial predicates which are worth detailed examination are computed by the refined techniques. The multi-level data processing saves computation effort especially when the computation of all possible spatial relations is time-consuming.

1.4.2 Indexing and clustering

Although the pre-computation offers best query response time, other approaches have also been considered to avoid the drawbacks of the technique (e.g. time-consuming maintenance, large storage requirements). The alternative approach consists in indexing and clustering (see Sections 3.1.1 and 3.1.2) to improve the performance of data analysis tasks [GG98, KMP03, MMNM03]. However, the character of SA (see Section 1.2) requires some modifications of the conventional indexing and clustering techniques which are inefficient in handling spatial objects. As a result, many specialised indexing and clustering techniques have been proposed for the data access and physical organisation of spatial data.

The spatial indexing has been discussed in the literature as a method supporting the processing of spatial joins and similarity queries on spatial objects. The spatial joins (definition of a join is based on topological relations) and similarity queries, such as nearest-neighbour query, are database primitives inherent to SA. Consequently, the application of spatial indexes – facilitating efficient spatial data management and the processing of spatial joins and similarity queries – can positively influence the overall performance of SA [Rot91, BKS93, HKS97, GG98, Vit01]. Because of its efficiency, the spatial index which is most commonly used in the evaluation of the spatial queries is the R-tree [BKS93]. Moreover, many specialised data access methods used for spatial data management and SA are, in fact, the modifications of the conventional R-tree, e.g. spatial join indexes [Rot91], bitmap R-trees [AT00] and the R-tree-like data structure for the physical organisation of spatial aggregates [PKZT01].

The spatial indexes attempt to store neighbouring spatial objects on a common disk page. The clustering strategy (see Section 3.1.2) improves query response time by reducing the number of physical I/O transfers in spatial queries which typically retrieve neighbouring objects (see Section 1.2) [BK94, GG98, Vit01]. For large spatial objects, however, the spatial indexes organize only object approximations in the form of a minimum bounding rectangle (MBR). The simplified representation is often exploited in the processing of complex spatial queries in the filter-and-refine manner [BKSS94]. The two-phase approach begins with the retrieval of spatial objects whose MBRs meet query criteria. This is followed by the final verification of the objects' properties with respect to the query restrictions. Since the first phase of the query processing strategy is the major performance

bottleneck, the application of clustering is necessary to minimize I/O cost. In order to benefit from the clustering, the proximity-preserving data organisation has been proposed for spatial access methods (the Hilbert R-tree [KF94]), processing of spatial queries [XZJ01], physical organisation of spatial data [BK94], spatial data mining [EFKS00], multidimensional data partitioning (declustering) for parallel and distributed data processing [FB93].

1.4.3 Conclusions

The implementation of SA algorithms on the top of a database provides many benefits, e.g. simplified implementation process, short development time, performance improvements etc. However, to make the SA algorithms efficient in data rich environments, a tight integration with a database system and the intelligent use of database primitives is required. The previous Sections of the Chapter surveyed a number of the state-of-the-art techniques integrating SA within the database technology. However, the detailed analysis of the subject reveals that – despite the technological progress and significant research effort – many performance-related challenges existing in SA have attracted so far only little attention of the scientific community. This includes the representation of complex spatial hierarchies, physical data organisation with respect to the spatial hierarchies, integrated processing of spatial and analytical data. Another conclusion that can be drawn from the study of the related work regards the recognition of the fact that most of the current database enhancements for SA have been proposed independently to address specific performance problems. Consequently, an aspect that seems to be particularly neglected concerns the cross-integration of database primitives for spatial and analytical data. Although both kinds of data have much in common (e.g. hierarchical character, multidimensionality) many partial approaches – rather than one general solution – have been proposed independently. The major difficulty resulting from the diversity is long-lasting and complex development necessary to implement a complete system for SA. This combined with the lack of one unified approach – leading to the weak integration of SA with the database technology – along with the unsatisfactory support of the existing techniques in terms of data modelling and processing have been the crucial insights necessary to formulate the objectives of the thesis.

1.5 Objective

The objective of the thesis is to develop the data model which efficiently supports spatial analysis (SA). Given the scope and character of SA (see Section 1.2), the major performance bottlenecks typical for the discipline (see Section 1.3) and the deficiencies of the existing performance-improvement approaches (see Section 1.4), the properties of the data model for SA which enable meeting the objective are defined as follows:

I. The data model should include both spatial and analytical data,

II. The data model should support various representations of spatiality, i.e. qualitative description, coordinates (location), geometries (shape),

III. The data model should represent the multidimensional character of SA,

IV. The data model should represent the hierarchical character of SA,

V. The data model should treat symmetrically all properties of the analysed phenomena, e.g. temporal and spatial data for movement analysis,

VI. The data model should be flexible to organise data on the physical level with respect to arbitrary semantics, e.g. spatial autocorrelation and heterogeneity,

VII. The data model should support the locality-preserving data organisation on the physical level,

VIII. The data model should support data partitioning,

IX. The data model should provide means to integrate other performance-related enhancements for SA.

In order to demonstrate the potential existing in the proposed data model for SA, the objectives of the thesis refer to the issues of optimising data access and query patterns typical for SA. The query patterns to which the thesis refers include:

X. Spatial queries, e.g. range queries, nearest neighbour queries, spatial joins,

XI. Aggregations at different levels of a spatial hierarchy,

XII. Multidimensional analytical queries, i.e. slice and dice, drill-down and roll-up.

1.6 Outline of the thesis

The thesis is organised into Chapters which address specific aspects of the author's doctoral work. The content of the Chapters is given below:

Chapter I introduces the subject of spatial analysis, focusing on the features of the discipline which are essential for the thesis (i.e. locality, multidimensionality, spatial hierarchies). After the introduction, the motivations for the thesis are presented, including the one resulting from the author's experience. This is followed by the overview of the related work which allows to benchmark contributions of the thesis to research in the area against the currently available techniques. The Chapter is concluded with the formulation of the objectives of the thesis.

Chapter II presents the author's analysis of space-filling curves in the scope relevant for the thesis. Since the proposed data model for SA exploits the curves for the proximity-preserving data organisation (clustering), the clustering properties of the curves and the compound (naive) curve are assessed and benchmarked. The efficiency of mapping algorithms of the space-filling curves is also analysed in the Chapter in order to identify a curve that most adequately meets the requirements of the data model for SA.

Chapter III presents the author's argumentation regarding the selection of data organisation and data access methods for the data model for SA. The Chapter describes the properties of the methods in the scope relevant for the thesis (i.e. multidimensional indexing and clustering), and discusses the principles of the indexes (the UB-tree and MHC/HI) which were adapted by the author for the proposed data model for SA.

Chapter IV describes the main contribution of the thesis which is the data model efficiently supporting SA. Although the Chapter focuses on the physical layer of the data model, the conceptual abstraction, especially for complex spatial hierarchies, is also discussed. Subsequently, the Chapter describes the relational representation of the data model using the data warehousing terminology. A dedicated physical schema, derived from the logical model for the needs of SA, is presented in the final part of the Chapter.

Chapter V refers to the issues of optimising query processing for SA, focusing mainly on those which concern multidimensional hierarchical clustering provided by the data model

for SA. In order to verify the adequacy of the proposed data model, the results of performance analysis are presented and discussed in the Chapter.

Chapter VI summarises the thesis with conclusions and overviews the future work.

Annex I describes the architecture of the system developed during the author's doctoral studies to verify the adequacy of the data model for SA. Since a single DBMS completely supporting all elements of the proposed data model does not exist, the development of the system was based on a federated architecture consisting of Transbase RDBMS and Smallworld GIS.

Annex II describes the schemata of an exemplary data model that was implemented in the system presented in Annex I and used for the performance analysis presented in Chapter V.

Chapter II

Space-filling curves

2.1 Introduction

Many computer applications demonstrate some degree of locality of memory references [Den80]. The pattern of memory references occurs when data of similar properties is accessed collectively and repeatedly before an application proceeds to other datasets. The locality of memory references is characteristic of spatial analysis (SA) because data processing associated with SA is typically local (see Section 1.1). Detecting the property in data processing or data analysis enables to improve the performance of workloads, especially when physical data organisation reflects semantical distance relations existing in the data. In other words, the proximity-preserving data organisation which complies with the locality of memory references inherent to data processing or data analysis leads to significant performance gains, e.g. by reducing a number of random I/O operations (see Section 3.1.2) [Vit01].

The following Sections of the Chapter describe the author's analysis of space-filling curves (SFC) which are commonly used for the proximity-preserving organisation of multidimensional data. The analysis focuses only on the properties of the curves which are essential for the needs of the thesis. This includes analysing the complexity of algorithms used for mapping multidimensional space onto a line segment as well as benchmarking the proximity-preserving properties of SFCs against the conventional data organisation ubiquitous in modern database systems. The conclusions summarising the Chapter provide the rationale necessary for selecting SFC that serves as a foundation of the physical data organisation in the proposed data model for SA (see Chapter IV). The development of the data model for SA is the main objectives of the thesis.

2.2 Terminology and basic concepts

The amount of research activity, intensified by the resent interest in fractals, indicates that the subject of space-filling curves (SFC) is still popular in the scientific literature. The most comprehensive and up-to-date mathematical treatment of the subject known to the author was attempted by H. Sagan [Sag94]. The terminology, formal introduction to the subject as well as general construction procedure applicable to all SFCs described in the thesis are based on the publication.

Definition 2.1 Let f be a function defined on a set $\mathcal{A}$ and taking values in a set $\mathcal{B}$. Then f is an injection (injective map) if, whenever $f(x) = f(y)$, it must be the case that $x = y$.

If the function is a linear operator which assigns a unique map to each value in vector space then the function is an injection. Specifically, given vector space V with $\mathbf{X}, \mathbf{Y} \in \mathrm{V}$, then a transformation T defined on V is an injection if $T(\mathbf{X}) \neq T(\mathbf{Y})$ for all $\mathbf{X} \neq \mathbf{Y}$.

Definition 2.2 Let f be a function defined on a set $\mathcal{A}$ and taking values in a set $\mathcal{B}$. Then f is an surjection (surjective map) if, whenever for any $b \in \mathcal{B}$, there exist an $a \in \mathcal{A}$ for which $b = f(a)$.

Let the function be an operator which maps points in the domain to every point in the range and let V be vector space with $\mathbf{X}, \mathbf{Y} \in \mathrm{V}$. Then a transformation T defined on V is a surjection if there is an $\mathbf{X} \in \mathrm{V}$ such that $T(\mathbf{X}) = \mathbf{Y}$ for all $\mathbf{Y}$.

Definition 2.3 A map is called bijective if it is both injective and surjective. A bijective map is also called a bijection. A function f admits an inverse f^{-1} if it is bijective.

Definition 2.4 If f is a function from subset of E^m into E^n, then

$$f_*(\mathcal{A}) = \{ f(x) \in \mathcal{R}(f) \mid x \in \mathcal{A} \cap \mathcal{D}(f) \}$$

where $\mathcal{A} \subseteq \mathrm{E}^m$, is called the direct image of $\mathcal{A}$ under f. $\mathcal{D}(f)$ denotes the domain, and $\mathcal{R}(f)$ the range of the function f.

Definition 2.5 If $f: I \to \mathrm{E}^n$ is continuous, then the image $f^*(I)$ is called a curve. The beginning is called $f(0)$ point of the curve and $f(1)$ is called its endpoint.

Definition 2.6 If $f: I \to \mathrm{E}^n$, $n > 2$, is continuous and passes through every point of n-dimensional region with positive Jordan content $J_n(f^*(I)) > 0$, then $f^*(I)$ is called a space-filling curve.

2.3 Overview of space-filling curves

In 1890 Giuseppe Peano and immediately after that in 1891, David Hilbert discussed curves which 'fill' a plane, i.e. given some patch of the plane, there is a curve which meets every point in that patch [Sag94]. A construction process of such a curve begins with a single line segment, the initiator, and then the segment is substituted by the generator curve. The process is continued ad infinitum and every iteration makes the generator curve fit better into a square. Figure 2.1 shows the first steps of the iterative construction of the Peano's original curve.

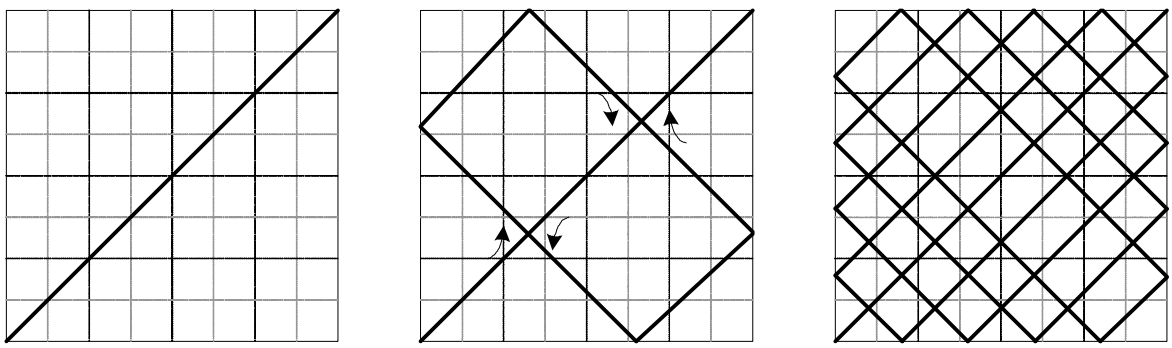


Fig. 2.1 The Peano's original curve.

Since the invention of the first SFC many other bijective mappings from a unit interval to a square and to a hyper-rectangle in general have been proposed, e.g. the Hilbert curve, Sierpiński curve, Lebesgue curve, Z-curve. All of the mappings share some properties, among which the most important are space-filling and distance-preserving[1] functions,

[1] Distance-preserving property is often referred to as clustering.

which are further discussed in following Sections of the Chapter. Because of its significance for the thesis, the main focus of the author's analysis of the properties concerns the Z-curve. The other mapping alternatives (such as the Hilbert curve) are also considered mainly as references for the comparison of clustering properties. Motivation for the comparative analysis results from the fact that SFCs cluster data differently which impacts data organisation and mapping as well as algorithms for query processing. The choice of a curve based on the analysis of the space-filling and distance preserving properties and is thus a crucial decision required to meet the objectives of the thesis.

2.3.1 Universal construction procedure

Although Peano discovered the first SFC, it was Hilbert who recognised a general geometrical procedure that allows a construction of an entire class of SFCs [Sag94]. The concept consisted in the following heuristic principle. If an interval I can be mapped continuously on a square Q, then after partitioning I into four congruent subintervals and the square into four congruent subsquares, each subinterval can be mapped continuously onto one of the subsquares. Afterwards, each subinterval is partitioned into four congruent subintervals and each subsquare is partitioned into congruent subsquers, and the mapping is repeated. The interval I and the square Q are partitioned after n iterations into 2^{2n} congruent replicas.

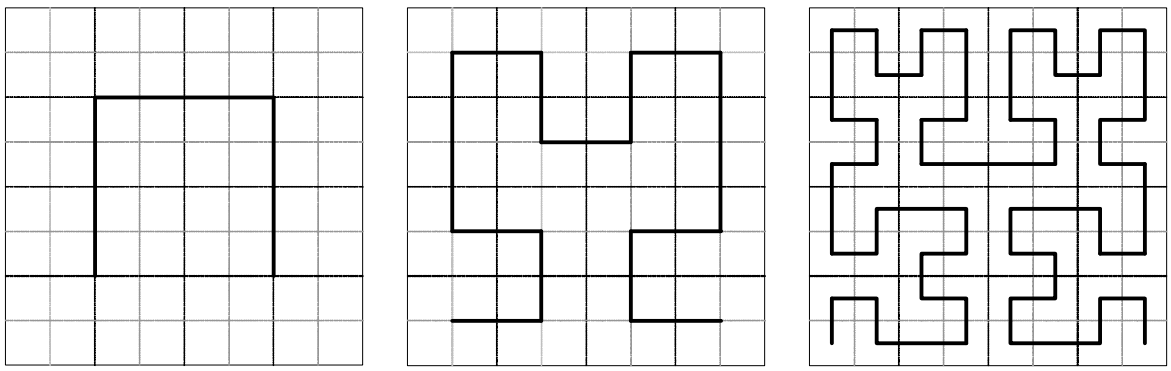
Definition 2.7 A number of steps, or iterations, for which the construction process is carried out is called an order of a curve, thus every iteration of the construction process increases the order of the curve by one.

Definition 2.8 An SFC of some finite order is an approximation of an SFC. The curve does not pass through every point in space but through all of the centre points of a finite number of equal-sized subsquares, a union of which comprises whole of the unit square.

Given the above assumptions a general algorithmical procedure for construction of a finite-order approximation of an SFC can be represented on an example of the Hilbert space-filling curve.

Step 1 Both the one-dimensional interval I and the square Q are initially divided into 4 congruent quarters. Each subinterval is then mapped to a different subsquare in such a way that subsquares mapped to from the adjacent subintervals share a common edge. The subsquares are thus ordered. This is expressed graphically (see Figure 2.2a) by drawing a line made up of straight segments and passing through centre points of subsquares in the sequence in which the subsquares are mapped to successive subintervals of the line. The line passing through the centre points of the subsquares is referred to as the first-order Hilbert curve.

Step 2 The process of division of intervals and squares is repeated for each of the four pairs of subintervals and subsquares produced in the first step. This results in four groups of four equal-sized subintervals and subsquares. Since the subsquares from which the groups are derived are ordered, the groups themselves are also ordered. Within each group, a mapping is established between subintervals and subsquares and a first-order curve is drawn in a similar manner to that described in the first step. The particular order in which subsquares are mapped to subintervals within a group is chosen such that the last subsquare shares a common edge with the first subsquare of the successor group. This results in a first-order curve within the group having a different orientation or being a reflection of the first-order curve drawn in the first step (see Figure 2.2b).



(a) (b) (c)

Fig. 2.2 The first- (a), second- (b) and third- (c) order Hilbert curves.

The above procedure transforms a first-order curve into a second-order curve and higher-order approximations are obtained in the following manner. A curve of order n, where $n>1$, is conceptually constructed by replacing each point on a curve of order n-1 size by a

scaled-down curve of order 1. These first-order curves are automatically ordered according to their corresponding points on the curve of order *n*-1. The curves are then suitably rotated or reflected so that the curves can be connected to each other in such a way that the last point on one curve is adjacent to the first point of its successor. The distance between any such pair of points is the same as the distance between any other pair of points on the resulting curve of order *n*. The process is equivalent to replacing every point on a curve of order 1 by a curve of order *n*-1. Figure 2.2c shows the third-order curves generated in this manner from the second-order curve.

For the original SFC the construction process is carried ad infinitum so that continuous surjective mapping from an interval I to a square Q is obtained. E. Netto demonstrated that a bijective mapping from an interval I to a square Q is necessarily discontinuous [Sag94]. However, if the condition of continuity is dropped a bijective mapping, i.e. one onto one correspondence among points on an interval I and a square Q, is possible. For many practical applications some finite-order mappings are often considered. The mappings are not SFC in strict mathematical definition (see Definition 2.6) but the term is still widely used in the literature even for the finite order mappings. The finite-order SFCs are bijective, i.e. the curves map points from an interval to a square or hyper-rectangle in one onto one correspondence.

2.3.2 Compound curve

A straightforward approach to the problem of filling a square with a curve is to draw a vertical line from one side of the square to the other alongside values of a first attribute. Once a limit value of the first attribute is reached the construction of the curve is continued in the similar manner for subsequent values of the second attribute until the limit value of all attributes is reached. In the first iteration of the process the curve intersects only two outermost values of the second attribute. In next iteration the procedure is continued with doubled resolution of the curve thus the next curve intersects twice as many points at half the distance as the curve in the previous iteration. Figure 2.3 shows the first tree stages of the construction process of the so-called compound curve.

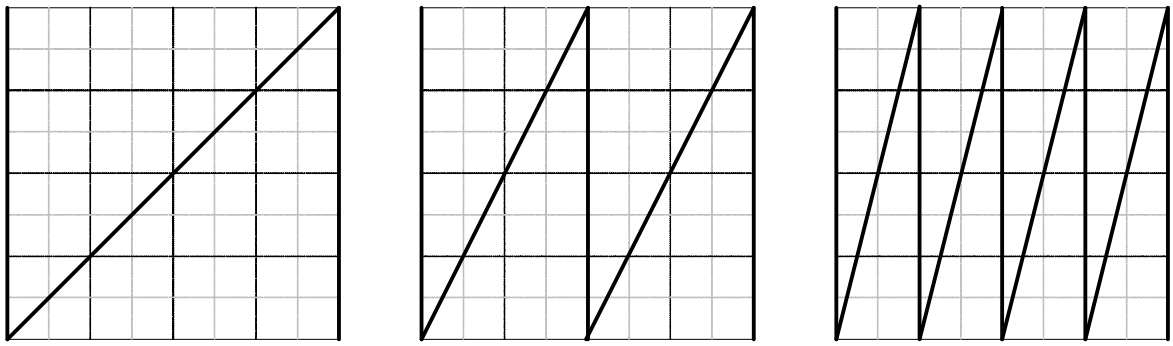


Fig. 2.3 The construction process of the compound curve.

The above construction process creates a curve but not an SFC in a sense of the strict mathematical definition (see definition 2.6). A typical SFC has infinite length, self-similarities and matches every point in the square. By contrast, the above construction process does not lead to a curve, although a curve is created at every stage of the process. The statement may seem contra-intuitive especially since for any given nonnegative resolution one can find a stage at which the generated curve passes through every point in the square with whatever-required small distance. Moreover, the curve may even seem a self-similar SFC because in each stage of the construction process the curve is composed of two curves of the previous stage scaled in the horizontal direction. The incorrectness of the above reasoning is proved by the theorem I.

Property 2.1 Let Q be a closed unit square, $x \in I^1$ and $y \in I^2$ unit intervals which are adjacent edges of the square and let $y^n(x)$ be the n-order image of the naive curve. The curve is not an SFC because

$$\bigwedge_{\alpha \in I^1} \lim_{n \to \infty} y^n(\alpha) \to \beta$$

does not exist for every α .

Given a $x \in I^1$ an increase in corresponding values $y^n(x)$ can be observed with every stage n . If the construction really leads to a well defined limit curve, then one must expect that the sequence of $y^1(x)$, $y^2(x)$, … also converges, namely to the y value of the limit curve at position x . The proposition is true only, however, for points x which allow a finite binary representation (such as $^1/_4$ or $^{139}/_{256}$) because the construction at such points have a y value of 0 provided the stage is large enough. The other points violate the crucial

convergence property, e.g. at $x = {}^{1}/_{7}$ the y values of the curve are ${}^{2}/_{7}$, ${}^{4}/_{7}$, ${}^{6}/_{7}$, ${}^{2}/_{7}$, ${}^{4}/_{7}$, ${}^{6}/_{7}$ ... and so on in a periodic fashion. Therefore there is no limit object for the point x, no self-similarity and consequently no SFC.

2.3.3 Hilbert curve

The construction process of a genuine SFC described in Section 2.3.1 is consistent with concepts presented in the original Hilbert paper [Sag94]. The initial mapping, i.e. ordering of subsquares, produced in step 1 of the process and illustrated in Figure 2.2a was arbitrarily selected from a finite number of available mapping alternatives. Different choices made in the step result in different but topologically equivalent orientations of the curve. Figure 2.4 shows mapping alternatives for a second-order SFC proposed by Hilbert.

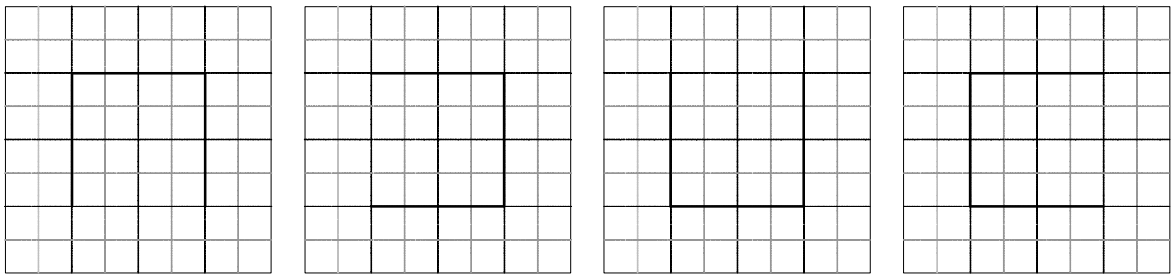
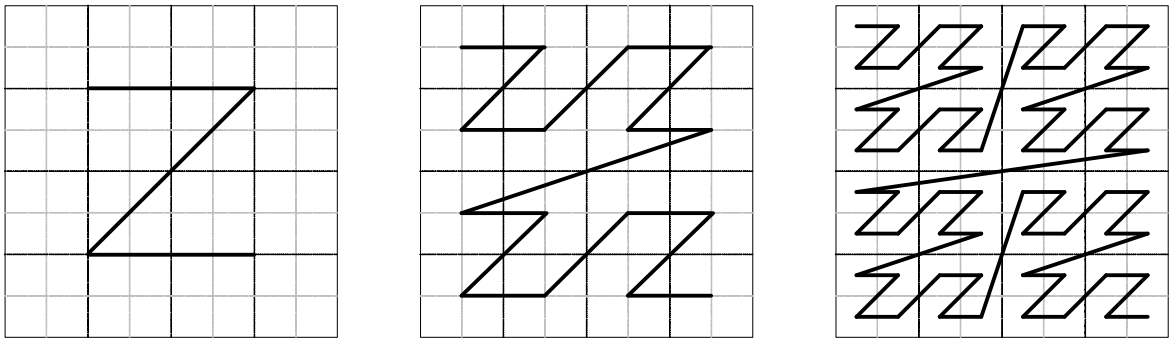


Fig. 2.4 The mapping alternatives for the first-order Hilbert curve.

As depicted in Figure 2.2(a) the second-order Hilbert curve comprises 4 ordered and connected first-order curves. The first and last sub-curves have different orientations to the curve produced in the first step while the middle two have the same orientation. Furthermore, Figure 2.2b shows that each of the middle two first-order curves within the second-order curve is direct transformation of a first-order curve of the same form and orientation. The regularity is also true for higher-order approximations of the Hilbert curve, e.g. the second-order curve (Figure 2.2b) within the third-order curve (Figure 2.2c). The above observation confirms self-similarity of the Hilbert curve, which provides an insight into how to express algorithmically such curves of some arbitrary order (see Section 2.4.2).

2.3.4 Z-curve

The construction of the Z-curve is performed in accordance with the procedure describes in Section 2.3.1. The mapping that defines the first-order curve, i.e. the generator curve, resembles letter Z, hence the name of the curve. A mathematical foundation for the curve is attributed to Lesbegue and the reinvention of the curve for modern computer science applications was done by Morton [Sag94]. Examples of the first three orders of the Z-curve in two dimensions are shown in Figure 2.5, which clearly illustrates self-similarity as the curve transforms from one order to the next.



(a) (b) (c)

Fig. 2.5 The first (a), second (b) and third (c) order Z-curves.

Unlike the Hilbert curve the Z-curve, though also passing through every point in space, is discontinuous, i.e. some points which are consecutive in their ordering are not adjacent in the original space from which the ordering was derived. The characteristic property of the Z-curve holds regardless of the number of dimensions in a space. Some of the discontinuities arise from the fact that the sub-squares within every square are always given the same orientation or ordering. Although the Z-curve is not an SFC in the limit according to the strict mathematical definition (see Definition 2.6), the curve at any finite stage of the construction demonstrates some clustering properties. The properties along with the ability to create bijective correspondence between a line segment and multidimensional space are fundamental for attaining the objectives of the thesis.

2.3.5 Space-filling curves in multidimensional space

The concept of SFC can be extended into higher-dimensional space. For three-dimensional space an interval I is divided in each step into 2^{3n} congruent subintervals and accordingly, the unit cube $\mathcal{W}$ into 2^{3n} congruent subcubes. The midpoints of the subcubes are lined up in such a way that adjacent subintervals are mapped onto adjacent subcubes with a common face and each mapping preserving the preceding one. Figure 2.7 represents the second-order tree-dimensional Z-curve obtained in accordance with the above algorithm.

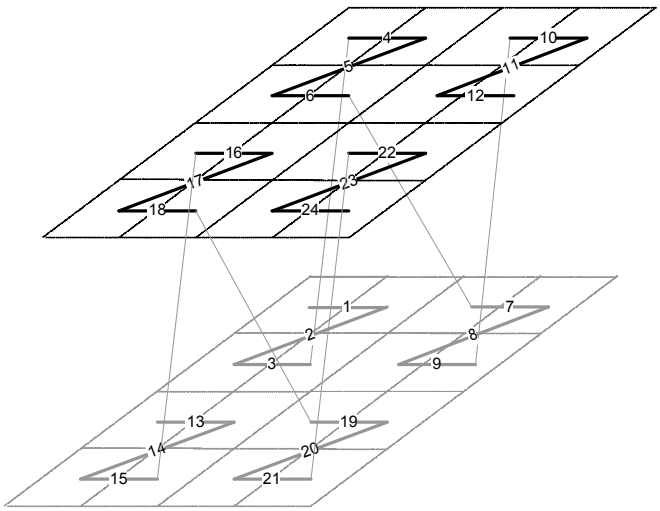


Fig. 2.7 The second-order tree-dimensional Z-curve.

The Hilbert curves in higher-dimensional space differ importantly from the two-dimensional curve as there are possible transformations of a first-order curve into a second-order curve. It is also possible to draw valid first-order curves which cannot be transformed into valid second-order curves, whereby the adjacency property of successive points is maintained [Sag94]. The existence of the alternatives increases the complexity of algorithms for mapping multidimensional space onto the Hilbert curve. The complexity of the algorithms is discussed in the following Section of the Chapter.

2.4 Mapping algorithms for space-filling curves

Many practical applications of SFCs are based on the binary representation of the curves. The binary representation is obtained by an ordering of sub-intervals and corresponding

sub-squares according to the procedure of a given space-filling mapping. An example of the binary ordering for the Hilbert space-filling curve is represented on Figure 2.8.

For a first-order SFC division of a square into four sub-squares results from the division of the intervals into four sub-intervals. These sub-intervals are placed in sequence and so that each one can be represented by a two digit integer number, i.e. by 00, 01, 10 and 11. These sequence numbers can now be used to assign coordinates to the sub-squares. The numeric identification of sub-squares and sub-intervals is shown in Figure 2.8a.

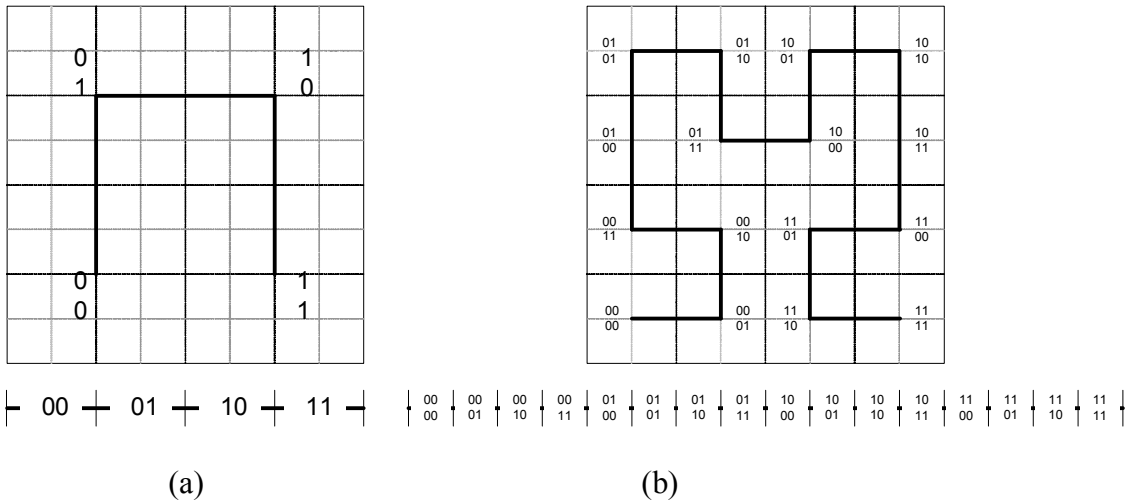


Fig. 2.8 The Binary representation of correspondence between points on an interval and the (a) fist- and (b) second-order Hilbert curve.

In the second step of the construction process, the number of sub-intervals and sub-squares increases four-fold, and thus additional two binary digits are required to represent the sequence numbers (Figure 2.8b). Since the group of four sub-squares produced in the second step is nested within the sub-square of the first step, the coordinates representing the latter become prefixes of the coordinates representing the former. For example, the sub-square produced in the first step and represented by coordinates 10 is divided into sub-squares represented by coordinates 10 00, 10 01, 10 10 and 10 11.

Property 2.2 The *k*-order *n*-dimensional space-filling curve can be represented by *k*n* binary digit number.

According to the construction process described in Section 2.3.1 the coordinate of a point on a first-order one-dimensional curve is expressed by a single binary digit. An additional

digit is required to represent distinct coordinate values with each increment in the order of the curve since the number of distinct values increases by a magnitude of 2. As a result, the number of digits required to represent the coordinate values of a point on a curve of order k is k. In n dimensions a first-order SFC passes through 2^n hyper-cubes. The construction process incrementing an order of the curve increases a granularity of the coordinate space by a magnitude of 2^n. Consequently, the number of points on a curve of order k is given by 2^{kn} so that the maximum one-dimensional sequence number requires $k*n$ binary digits to represent a distinct coordinate value.

2.4.1 Mapping algorithm for the Hilbert curve

As a result of the recursive fashion in which SFC is constructed (see Section 2.3.1), a mapping to SFC can be expressed as a tree structure. The tree structure is based on a binary representation of both coordinates and points lying on the curve, i.e. n-bit coordinates concatenated into a single distinct n-bit value (a derived key), as described in Section 2.4.

For the Hilbert space-filling curve[2], the construction process of the tree begins by placing a first-order curve at the root of the tree. Root node comprises the set of ordered pairs: $< 00,00 >$, $< 01,01 >$, $< 10,11 >$ and $< 11,10 >$, where the first value of each pair is the derived-key of a sub-interval in the domain of the mapping and the second value is the bit representation of a sub-square point (see Figure 2.8a). In the transformation to a second-order curve, each of these ordered pairs becomes the parent of a node similar to the root and also comprising a set of ordered pairs so the height of the tree increases to 2. The set of ordered pairs in all the leaves corresponds to the finite set of points through which the curve passes while non-leaf nodes correspond to sub-squares which contain a sub-set of points at the leaf level. The process of growing the tree can be continued for each node at the lowest level as the order of the curve increments. The fan-out of the tree equals the number of points on a first order curve (e.g. 4 for two-dimensional curve), while the height of a tree for a finite order SFC is finite and equals to the order of the curve. Figure 2.9 shows the tree representation of mappings between subintervals and sub-squares for the second-order two-dimensional SFC.

[2] The construction process is also applicable to other space-filling curves.

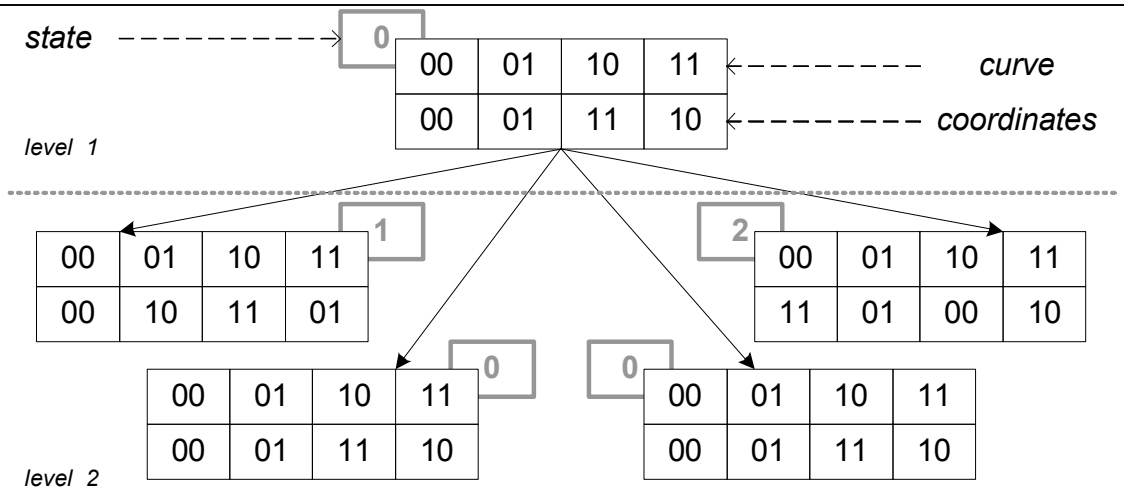


Fig. 2.9 The tree representation of the second-order two-dimensional Hilbert curve.

Figure 2.9 contains two of the child nodes expressing a first order mapping which is the same as that of their parent (root) while the other two express mappings which are different. The fact that some nodes are the same as others is exploited in the modelling a state diagram which is a structure that can be used for the compact representation of the tree. Algorithms for the efficient construction of state diagrams were addressed in many papers. The first attempt known to the author was Bially [Bia67] who described an algorithm for the construction of state diagrams not limited to any specific SFC. However, the technique required the generator table for the state diagram to be constructed manually when the number of dimensions exceeds 2. Additionally, space complexity grows exponentially as the number of dimensions in space increases, both in terms of the number of states in a diagram and in terms of the size of individual states. Butz [But71] overcomes this limitation by describing how to calculate mappings from one-dimensional values to points on the Hilbert curve in any number of dimensions. Lawdel [Law01] makes improvements to the Butz's algorithm by reducing the constant element of computational complexity and extended the algorithm by presenting additional rules which enable the automated generation of state diagrams.

2.4.3 Mapping algorithm for the Z-curve

A mapping for the Z-curve is performed by a bit-interleaving algorithm [Mar99]. In discrete space containing a finite number of points all coordinates of the points are expressed in the binary notation (see Section 2.4.1). In the bit-interleaving process bits

from each coordinate (form least to most significant) are taken interchangeably and concatenated into a single value. For example, coordinates of a two-dimensional point $\mathcal{P}$ lying on the Z-curve of order k, are given as:

$$\mathcal{P} = \{x_1\, x_2\, x_3 \ldots x_k : y_1\, y_2\, y_3 \ldots y_k\}$$

where x_1 and x_k are the most and least significant bits respectively. The $\mathcal{Z}$ value for the $\mathcal{P}$ being the result of bit-interleaving is:

$$\mathcal{Z} = \{x_1\, y_1\, x_2\, y_2\, x_3\, y_3 \ldots x_k\, y_k\}$$

Generally, in n dimensions, the coordinates of $\mathcal{P}$ can be expressed as:

$$\mathcal{P} = \{x^1{}_1\, x^1{}_2\, x^1{}_3 \ldots x^1{}_k : x^2{}_1\, x^2{}_2\, x^2{}_3 \ldots x^2{}_k : \ldots : x^n{}_1\, x^n{}_2\, x^n{}_3 \ldots x^n{}_k\}$$

where each $x^i{}_j$ is a value of j bit of the coordinate in dimension i, where $i \in [1, n]$ and interval $j \in [1, k]$. The $\mathcal{Z}$ value of the $\mathcal{P}$ coordinate can be thus expressed as:

$$\mathcal{Z} = \{x^1{}_1\, x^2{}_1\, x^3{}_1 \ldots x^n{}_1\, x^1{}_2\, x^2{}_2\, x^3{}_2 \ldots x^n{}_2 \ldots x^1{}_k\, x^2{}_k\, x^3{}_k \ldots x^n{}_k\}$$

The $x^1{}_1$ is the most significant bit of the coordinate in dimension i and $x^n{}_k$ is the least significant bit in the coordinate.

The process of bit-interleaving results in a direct relationship between values of bit digits in particular positions within particular coordinates and the values of bit digits in particular positions within derived-keys, and vice-versa. Furthermore, a value of any bit within a derived-key is independent of a value of any higher bits what is not true for the Hilbert curve. Consequently, Z-curve algorithms provide not only an advantage of lower computational complexity but also offer a greater degree of flexibility in implementation.

2.5 Clustering properties of space-filling curves

The ability of SFC to preserve proximity between data points when mapping multidimensional space onto a line segment is the most important property of the curves in terms of the objectives of the thesis. None of the mappings, however, is ideal because the

total ordering of multidimensional space does not exist, i.e. not all adjacent points in the multidimensional space are adjacent on the curve [Sag94]. Although proximity is not fully preserved, the extent to which SFC cluster data justifies the application of the curves for the linearization of the multidimensional space. In order to quantify the clustering properties, the analysis of the property of SFCs is presented in the following Sections of the Chapter. The intention of the analysis is to provide the rationale for selecting the Z-curve as a foundation of the data model for SA (see Chapter IV).

2.5.1 Comparison of clustering properties

Because of its significance, the proximity-preserving property of SFC has been addressed in many academic papers. The examination of the property presented in the papers consists in the theoretical evaluation of some locality measures or the experimental performance analysis of simulated query workloads.

Perez et al. [PKK92] conducted the theoretical evaluation of clustering using an average locality measure and describing the construction of a curve which comes close to minimizing the proposed measure. Mitchison et al. [MD86] investigated similar measures but took into account unit Euclidean distances considering discrete grid frameworks used in similar studies as an abstract lattice graph ignoring underlying geometry. The paper also presented a mapping with suboptimal clustering properties as well as the authors' conjecture that the optimal mapping must define a curve of a "fractal character". Voorhies [Voo91] defined a heuristic measure of locality related to computer graphics and experimentally compared the obtained measures for a variety of SFCs. Gotsman et al. [GL96] addressed a converse question, i.e. to what extent two points, which are close in a traversal order along a curve, are distant in a multi-dimensional Euclidean metric. To quantify the problem the authors estimated the lower and upper bounds of a locality measure and concluded (as all aforementioned papers) that the Hilbert curve demonstrated superior clustering properties. The lower and upper bound estimated in the papers prove, however, that there is still a gap between the theoretically optimal clustering and that of the currently existing curves.

The experimental analysis of clustering properties of SFC was conducted by Faloutsos et al. [FR89]. The experiments consisted in counting contiguous curve sections which pass

through hyper-cubes placed in all possible locations of space containing finite numbers of points. Jagadish [Jag90] performed another comparison by executing partial match and range queries over the Hilbert, Gray-code, Z-curve, and compound curves in two dimensional space. The performance of the curves was measured by counting retrieved data blocks (curve sections) that contained points which did not lie within query regions. Faloutsos et al. [FR91] studied a number of contiguous curve sections retrieved during an execution of range queries over straight line segments mapped to two dimensional curves. Similar simulations were carried out by Kumar [Kum94] for the Hilbert, Gray-code and Z-curve. In the analysis, coordinate space was divided into data blocks and number of blocks intersected by rectangular range queries of different size was counted. The most extensive analysis was performed by Moon et al. [MJFS99], who provided closed-form expressions for a number of contiguous curve Sections retrieved in an execution of range queries of arbitrary shape and size. The analysis was verified by experiments for two and three dimensional the Hilbert curve for queries not confined to hyper-rectangular form. Most recently, Mokbel et al. [MAK02] analysed SFCs by developing formulas for a vector of estimates (jump, contiguity, reverse, forward and still) encapsulating clustering properties of the curves.

2.5.2 Clustering for multidimensional partitioning

The proximity-preserving property of SFCs is a consequence of the recursive fashion in which the curves are constructed. The recursive partitioning of multidimensional space maps points within a particular subsquare closer to each other then to points which are located within any other subsquare (see Section 2.3.1). Moreover, the subsquares themselves are arranged in such a way that adjacent subintervals correspond to adjacent subsquares with one edge in common. Such mapping preserves inclusion relationships, i.e. if a square corresponds to an interval then its subsquares correspond to the subintervals of the interval. Another consequence of the arrangement is the fact that given some part of an SFC (certain percentage of the curve's length) the area covered by the partial curve is always bounded. For example, if one considers a part of the first-order Z-curve, the ending point of the curve is mapped to a certain subsquare. The analysis of the following iterations reveals that the ending point of subsequent orders of the Z-curve is always mapped to the same subsquare. Moreover, the ending points are also mapped to the corresponding

subsquers of the first and following subsquares considered in the proceeding iterations. This indicates that for a certain part of the Z-curve there exists a closed area of the original square completely covered by the curve. The graphical interpretation of the property is represented in Figure 2.10.

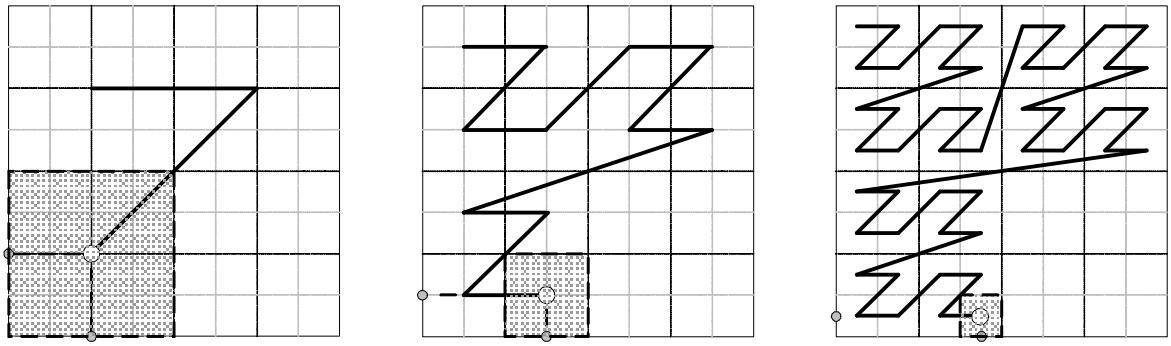


Fig. 2.10 The parameterisation of a square defined by the Z-curve.

The most important consequence of the property in terms of the objectives of the thesis is that given a subset of multidimensional space, there exists an upper bound for the SFC that completely covers the subspace. Consequently, there exists a parameterisation of the square defined by the Z-curve that enables disjoint partitioning of multidimensional space.

2.6 Conclusions

The author's study of SFCs confirms the adequacy of the curves as underlying data organisation in the data model supporting SA (see Chapter IV). The decisive conclusion is derived from the analysis of space-filling (see Section 2.3.1) and proximity-preserving properties of the curves (see Section 2.5). The space-filling property allows to map multidimensional space – defined by domains of attributes involved in SA – onto a line segment that resembles linear data organisation on disk drives (see Section 2.3.1 and 2.3.5). Although the bijective correspondence ensures the complete coverage of multidimensional space, the SFC mappings are practical only for moderately multidimensional spaces. The limitation results from the deterioration of the clustering property observed with increasing dimensionality of original space and overhead generated by the computation of the mappings [Mar99]. The second problem is especially severe for

curves (e.g. the Hilbert, Grey-code curves) whose mapping algorithms are complex and inefficient in practical applications (see Section 2.4.2). On the contrary, the bit-interleaving algorithm operating on the binary representation of coordinates (see Section 2.4.1) has nearly linear complexity and thus can be efficiently used for the mapping of the Z-curve (see Section 2.4.3).

Anther property of SFC which is relevant for the objectives of the thesis is clustering. The property allows mapping points form multidimensional space onto a line segment while preserving proximity relations existing among the points (see Section 2.5). The degree of clustering varies among curves but is considered to be finest for the Hilbert curve (see Section 2.5.1). The Z-curve, though not as good as the Hilbert curve, also demonstrates clustering properties which are superior to that of the ordinary compound curve (see Section 2.3.2). The experimental quantifications of differences between the Hilbert and Z-curve indicate that several percent of data blocks are more likely to be required during the execution of queries over data mapped to the Z-curve as opposed to the Hilbert curve (see Section 2.5.1).

The above conclusions provide sufficient arguments for the choice of the Z-curve as the most adequate SFC in terms of the objectives of the thesis. The Z-curve balances reasonable clustering with efficient algorithms for the calculations of the mappings. Other curves mentioned in the Chapter, though demonstrating slightly better clustering properties, are inefficient in the computation of the mappings and are consequently impractical in query-intense applications.

Chapter III

The UB-tree and MHC/HI

3.1 Introduction

The problem of effective I/O communication between random access memory and magnetic disks results from a difference in access speeds between the two memory levels [Vit01]. In order to improve the effectiveness of the communication, many specialised data organisation and data access methods have been proposed [Com79, GG98]. Most of the methods, however, are based on a concept of a balanced tree which was originally created for the B-tree index [Com79]. The main advantages of the balanced data structure are the guaranteed logarithmic complexity of basic operations and the efficient processing of one-dimensional range queries. The second property of the B-tree is a consequence of lexicographic ordering, natural for one-dimensional data, which is exploited by the data structure. Since such ordering does not exist for multidimensional data, an application of the conventional B-tree for indexing the data may result in unsatisfactory performance [AE99, MB97]. The author's study of spatial analysis (SA) (see Section 1.1.1) and the thesis related work (see Section 1.4) also confirms the inefficiency of the indexing approach in the analysis of complex spatial phenomena. The data processing patterns inherent to SA (e.g. multidimensional range queries, nearest neighbour queries, spatial joins, spatial aggregation queries etc.) require dedicated data organisation and data access methods to improve query response time. The following Sections of the Chapter present two such methods, namely the UB-tree and MHC, which were adapted and enhanced during the author's doctoral work to achieve the objectives of the thesis. Before the methods are presented, the subjects of multidimensional indexing and clustering are briefly surveyed. The intention of the survey is to explain the rationale for the choice of the UB-tree and MHC as the data model's underlying data organisation and data access methods.

3.1.1 Multidimensional indexing

The main objective of data indexing is to store data compactly while enabling efficient query processing. In order to achieve the objective a strategy for access and organisation of data space or the data itself is required. Specifically, the properties of an index for secondary memory can be defined as follows: (a) good utilisation of data and index pages, (b) high fan-out, (c) fair clustering for all attributes, (d) efficient algorithms with worst case performance guarantee for basic operations, and (e) efficient exact match and range searching [SL91]. Data structures that initially attempted to meet the above requirements for multidimensional data depended on the decomposition of the multidimensional data space using the binary search tree (e.g. quad-tree, k-d-tree, hB-tree) or the multidimensional hashing (e.g. grid-file, EXCELL). There are, however, many weaknesses of the indexes resulting, among others, from an unbalance structure of the indexes for non-uniformly distributed data, potential reorganisation of the index structure in unfavourable circumstances and low storage utilisation [SL91, GG98]. There exist also some alternative methods of multidimensional indexing which are based on balanced data structures similar to the B-tree. However, the methods are able to benefit from the balanced data organisation only if the multidimensional indexes create a disjoint partitioning of underlying data. The disjoint partitioning can be relatively simply created and maintained for point data (point access methods, e.g. the UB-tree [Mar99]) but specialised indexes (spatial access methods, e.g. the R-tree [GG98]) are required to deal with complex data (e.g. spatial objects). The management of complex data by spatial indexes depends mainly on the properties of the indexes and typically consists in one of the following: approximations of spatial objects with minimal, possibly overlapping boundary regions (R-tree, R*-tree), clipping ensuring disjoint data partitioning (R+-tree), transformations of spatial objects into a higher-dimensional point or a set of one-dimensional intervals [GG98]. Because of the complexity of spatial objects, none of the strategies is always superior to the others, which confirms the lack of the worst case performance guarantee comparable to the B-tree. The lack of one dominant indexing strategy implies that the spatial indexes either attempt to balance query performance with maintenance costs or refine some operations in order to efficiently support specific query workloads. The first strategy leads to stable but average performance while the other enables better efficiency at the expense of high maintenance or poor worst case performance.

Besides the compact data representation and efficient support for basic operations, an index should execute similar queries with similar efficiencies regardless of database size, data distribution, query type etc. In the case of the analytical processing of multidimensional data the objective becomes more challenging due to the character of queried data and the complexity of query patterns involved in SA (see Section 1.2). The query patterns which are essential for SA, and thus require special consideration, include: spatial data processing (e.g., nearest neighbour searching, topology queries, spatial joins), multidimensional analytical processing (e.g., multidimensional range searching, aggregations) and combined processing of analytical and spatial data. Since the conventional indexing does not provide satisfactory performance for any of the query patterns, many new indexes have been proposed in recent years to deal with the specifics of multidimensional and spatial data analysis [MB97, GG98, Vit01]. Although most of the indexes are able to handle the aforementioned query patterns, the complexity and diversity of the patterns entailed further specialisation of the data structures[3]. In consequence, the examination of data and query patterns associated with the analytical workloads is a necessary prerequisite to select the most appropriate indexing approach. In order to meet the objectives of the thesis such examination regarding SA was conducted by the author and presented in Section 1.2. The outcomes of the work allowed to select indexes that most adequately serve as the data organisation and data access methods underlying the data model for SA. The following Sections of the Chapter overview the indexes, focusing only on the properties of the data structures which are relevant for the thesis. The adaptation of the indexes to the data model for SA, the methodology for the model design and tuning are thoroughly discussed in the Chapter IV.

3.1.2 Multidimensional data clustering

Many algorithms and data structures are only effective when all relevant data fits in the main memory. Deploying the same algorithms in an eternal memory setting may result in performance deterioration (memory swapping) even if the locality of data references exists

[3] Spatial indexes (e.g., R-tree), metric/distance indexes (e.g., M-tree), indexes for analytical processing (e.g., Bitmap indexes, join indexes).

and data caching at higher memory levels is applied. Since the pattern of memory references is inherent to many data processing domains, e.g. Spatial analysis (see Section 1.1.1), exploiting the property creates an opportunity for the rationalisation of I/O communication which can lead to significant performance improvements [Den80, GG98].

Incorporating locality directly into data management consists in the proximity-preserving data organisation (clustering) on a secondary storage device [Vit01]. The main benefit of the clustering organisation is a reduced number of physical I/Os (improved caching effectiveness) and random access operations (improved pre-fetching effectiveness) which are the most common bottlenecks in large data processing workloads. The improved cache utilisation and continuous data access take advantage of the tuple and page clustering that is typically maintained in data files by clustering indexes (e.g. clustering B-tree) [Mar99]. The tuple clustering requires the proximity-preserving arrangement of tuples within a data page while page clustering requires physical proximity among data pages which contain semantically neighbouring data. The finest query performance, e.g. in terms of query evaluation time, is obtained when both tuple and page clustering is preserved. The dual clustering is, however, difficult to maintain in transactional environments where numerous data manipulation queries are executed. In order to deal with frequent inserts, updates, deletes an occasional data reorganisation or bulk data loading is necessary to restore clustering in the data files. Contrary to one-dimensional data, the proximity-preserving organisation of multidimensional data on the secondary storage is only feasible to limited extent. The property of multidimensional data, resulting form the lack of total ordering inherent to the data, allows only for the approximate data clustering, e.g. by the application of finite-order SFC (see Section 2.3.1). The finite-order SFCs are proximity-preserving functions defining one-to-one correspondence between the original multidimensional data and the one-dimensional data arrangement on disk drives. The clustering properties (see Section 2.5) and efficient mapping algorithms (see Section 2.4.3) motivated the incorporation of one such curve (the Z-curve) into the UB-tree. The adaptation of the clustering index to the data model for SA is one of the main contributions of the thesis. The performance analysis presented in Chapter VI proves that the proximity-preserving organisation of multidimensional data efficiently supports SA in which locality underlies most data analysis and data processing patterns (see Section 1.1).

3.2 UB-tree

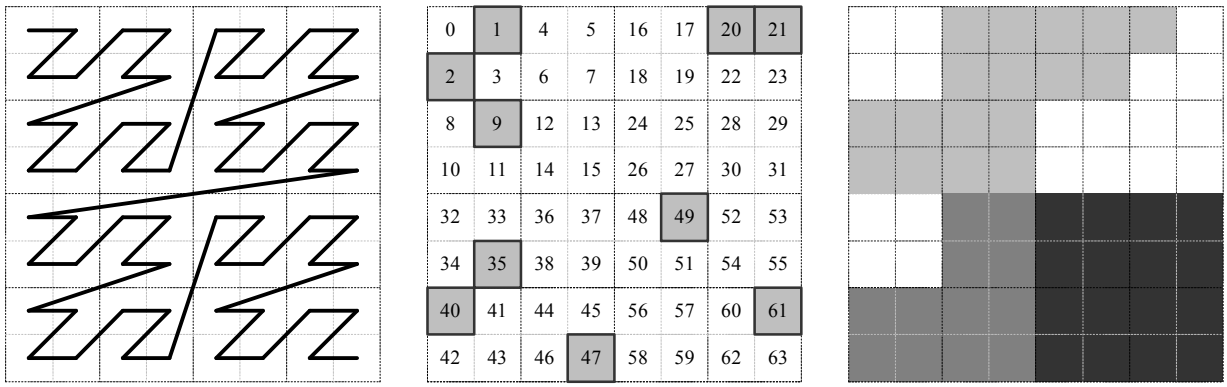
Let **D** be a relation defined over a set of attributes $\mathbf{A} = \{a_0,...,a_m\}$ and containing a set of tuples $\mathbf{T} = \{t_0,...,t_n\}$.

Definition 3.1 A Z-address is a value obtained in the bit-interleaving (see Section 2.4.3) of the binary representations (see Section 2.4) of values in a set of attributes of a tuple.

Definition 3.2 A Z-ordering of multidimensional space is defined by Z-addresses of tuples in the tuple set **T**.

Definition 3.3 A Z-region $[\alpha : \beta]$ is space covered by an interval of the Z-curve and defined by two Z-addresses α and β.

Figure 3.1a shows the third-order two-dimensional Z-curve (see Section 2.3.5) and Figure 3.1b shows the Z-ordering defined by the curve. Figure 3.1c shows a partitioning with five Z-regions $[0 : 3]$, $[4 : 20]$, $[21 : 35]$, $[36 : 47]$ and $[48 : 63]$ which are defined by ten points (Figure 3.1b) assuming the page capacity of two points per page.



(a) (b) (c)

Fig. 3.1 The partitioning of two-dimensional space with Z-regions.

Since the Z-curve is not continuous two neighbouring points on the Z-curve may not be neighbouring points in the multidimensional space, e.g. Z-region $[21 : 35]$ on Figure 3.1c. Volker [Mar99] proves that any Z-region consists of at most two spatially disconnected sets of points.

Definition 3.4 A page is a fixed-size container to store objects or object identifiers in a Z-region.

The UB-Tree [Bay96] exploits the Z-curve to create the proximity-preserving partitioning of multidimensional space. The UB-Tree is based on the B*-Tree [Com79] index to store Z-addresses of a Z-region which is mapped onto one disk page.

Definition 3.5 The UB-Tree is a variant of the B-Tree in which keys are the Z-addresses of Z-regions ordered by the Z-ordering. Each leaf page of the UB-tree holds tuples belonging to a corresponding Z-region while for the secondary UB-Trees only tuple identifiers are stored on a corresponding leaf page.

Since the UB-tree is based on the B*-tree, the multidimensional index inherits logarithmic complexity for point queries. The only overhead specific to the UB-tree is the computation of the Z-address $\xi = Z(t)$ identifying the queried tuple *t*. However, the computation is only performed with precision that suffices to locate the unique Z-region $[\alpha : \beta]$ for which $\alpha < \xi < \beta$. In the next step, the corresponding page $(\alpha : \beta)$ of the Z-region $[\alpha : \beta]$ is fetched from the UB-tree and searched for the tuple *t*. Since the algorithm for the computation of the Z-address has linear complexity, the whole operation is bound by the binary search of the tree structure. Similarly to the point query, an update operation also fetches a relevant page from the UB-tree but additionally requires a modification and writing-back the page to the index. An insert operation follows the same steps as the update but may require splitting a full page. In the case, the full Z-region $[\alpha : \beta]$ is split into two Z-regions by introducing a new Z-address γ for which $\alpha < \gamma < \beta$. The new Z-address γ is chosen so that the first half of the tuples stored on Z-region $[\alpha : \beta]$ is distributed to the Z-region $[\alpha : \gamma]$ and the second half is stored on $[\gamma : \beta]$. Alternatively, when a delete operation results in a page $(\alpha : \beta)$ containing less than half of tuples, the page is merge with following page $(\beta + 1 : \varsigma)$ and the Z-region $[\beta + 1 : \varsigma]$ disappears. The final split of regions and pages is similar to the underflow technique between pages of the B-tree [BC79]. The inherited properties of the underlying B-tree guarantee the worst case storage utilization of 50%. There exists, however, a trade-off between the storage utilisation and the performance of range queries that the index structure may attempt to exploit. In order to reduce a number of intersected query boxes during a range query, the split and merge

algorithms may attempt to maintain the rectangular partitioning of regions at the cost of worse space utilisation [Mar99, RMFZEB00].

The processing of a multidimensional range query over the UB-tree requires only Z-regions which intersect a query box to be fetched from a disk drive [Mar99]. In the initial phase of the range query algorithm the first Z-region that is overlapped by the query box is calculated and retrieved. Subsequently, the next intersecting Z-regions are calculated and retrieved until a minimal cover for the query box has been constructed, i.e. the region that contains the ending point of the query box has been retrieved. Since the complexity of fetching the first Z-region is equivalent to a point query, the calculation of other relevant Z-regions is the most important operation that influences the performance of range queries over the UB-tree (Figure 3.2).

Fig. 3.2 The graphical interpretation of the range query processing [RMFZEB00].

Unlike the standard one-dimensional index built over compound attributes (see Section 2.3), the Z-curve underlying the UB-tree creates the symmetrical partitioning of multidimensional data (see Section 2.5). Since the compound index favours the leading attributes of the structure, the access method is not able to maintain equal performance when some subsets of the attributes are queried. The symmetrical partitioning of the UB-tree guarantees stability regardless of a number of attributes involved in query restrictions. An alternative approach for the indexing of multidimensional space consists in creating multiple secondary indexes. However, this entails high maintenance costs in comparison with the efficient update, insert, and delete operations of the UB-tree. Moreover, the

performance of multiple secondary indexes deteriorates with an increasing number of dimensions whereas the performance of the UB-tree improves. There is also a positive correlation between query evaluation time and a selectivity of the query as processing a range query over the UB-tree requires time proportional to the query size result.

3.3 Multidimensional hierarchical clustering (MHC)

Let **D** be a relation defined over a set of attributes $\mathbf{A} = \{a_0,...,a_m\}$ and containing a set of tuples $\mathbf{T} = \{t_0,...,t_n\}$.

Definition 3.6 Assuming there exist hierarchical relationships between the attributes of the dimension **D**, a hierarchy **H** of a depth $l+1$ over the relation **D** is an ordered set of levels $\mathbf{H} = \{a_0,...,a_l\}$, where $l \leq m$.

Each level j of a hierarchy instance is a set $h_j = \{m_j^0,...,m_j^k\}$. The symbol $\rightarrow$ between the instances of subsequent levels denotes the hierarchical relationship between the instances.

Definition 3.7 The children of an instance m_j^k of a level h_j are all instances m_{j-1}^l of a level h_{j-1}, so that $childern(m_j^k) = \{m_{j-1}^l \in h_{j-1} : m_j^k \rightarrow m_{j-1}^l\}$.

Definition 3.8 The parent of an instance m_j^k of a level h_j is an instance m_{j+1}^l of a level h_{j+1}, so that $parent(m_j^k) = \{m_{j+1}^l \in h_{j+1} : m_{j+1}^l \rightarrow m_j^k\}$.

Definition 3.9 A hierarchy path $P_{\mathbf{H}}$ between an instance m_i^k of a level h_i and a instance m_j^l of a level h_j is sequence $P_{\mathbf{H}} = (m_i^k, m_{i-1}^x, m_{i-2}^y,...,m_j^l)$, where the instance $m_{z+1}^x = parent(m_z^y)$, for $z = i, \ldots, j-1$.

Let ord_m be a bijective function defining a numbering scheme for the children of m, so that ord_m assigns to every child of m a number between 0 and a total number of the children. In order to ensure the uniqueness of every hierarchy path, a unique compound surrogate for each instance of the path can be recursively calculated by concatenating a compound surrogate of the instances's parent with a value of ord_m function for the instance.

Definition 3.10 A compound surrogate $cs(m_i^j)$ for an instance m_i^j of a level h_i of a hierarchy instance having l levels is defined as:

$$cs(m_i^j) = \begin{cases} ord_m(m_i^j) & , i = l \\ cs(m_{i+1}^k) \,|\, ord_m(m_i^j) & , i < l \text{ and } m_{i+1}^k = parent(m_i^j) \end{cases}$$

Example 3.11 Let the *Location* relation contain a set of attributes $\mathbf{A}_{\text{Location}} = \{a_0 = \text{City}, a_1 = \text{Country}, a_2 = \text{Continent}\}$ and let the hierarchical dependencies between the attributes form a hierarchy **H** consisting of levels $h_0 = \text{City}$, $h_1 = \text{Country}$ and $h_2 = \text{Continent}$. Figure 3.3 represents an exemplary instance of the hierarchy **H**.

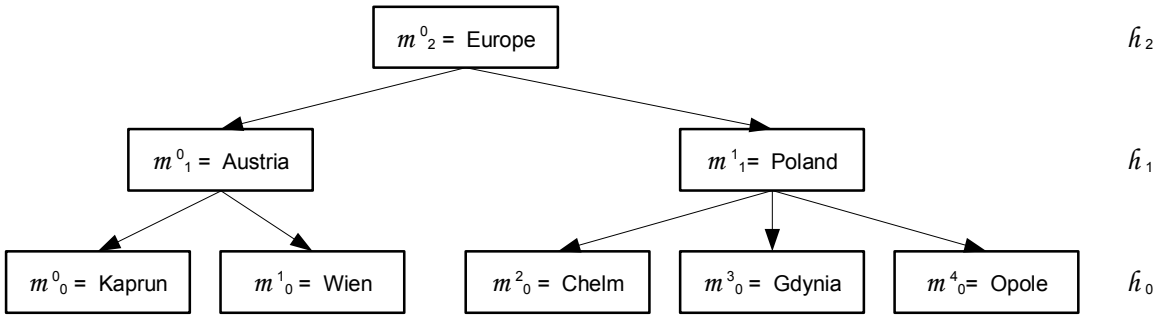


Fig. 3.3 The exemplary instance of three-level *Location* hierarchy.

A value of ord_m function can be obtained for every instance of all three levels of the hierarchy instance.

$$ord_m(m_0^0) = 0\ ,\ ord_m(m_0^1) = 1\ ,\ ord_m(m_0^2) = 0\ ,\ ord_m(m_0^3) = 1\ ,\ ord_m(m_0^4) = 2\ ,$$
$$ord_m(m_1^0) = 0\ ,\ ord_m(m_1^1) = 1\ ,\ ord_m(m_2^0) = 0$$

A compound surrogate for the hierarchy path $P_{\mathbf{H}}$ between the instance m_2^0 of the level h_2 and the instance m_0^4 of the level h_0 is as follows:

$$P_{\mathbf{H}} = (m_2^0, m_1^1, m_0^4) \Rightarrow cs(m_0^4) = cs(m_1^1) \mid ord_m(m_0^4) = cs(m_1^1) \mid 2 =$$
$$cs(m_2^0) \mid ord_m(m_1^1) \mid 2 = ord_m(m_2^0) \mid 1 \mid 4 = 0 \mid 1 \mid 2$$

Although the compound surrogate is created by concatenation, the numerical interpretation of the compound surrogate is possible. This allows to derive a binary representation of the

compound surrogate while still being able to identify the bit-stings of the contributing hierarchy instances. For example, the binary representation of the compound surrogate $cs(m_0^4)$ equals:

$$cs(m_0^4) = 0 \mid 1 \mid 2 = 000_2 \mid 001_2 \mid 010_2$$

The binary representation of a hierarchy path requires bit-strings of a length sufficient to represent the encoding. A bit-string for a level h_i must store a value equal to the maximum number of instances at the hierarchy level which have the same parent. As a result, the length of the bit-string can be computed as:

$$\log_2(\max(card(m_i^j))), \text{ where } m_{i+1} = parent(m_i^j) \text{ and } j = 0, \ldots, card(m_i^j)$$

The minimum length of bit-strings is adequate only for static hierarchies. If a growth rate of a hierarchy is known in advance, an extended length of bit-strings may be determined.

The lexicographic ordering defined by the ord_m function allows to efficiently exploit compound surrogates in queries. A point restriction on the leaf level corresponds to a point restriction on the compound surrogate whereas a restriction on an upper hierarchy level results in range restrictions on the finest granularity of the hierarchy. For instance, the point restriction City = "*Gdynia*" maps to restriction

$$cs(m_0^3 = Gdynia) = 0 \mid 1 \mid 1 = 000_2 \mid 001_2 \mid 001_2 \Rightarrow 000010001_2 = 17_{10}$$

The restriction Country = "*Poland*" maps to a range restriction between values obtained from $cs(m_1^1 = Poland)$ suffixed with the minimum and maximum bit-string values of instances on the lower levels of the hierarchy. The restriction maps to the range between

$$cs(m_1^1 = Poland) \mid 000_2 = 000_2 \mid 001_2 \mid 000_2 \Rightarrow 000001000_2 = 8_{10}$$

and

$$cs(m_1^1 = Poland) \mid 111_2 = 000_2 \mid 001_2 \mid 111_2 \Rightarrow 000001111_2 = 15_{10}.$$

The multidimensional variant (MHC/HI) and adaptation of the hierarchical encoding for the proximity-preserving data organisation in the proposed data model for SA is thoroughly discussed in Section 4.4.

3.4 Conclusions

The main objective of the thesis is the design of the data model efficiently supporting SA. The objective, specified in Section 1.4, is realised by the dedicated physical data model which serves as a basis for the performance improvement of the spatial and analytical processing. Performance gains provided by the data model also result from the application of the UB-tree, MHC/HI, R-tree indexes which were designed specifically for the multidimensional and spatial data processing. The properties of the indexes, characterised in Sections 3.1.1 and 3.1.2, enable the efficient handling of query patterns typical of SA and balancing the costs related to maintenance operations. The properties, which have been identified by the author as essential for the efficient database support for SA, are as follows:

1. ability to handle multidimensional data, spatial (the R-tree) and non-spatial (the UB-tree),
2. multidimensional hierarchical data clustering (the UB-tree and MHC/HI),
3. ability to handle spatial query patterns (the R-tree),
4. balanced structure and disjoint multidimensional data partitioning which guarantees the logarithmic complexity of maintenance operations and exact-match queries (the UB-tree),
5. efficient processing of multidimensional range queries (the UB-tree and the Z-curve).

The proposed data model for SA (see Chapter IV) integrates the indexes into a fully-functional and efficient data organisation and data access layer which is transparent for SA techniques so that the reimplementation of the techniques is not necessary.

Chapter IV

The data model for spatial analysis

4.1 Introduction

The main objective of the thesis concerns the development of the data model efficiently supporting spatial analysis (SA). The author's study of SA (see Section 1.1) reveals some characteristic features of the discipline which must be particularly considered in order to meet the above objective. Three of the features are described below with an intention to explain assumptions which mainly determine the properties of the data model for SA. The features in question are:

1. **Coexistence of spatial and non-spatial data** – SA requires the complete representation of spatial objects and phenomena and unrestricted access to all kinds of data which are involved in the analysis,
2. **Multidimensionality** – SA requires unbiased access to all features of modelled spatial objects and phenomena, e.g. spatiality should not be favoured over time which may be necessary for the temporal or movement analysis,
3. **Hierarchical data organisation and analysis** – SA requires hierarchies in order to support the multidimensional data analysis at the various levels of granularity, e.g. aggregation, generalisation etc.

The above characterisation indicates many similarities between the data-related features of SA and data models exploited in multidimensional databases [Raf03]. The existence of the similarities is not coincidental because cooperation between multidimensional databases and analytical systems is very common [Kim96, BS97, CD97, Raf03]. In particular, spatial

data warehousing and spatial OLAP exemplify the application of the multidimensional models for the spatial analytical processing which is an integral part of SA (see Section 1.1.3). The examples of cooperation and the maturity of the technology provide sufficient rationale for adapting the multidimensional modelling to design the data model for SA. Because of that, the data model for SA presented in the thesis is characterised by many features of the generic multidimensional models but also provides several enhancements designed specifically for SA.

The following Sections of the Chapter describe in detail the data model for SA and a design methodology created especially for the data model. The proposed methodology is consistent with the standard database design approach and includes guidelines for the modelling of conceptual, logical and physical data schemata. However, since the main objective of the thesis is performance-related, the primary focus of the proposed methodology concerns the physical schema which was designed specifically for the objective. In consequence, the conceptual abstraction of the data model is only discussed in the scope which is relevant for objectives of the thesis. Besides the lack of a complete conceptual model, the proposed methodology does not comprise some other design phases which practical implementations may require. Most general database design methodologies, especially the ones for analytical systems [LBMS02, Raf03], cover such aspects of the development process as: analysis of functional and technical requirements, metadata design, interoperability and ETL, security etc. Although that part of the system development is out of the scope of the thesis, the methodology proposed in the Chapter can be easily integrated with the other general methodologies to constitute a complete solution.

4.2 Conceptual data model

The benefits of conceptual data modelling have been repeatedly acknowledged in many publications [EN00, MZ04]. Because of the benefits, a significant research effort has been dedicated to the adaptation of the conventional conceptual models (ER, Object-oriented models) to design specialised databases. In particular, several conceptual models have been proposed for spatial and multidimensional databases in order to reflect the specifics inherent to applications domains which are modelled in the databases [MZ04]. However, a conceptual model for a database supporting SA is not a simple integration of the two but

requires some enhancements of the existing models with features unique for the discipline. Although the development of a complete conceptual model for SA is out of the scope of the thesis, the following Sections of the Chapter outline some features of such a model. The intention of the outline is to indicate the features which are especially relevant for SA and which have implications for designing the logical and physical layers of the proposed data model (see Sections 4.3 and 4.4), i.e. the representation of geography in the data model, complex spatial hierarchies, aggregation semantics.

4.2.1 Representation of geography for spatial analysis

Prior to the development of the data model for SA, a decision regarding the representation of geography in the model must be made. The representation is tightly coupled with SA because SA techniques are not able to extract more information than a model of geographic space captures. Since the limitations of geo-representations can lead to a failure of the model to represent complex geographic process or even bias SA, a selection of geo-representation for SA requires special consideration [MW03]. Despite the importance of the aspect of the conceptual modelling, there are no commonly accepted standards for geo-representation in SA. However, there exists a theory of geographic space proposed by Beguin et al. [BT79] which makes fundamental assumptions necessary to measure geographic phenomena in a manner supporting analysis. The foundation of the theory is a geospatial measurement framework which is defined by a set of locations, a length-metric relation and an area measure. The Euclidean space is one of geo-representations that can support SA in consistence with the above framework [RSV02, MW03]. Models based on the representation are able to support the measurement and inference of quantitative properties and relations while providing an adequate approximation of geographic space.

Besides the reference framework, the representation of geography in conceptual models also concerns modelling the spatial properties of objects and phenomena. The geo-representations of spatial objects consist of descriptive components (ordinary data types, such as integers and strings) and spatial components (complex data types, such as coordinates, geometries and topology). The spatial components are typically expressed on the basis of a reference system (e.g. Euclidian space) either as a zero- (point), one- (line, curve) two- (surface), three- (volume) dimensional object. The geometric types can also be

extended into the collections of homogeneous types called multi-point, multi-line, and multi-surface to create a collection of spatial objects corresponding to a particular topic (theme). Modelling spatial phenomena is typically realised by an application of continuous fields relevant for each point of the space. The geo-representation is usually used to describe the properties of spatial objects (e.g. values collected inside a geometry) or the space itself (e.g. temperature, land use). The data model for SA proposed in the thesis supports both geo-representations using Euclidian space as a reference framework.

4.2.2 Conceptual modelling for spatial analysis

Although a complete conceptual model for SA has not been proposed yet, there exist several conceptual models for spatial data warehousing (SDW). The idea underlying the conceptual models for SDW consists in enhancing the conventional multidimensional data models [Raf03] with features characteristic of spatial data and spatial analytical processing [MZ04]. The existing models, however, are not able to capture the entire specifics of SA and thus provide only partial support for many practical applications. The functional discrepancies become especially evident during the integration of the multidimensional models with complex spatially-oriented semantics inherent to SA. The author's analysis of the discrepancies indicates that features which require special consideration but are not currently available in the existing conceptual models include: (a) consistent formalism necessary to automate mappings from the conceptual abstraction to a logical schema, (b) well-researched semantics of spatiality in the context of analytical processing (e.g. spatial aggregation and hierarchies), (c) logic of operators referencing both spatial and non-spatial data, (d) interoperability of spatial and analytical data, (e) unified metadata models. Section 4.2.3 proposes formalism for the features which are especially relevant for the proposed data model, i.e. semantics of spatial hierarchies and aggregations. The consideration of the features in the design of the conceptual model for SA is meant to facilitate the integration of analytically-oriented spatiality with existing multidimensional models. The semantical analysis of spatial hierarchies provides a foundation for algorithms which enable an integration of the complex spatial structures in the relational model (see Section 4.4).

4.2.3 Conceptual modelling of hierarchies

The semantical analysis of hierarchies is essential for any conceptual model using hierarchies as an organising principle, e.g. the conceptual model for SA. The theoretical background for the hierarchical abstraction can be based on the general hierarchy theory, and in the case of SA, on the adaptation of the theory for the hierarchical spatial reasoning [Car98a]. The paper by Car [Car98a] provides a classification of spatial hierarchies derived from the theory and describes the structures, behaviour and construction methods of functional and structural hierarchies. The proposed data model for SA exploits the structural spatial hierarchies to represent a subdivision of a spatial domain consistently with the requirements of SA (see Sections 1.1.3 and 1.1.4). In the most ordinary variant[4], a structural spatial hierarchy organises spatial objects of the same type into levels. The lowest (leaf) level corresponds most primitive spatial objects which determines the granularity of the hierarchy. The exact semantics of the spatial hierarchy depends on a type of an abstraction process modelled by relationships between the levels of the spatial hierarchy. There are four most common types of the aggregation process:

I. **Classification** is a form of abstraction in which an object type is defined as a set of instances (*instance-of* relationship),

II. **Generalisation** is a form of abstraction in which similar objects are related to the higher-level generic objects (*is-a* relationship),

III. **Association** is a form of abstraction in which a relationship between objects is considered as a higher-level set of objects (*member-of* relationship),

IV. **Aggregation** is a form of abstraction in which a relationship between objects is considered as a higher-level aggregation objects (*part-of* relationship).

Section 4.2.3.1 proposes the formal and implementation-independent definitions of spatial structural hierarchies. Section 4.2.3.2 discusses aggregation semantics in the context of spatiality and spatial hierarchies.

[4] The simple-symmetric hierarchy as described in definition 4.1.

4.2.3.1 Complex hierarchies

For the conceptual description of the domains of interest the basic notions are introduced. Let **E** be a set of entity types and let **R** be a set of binary relationship types, thus a relationship type $R \in \mathbf{R}$ is a pair $R = (E_i, E_j)$. For each entity type $E \in \mathbf{E}$ a set of attributes is defined as $\mathbf{A} = \{a_0,...,a_k\}$. Given the sets of entity types **E** and relationships types **R**, a conceptual model is defined as a directed labelled graph $\mathbf{M} = (\mathbf{E}, \mathbf{R})$, where $\mathbf{R} \subseteq \mathbf{E} \times \mathbf{E}$ is a relation over **E**. In the model **M**, each relationship type $R \in \mathbf{R}$ represents an abstraction process as described in Section 4.2.3, e.g. aggregation, association etc. A relationship type $R \in \mathbf{R}$ is *functional* if it is of the ratio 0:N or 1:N, whereas a relationship type $R \in \mathbf{R}$ is *partial* if it is of the ratio 0:N. The functional relationship which is not partial is called *complete*. A functional relationship $R \in \mathbf{R}$, where $R = (E_i, E_j)$, determines the direction from E_i to E_j. The closure of the relation **R** will be denoted by R^*, i.e.:

$$R^* = \bigcup_{i>0} \mathbf{R}^i$$

If the relation R^* is a linear ordering in **E** it is denoted by the symbol $<_{\mathbf{R}}$ whereas the relation R^* of a partial order is denoted by the symbol $\prec_{\mathbf{R}}$.

Example 4.1 A conceptual model $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ contains a set of entity types:

$$\mathbf{E} = \{Region\ ,\ Country\ ,\ Continent\}$$

and a set of relationship types

$$\mathbf{R} = \{\ (Country\ ,\ Region)\ ,\ (Continent\ ,\ Country)\ \}$$

In this example R^*, denoted further by $<_{\mathbf{R}}$, is a linear ordering relation:

$$Region <_{\mathbf{R}} Country <_{\mathbf{R}} Continent$$

A notion of an instance graph, which is consitent with the conceptual model is introduced below. For each entity type $E \in \mathbf{E}$ there exists a set of instances I_E. For each relationship type $R \in \mathbf{R}$, a set of relationship instances r_R is defined as a relation $r_R \subseteq I_{E_i} \times I_{E_j}$. The instance graph is defined as $DB = (I, r)$, where

$$I = \bigcup_{E \in \mathbf{E}} I_E \qquad\qquad r = \bigcup_{R \in \mathbf{R}} r_R$$

The closure of the relation r will be denoted by r^*, i.e.:

$$r^* = \bigcup_{i>0} r^i$$

If the relation r^* is a linear ordering in I it is denoted by the symbol $<_r$ whereas the relation r^* of a partial order is denoted by the symbol $\prec_r$.

Example 4.2 For the conceptual model **M** described in Example 4.1 the corresponding instance graph $DB = (I, r)$ exists, where

$$I_{REGION} = \{Silesia\ ,\ Mazovia\ ,\ Brandenburg\ ,\ Saxony\}\ ,$$

$$I_{COUNTRY} = \{Poland\ ,\ Germany\}\ ,\ I_{CONTINENT} = \{Europe\}$$

and

$$r = r_{COUNTRY,REGION} \cup r_{CONTINENT,COUNTRY} = \{\ (Poland, Silesia), (Germany, Brandenburg), (Germany, Saxony), (Poland, Mazovia)\ \} \cup \{\ (Europe, Germany), (Europe, Poland)\ \}$$

In this example r^*, denoted further by $\prec_r$, is a partial ordering relation:

$$Silesia \prec_r Poland \prec_r Europe\ ,\ Mazovia \prec_r Poland \prec_r Europe\ ,$$

$$Brandenburg \prec_r Germany \prec_r Europe\ ,\ Bavaria \prec_r Germany \prec_r Europe$$

Property 4.1 If for the instance graph $DB = (I, r)$ the relation r^* is a partial ordering relation, then *DB* is a directed acyclic graph.

Simple-asymmetric hierarchy

Definition 4.1 A conceptual model $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ is called a *simple-asymmetric hierarchy* if R^* is a linear ordering relation and all relationship types $R \in \mathbf{R}$ are functional.

Property 4.2 Every instance graph *DB* consistent with **M** which is a simple-asymmetric hierarchy is a set of trees.

The entity type $E_\top$ is the greatest element of **E** with respect to the linear ordering relation R^* iff $E <_{\mathbf{R}} E_\top$ for each element $E \in \mathbf{E}$. The entity type $E_\perp$ is the least element of **E** with respect to the linear ordering relation R^* iff $E_\perp <_{\mathbf{R}} E$ for each element $E \in \mathbf{E}$. For the model

M the greatest element $E_{\top}$ is the root of the hierarchy whereas the least element $E_{\perp}$ is the leaf of the hierarchy.

For each relationship type $R \in \mathbf{R}$, where $R = (E_i, E_j)$, E_i is a *parent* of E_j and E_j is a *child* of E_i. The *out-degree* (E_i) is a number of relationship types in **R** in which E_i is a parent, e.g. *out-degree* $(E_{\perp}) = 0$. The *in-degree* (E_i) is a number of relationship types in **R** in which E_i is a child, e.g. *in-degree* $(E_{\top}) = 0$. Assuming there exists a relationship type $R \in R^*$, where $R = (E_i, E_j)$, a *path* between E_i and E_j is a sequence $E_j <_{\mathbf{R}} \ldots <_{\mathbf{R}} E_i$. A *branch* is a path between the root $E_{\top}$ and the leaf $E_{\perp}$ entity types.

Property 4.2 results from the linear ordering of entity types in **E**, which explicitly define the root $E_{\top}$ and the leaf $E_{\perp}$ levels in the model **M**, and from the functional relationship types in **R**, which guarantee that there always exists exactly one parent for every entity type $E \in \mathbf{E}$ except the root $E_{\top}$. Consequently, every non-root instance of the instance graph *DB* consistent with **M** has *in-degree* = 1, thus there exists only one path between instances of the instance graph *DB*. In the case *card* $(I_{E_{\top}}) = 1$ there exist only one root instance which implies that the instance graph *DB* is a tree. Since instances are not allowed to have many parents, the instance graph *DB* is a set of disjoint trees when *card* $(I_{E_{\top}}) > 1$.

Since some $E \in \mathbf{E}$ are optional in **M** which is a simple-asymmetric hierarchy, the instance graph *DB* consistent with **M** may happen to be an unbalanced tree. Figure 4.1 represents an example of a conceptual model being a simple-asymmetric hierarchy and its instance.

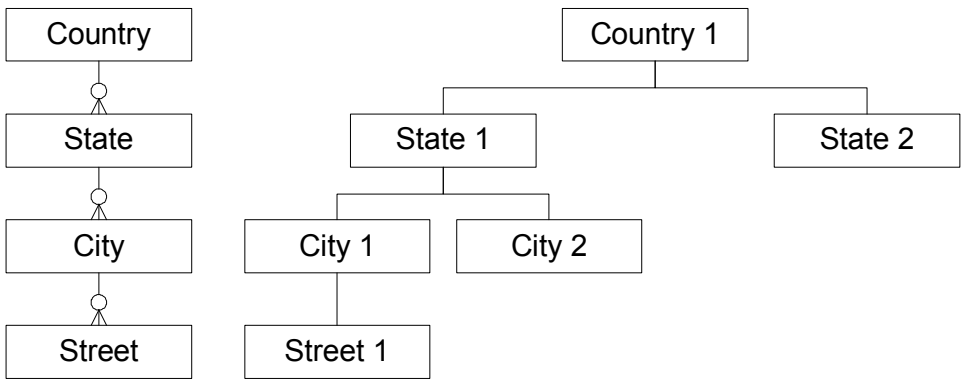


Fig 4.1 The conceptual model and the instance of a simple-asymmetric hierarchy.

Simple-symmetric hierarchy

Definition 4.2 A conceptual model $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ is called a *simple-symmetric hierarchy* if R^* is a linear ordering relation and all relationship types $R \in \mathbf{R}$ are functional complete.

Property 4.3 Every instance graph *DB* consistent with **M** which is a simple-symmetric hierarchy is a set of balanced trees.

Similarly to the simple-asymmetric hierarchy, the instance graph *DB* consistent with **M** is a tree. Since the relationship types in **R** are functional complete, all entity types in **E** are mandatory, thus every branch in the instance graph *DB* is of the same length. This implies that the instance graph *DB* consistent with **M** is a balanced tree. Figure 4.2 represents an example of a conceptual model being a simple-symmetric hierarchy and its instance.

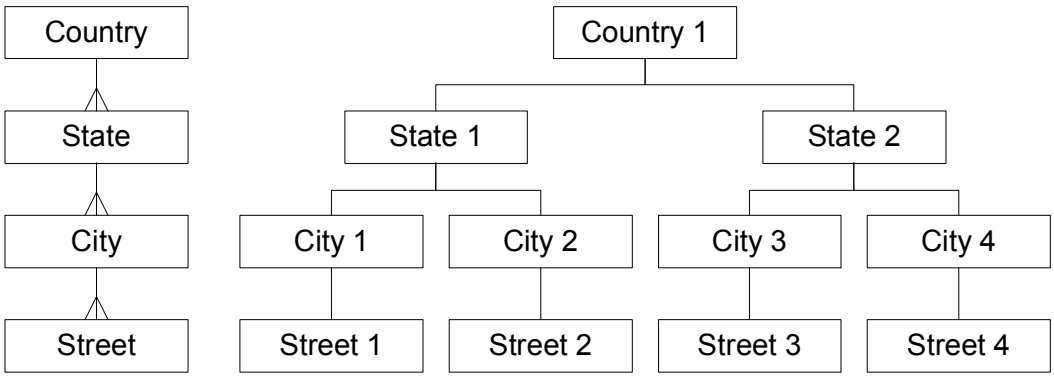


Fig 4.2 The conceptual model and the instance of a simple-symmetric hierarchy.

Union of hierarchies

Definition 4.3 A conceptual model $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ is defined as an order-preserving union of *n* conceptual models of simple hierarchies $\mathbf{M}_i = (\mathbf{E}_i, \mathbf{R}_i)$ iff:

1. for every hierarchy $\mathbf{M}_i$, $\mathbf{E}_i \cap \bigcup_{j \neq i} \mathbf{E}_j \neq \emptyset$,
2. the model $\mathbf{M}_U$ is built of $\mathbf{E}_U = \bigcup_{i \leq n} \mathbf{E}_i$ and $\mathbf{R}_U = \bigcup_{i \leq n} \mathbf{R}_i$,
3. the relation R_U^* being the closure of the relation $\mathbf{R}_U$, i.e.: $R_U^* = \bigcup_{i>0} \mathbf{R}_U^i$.

Multiple hierarchy

Definition 4.4 A conceptual model $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ is called a *multiple hierarchy* iff:

1. the model $\mathbf{M}_U$ is an order-preserving union of hierarchies,
2. all relationship types $R_U \in \mathbf{R}_U$ are functional,
3. there exist the greatest element $E_\top \in \mathbf{E}_U$ and the least element $E_\perp \in \mathbf{E}_U$.

The multiple hierarchy contains a single root $E_\top$ (the greatest element) and a single leaf $E_\perp$ (the least element) entity type. Since some entity types in $\mathbf{E}_U$ may be shared in the multiple hierarchy, e.g. the leaf entity type $E_\perp$, a single instance may be related to many instances belonging to different entity types in $\mathbf{E}_U$. Figure 4.4 represents an example of a conceptual model being a multiple hierarchy and its instance.

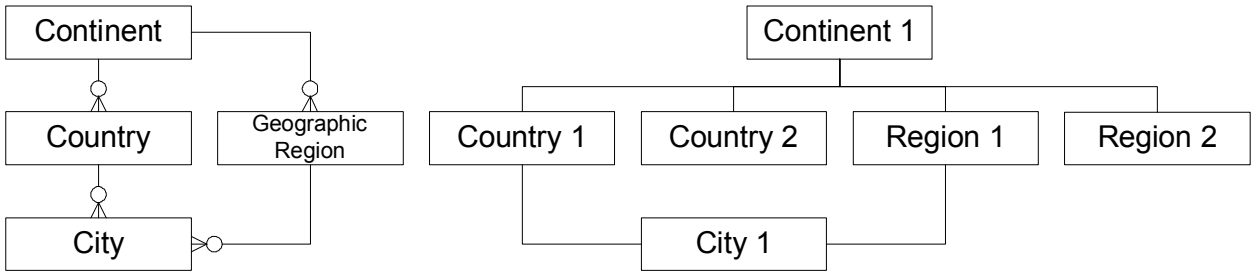


Fig 4.4 The conceptual model and the instance of a multiple hierarchy.

Generalised and non-covering hierarchies

Definition 4.5 A conceptual model $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ is called a *generalised hierarchy* iff:

1. the model $\mathbf{M}_U$ is an order-preserving union of hierarchies,
2. all relationship types $R_U \in \mathbf{R}_U$ are functional,
3. there exist the greatest element $E_\top \in \mathbf{E}_U$ and the least element $E_\perp \in \mathbf{E}_U$,
4. Let $E_i \in \mathbf{E}_U$ be an entity type originating a multiple-exclusive paths and let $\mathbf{S}(E_i)$ be a set of related entity types, where $\mathbf{S}(E_i) = \{E : E \in \mathbf{E}_U, (E_i, E) \in \mathbf{R}_U\}$, so that $card(\mathbf{S}(E_i)) > 1$. Let $\rho(x, E_i, E) = \{(x, y) \in r_{E_i,E} : E \in \mathbf{S}(E_i)\}$, if $\rho(x, E_i, E_j) \neq \emptyset$ then for every $E \in \mathbf{S}(E_i) - \{E_j\}$, $\rho(x, E_i, E) = \emptyset$.

The generalised hierarchy contains a number of multiple-exclusive paths between some entity types of the hierarchy. In the instance graph *DB* consistent with $\mathbf{M}_U$, assuming an instance $x \in I_{E_i}$, where $E_i \in \mathbf{E}_U$ originates an exclusive path, the instance is related to instances belonging to I_{E_j} of only one entity type E_j, where $(E_i, E_j) \in \mathbf{R}_U$, ending the path. A generalised hierarchy consisting of component hierarchies with a different number of levels is called a *non-covering hierarchy*. Figure 4.5 represents an example of a conceptual model being a generalised hierarchy and its instance.

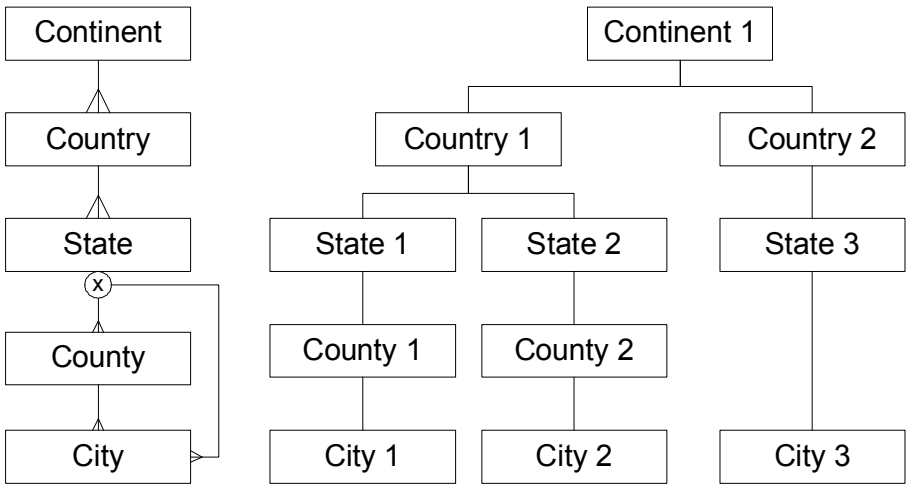


Fig. 4.5 The conceptual model and the instance of a non-covering hierarchy.

Parallel independent hierarchy

Definition 4.6 A conceptual model $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ is called a *parallel independent hierarchy* iff:

1. the model $\mathbf{M}_U$ is an order-preserving union of hierarchies,
2. all relationship types $R_U \in \mathbf{R}_U$ are functional,
3. all entity types of the component hierarchies are disjoint except the least element $E_\perp \in \mathbf{E}_U$, i.e. for every $\mathbf{M}_i$, $\mathbf{E}_i \cap \bigcup_{j \neq i} \mathbf{E}_j = \{\mathbf{E}_\perp\}$.

The parallel-independent hierarchy may consist of simple-symmetric, simple-asymmetric, multiple or generalized hierarchies. These hierarchies share the leaf entity type while all other entity types of the component hierarchies are different. Figure 4.6 represents an example of a conceptual model being a parallel independent hierarchy and its instance.

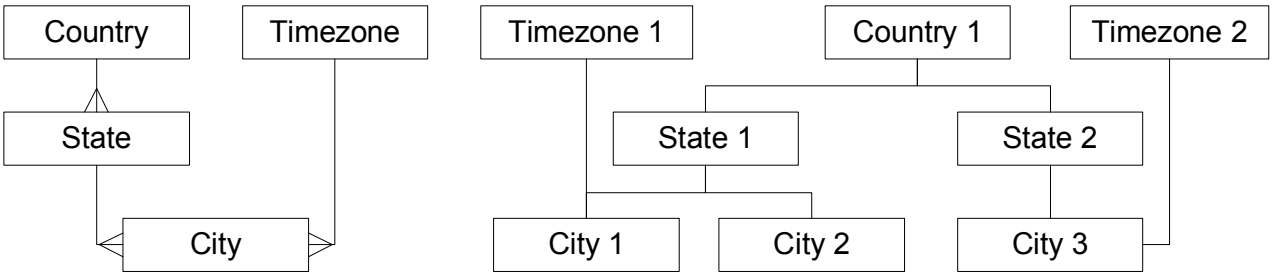


Fig. 4.6 The conceptual model and the instance of a parallel independent hierarchy.

4.2.3.2 Semantics of spatial aggregates

One of the main requirements of the conceptual modelling for SA is the comprehensive support of aggregation semantics. The identification of meaningful aggregates is essential for the completeness of the conceptual model and for the correct computation of query results. The aggregation semantics is typically analysed in the context of measures and hierarchies. In the first case, the analysis of a measure consists in verification if the measure can be meaningfully aggregated with respect to all (fully additive measure) or only some (semi-additive measure) referenced hierarchies. The second aspect of the aggregation semantics concerns verification if individual aggregate results can be directly combined to produce new aggregate results. The property of summarizability is thus strictly related to hierarchies (in particular to the semantics of entity types which constitute the aggregation hierarchy) and to the relationships types between the entity types. Definitions 4.10, 4.11 and 4.12 formalise conditions required by the proposed conceptual models to correctly support aggregations. The definitions are based on the conditions identified by Lorenz and Shoshanie [LS97] as necessary to support summarizability. For the needs of the thesis, the conditions are additionally considered with respect to the specifics of SA, i.e. representation of geography, spatial hierarchies, and spatial measures.

The following analysis of summarizability requires, however, a definition of an aggregation framework for which the analysis is performed. Prior to that, the notions of a data cube and a space of aggregation components are defined in order to provide a structure for the aggregation framework.

Definition 4.7 A data cube is $\mathbf{C} = (\Omega, S, \mathbf{m}, f)$, where

1. $\Omega = \{\mathbf{M}^0, ..., \mathbf{M}^d\}$ is a set of the conceptual models $\mathbf{M}^i = (\mathbf{E}^i, \mathbf{R}^i)$,
2. for each model $\mathbf{M}^i \in \Omega$, there exists a corresponding instance graph (dimension) $DB^i = (I^i, r^i)$,
3. **m** is a set of measures,
4. $S \subseteq I^0 \times ... \times I^d$ is a relation defined over the sets of the instances $I^0, ..., I^d$ of the corresponding dimensions. Additionally, $S_\perp \subseteq I_{E_\perp^0} \times ... \times I_{E_\perp^d}$ is a relation defined over the sets of the leaf-level instances of the corresponding dimensions,
5. leaf-level instances of the dimensions functionally determine measures, so that *f* is a function defined as $f : S_\perp \to \mathbf{m}$.

Definition 4.8 Given a data cube **C**, the space of elementary aggregation components *A* is defined as $A = \{(s, m) : s \in S_\perp, m = f(s)\}$.

Example 4.3 Let **C** be a data cube consisting of three dimensions corresponding to the following conceptual models:

$$Country <_{\mathbf{R}^0} Region <_{\mathbf{R}^0} Continent\ ,\ Season <_{\mathbf{R}^1} Year\ ,\ Bird$$

Assuming there exist the sets of the leaf-level instances of the dimensions

$$I_{Country} = \{Poland, Slovakia\}\ ,\ I_{Season} = \{Winter, Spring\}\ ,\ I_{Bird} = \{Eagle, Falcon\}$$

and **m** is a set of natural numbers, the relation $S_\perp$ contains the following tuples

$$S_\perp = \{\ (Poland, Winter, Eagle)\ ,\ (Slovakia, Winter, Eagle)\ ,\ (Poland, Winter, Falcon)\ ,$$
$$(Slovakia, Winter, Falcon)\ ,\ (Poland, Spring, Eagle)\ ,\ (Slovakia, Monday\ , Eagle)\ ,$$
$$(Poland, Spring, Falcon),\ (Slovakia, Tuesday, Falcon)\ \}$$

For each tuple from $S_\perp$, an elementary aggregation component can be defined, e.g. assuming $s_1 = (Poland, Winter, Eagle)$ is a tuple from $S_\perp$ and $f(s_1) = 55$, the tuple $(Poland, Winter, Eagle, 55)$ is an elementary aggregation component, and is semantically interpreted as "55 Eagles in Poland in Winter". Similarly, $s_2 = (Slovakia, Winter, Eagle)$ is a tuple from $S_\perp$ and $f(s_2) = 65$, the tuple $(Slovakia, Winter, Eagle, 65)$ is an elementary aggregation component, and is semantically interpreted as "65 Eagles in Slovakia in Winter".

Definition 4.9 An aggregation framework is $\Pi = (\mathbf{C}, \Theta)$, where

1. **C** is a data cube,
2. $\Theta : S \to P(A)$ is a function, where $P(A)$ is a power set of A, and for
   a. an instance $x \in I_{E^i}$, where $E^i \in \mathbf{E}^i$ of the conceptual model $\mathbf{M}^i$,
   b. a set of constants α^j, for each $j \neq i$, where $\alpha^j \in I_{E^j}$ and $E^j \in \mathbf{E}^j$ of the conceptual model $\mathbf{M}^j$,

$$\Theta(\alpha^0,...,x,...,\alpha^d) = \begin{cases} \{(\alpha^0,...,x,...,\alpha^d, m_x)\} & , x \in I_{E^i_\perp} \text{ and } \alpha^j \in I_{E^j_\perp} \\ \bigcup_{y \in Y} \Theta(\alpha^0,...,y,...,\alpha^d) & , Y = \{y : (x,y) \in r^i\} \end{cases}$$

Example 4.4 Let **C** be the data cube for which there exist the elementary aggregation components as described in Example 4.3. Assuming $I_{Region} = \{Central\ Europe\}$ is a set of the *Region*-level instances and $s = (Central\ Europe, Winter, Eagle)$ is a tuple from *S*, the aggregation framework $\Pi = (\mathbf{C}, \Theta)$ defines

$$\Theta(s) = \Theta(Central\ Europe, Winter, Eagle) =$$

$$\Theta(Poland, Winter, Eagle) \cup \Theta(Slovakia, Winter, Eagle) =$$

$$\{\ (Poland, Winter, Eagle, 55)\ ,\ (Slovakia, Winter, Eagle, 65)\ \}$$

Given the concepts of a data cube, elementary aggregation component and aggregation framework, Definitions 4.10, 4.11 and 4.12 describe the summarizability conditions.

Definition 4.10 Let $\Pi = (\mathbf{C}, \Theta)$ be an aggregation framework. Assuming there exists a tuple $s = (\alpha^0,...,x,...,\alpha^d)$ from *S*, defined by an instance $x \in I_{E^i}$, where $E^i \in \mathbf{E}^i$, and a set of constants α^j, where $\alpha^j \in I_{E^j}$ and $E^j \in \mathbf{E}^j$, for each $j \neq i$, the condition of *disjointness* is satisfied iff

$$\Theta(\alpha^0,...,y_k,...,\alpha^d) \cap \Theta(\alpha^0,...,y_l,...,\alpha^d) = \emptyset$$

for every two distinct instances y_k and y_l, where $(x, y_k) \in r^i$ and $(x, y_l) \in r^i$.

The summarizability condition implies that elementary aggregation components must be disjoint. In the case the leaf-level instances of the aggregated dimension are individual

objects, the disjointness of elementary aggregation components suffices to exclude double-counting. However, when the aggregation is performed on pre-aggregated values and corresponding instances are not individual objects, the verification of disjointness requires the semantical analysis of the data cube. In Example 4.4, the aggregation framework produces correct results for the *Region*-level instances (assuming *Country*-level elementary aggregation components are disjoint) but might result in double-counting for *Year*-level instances because some *Eagles* observed in *Spring* might also have been observed in *Winter*. In order to verify the correctness of the aggregation, the conceptual model of the aggregated dimension must be extended to the level which corresponds to individual objects, i.e. individual *Eagles*. The elementary aggregation components of the extended model can then be analysed for the compliance with the disjointness condition. Assuming, however, that the semantics in Examples 4.3 and 4.4 concerns a change in the bird's population (e.g. "the population of Eagles in Poland in Winter increased by 55"), the correctness of the *Year*-level aggregation can be confirmed immediately without the semantically analysis. The detailed discussion of summarizability for various measure types is presented in Definition 4.12.

Definition 4.11 Let $\Pi = (\mathbf{C}, \Theta)$ be an aggregation framework. Assuming there exists a tuple $s = (\alpha^0, ..., x, ..., \alpha^d)$ from *S*, defined by an instance $x \in I_{E^i}$, where $E^i \in \mathbf{E}^i$, and a set of constants α^j, where $\alpha^j \in I_{E^j}$ and $E^j \in \mathbf{E}^j$, for each $j \neq i$, the condition of *completeness* is satisfied iff

$$\bigcup_{y \in Y} \Theta(\alpha^0, ..., y, ..., \alpha^d)$$

constitutes the entire set of the elementary aggregation components of the instance *x*, where $Y = \{y : (x, y) \in r^i\}$.

The condition of completeness implies that there exists a full mapping between instances and corresponding aggregation components on subsequent hierarchical levels. In Example 4.4, the aggregation framework may produce incorrect results because the union of the *Country*-level instances does not semantically constitute the entire set of the *Region*-level instances. Unless the semantics of the framework is chosen deliberately, i.e. only *Poland* and *Slovakia* are considered; the set of the *Country*-level instances and their corresponding aggregation components must be extended to include other *Central Europe* countries. In

the case of a non-covering aggregation framework the semantics of missing values must be additionally considered. The *unknown* and *non-existing* cases are typically distinguished on the conceptual level as *non-available* and *structural zero* values. For example, there are *Countries* in *Central Europe* for which a number of *Eagles* is unknown (non-available) and there are *Countries* in *Central Europe* where *Eagles* do not exist (structural zero).

The third condition of summarizability concerns aggregation functions operating on elementary aggregation components. Assuming **F** is a set of aggregation functions, e.g. *sum*, *count*, *avg*, etc., **T** = {*flow*, *stock*, *unit-per-value*} is a set of measure types and $\Psi_{\mathbf{E}} = \{temporal, non\text{-}temporal\}$ is a classification of entity types in **E**, the following condition verifies the compatibility of the elements in terms of the correctness of the overall aggregation framework Π.

Definition 4.12 Let $\Pi = (\mathbf{C}, \Theta)$ be an aggregation framework and let $s = (\alpha^0, ..., x, ..., \alpha^d)$ be a tulpe from *S*, defined by an instance $x \in I_{E^i}$, where $E^i \in \mathbf{E}^i$, and a set of constants α^j, where $\alpha^j \in I_{E^j}$ and $E^j \in \mathbf{E}^j$, for each $j \neq i$. Assuming there exist a category type $\Psi_{\mathbf{E}} = (E^j)$, a function from **F** operating on the measures of a type $\mathbf{T}(m)$, the condition of *compatibility* is satisfied iff the function is compatible with the measure type, i.e. the aggregation is semantically meaningful.

The third condition concerns a distinction between the summarization for temporal (e.g. *Season*, *Year*) and non-temporal entity types (e.g. *Country*, *Region*). Semantically, *flow* refers to periods and records cumulative effects at the end of periods (e.g. increase in quantity). *Stock* is measured and recorded at particular points in time (e.g. quantity or amount). *Value-per-unit* is also determined by a fixed time but is not similar to *Stock* because of a difference in unit (e.g. item price, cost per unit). The compatibility condition is semantically analysed for a given combination of the elements of the aggregation framework. For example, the temporal summarizability of *flows* is semantically meaningful (e.g. increase in quantity during a month as a sum of daily flows) while the temporal summarizability of *stock* and *value-per-unit* measure types depends on the aggregation function (e.g. quantity at the end of a month is not a sum of daily quantities).

Besides Definitions 4.10, 4.11 and 4.12, the aggregation semantics in the conceptual modelling for SA concerns the conditions which are specific for the discipline. As a result,

the completeness and correctness of the proposed conceptual model for SA requires the verification of the following conditions:

1. consistency of geo-representations used for the modelling of spatial properties,
2. completeness, disjointness and consistency of definition domains of spatial instances (e.g. consistency of a unit measure),
3. semantics of spatial measures (e.g. disjointess of spatial measures),
4. semantics of operators (aggregation functions) referencing both spatial and non-spatial data (e.g. aggregation function over spatial measures, groupings of geometries etc).

4.3 Logical data model

The argumentation presented in Section 4.1 provides rationale for the adaptation of the multidimensional modelling to design the data model for SA. Although the conceptual abstraction of the proposed data model for SA does not assume any implementation details, similarities to the entity-relationship (ER) model incline towards an application of the relational model. The selection of the relational model for designing the logical schema is additionally justified by the fact that most SA methods operate on tabular data or directly on tables in relational databases. As a result, all performance-improvement techniques proposed in the thesis were designed specifically for the relational data representation. In order to maintain the consistency of the thesis, the following presentation of the data model for SA is based on the relational and data warehousing terminology. This enables to highlight the elements which are unique for the proposed data model while referring to the definitions of basic concepts in the literature [Kim96, CD97, Raf03].

4.3.1 Dimensions

The relational representation of dimensions is based on the conceptual model presented in Section 4.2.3.1. Designing the logical schemata which model complex hierarchies requires, however, the consideration of options which the relational model provides. A dimension

modelling a hierarchy can be represented either as a denormalised table or as a set of normalised tables. The structure of a spatial dimension is additionally determined by spatial attributes which may be of heterogeneous data types. Given the above assumptions, the Section presents the logical schemata of dimensions modelling complex spatial hierarchies designed specifically for the proposed data model for SA.

The relational model of a dimension **D** is based on the conceptual model $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ described in Section 4.2.3.1. In the denormalised schema the dimension **D** is modelled as a single table and the attributes of entity types $E \in \mathbf{E}$ are the attributes of the dimension table $\mathbf{D} = \{a_0,...,a_m\}$. It is assumed that a spatial dimension models a hierarchy of spatial objects which are characterised by an identifier (e.g. a numerical value, a name), a location (two coordinates *x* and *y*), an area (a geometry *g*) and other descriptive attributes. An set of the identifiers of the spatial objects corresponding to the hierarchical levels is labelled as $\mathbf{H} = \{h_0,...,h_l\}$. The remaining (feature) attributes of the spatial dimension are labelled as $\mathbf{F} = \{f_0,...,f_k\}$. The default geo-representation is the Euclidian space consistent with the framework described in Section 4.2.1. The dimension table contains a set of tuples $\mathbf{T} = \{t_0,...,t_t\}$ and each tuple *t* is uniquely identified by the primary key attribute h_0.

The normalised schema of a dimension **D** is based on the conceptual model $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ described in Section 4.2.3.1. In the normalised schema, the dimension **D** is modelled as a set of normalised tables, where each table $\mathbf{D}_i$, for $i = 0, \ldots, l$, corresponds to a single entity type $E \in \mathbf{E}$ of the conceptual model **M**. In the normalised spatial dimension, each table contains the primary key h_i, foreign key h_{i+1} referencing the subsequent level, and feature attributes which characterise the spatial object modelled in the table. The exact structure of the spatial dimension **D** is determined by the type of the spatial hierarchy and depends on the corresponding conceptual model **M**.

4.3.2 Relational representation of spatial hierarchies

Simple-symmetric hierarchy

Let **D** be a dimension table modelling a simple-symmetric spatial hierarchy. The denormalised schema of the dimension **D** is represented as a single table

$$\mathbf{D} = \{h_0, x_0, y_0, g_0, f_0, ..., h_l, x_l, y_l, g_l, f_l, ..., f_k\}$$

Since the attribute h_i uniquely identifies its corresponding spatial object, the relationships between spatial objects are modelled in the normalised schema as foreign keys h_{i+1} which reference the subsequent hierarchical level. As a result, the normalised schema of the dimension **D** is defined as a set of normalised tables

$$\mathbf{D}_0 = \{h_0, h_1, x_0, y_0, g_0, f_0, ...\} \ , \ \ldots \ , \ \mathbf{D}_l = \{h_l, x_l, y_l, g_l, f_l, ...\}$$

Figure 4.7 represents the normalised schema of the dimension **D** modelling the simple-symmetric spatial hierarchy.

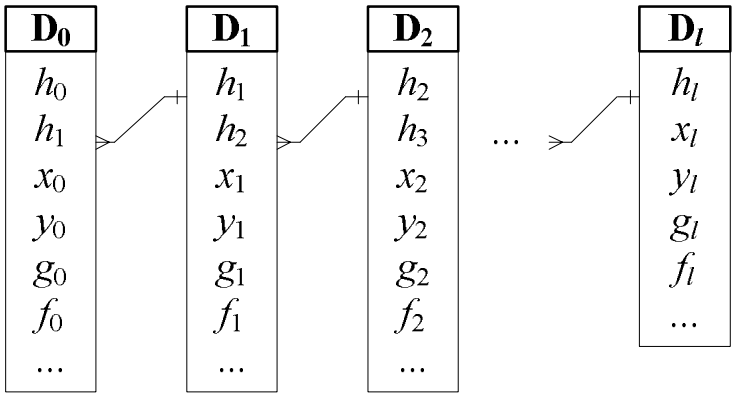


Fig. 4.7 The normalised schema of the dimension modelling the simple-symmetric spatial hierarchy.

Simple-asymmetric hierarchy

A dimension **D** modelling a simple-asymmetric hierarchy is similar to the dimension modelling a simple-symmetric hierarchy. In the denormalised schema, however, some tuples may contain *nulls* when instances on the lower levels of the asymmetric hierarchy do not exist. As discussed in Section 4.2.3.2, the proper handling of *nulls* requires a distinction between the *non-available* and *structural-zero* cases. An alternative approach, eliminating the problem, was proposed by Pedersen [Ped00]. In the approach, a complex hierarchy (e.g. simple-asymmetric spatial hierarchy) is transformed into a simple one by making relationships in the hierarchy *covering* (by introducing extra intermediate values), *onto* (by introducing placeholder values at lower levels for values without any children),

and *strict* (by fusing values together). After the transformation, the complex hierarchy can be modelled in the denormalised schema under the condition that the transformation does not change the semantics of the hierarchy.

In the case of the normalised schema, a dimension **D** modelling a simple-asymmetric spatial hierarchy is represented as a set of normalised tables.

$$\mathbf{D}_0 = \{h_0, h_1, x_0, y_0, g_0, f_0, ...\} \ , \ \dots \ , \ \mathbf{D}_l = \{h_l, x_l, y_l, g_l, f_l, ...\}$$

Since an instance of the simple-asymmetric spatial hierarchy may contain instances for which child instances do not exist, relationships between the normalised tables modelling the hierarchy are of 0:N cardinality. Figure 4.8 represents the normalised schema of the dimension **D** modelling the simple-asymmetric spatial hierarchy.

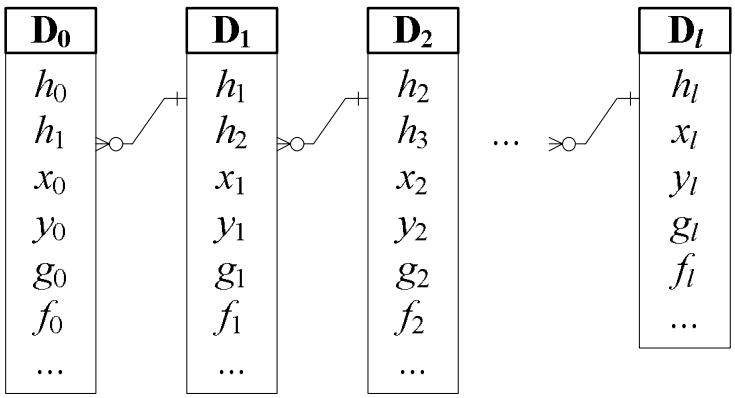


Fig. 4.8 The normalised schema of the dimension modelling the simple-asymmetric spatial hierarchy.

Modelling the simple-asymmetric spatial hierarchy can thus be performed according to the three approaches: a denormalised schema with the extended semantics of *nulls*, a denormalised schema with a transformed hierarchy, and a normalised schema. The three approaches are also feasible for other complex spatial hierarchies.

Multiple hierarchy

A multiple hierarchy is defined in the conceptual model as a union of simple hierarchies which share at least the root and leaf levels (see Section 4.2.3.1). A denormalised schema

of a dimension **D** modelling the multiple hierarchy, which consist of at least one simple-asymmetric hierarchy, requires the support of the extended semantics of *nulls* or the transformation of the asymmetric hierarchy into a symmetric one [Ped00].

The normalised schema of a dimension **D** modelling the multiple spatial hierarchy is represented as a set of normalised tables. In the normalised schema, tables representing the shared levels reference the higher-level tables of the component hierarchies. For example, let $\mathbf{D}_k$ be a shared *k* level table of the dimension **D** modelling the multiple spatial hierarchy

$$\mathbf{D}_k = \{h_k, h^1_{k+1}, ..., k^n_{k+1}, x_k, y_k, g_k, f_k, ...\}$$

Assuming the multiple spatial hierarchy is a union of *n* component hierarchies, the table $\mathbf{D}_k$ contains the primary key h_k and a set of foreign keys h^i_{k+1}, for $i = 1, \ldots, n$, which reference the tables $\mathbf{D}^i_{k+1}$ of the *k*+1 non-shared level of the hierarchy

$$\mathbf{D}^1_{k+1} = \{h^1_{k+1}, h^1_{k+2}, x^1_{k+1}, y^1_{k+1}, g^1_{k+1}, f^1_{k+1}, ...\}, \ldots, \mathbf{D}^n_{k+1} = \{h^n_{k+1}, h^n_{k+2}, x^n_{k+1}, y^n_{k+1}, g^n_{k+1}, f^n_{k+1}, ...\}$$

Figure 4.9 represents the normalised schema of the dimension **D** modelling the multiple spatial hierarchy, where the foreign keys h^i_1 in the shared leaf-level table $\mathbf{D}_0$ reference the tables $\mathbf{D}^i_1$, for $i = 1, \ldots, n$, of the non-shared level 1 of the hierarchy.

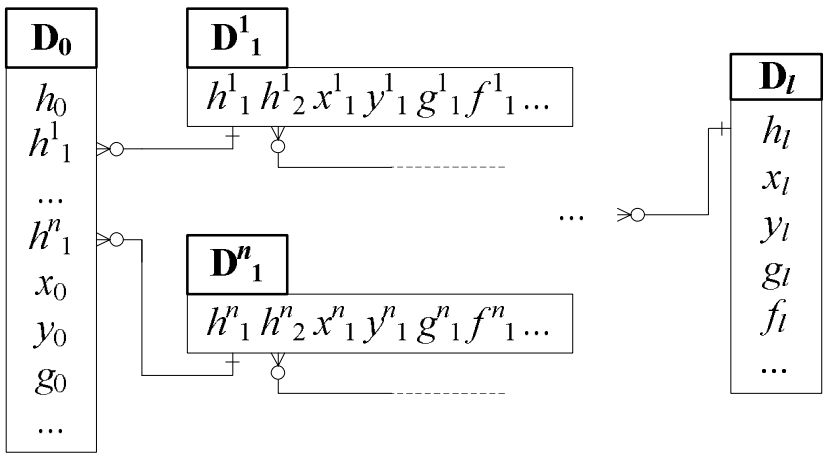


Fig. 4.9 The normalised schema of the dimension modelling the multiple spatial hierarchy.

Generalised and non-covering hierarchy

The relational representation of a dimension **D** modelling a generalised hierarchy is similar to a dimension modelling a multiple hierarchy. The similarity between the dimensions concerns the levels which are not part of exclusive paths. The levels which are part of the exclusive paths are represented as tables exclusively referencing higher-level tables participating in the exclusive paths. For example, let $\mathbf{D}_k$ be a k level table of the dimension **D** modelling a generalised spatial hierarchy

$$\mathbf{D}_k = \{h_k, h_{k+1}, x_k, y_k, g_k, f_k, ...\}$$

The tables $\mathbf{D}^i_{k+1}$, for $i = 1, \ldots, n$, are mutually exclusive components of the k+1 level of the generalised spatial hierarchy

$$\mathbf{D}^1_{k+1} = \{h^1_{k+1}, h^1_{k+2}, x^1_{k+1}, y^1_{k+1}, g^1_{k+1}, f^1_{k+1}, ...\}, \ldots, \mathbf{D}^n_{k+1} = \{h^n_{k+1}, h^n_{k+2}, x^n_{k+1}, y^n_{k+1}, g^n_{k+1}, f^n_{k+1}, ...\}$$

The table $\mathbf{D}_{k+1}$ contains the primary key h_k and the foreign key h^i_{k+1} which references one of the k+1 level tables $\mathbf{D}^i_{k+1}$, $i = 1, \ldots, n$. The proposed modelling of a generalised hierarchy requires maintaining unique h^i_{k+1} identifiers in the tables participating in the exclusive paths. Figure 4.10 represents the normalised schema of the dimension **D** modelling the generalised spatial hierarchy, where the tables $\mathbf{D}^i_1$, for $i = 1, \ldots, n$, are part of the mutually exclusive paths which end in the shared table $\mathbf{D}_0$.

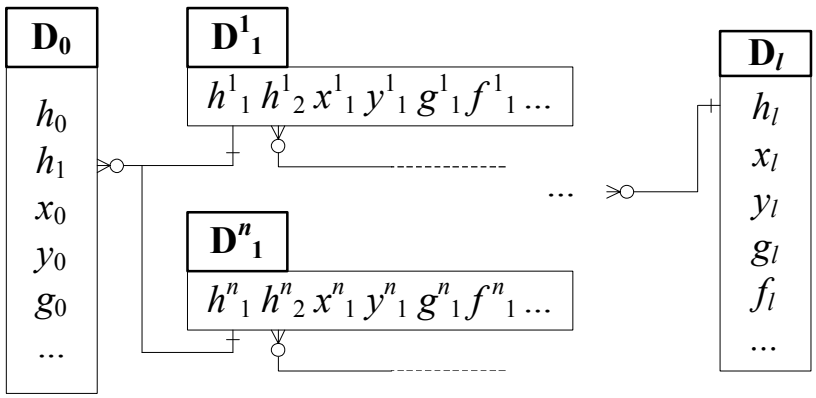


Fig. 4.10 The normalised schema of the dimension modelling the generalised spatial hierarchy.

Parallel independent hierarchy

Since the parallel independent hierarchy share only the leaf level, the denormalised schema is feasible but very ineffective. The ineffectiveness results from a potential large number of *nulls* in every tuple of the denormalised table. The normalised schema of the dimension **D** modelling the parallel independent spatial hierarchy contains a leaf-level table

$$\mathbf{D}_0 = \{h_0, h_1, ..., h_n, x_0, y_0, g_0, f_0, ...\}$$

consisting of the primary key h_0 and a set of foreign keys h_i, for $i = 1, \ldots, n$, referencing the level 1 of the hierarchy. Since all higher levels of the hierarchy are independent, every component hierarchy is represented as a set of normalised tables:

$$\mathbf{D}_0 = \{h_0, h_1, ..., h_n, x_0, y_0, g_0, f_0, ...\},$$

$$\mathbf{D}_1 = \{h_1, h_2, x_1, y_1, g_1, f_1, ...\}, \ldots, \mathbf{D}_l = \{h_l, x_l, y_l, g_l, f_l, ...\}$$

Figure 4.11 represents the normalised schema of the dimension **D** modelling the parallel independent spatial hierarchy

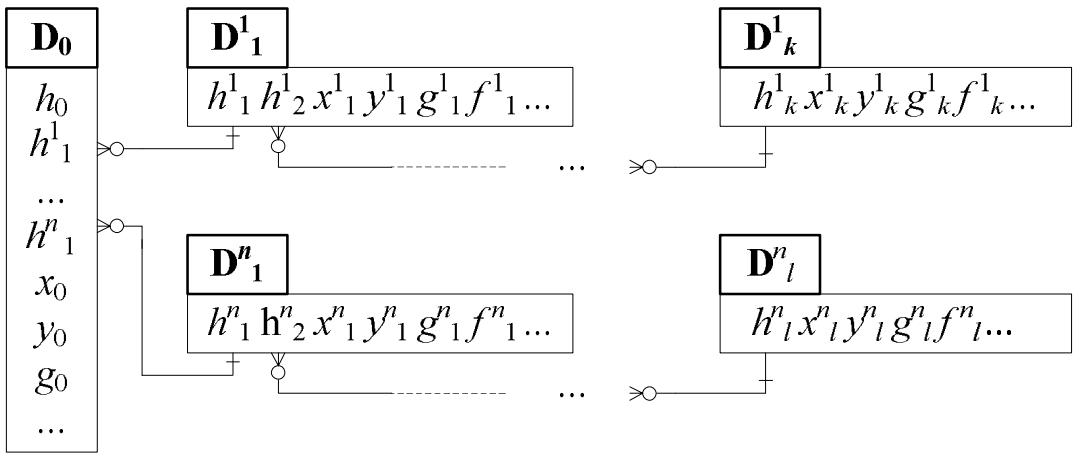


Fig. 4.11 The normalised schemata of the dimension modelling the parallel independent spatial hierarchy.

4.3.3 Logical database schema

The star and snowflake schemata are the most common relational representations of the multidimensional data models [Kim96]. The star schema organises data in a fact table referencing denormalised dimension tables, whereas the snowflake schema allows for normalised dimensions. The logical schema of the data model for SA is similar to the star and snowflake schemata because analytical data for SA are modelled in a central table which references dimensions of a structure determined by hierarchies (see Section 4.3.2). Because of the similarity, the logical schema of the data model for SA may benefit from the properties and modelling techniques characteristic of the multidimensional data models, e.g. handling changing dimensions, temporal models etc.

Denormalised database schema

The relational representation of a data cube is based on the conceptual model described in Definition 4.7. As a result, the logical schema of the data cube includes a set of dimensions $\Omega = \{\mathbf{D}^0,...,\mathbf{D}^d\}$ and a central (fact) table $\mathbf{FT} = \{h_0^0,...,h_0^d, m_0,...,m_m\}$ consisting of the foreign keys h_0^i referencing the dimensions $\mathbf{D}^i$, for $i = 0$, … , d , and a set of measures m_i, for $i = 0$, … , m. In the case of the denormalised (star) schema, each dimension is modelled as a denormalised table (see Section 4.3.2)[5]. In the logical schema of the data model for SA, at least one of the dimensions is spatial, i.e. the dimension models a hierarchy of spatial objects. Spatiality is typically represented in the logical schema as attributes in the spatial dimension (i.e. spatial hierarchies, coordinates, geometries) or as spatial measures in the fact table. The former enables the modelling of spatial objects for which numerical measures are collected, i.e. a numerical measure (e.g. number of *Cars*) collected for a set of spatial objects (e.g. *City*) in a spatial dimension modelling a spatial hierarchy (e.g. *City*, *Region*, *Country*). The later enables the modelling of spatial phenomena with respect to some dimensions of interest, i.e. spatial measure (e.g. iso-geometries of temperature) collected in a measurement framework (e.g. dimension with a

[5] The denormalised schema of the data model for SA is only feasible for dimensions containing hierarchies which can be modelled as denormalised tables or hierarchies for which appropriate transformations exist.

measurement scale and unit, such as *Celsius degrees*). The two cases confirm that the data model for SA supports the geo-representations of spatiality described Section 4.1.2. Figure 4.12 represents the denormalised database schema containing a fact table with measures and foreign keys referencing four denormalised dimensions.

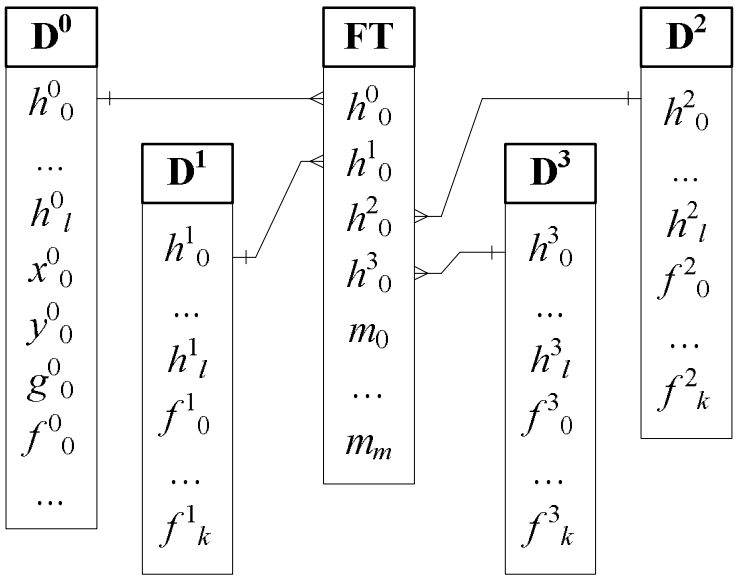


Fig. 4.12 The denormalised database schema.

Normalized database schema

Although the normalised schema has a more sophisticated structure, the type of schema is more flexible and adequate for the representation of complex hierarchies. The normalised dimensions modelling complex spatial hierarchies (see Section 4.3.2) determines the structure of the normalised schema. Besides the representation of complex hierarchies, the normalised schema also facilitates the modelling of aggregations for the non-leaf levels of the hierarchies. For example, let $\mathbf{D}^0$ be the normalised spatial dimension modelling a simple-symmetric spatial hierarchy

$$\mathbf{D}_0^0 = \{h_0^0, h_1^0, x_0^0, y_0^0, g_0^0, f_0^0, ...\} \ , \ \ldots \ , \ \mathbf{D}_l^0 = \{h_l^0, x_l^0, y_l^0, g_l^0, f_l^0, ...\}$$

Assuming that fact data are collected for the leaf and l levels of the spatial dimension $\mathbf{D}^0$, the corresponding fact tables $\mathbf{FT}_0$ and $\mathbf{FT}_l$ referencing the identifiers h_0^0 and h_l^0 of the dimension tables $\mathbf{D}_0^0$ and $\mathbf{D}_l^0$ are modelled as

$$\mathbf{FT}_0 = \{h_0^0,...,h_0^d,m_0,...,m_m\} \ , \ \mathbf{FT}_l = \{h_l^0,...,h_0^d,m_0,...,m_m\}$$

Figure 4.13 represents the normalised schema consisting of two fact tables for the leaf and l levels of the simple-symmetric spatial hierarchy.

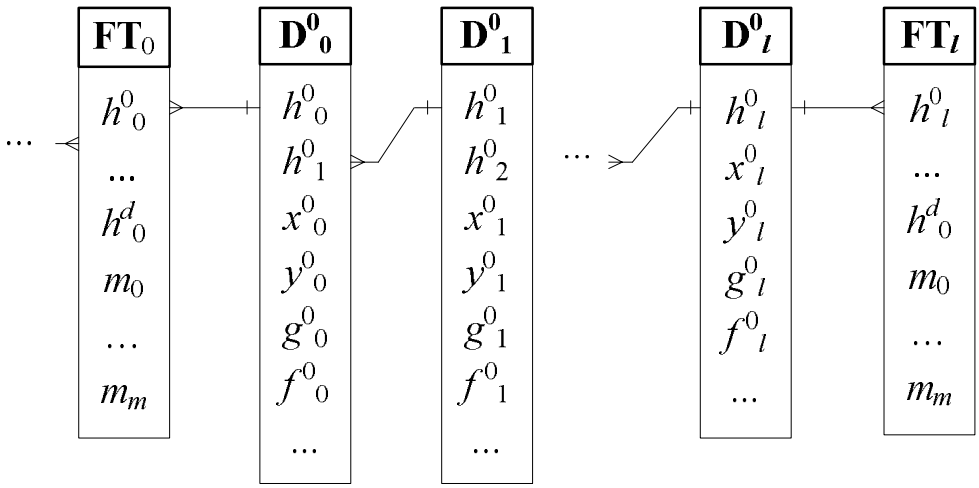


Fig. 4.13 The normalised schema with two fact tables.

In addition to the normalisation of the logical schema resulting from the modelling of complex hierarchies, the logical schema for the proposed data model for SA supports two other normalisations (*path-* and *field- normalised* schemata).

Both the normalisations require the leaf-level table $\mathbf{D}_0^0$ of the dimension $\mathbf{D}^0$ to include all hierarchical identifiers h_i^0 , for $i = 0$, … , l. The path-normalised schema is modelled as a set of sequentially referenced tables $\mathbf{D}_i^0$, for $i = 0$, … , l , corresponding to the levels of the hierarchy. Each of the tables contains feature attributes relevant to the level as well as hierarchical identifiers of the level and all higher levels. For example, let $\mathbf{D}^0$ be the path-normalised dimension modelling a simple-symmetric spatial hierarchy. The tables $\mathbf{D}_i^0$, for $i = 0$, … , l , corresponding to the levels of the hierarchy are defined as

$$\mathbf{D}_0^0 = \{h_0^0,...,h_l^0,x_0^0,y_0^0,g_0^0,f_0^0,...\} \ , \ \mathbf{D}_1^0 = \{h_1^0,...,h_l^0,x_1^0,y_1^0,g_1^0,f_1^0,...\} \ , \ \dots \ ,$$
$$\mathbf{D}_l^0 = \{h_l^0,x_l^0,y_l^0,g_l^0,f_l^0,...\}$$

Figure 4.14 represents the path-normalised schema containing the spatial dimension modelling the simple-symmetric spatial hierarchy.

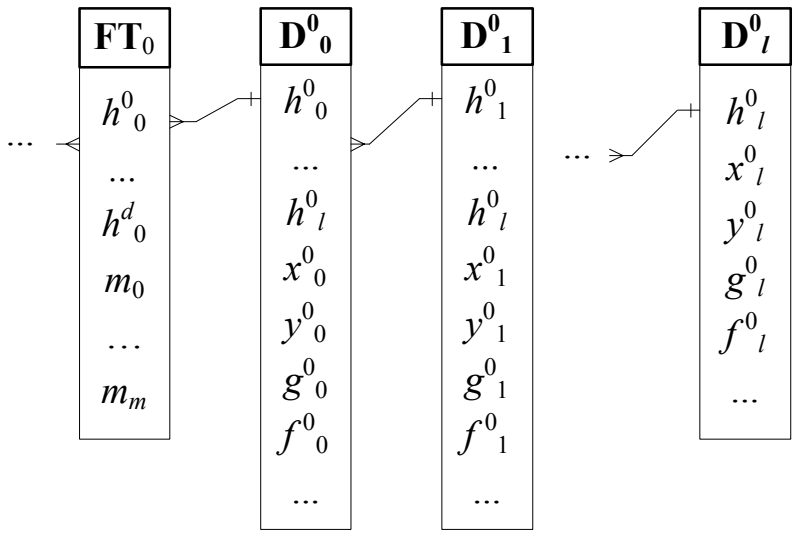


Fig. 4.14 The path-normalised schema.

Similarly to the path-normalized schema, the field-normalized schema requires the leaf-level table $\mathbf{D}_0^0$ to contain all hierarchical identifiers h_i^0, for $i = 0, \ldots, l$. The field-normalised schema is represented as a set of tables $\mathbf{D}_i^0$, for $i = 1, \ldots, l$, corresponding to the levels of the hierarchy referenced by the leaf-level table. Each non-leaf-level table contains the hierarchical identifier of the level and feature attributes relevant to the level. For example, let $\mathbf{D}^0$ be the field-normalised dimension modelling the simple-symmetric hierarchy. The leaf-level table is represented as

$$\mathbf{D}_0^0 = \{h_0^0,...,h_l^0,x_0^0,y_0^0,g_0^0,f_0^0,...\}$$

whereas the non-leaf level tables are defined as

$$\forall\ 1 \le i \le l \qquad \mathbf{D}_i^0 = \{h_i^0,x_i^0,y_i^0,g_i^0,f_i^0,...\}$$

Figure 4.15 represents the field-normalised schema containing the spatial dimension modelling the simple-symmetric spatial hierarchy.

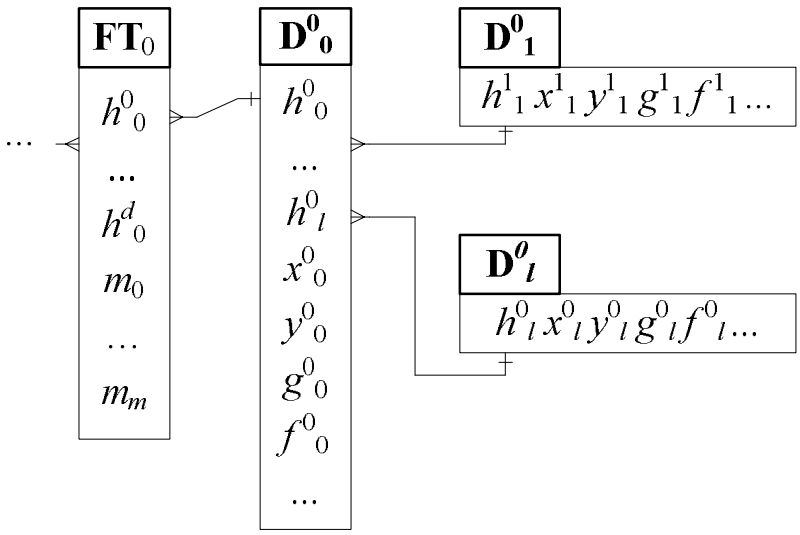


Fig. 4.15 The field-normalised schema.

4.4 Physical data model

Indexing multidimensional data with secondary indexes (e.g. the B-trees or bitmap indexes) is the most common performance improvement approach applied in physical data models for analytical processing. The main advantage of the approach is a reduced number of retrieved disk pages in comparison with a number of tuples specified by query restrictions. However, the secondary indexes do not always guarantee satisfactory performance which may deteriorate in the case of complex data models and diversified query processing workloads. Other drawbacks of the approach are high storage requirements and costly maintenance.

The data model for SA attempts to solve the performance problems typical of SA. The objective is accomplished by an application of enhancements, embedded into the proposed data model, which make SA methods more efficient. The main idea underlying the enhancements consists in ensuring the compatibility of the physical data organisation and data access with the profile of the spatial analytical processing (see Section 1.2). The proposed physical data model provides the proximity (locality)-preserving hierarchical organisation of analytical and spatial data. The data access methods benefit from the physical data organisation, thus making physical I/O transfers more effective. The following Sections of the Chapter present the physical layer of the data model for SA, focusing on the performance enhancements which were design specifically for SA.

4.4.1 Physical data modelling prerequisites

Physical data modelling provides better results when information about data processing patterns and data distributions is available. Analysing the scope and character of SA facilitates the identification of most frequently accessed attributes and most common data processing patterns. Analysing data distributions facilitates the estimation of selectivity for attributes and queries. The following parts of the Section discuss the role of the prerequisites in designing the physical layer of the data model for SA.

Data processing analysis

Although SA methods differ in scope and objective, there are many data processing primitives which are common for all the methods (see Sections 1.2). The primitives are instances of query patterns which can be further generalised to query templates. For instance, if a workload of queries aggregates a measure and restricts only hierarchical attributes, a query template can be derived from the workload and optimised. Contrary to the standard query patterns, the ad-hoc queries (typical of interactive data analysis) are more difficult to handle especially when the queries are significantly different from the standard queries. In the case of ad hoc queries which are similar to the optimised query templates, the evaluation plan of the ad hoc queries can be rewritten in order to benefit from the optimised templates .

The query patterns identified by the author as common for SA can be grouped into two categories: spatial queries (e.g. NN-queries, spatial joins, topology queries, spatial aggregations, network queries, etc.) and analytical queries (e.g. multidimensional hierarchical queries, range queries, aggregations, etc.). All data processing profiles which are specific for SA can be defined as a combination of query patterns from the two categories, e.g. a multidimensional hierarchical query with spatial predicates. Such decomposition of complex queries into data processing primitives improves query execution plans through the independent or cross-optimization of the primitives. The detailed analysis of data processing profiles is always performed in the context of a data model and underlying data distributions.

Data distribution analysis

Exact or estimated data distribution statistics enable the identification of most selective attributes which contribute to the overall selectivity of queries. Since the effectiveness of the proposed data model for SA correlates with selectivity, the design of the physical model must consider underlying data distributions. The definition 4.13 formalises the concept of selectivity.

Definition 4.13 A selectivity of a query φ with respect to a relation **D** is a number of tuples in a result set of the query compared to a number of tuples stored in the relation **D** :

$$selectivity(\mathbf{D},\varphi) = \frac{|RS(\mathbf{D},\varphi)|}{|\mathbf{D}|}$$

A selectivity of a query restriction is defined accordingly as a number of tuples when attributes are restricted compared to a number of tuples without the restriction. The design of the physical data model which take into account most selective restrictions leads to more effective query evaluation plans. The analysis of data distributions supported by the analysis of definition domains is also necessary for the multidimensional hierarchical clustering (MHC/HI). The definitions of surrogates encoding a hierarchy require information about a number of distinct values for each level of the hierarchy. In the case of complex data models, the analysis of dependencies among attributes provides an insight about correlations between data distributions. The main advantage of the analysis is an efficient execution of queries which were not directly optimised but are similar to the implemented query optimization strategies.

4.4.2 Physical schema of spatial dimension

The proposed physical data model is meant to improve the performance of SA algorithms operating on large data sets. Since the data sets are usually larger than internal memory, the SA algorithms are typically I/O bound, i.e. data access is the main performance bottleneck. As a result, improving the effectiveness of I/O operations is expected to deliver significant performance gains. For the needs of the thesis the problem is defined as follows.

Let **D** be a space of all possible distributions of a certain data set and let S be a space of data access paths which are covered by some data analysis algorithms. Let $P(s)$ be a probability that an algorithm covers a path $s \in S$ and let $C(s)$ be a cost related to covering the path s. Assuming the cost $C(s)$ depends on an underlying data distribution D, finding a data distribution which results in the most efficient execution of the data analysis algorithms is modelled as a following optimisation problem:

$$\min_{D \in \mathbf{D}} \left(\sum_{s \in S} P(s) \times C(s) \right)$$

Although there exists a data distribution which minimizes the cost of covering the data access paths, there does not exist an efficient method for finding the globally optimal solution. As a result, the objective is to find a data distribution which is suboptimal but adequate for all data analysis algorithms. Without defining a detailed model of a disk drive, the properties of the device [Vit01] enable expecting some performance improvements of I/O operations when data access is exact (only relevant data pages are retrieved) and continuous (number of random access operations is reduced). In order to achieve the data access characteristics, the physical data model must provide a data distribution so that the paths of data access operations of the algorithms follows the cost-efficient patterns.

In order to provide a common data organization strategy, the proposed physical data model exploits the property inherent to all methods of spatial analysis, i.e. locality of data references (see Section 1.1). Implementing the locality-preserving data distributions is expected to reduce the cost related to I/O transfers because data clusters can be retrieved in a single (exact) or few continuous data accesses. Given the above assumptions, the heuristic consisting in the locality-preserving physical data organisation is the main assumption underlying the proposed physical data model.

4.4.2.1 Proximity-preserving encoding spatial hierarchies

The data model for SA is based on the proximity-preserving encoding of spatial hierarchies. The heuristics underlying the encoding assumes that spatial hierarchies and other spatial attributes (locations and geometries) are clustered in a manner reflecting distance semantics. The clustering makes the physical data organisation consistent with the character of SA which demonstrates the locality of data references (see Section 1.1). As a

result, the proposed physical data model arranges data in a spatial dimension so that the tuples which correspond to near spatial objects are closely located on disk drives. Data access operations referencing the tuples are thus more likely to retrieve a single data page (reduction of physical I/O operations) or consecutive data pages in a sequential scan (reduction of random access operations).

As described in the conceptual and logical models, a spatial hierarchy may be defined over spatial objects which are characterized by heterogeneous attributes (see Section 4.3.2). Although the representation accurately reflects the semantics of the spatial objects, the heterogeneity of the attributes may complicate the physical data modelling because some types of the attributes do not reflect distance relations existing among the spatial objects. For examples, assuming there exists a spatial hierarchy defined over spatial objects which contain only one alphanumeric attribute (e.g. name), the lexicographic ordering of values of the attribute does not reflect distance and topology semantics. As a result, the proximity-preserving encoding of spatial hierarchies must be based on spatial attributes (e.g. coordinates, geometry) in order to create a surrogate hierarchy which properly reflects spatial properties. In the proposed physical data model, the proximity-preserving encoding is defined by the $ord_m^{\#}$ function which enhances the standard ord_m function of MHC/HI method (see Section 3.3). The proximity-preserving encoding guarantees that the values of the $ord_m^{\#}$ function obtained for any spatial objects located close to each other are also near. Assuming information about distance relations is available, the $ord_m^{\#}$ function defines the proximity-preserving encoding according to the following algorithm.

```
EncodingOfSpatialHierarchies:

H0 := spatial hierarchy
H1 := surrogate hierarchy

/* iterate from the leaf level of the spatial hierarchy */
for i = 0 to H.numOfLevels ()
/* iterate for all instances of the level */
      for j = 0 to H.numOfInstances (i)
/* check if the attribute is a geometry */
            if IsGeometry (H.Instance (i,j)) then
                  H1.ord (H.Instance (i,j)->g)
            else
                  H1.ord (H.Instance (i,j)->x, H.Instance (i,j)->y);
```

The algorithm produces the proximity-preserving encoding independently for all levels of the spatial hierarchy using the coordinates *x*, *y* and geometries *g* of spatial objects. In order to further improve the quality of clustering encoding, the $ord_m^{\#}$ function needs to take into

account not only distance relations among the instances of the same level of the spatial hierarchy but also distance relations among the parents of the instances. The proximity-preserving encoding of sub-trees of the spatial hierarchy is represented by compound surrogates which are subsequently used in the clustering of the spatial dimension.

Example 4.5 Let $\mathbf{M} = (\mathbf{E}, \mathbf{R})$ be the conceptual model defined in Example 4.1 and let $DB = (I, r)$ be its corresponding instance graph. Given the sets of *Region* and *Country*-level instances $I_{REGION} = \{x_0,...,x_m\}$ and $I_{COUNTRY} = \{y_0,...,y_n\}$, $I^{\#}_{REGION} = \{x^{\#}_0,...,x^{\#}_m\}$ and $I^{\#}_{COUNTRY} = \{y^{\#}_0,...,y^{\#}_n\}$ are the sets of the corresponding proximity-preserving encodings, where $x^{\#}_i = ord^{\#}_m(x_i)$, for i = 0 , … , m, and $y^{\#}_j = ord^{\#}_m(y_j)$, for j = 0 , … , n. Assuming $\mathbf{X}(y) = \{x \in I_{REGION} : (y, x) \in r\}$ is a set of *Region*-level instances which are the children of a *Country*-level instance y, the compound surrogates encoding sub-trees originating from the instance y are defined as:

$$\forall x \in \mathbf{X}(y) \qquad cs(x) = cs(y) \mid ord^{\#}_m(x)$$

Since compound surrogates has numerical interpretation (see Section 3.4), the ordering represented by the compound surrogates is also proximity preserving. Figure 4.16 shows the graphical interpretation of the proximity-preserving encoding defined by compound surrogates.

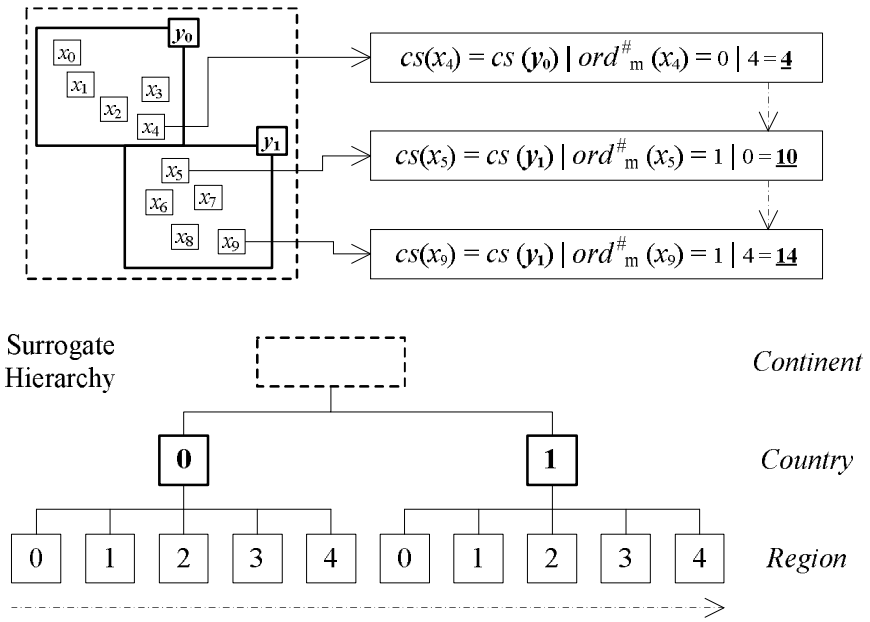


Fig. 4.16 The proximity-preserving encoding defined by compound surrogates.

The proposed physical data model organises tuples in a spatial dimension according to compound surrogates representing the proximity-preserving encoding of a spatial hierarchy. The exact encoding, however, may depend on arbitrary semantics resulting from the properties of spatial objects and from the character of SA. For example, if a given SA method analyses spatial phenomena which demonstrate high autocorrelation, the physical data model may organise data in a regular proximity-preserving manner. However, if the analysed phenomena also demonstrates spatial heterogeneity, regular data clustering may not be fully consistent with the character of the phenomena and consequently with the SA method. Figure 4.17 represents six data distributions which have the same spatial autocorrelation (Moran coefficient equals 0.8) but demonstrate different heterogeneity.

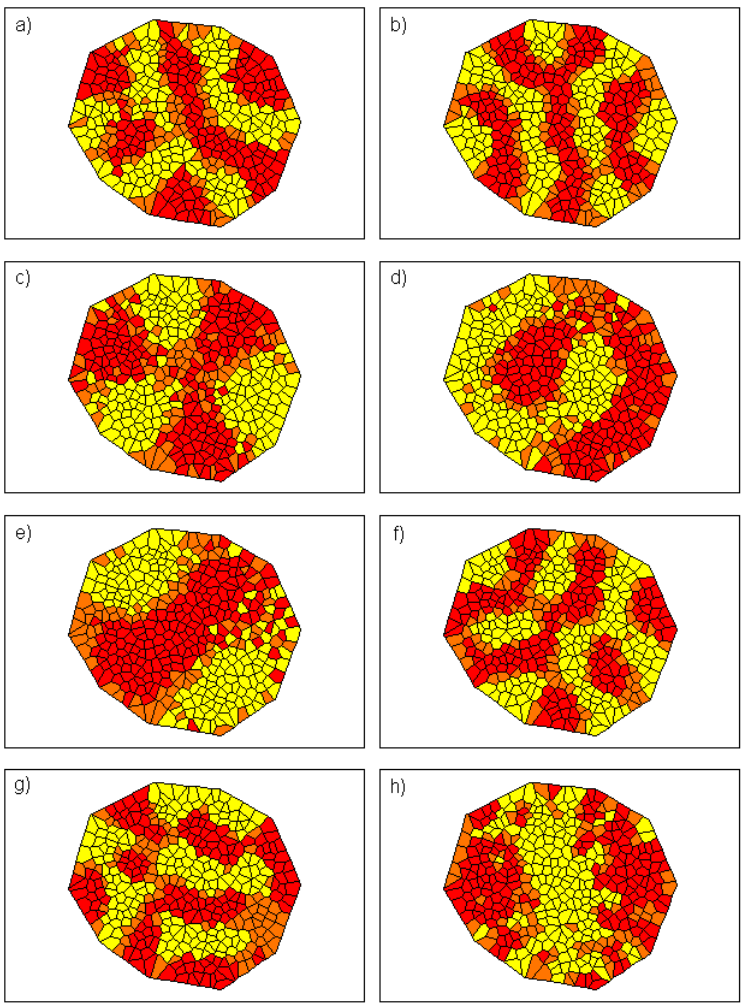


Fig. 4.17 The data distributions with the same spatial autocorrelation but different spatial heterogeneity.

Figure 4.17 confirms that the physical data organisation for all the data distributions should consider not only distance relations but also other properties of spatial objects, e.g. heterogeneity, dispersion, etc. As a result, the logic of the $ord_m^{\#}$ function encoding a spatial hierarchy and defining compound surrogates must be proximity-preserving as well as adaptive to the arbitrary semantics resulting from the specifics of the spatial objects or the requirements of SA. In order to provide a regular proximity-preserving organisation of spatial objects (e.g. autocorrelation without spatial heterogeneity), the encoding can be based on a space-filling curve, e.g. $ord_m^{\#} = Z(x, y)$, where Z is a mapping of the Z-curve (see Section 2.4.3). In the case of complex data distributions, the compound surrogate encoding a spatial hierarchy can regularly cluster leaf-level data but arrange higher levels so that the global data organisation accurately adapts to the required semantics.

4.4.2.2 Physical modelling of complex spatial hierarchies

The physical model of a spatial dimension is based on compound surrogates representing the proximity-preserving encoding of a spatial hierarchy. In the proposed data model for SA, a physical schema of a denormalised spatial dimension modelling a simple-symmetric hierarchy is defined as:

$$\mathbf{D} = \{h_0, x_0, y_0, g_0, f_0, ..., h_l, x_l, y_l, g_l, f_l, ..., f_k, h_0^{\#}, ..., h_1^{\#}, cs\}$$

where h_i and $h_i^{\#}$, for $i = 0$, … , l, are the levels of the original and surrogate hierarchy respectively. However, the levels $h_i^{\#}$ of the surrogate hierarchy are optional when the proximity-preserving encoding of compound surrogates is directly derived from the original hierarchy. Attributes x_0 and y_0 are the coordinates of leaf-level spatial objects and g_0 is an attribute representing a geometry of the objects. The clustering of the spatial objects is based on the Z-curve, so that $h_0^{\#} = Z(x_0, y_0)$, whereas the values of the other attributes of the surrogate hierarchy are obtained from the $ord_m^{\#}$ function. The attribute *cs* is the compound surrogate representing the proximity-preserving encoding of the spatial hierarchy and clustering the dimension **D**.

Although the physical model of a spatial hierarchy described in Section 4.4.2.1 assumes the spatial hierarchy to be simple-symmetric, the proposed data model for SA also supports

complex hierarchies. The physical modelling of the complex hierarchies is based on the linearization or exploitation of the interdependencies of component hierarchies.

4.4.2.2.1 Linearization of complex spatial hierarchies

The linearization of a complex hierarchy is feasible on the conceptual level but may interfere with the semantics of the complex hierarchy. An alternative approach consists in a linearization of a surrogate hierarchy so that the semantics of the original hierarchy remains unaffected. The following part of the Section describes the linearization which operates only on the physical level and proposes the modifications of the linearization algorithm HINTA in order to improve the quality of the hierarchal data clustering.

Generally, the linearization of a hierarchy is a special case of the topological sorting of the hierarchy. Taking into account the objectives of the thesis, the most adequate linearization algorithm for the physical clustering of complex hierarchies known to the author was proposed by Peringer et al. [PMRB01]. The algorithm (HINTA) transforms an instance of a complex hierarchy into an instance of a simple-symmetric hierarchy in a way that preserves hierarchical dependencies. Prior to the actual linearization, HINTA decomposes an instance of a complex hierarchy into a set of primitive hierarchy instances (`phi`) which are the instances of a simple-symmetric hierarchy. The publication defines all possible variants of `phi` and a procedure transforming an arbitrary `phi` into an instance of a simple-symmetric hierarchy. The *TransformPhiToSimpleHierarchy* procedure is a basic step in the iterative *HINTA* algorithm. The algorithm assumes that there exists a binary relationship *isPreferred* which is a total order on a set of the instances of simple-symmetric hierarchies forming the instance of the complex hierarchy. The ordering of the hierarchies determines the structure of the outcome hierarchy generated by *HINTA*. Taking as input an instance of a complex hierarchy consisting of an arbitrary number n, where $n \geq 2$, of the instances of simple-symmetric hierarchies, the algorithm generates a single instance of a simple-symmetric hierarchy. A high-level pseudo code of the HINTA algorithm is presented below.

```
HINTA(H):
S := stack of hierarchies order by isPreferred relationship;

/* iterate over simple hierarchies */
while numOfElements(S)>1 {
      H0 := pop(S);           /* first most preferred hierarchy */
      H1 := pop(S);           /* second most preferred hierarchy */
      H := sum(H0,H1);        /* union of two subsequent hierarchies */
      phis := split(H);       /* split H into an ordered set of primitive
                              hierarchy instances */

      /* iterate over primitive hierarchy instances */
      for j:=1 to numOfElements(phis) {

            /* transform each phi into a simple-symmetric hierarchy */
            phis[j] := TransformPhiToSimpleHierarchy(phis[j]);
      }

            H := merge(phis);       /* merge into a single hierarchy */
            push(S, H);             /* store merged hierarchy on stack */
}

HINTA := pop(S);              /* return remaining hierarchy from stack */
```

Example 4.6. Let $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ be the conceptual model of a multiple hierarchy defined by a union of two simple-symmetric hierarchies $\mathbf{M}^0 = (\mathbf{E}^0, \mathbf{R}^0)$ and $\mathbf{M}^1 = (\mathbf{E}^1, \mathbf{R}^1)$, where $E_0 <_{\mathbf{R}^0} E_1^0 <_{\mathbf{R}^0} E_2^0$ and $E_0 <_{\mathbf{R}^1} E_1^1 <_{\mathbf{R}^1} E_2^1$, such that $\mathbf{E}^0 \cap \mathbf{E}^1 = E_0$. Let $DB^0 = (I^0, r^0)$ and $DB^1 = (I^1, r^1)$ be the corresponding instance graphs and let $I_{E_0} = \{e_0\}$, $I_{E_1^0} = \{e_1^0\}$, $I_{E_2^0} = \{e_2^0\}$, $I_{E_1^1} = \{e_1^1\}$, $I_{E_2^1} = \{e_2^1\}$ be the sets of instances, such that $e_0 <_{r^0} e_1^0 <_{r^0} e_2^0$ and $e_0 <_{r^1} e_1^1 <_{r^1} e_2^1$. Assuming $DB^0 = (I^0, r^0)$ is the preferred hierarchy instance, the graphical interpretation of linearization performed by HINTA is represented on Figure 4.18.

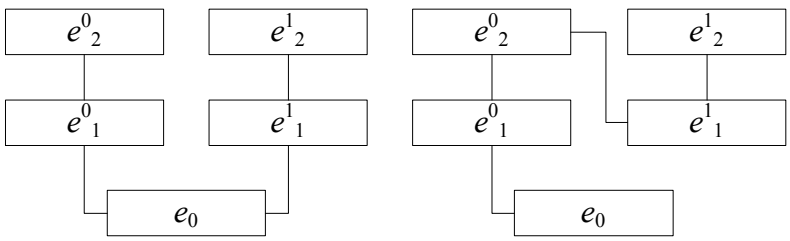


Fig. 4.18 The HINTA transformation of the instances of two component hierarchies.

One of the main assumptions of HINTA is the prioritisation of primitive hierarchy instances which determines a structure of the outcome hierarchy instance. A detailed analysis of the algorithm indicates that the prioritisation of individual levels, as opposed to the primitive hierarchy instances as a whole, makes the linearization of the complex

hierarchy more flexible. The proposed modification of HINTA algorithm improves the linearization by prioritizing individual levels prior to the actual transformation. The level-by-level prioritisation takes into account the properties of spatial objects corresponding to the levels of the component hierarchies in order to create the most adequate ordering. The properties includes the semantics and granularity of spatial objects for the proximity-preserving positioning of the spatial objects in the outcome hierarchy instance.

For example, assuming levels E_1^0 and E_1^1 as well as levels E_2^0 and E_2^1 of the complex hierarchy $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$ from Example 4.6 refer to similar concepts, e.g. similar spatial objects, the transformation represented on Figure 4.18 may not be adequate. Since HINTA does not consider the semantics of the individual levels of the complex hierarchy $\mathbf{M}_U = (\mathbf{E}_U, \mathbf{R}_U)$, interleaving the levels may lead to a better linearization. Following Example 4.6, let $\mathbf{M}^0 = (\mathbf{E}^0, \mathbf{R}^0)$ be an administrative hierarchy of a country, e.g. $District <_{\mathbf{R}^0} City <_{\mathbf{R}^0} State$, and let $\mathbf{M}^1 = (\mathbf{E}^1, \mathbf{R}^1)$ be another geographical hierarchy of the country, e.g. $District <_{\mathbf{R}^1} County <_{\mathbf{R}^1} Region$. Although the corresponding levels of the component hierarchies are not identical, the levels refer to the spatial objects of similar semantics or granularity. Taking into account the semantics, the surrogate hierarchy $\mathbf{M}^{\#} = (\mathbf{E}^{\#}, \mathbf{R}^{\#})$ resulting from the interleaving linearization of the component hierarchies and providing better clustering than the hierarchy produced by HINTA is defined as $District <_{\mathbf{R}^{\#}} City <_{\mathbf{R}^{\#}} County <_{\mathbf{R}^{\#}} State <_{\mathbf{R}^{\#}} Region$.

4.4.2.2.2 Interdependencies between component hierarchies

In some cases, more than one hierarchy of a dimension is required for clustering, e.g. a dimension table referenced by several fact tables. The physical schema of the data model for SA enables an independent encoding of component hierarchies resulting in a distinct compound surrogate for every hierarchy. Although a number of clustering hierarchies is restricted due to the properties of the clustering index, interdependences between component hierarchies enable an efficient evaluation of queries restricting the attributes of the non-clustering hierarchies. The interdependencies concern the levels which are shared between the component hierarchies as well as the correlations in the data distributions of attributes belonging to different component hierarchies. Because of the fact that the interdependencies may also concern non-hierarchical attributes (e.g. coordinates), the

evaluation of queries accessing the non-clustering hierarchical and non-hierarchical attributes is discussed collectively in Section 4.4.2.2.

4.4.2.2 Physical schema of spatial dimension

The physical schema of a denormalised spatial dimension modelling a complex spatial hierarchy is represented in the proposed data model for SA as:

$$\mathbf{D} = \{h_0, x_0, y_0, g_0, f_0, ..., h_l, x_l, y_l, g_l, f_l, ..., f_k, h_0^{\#}, ..., h_1^{\#}, cs_0, ..., cs_n\}$$

where h_i and $h_i^{\#}$, for i = 0 , … , l, are the levels of the original and surrogate hierarchy consisting of n component hierarchies. Attributes x_0 and y_0 are the coordinates and g_0 is a geometry of the leaf-level spatial objects. The dimension **D** contains the proximity-preserving compound surrogates cs_i, for i = 0 , … , n, defined for all component hierarchies and cs_0 is an encoding of the clustering hierarchy. The clustering of the leaf-level spatial objects is based on the Z-curve, so that $ord_m^{\#}(h_0) = Z(x_0, y_0)$, whereas the surrogates of the other attributes of the component hierarchies are the results of the proximity-preserving $ord_m^{\#}$ function. In order to improve the effectiveness of maintenance operations (e.g. computation and lookups of surrogates) and data access in the dimension **D**, the following B-tree indexes are defined:

1. clustering index `PK_indx(h0)` defined on the primary key attribute,
2. secondary index `CS_indx(csi)` defined for every compound surrogate,
3. composite secondary index `HCS_indx(hl, … , h0, csi)` defined for all levels of a component hierarchy and a compound surrogate encoding the hierarchy,
4. depending on the analysis of query workloads and selectivity of the hierarchical attributes (see Section 4.4.1), secondary indexes `H_indx(hi)` are defined on the attributes.

The physical model of the spatial dimension is based on the proximity-preserving encoding of the spatial hierarchy as well as indexes `PK_indx`, `CS_indx`, `HCS_indx` , `H_indx` improving the performance of queries accessing the hierarchical attributes. Because of interdependencies between the non-clustering and clustering component hierarchies,

queries accessing the non-clustering attributes can also benefit from the proximity-preserving hierarchical organisation of the spatial dimension. The interdependencies concern the levels which are shared between the component hierarchies as well as correlations between the data distributions in attributes belonging to the different component hierarchies. The compound surrogate of the clustering hierarchy is obtained in a concatenation of surrogates corresponding to the levels of the hierarchy. As a result, queries restricting the levels of component hierarchies which are shared with the clustering hierarchy are rewritten to take advantage of the clustering compound surrogates.

In the case of the non-shared and non-clustering attributes, the data access performance improvement results from the correlation between data distributions of the non-clustering and clustering attributes. Processing queries restricting the non-clustering attributes is initiated by index access `H_indx(hi)` retrieving identifiers of tuples satisfying criteria of the restriction[6]. In the proposed approach, the locality of query restrictions and the correlation between the data distributions of the restricted non-clustering attributes and the corresponding clustering attributes is assumed. The locality of data references and the proximity-preserving property of the proposed physical data model guarantee that spatial objects selected during index access `H_indx(hi)` are likely to be clustered. As a result, the tuples which satisfy query criteria, though restricting non-clustering attributes, are also likely to be clustered on a disk drive which allows to rewrite the query so that the random access retrieval of a single tuple is transformed to continuous scans.

Similarly, queries restricting the coordinates of spatial objects can be rewritten to take advantage of the proximity-preserving data organisation provided by the physical data model. Since restrictions on the coordinates are typically local in SA (e.g. range query), the restrictions are consistent with the hierarchical clustering defined by compound surrogates encoding the spatial hierarchy. The consistency is additionally intensified when leaf-level spatial objects are clustered using Z-ordering of the coordinates, i.e. $ord_m^{\#}(h_0) = Z(x_0, y_0)$. As a result, spatial predicates referencing near spatial objects are likely to reference clustered data pages containing the objects. Figure 4.19 represents the graphical interpretation of data access to non-clustering attributes which data distribution correlates with data distribution of the clustering attribute.

[6] Full table scan may provide better performance for low-selectivity restrictions.

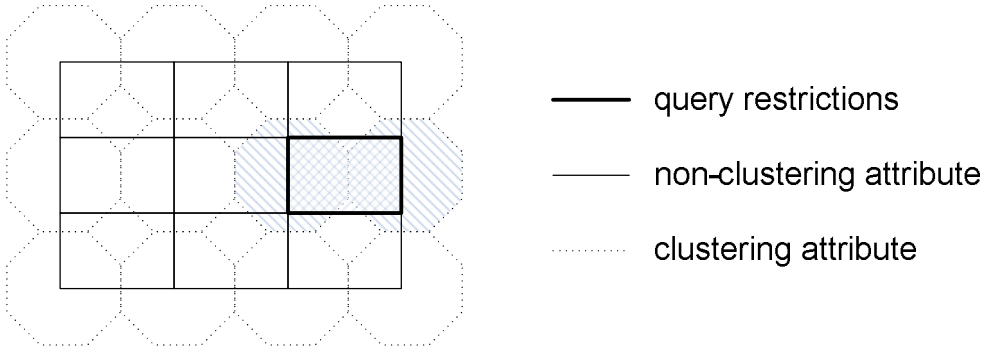


Fig. 4.19 The data access for non-clustering attributes.

In order to improve the evaluation of complex spatial predicates, additional indexes are defined in the proposed physical model. A single UB-tree index `XY_indx(x,y)` is created for coordinates represented in the data model for SA as two numerical attributes. A single R-tree index `R_indx(gi)` is created for geometries to efficiently handle spatial queries. Data access to the complex spatial attributes may be additionally improved with dedicated data structures which are designed specifically for SA, e.g. indexed proposed by Egshar et al. [EGKS99] for NN operations in form required by SA algorithm presented in the publication.

4.4.3 Physical schema of fact table

The logical schema of the data model for SA consists of a fact table and several dimension tables including a spatial dimension modelling a hierarchy of spatial objects (see Section 4.3.1). Since the spatial dimension is referenced by the fact table, the geo-locations of spatial objects determine where a given fact took place or where a given object to which a fact refers is located. As a result, both spatial objects in the spatial dimension as well as facts (measures) in the fact table can be undoubtedly geographically positioned. The conclusion implies that the proximity-preserving data organisation in the spatially-referenced fact table is feasible. Clustering the fact table with respect to the spatial dimension guarantees that tulpes in the fact table are physically located in the distance proportional to the distance between tuples in the referencing spatial dimension.

The physical schema of the data model for SA is a set of dimensions $\Omega = \{\mathbf{D}^0,...,\mathbf{D}^d\}$, where at least one of the dimensions is a spatial dimension as described in Section 4.4.2, and a fact table *FT* referencing the dimensions. Each dimensions table $\mathbf{D}^i$, for $i = 0, \ldots, d$, is hierarchically clustered by a compound surrogate encoding a hierarchy. In particular, the spatial dimension is hierarchically clustered by the proximity-preserving compound surrogate (see Section 4.4.2.1). All primary keys h_0^i of dimensions $\mathbf{D}^i$, for $i = 0, \ldots, d$, are referenced by the fact table defining the multidimensional data space of measures m_i, for $i = 0, \ldots, n$. Since each of the primary keys h_0^i is a placeholder in the corresponding compound surrogate cs^i, the compound surrogates of the dimensions which are referenced by the fact table are adequate to uniquely identify measures in the fact table. A reference surrogate rcs^i is thus an attribute in the fact table *FT* which is logically connected to the compound surrogate cs^i of the dimension $\mathbf{D}^i$. As a result, the physical schema of the fact table is defined as $FT = \{h_0^0,...,h_0^d, m_0,...,m_n, rcs^0,...,rcs^d\}$, where rcs^i, for $i = 0, \ldots, d$, is a reference compound surrogate of the dimension $\mathbf{D}^i$. Figure 4.20 represents relations between the hierarchy of the dimension $\mathbf{D}^0$ and the compound surrogate cs^0 encoding the hierarchy as well as the reference compound surrogate rcs^0 in the fact table *FT* corresponding to the compound surrogate cs^0 of the dimension $\mathbf{D}^0$.

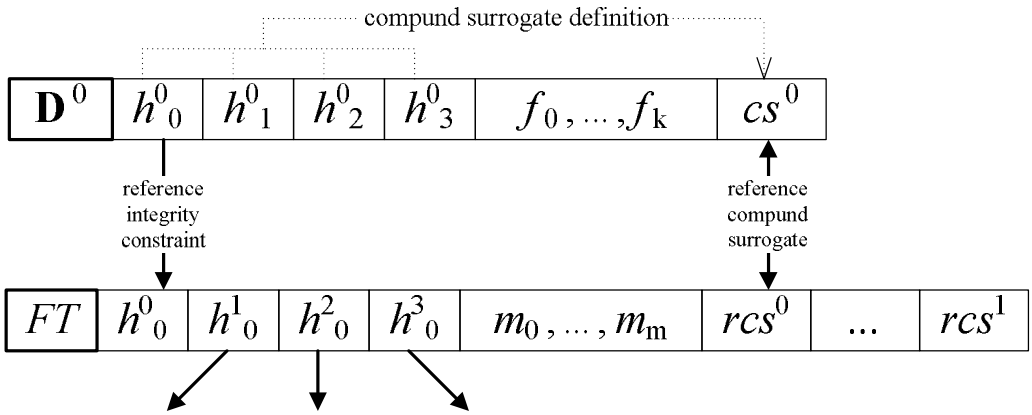


Fig. 4.20 The relationships between the hierarchy, the compound surrogate and the reference compound surrogate.

The inclusion of the reference compound surrogates in the fact table enables the multidimensional hierarchical clustering of the fact table, which is realised by the UB-Tree index defined over the reference compound surrogates (MHC/HI). The adaptation of

MHC/HI to the data model for SA extends the proximity-preserving data organisation in dimensions onto the fact table so that tuples in the fact table are located on a disk drive in a distance proportional to the distance between tuples in the corresponding dimensions. Figure 4.21 represents the graphical interpretation of the multidimensional hierarchical clustering of the fact table. Clustering in the dimension tables is represented by subscripted letters, whereas clustering in the fact table is represented by tuples containing the subscripted letters, such that the subscripts in every two adjacent letters differ by one.

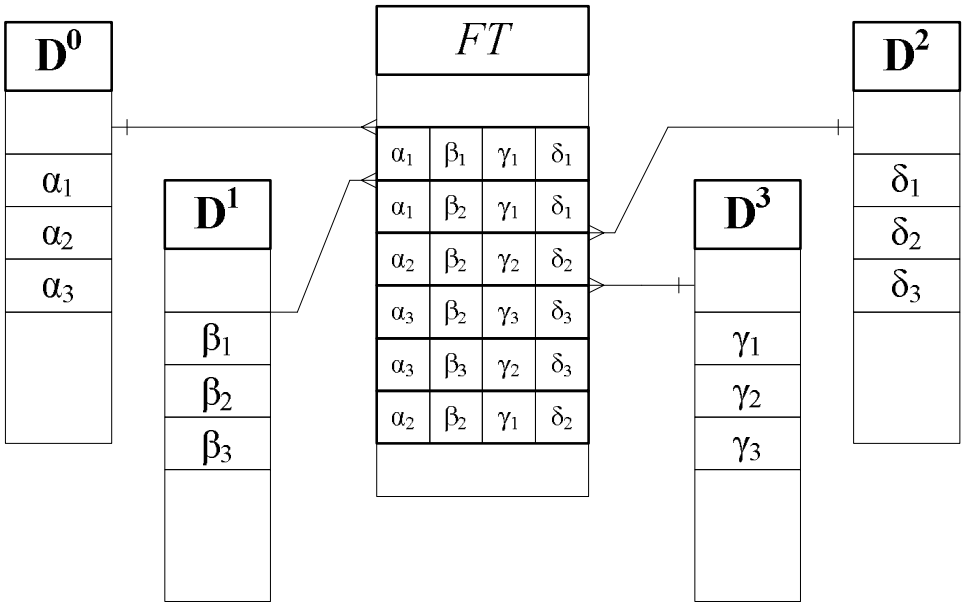


Fig. 4.21 The graphical interpretation of multidimensional hierarchical clustering.

The proposed physical model of the fact table improves the performance of fact tables data access with respect to the dimensions. The multidimensional hierarchical clustering guarantees that the selectivity of restrictions on dimensional attributes is transfered proportionally in the fact table. The reference compound surrogates enable the efficient evaluation of queries joining the dimensions with the fact table by transforming the joins into restrictions on the reference compound surrogates. A detailed discussion of query processing algorithms for a hierarchically clustered fact table is presented in Chapter V.

The current implementation of MHC/HI in the Transbase DBMS supports also the multidimensional hierarchical clustering of a fact table on the basis of a multiple hierarchy of a single dimension. For example, let the physical schema of the data model for SA consist of dimensions $\mathbf{D}^0, \mathbf{D}^1, \mathbf{D}^2$ and a fact table FT defined as follows:

$$\mathbf{D}^0 = \{h_0^0, h_1^0, h_2^0, f_0^0, f_1^0, cs_0^0\} \ , \ \mathbf{D}^1 = \{h_0^1, h_1^1, h_2^1, f_0^1, f_1^1, cs_0^1\} \ ,$$

$$\mathbf{D}^2 = \{h_0^2, h_1^2, h_2^2, h_3^2, x_0^2, y_0^2, g_1^2, g_2^2, f_0^2, f_1^2, cs_0^2, cs_1^2\}$$

$$FT = \{h_0^0, h_0^1, h_0^2, m_0, m_1, rcs_0^0, rcs_0^1, rcs_0^2, rcs_1^2\}$$

The dimensions $\mathbf{D}^0$ and $\mathbf{D}^1$ are ordinary dimensions (e.g. time dimension), containing two simple-symmetric hierarchies which are encoded by compound surrogates cs_0^0 and cs_0^1. The dimension $\mathbf{D}^2$ is a spatial dimension containing a complex-spatial hierarchy defined by two component hierarchies which each are encoded by the compound surrogates cs_0^2 and cs_1^2. The fact able *FT* references the primary keys h_0^0, h_0^1, h_0^2 and the corresponding compound surrogates rcs_0^0, rcs_0^1, rcs_0^2, rcs_1^2. The reference compound surrogates are used for the clustering of the fact table. A DDL SQL statement of the physical schema is following:

```
create table D^0 (
      h^0_0 char(*) not null,
      h^0_1 char(*) not null,
      h^0_2 char(*) not null,
      f^0_0 integer,
      f^0_1 char(*),
      SURROGATE cs^0_0 COMPOUND (h^0_2 SIBLINGS 10, h^0_1 SIBLINGS 50,
                                 h^0_0 SIBLINGS 1000)
) key is h^0_0 ;

create table D^1 (
      h^1_0 char(*) not null,
      h^1_1 char(*) not null,
      h^1_2 char(*) not null,
      f^1_0 char(*) integer,
      f^1_1 char(*),
      SURROGATE cs^1_0 COMPOUND (h^1_2 SIBLINGS 5, h^1_1 SIBLINGS 20,
                                 H^1_0 SIBLINGS 100)
) key is h^1_0 ;

create table D^2 (
      h^2_0 char(*) not null,
      h^2_1 char(*) not null,
      h^2_2 char(*) not null,
      h^2_3 char(*) not null,
      x^2_0 integer,
      y^2_0 integer,
      g^2_1 geometry,
      g^2_2 geometry,
      f^2_0 integer,
      f^2_1 char(*),
      SURROGATE cs^2_0 COMPOUND (h^2_2 SIBLINGS 10, h^2_1 SIBLINGS 100,
                                 H^1_0 SIBLINGS 500),
      SURROGATE cs^2_1 COMPOUND (h^2_3 SIBLINGS 30, h^2_1 SIBLINGS 100,
                                 H^2_0 SIBLINGS 500)
) key is cs^2_0 ;
```

```
create table ft (
      h⁰₀ char(*) references product (D⁰),
      h¹₀ char(*) references product (D¹),
      h²₀ char(*) references product (D²),
      m₀ integer,
      m₁ integer,
      SURROGATE rcs⁰₀ FOR D⁰ REFERENCES D⁰(cs⁰₀),
      SURROGATE rcs¹₀ FOR D¹ REFERENCES D¹(cs¹₀),
      SURROGATE rcs²₀ FOR D² REFERENCES D²(cs²₀),
      SURROGATE rcs²₁ FOR D² REFERENCES D²(cs²₁)
) hckey is rcs⁰₀ , rcs¹₀ , rcs²₀ , rcs²₁ ;
```

The dimensions $\mathbf{D}^0, \mathbf{D}^1, \mathbf{D}^2$ and the fact table *FT* are clustered by indexes (`PK_indx`) defined on the the primary keys. The compound surrogates (`CS_indx`) and hierarchical attributes (`HCS_indx`) of the dimensions are additionally indexed as described in Section 4.4.2.2. Depending on a query workload defined by SA, the selectivity of the queries and the data distributions of restricted attributes, the DDL statement can be additionally enhanced with definitions of the following indexes:

```
/* UB-tree index on coordinates */
CREATE INDEX D²_XY_indx ON D²(x²₀, y²₀, h²₀) HCKEY NOT UNIQUE IS x²₀, y²₀;

/* R-tree index on geometries */
CREATE INDEX D²_R_indx ON D²(g²₀, h²₀);

/* B-tree index on component non-clustering hierarchy*/
CREATE INDEX D²_H_indx ON D²(h³₀);
```

4.4.4 Partitioning

Data partitioning consists in the physical splitting of a large datasets into smaller partitions in order to facilitate parallel query processing and improve the performance of I/O transfers and maintenance operations [Vit01, EN00]. Since the data model for SA is based on the relational data representation, all standard partitioning methods available for the representation can also be applied to the physical schema of the data model.

For example, partitioning by columns allows to separate geometries from other attributes in dimensions and fact tables. Otherwise, when all attributes are physically stored in the same table, complex attributes (e.g. geometries) increase storage overhead and distance between tuples on disk drives. Assuming $\mathbf{D} = \{a_0,...,a_m,g\}$ is a dimension table containing attributes a_i, for $i = 0$, … , m, and a geometry g, the distance between tuples equals a size of a physical tuple and can be computed as:

$$SizeOf(t) = TupleOverhead + \sum_{i=0..m} SizeOf(a_i) + SizeOf(g)$$

The size of a physical tuple can be significantly reduced when the geometry *g* is relocated to another table and referenced to the dimension **D** with an identifier g^*. The modified schema consists of a dimension $\mathbf{D} = \{a_0,...,a_m,g^*\}$, a table $\mathbf{G} = \{g^*,g\}$ and referential integrity constraint $\mathbf{D}(g^*) \rightarrow \mathbf{G}(g^*)$. Since a size of the identifier g^* is smaller than a size of the geometry *g*, the capacity of a data page and consequently the effectiveness of I/O transfers are increased. The modification of the logical schema necessitates, however, a modification of queries referencing the geometry *g*, e.g. by including a join predicate between the dimension **D** and the table **G**. Alternatively, the physical partitioning of the dimension **D** by the geometry *g* does not require any modifications of the logical schema. The partitioning is automatically maintained by a DBMS which improves the execution plans of queries referencing the partitioned data.

Besides the standard partitioning, the data model for SA also provides two additional partitioning methods, i.e. hierarchical and Z-partitioning. Both methods are especially practical for SA because the methods are consistent with the character of SA and with the multidimensional hierarchical data organisation. The physical schema of the data model for SA is based on compound surrogates obtained as a result of the proximity-preserving encoding of a spatial hierarchy (see Section 4.4.2.1). Properties of the encoding allow to identify all sub-trees of the spatial hierarchy in the numerical interpretation of the compound surrogates. For example, assuming a physical schema consists of a dimension $\mathbf{D} = \{h_0,...,h_l,cs\}$, the compound surrogate *cs* encoding the hierarchy maps all sub-trees of the hierarchy on a level h_j, where $0 < j < l$, as a set of ranges $cs(h_j) \mid 000_2 \div 111_2$ obtained for all distinct values of h_j. Since the ranges on the compound surrogates have a constant prefix, the volatility of the suffix is known, the only variable in the compound surrogate is a value of the $ord_m^{\#}(h_j)$ function:

$$cs(h_j) \mid 000_2 \div 111_2 = cs(h_{j-1}) \mid ord_m^{\#}(h_j) \mid 000_2 \div 111_2$$

Since the conventional (range, list, hash) partitioning of the dimension **D** is not proximity-preserving, the hierarchical and clustering partitioning of the dimension can be based on the compound surrogates. Partitions, corresponding to the sub-trees of a hierarchy, can be defined as a list or ranges of instances which map bijectively onto the compound

surrogates. Additionally, since tuples in a fact table are referenced by the compound surrogates, the hierarchical partitioning of the dimensions can be extended onto the fact table to create equi-partitioning. The equi-partitioning increases the independence of maintenance operations, improves the performance of parallel query processing, large data sorts and joins. Figure 4.22 represents the dimension **D** containing a hierarchy equi-partitioned with a fact table *FT* based on the compound surrogates encoding the hierarchy.

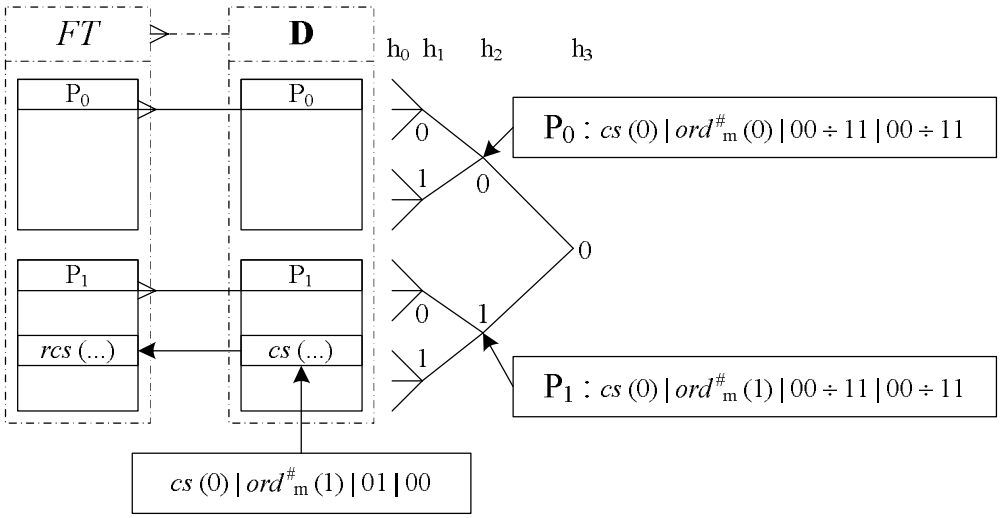


Fig. 4.22 The hierarchical equi-partitioning of a dimension and a fact table.

Another data partitioning method feasible in the data model for SA is based on the parameterisation of multidimensional space realised by a finite-order space-filling curve, e.g. the Z-curve. The main advantage of the Z-partitioning is the proximity-preserving partitioning defined by multiple attributes. The conventional partitioning methods operating on multiple attributes (composite partitioning) create either compound or hash data distributions, e.g. hash sub-partitions within range partitions. Contrary to the conventional partitioning, the Z-partitioning maintains the symmetrical clustering within partitions thus providing better support for multidimensional data processing. The proposed method partitions tuples by rows by taking into account the values of partitioning attributes. Since the Z-ordering is already available in the proposed data model, partitioning by the ordering can be directly applied. For example, let a physical schema consist of a spatial dimension $\mathbf{D} = \{h_0,...,h_l,x,y,cs\}$, where *x* and *y* are the coordinates of a leaf-level spatial object and a compound surrogate *cs* encoding the hierarchy. As described

in Section 2.5.2, the parameterisation of multidimensional data space defined by the Z-curve is limited for every point belonging to the curve. As a result, given the scope and resolution of the multidimensional data space (domains of partitioning attributes), parameterisation (partitioning) of the space can be precisely defined. Following the above example, assuming a domain of x and y corresponds to 3-bit integer data type, the coordinates partition the dimension **D** by dividing data space into n equal parts covered by $^1/_n$ of the Z-curve. The equal Z-partitioning is adequate for normal data distributions but does not guarantee optimal I/O balancing in the case of skewed data distributions. The unsymmetrical data distributions can be partitioned using the arbitrary selected ranges of Z-values corresponding to physical partitions. As opposed to the equal Z-partitioning which partitions the data space, the adaptive Z-partitioning partitions data itself and consequently scales better on both sparse and dense data distributions. Figure 4.23 represents (a) equal (4 partitions) and (b) adaptive (5 partitions) Z-partitioning.

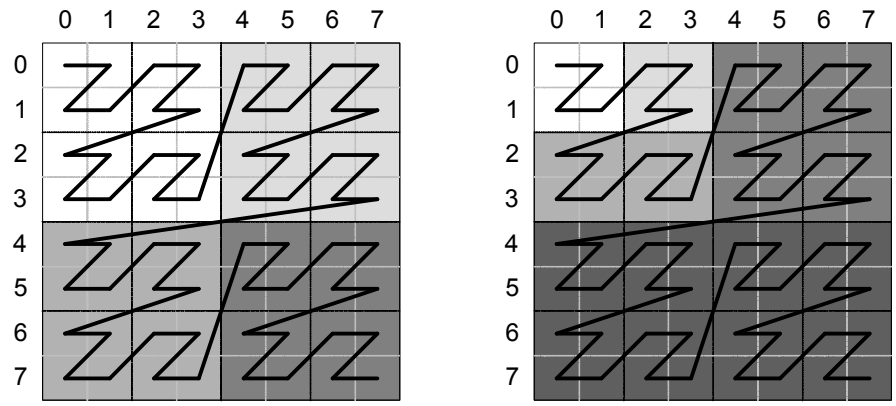


Fig. 4.23 The Z-partitioning (a) equal and (b) adaptive.

The Z-partitions represented in Figure 4.23 parameterise the multidimensional data space according to the rule $P_z(x_1 \div x_2, y_1 \div y_2) \rightarrow Z(x_1, y_1) \div Z(x_2, y_2)$. For example, the partition covering the area $[0,4] \times [7,7]$ corresponds to the Z-partition defined by the following Z-values $Z(0,4) \div Z(7,7)$.

Chapter 5

Query processing and performance analysis

5.1 Query processing

The motivation for the proximity-preserving data organisation results from the locality of data references which is characteristic of queries implementing SA (see Section 1.2). Query evaluation can benefit from the proposed data model by recognising the underlying data clustering and choosing access paths which improve the performance of I/O operations. Considerable performance gains can thus be attained in relation to tuple clustering which enables to reduce a number of required physical disk page accesses because one retrieved disk page usually contains many tuples of a result set. Page clustering enables to reduce a number of random access operations because consecutive data pages are retrieved in a sequential scan. Figure 5.1 represents an example of a query accessing multidimensional data space with tuple and page clustering based on the Z-curve (pages capacity is 2 tuples). The retrieval of the query result set requires only two sequential scans of six data pages.

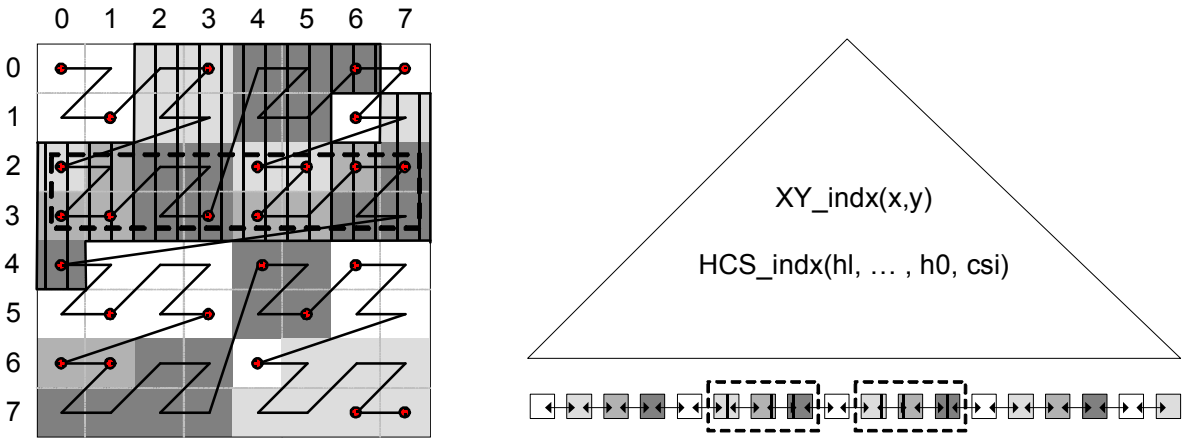


Fig. 5.1 The processing of a query accessing clustered data.

In order to fully benefit from the data model for SA, few modifications of the conventional query evaluation techniques are required. The following Sections of the Chapter discuss the modifications which concern data access for hierarchically-clustered spatial dimensions. For the sake of brevity, the discussion does not cover aspects which are also characteristic of other query evaluation techniques and which are discussed in the literature, e.g. processing of spatial queries using spatial access methods [BKSS94, GG98]. As a result, the following Sections reference the query evaluation approach for a MHC/HI-organised schema (i.e. processing of star-queries accessing a hierarchically clustered fact table [KTSPMRFEB02, RMFZEB00, RMFZEB00]) and describe improvements which are feasible in the context of the proximity-preserving hierarchical data organisation. In order to verify the adequacy of the proposed data model for the needs of SA, the performance analysis of some query workloads is presented in Section 5.2. The query workloads are not the complete implementations of SA techniques but rather the simulations of database primitives (data access patterns) characteristic of SA. This performance analysis approach enables to examine an impact of the proposed physical data organisation on the performance of individual data processing elements of SA.

5.1.1 Overview of query template

Since the logical schema of the data model for SA is similar to the typical data warehouse schema, the most complex variant of queries implementing SA and accessing the proposed data model is similar to the star-query which is most typical query pattern in data warehousing. As a result, the following Sections of the Chapter concentrate on the following query template:

```
SELECT      {D^i_k.h_j},
            {D^i_k.f_j},
            {aggr(FT.m_i)AS AM_j},
            {aggr(D^i_k.h_j)AS AH_j},
            {aggr(D^i_k.f_j)AS AF_j}

FROM        FT,
            {D^0_i} , … , {D^d_k}

WHERE       FT.d_0 = D^0.h_0
            AND …
            AND FT.d_d = D^d.h_0
            AND DPRED({D^0_i}, … , {D^d_k})
            AND LOCPRED({D^o_i})
            AND …
            AND LOCPRED({D^d_k})
            AND MPRED({FT.m_i})

GROUP BY    {D^i_k.h_j},
            {D^i_k.f_j},
            {FT.m_i}

HAVING      <having clause>
ORDER BY    <ordering fields>
```

The query template contains the following components:

I. `aggr` is a standard SQL aggregate function (MIN, MAX, SUM, COUNT, AVG).

II. `DPRED({D^0_i},… ,{D^d_k})` is a predicate depending on an actual structure of dimensions (denormalised, path- or field-normalised) because the predicate defines join criteria among the component tables `{D^n_k}` of each dimension.

III. `LOCPRED(D^i)` is a local predicate on a dimension table `D^i` producing compound surrogate intervals for fact table access. The predicate is defined as follows:

```
LOCPRED(Di) ::= PRED(Di) [ AND PRED(Di) ] …

PRED(Di) ::= Di.hj op xj | Di.fm op xm |
             Di.hj IN Sj | Di.fm IN Sm |
             Di.hj BETWEEN xj1 AND xj2 |
             Di.fm BETWEEN xm1 AND xm2

op ::= = | <> | < | <= | > | >= topology_operator

xi ::= scalar expression not containing fields of Di

Si ::= set of constant values
```

IV. `MPRED({FT.m_i})` is a predicate constraining the measure of the fact table.

The query pattern does not contain INs and ORs because such a directive can be transformed into a series of disjunctions of equality comparisons. Similarly, a BETWEEN directive can be transformed into a conjunction of two inequality comparisons. As a result, only queries with the conjunctions of predicates are considered in the following Sections.

5.1.2 Query processing for MHC/HI-organised fact table

As explained in Section 5.1.1, the query template contains a set of restrictions on dimensions joined with a fact table as well as some groupings and aggregations of measures. The main computational cost related to processing such queries concerns the evaluation of the star-join. In the case of MHC/HI, the presence of compound surrogates in the dimensions and the fact table enables transforming the star-join into a multidimensional selection followed by residual joins. An abstract plan for the star-query processing on the hierarchically clustered fact table is represented in Figure 5.2.

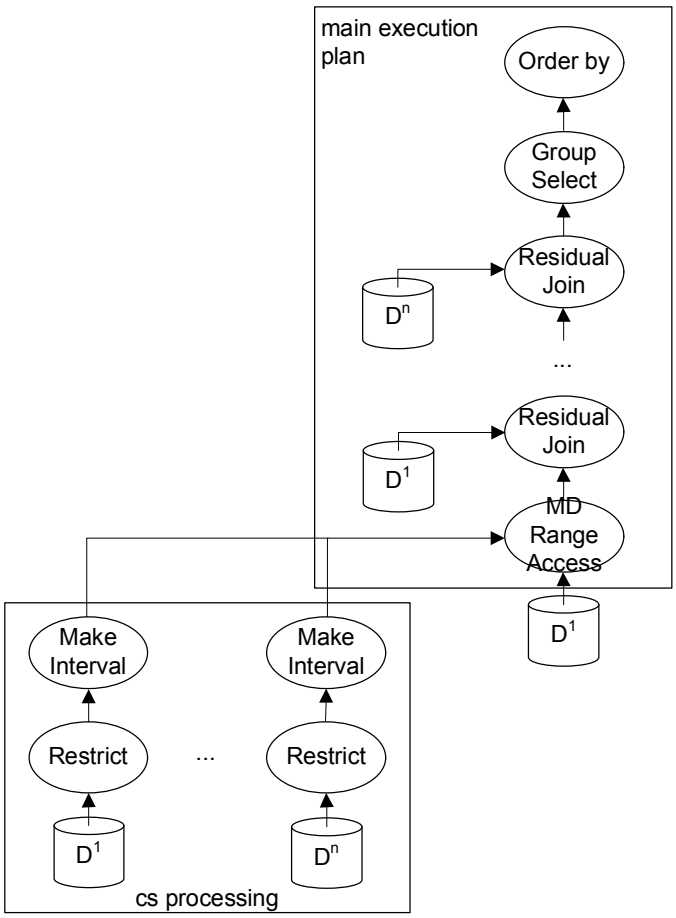


Fig. 5.2 The query execution plan for a MHC/HI-organised fact table [RMFZEB00].

A set of physical operators is used to implement the abstract plan. The choice of the physical operators depends on a query type, selectivity and possibility of pipelining operators in order to avoiding intermediate tuple materialisation. The implementation rules define the way in which the physical operators are combined to implement a particular abstract operator. The optimization of the abstract execution plan is based on a set of heuristic rules which guide the selection of the most appropriate implementation rules and transformations. In addition to the heuristic optimization, cost estimations or statistics may also be used for selecting the most adequate execution plan.

The abstract execution plan represented in Figure 5.2 can be used to answer the query template described Section 5.1.1. The plan recognises the MHC/HI-organised schema and uses compound surrogates to replace the star-join with the multidimensional range selection on the fact table. In order to perform the range selection, the query processor transforms restrictions on the dimensions into a sets of surrogate intervals. The process is performed by the `Restrict` and `Interval Make` operators in the `cs processing` phase of the plan. The surrogate intervals are subsequently passed to the `MD Range Access` operator which retrieves corresponding tuples from the fact table. The fact table tuples may then be further selected or joined with dimensions and the result is usually grouped, aggregated and sorted. The execution plan in Figure 5.2 represents the most complex possible case because not all operators may be required for executing a query that implements a particular SA technique. For example, if the output tuples of query are not required in a specific order, then the `Order` operator is not applied. Similarly, queries restricting only a subset of dimensions do not require feature or hierarchical attributes from all dimension tables. As a result, only some `Restrict`, `Interval Make` and `Residual Join` operators are used for the relevant subset of the dimension. In the simplest possible query only the `MD Range Access` operator is needed.

5.1.3 Data access for spatial dimensions

The proximity-preserving data organisation guarantees that tuples corresponding to spatial objects which are semantically local are also located in proximity on disk drives. The Section proposes data access paths which exploit the physical data organisation in order to

minimise the cost of I/O operations. The data access paths for spatial dimensions are listed in Table 5.1.

Tab. 5.1 The data access paths for spatial dimensions.

#	Data access	Attribute	Data access implementation rule
1	Point	h^0_0	Clustering index PK_indx
2	Range	h^0_0	Clustering index PK_indx
3	Point	h^0_i, i>0	Index H_indx or HCS_indx creating ranges on the compound surrogate(#2)
4	Point	h^i_0, i>0	$h^0_0=h^i_0$, for i>0 (#1)
5	Range	h^i_0, i>0	$h^0_0=h^i_0$, for i>0 (#1)
6	Point	h^i_j, i>0, j>0	if $h^i_j = h^0_j$ then (#3) else index HX and (#2) or full table scan
7	Range	h^i_j, i>0, j>0	if $h^i_j = h^0_j$ then (#5) else index HX and (#2) or full table scan
8	Point	f^i, i>0	Secondary index (#3) or full table scan
9	Range	f^i, i>0	Secondary index (#3) or full table scan
10	Point	x, y	Index XY_indx (#1)
11	Range	x, y	Index XY_indx creating ranges on the compound surrogate(#2)
12	Spatial join	G	Index R_indx creating ranges on the compound surrogate(#2)

5.2 Performance analysis

Contrary to the transactional processing, maintenance operations in the context of data analysis are usually narrowed to bulk loading. As a result, the following performance analysis does cover the maintenance operations because the operations are of minor relevance and their impact on the overall performance is typically insignificant[7]. Similarly, since the evaluation of general analytical and spatial queries have already been discussed in many publications, the performance analysis of the proposed data model was performed only in scope relevant for SA. The following Sections of the Chapter present the detailed performance analysis performed in the environment described in Annex I and using the schema instance described in Annex II. The scope of the analysis was based on two performance measures – query evaluation time for cold and hot cache as well as a number of logical and physical I/O operations – and the following measurement variables:

[7] The performance analysis of maintenance operations for the general hierarchically clustered fact tables was published in the deliverables of the EDITH project [GDW01].

1. **Query patterns (hierarchical, non-hierarchical, spatial, complex)** – query patterns which are the instances of the query template described in Section 5.1.1,
2. **Selectivity** – various restrictions in the query patterns for the spatial and time dimensions, i.e. the spatial hierarchy HDC, ESTATE, REGION, CITY and the time hierarchy DAY, FORTNIGHT, MONTH, YEAR,
3. **Database size** – two database instances containing one-year and two-year datasets,
4. **Hardware configurations** – two environments with large and small memory configurations.

The intention of the varying measurement setting was to analyse the scalability of the proposed data model. In the approach, the influence of each variable is measured in a configuration of other variable being constant.

5.2.1 Hierarchical queries

The first category of tests concerns queries with predicates involving hierarchical and feature attributes. The intention of the analysis is to analyse the performance of queries with restrictions consistent with the proximity-preserving data organisation.

The Hierarchical Query Temple 1 (HQT_1) contains only restrictions on hierarchical attributes which are included in the definitions of compound surrogates. Because of the fact that the hierarchical attributes are directly involved in the hierarchical data clustering, the analysis of results obtained for the HQT_1 query template provides the performance assessment for the queries which are fully compatible with the proposed data model. The Hierarchical Query Temple 1 is as follows:

```
SELECT SUM(ch_energy), SUM(hw_energy), COUNT(*)
FROM  heat_consumption h, hdc, day, building, customer
WHERE h.hdc_number = hdc.hdc_number AND h.day_number = day.day_number
AND   h.building_id = building.building_id AND h.customer_id =
      customer.customer_id
AND   <hierarchical restrictions>;
```

The Hierarchical Query Temple 2 (HQT_2) restricts the hierarchical attributes which are included in the definitions of compound surrogates. The template also contains the restriction of the feature attribute *building_technology* from the spatial dimension *Building*.

Because of the fact that the values of the feature attribute are not correlated with the proximity-preserving data optimization, the additional restriction leads to a set of non-overlapping intervals and consequently to a set of non-overlapping query boxes. The analysis of the HQT_2 queries provides the performance assessment for the queries which are not fully compatible with the proposed the data model. The Hierarchical Query Temple 2 is as follows:

```
SELECT SUM(ch_energy), SUM(hw_energy), COUNT(*)
FROM  heat_consumption h, hdc, day, building, customer
WHERE h.hdc_number = hdc.hdc_number AND h.day_number = day.day_number
AND   h.building_id = building.building_id AND h.customer_id =
      customer.customer_id
AND   <hierarchical restrictions> and building_technology = "wood";
```

Sections 5.2.1.1 and 5.2.1.2 represent the results obtained for the Hierarchical Query Template 1 (HQT_1) and the Hierarchical Query Template 2 (HQT_2) queries respectively.

In some of the following Figures, the results for queries with lowest selectivity restrictions are not shown because the results are significantly higher than the results obtained for queries with higher selectivity restrictions. The processing of the queries with lowest selectivity restrictions requires the retrieval of nearly entire data set (full table scans), thus making the results irrelevant in terms of the objectives of the thesis. Because of the irrelevance and the fact that including the results in the following Figures would result in the deformation of the characteristics of the remaining results, the results for the queries with lowest selectivity restrictions are intentionally skipped.

5.2.1.1 Results for HQT_1 queries

The results for the HQT_1 queries for the database one on the primary hardware configuration are represented in Figure 5.3 and Figure 5.4.

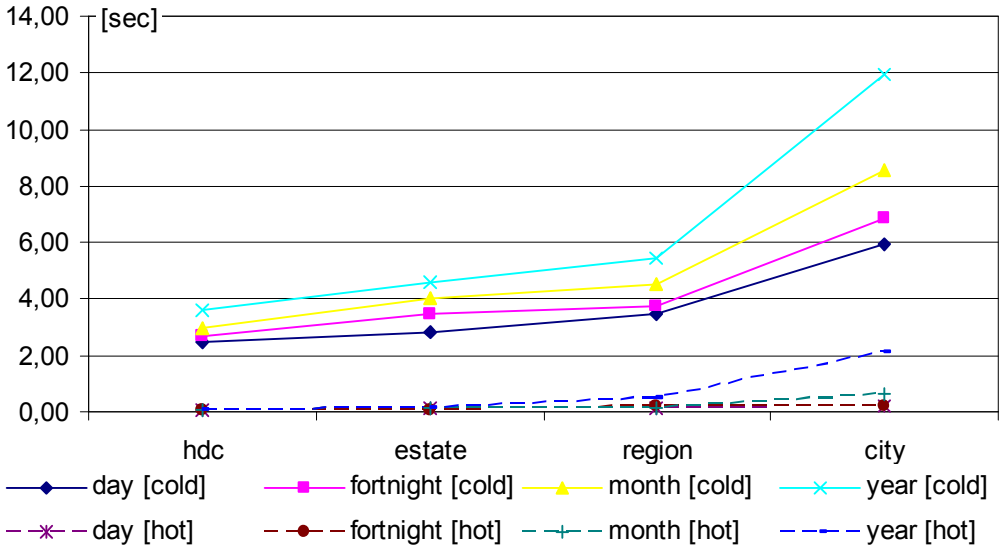


Fig. 5.3 The query evaluation time for the HQT_1 queries for the database one on the primary hardware configuration.

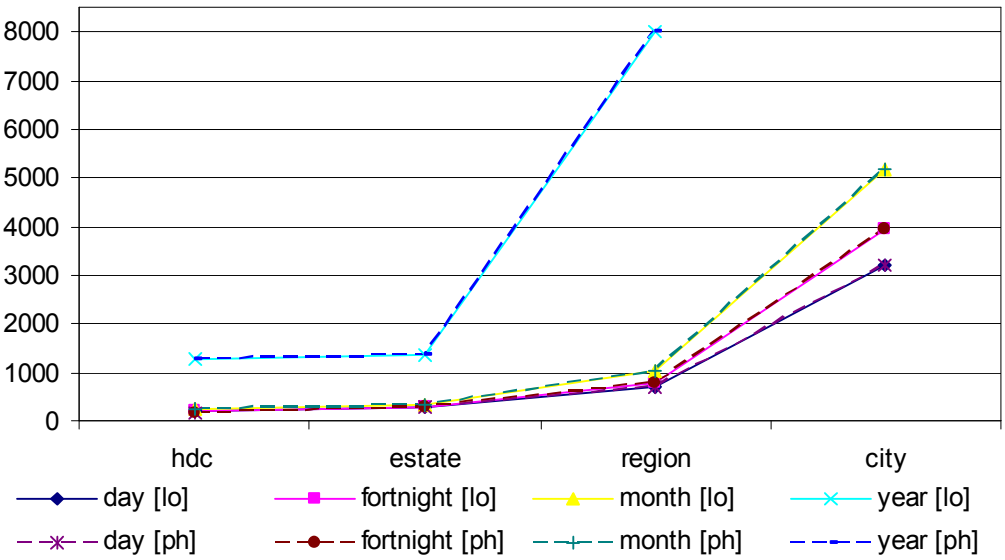


Fig. 5.4 The number of I/O operations (hot cache) for the HQT_1 queries for the database one on the primary hardware configuration.

The results for the HQT_1 queries for the database one on the secondary hardware configuration are represented in Figure 5.5 and Figure 5.6.

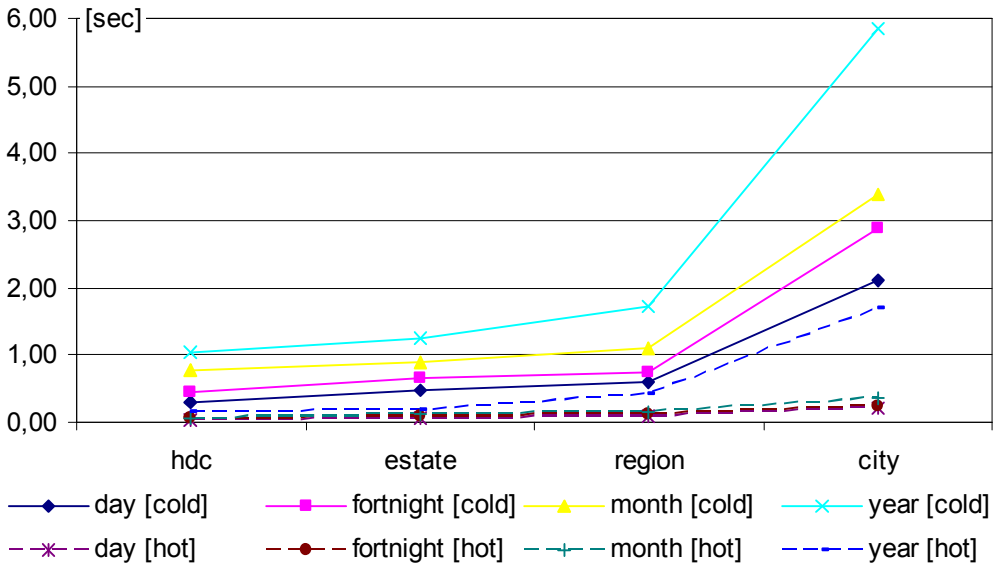


Fig. 5.5 The query evaluation time for the HQT_1 queries for the database one on the secondary hardware configuration.

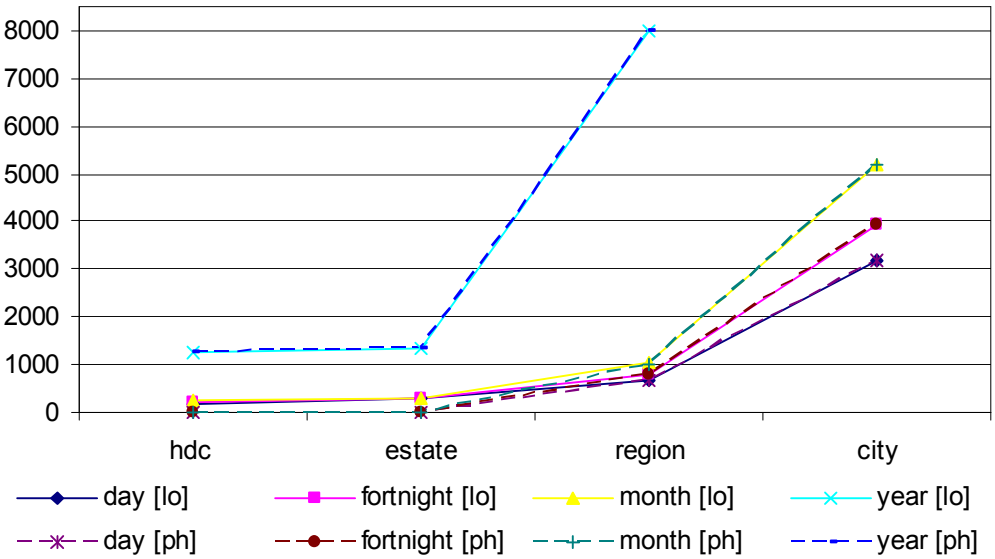


Fig. 5.6 The number of I/O operations (hot cache) for the HQT_1 queries for the database one on the secondary hardware configuration.

The results for the HQT_1 queries for the database two on the primary hardware configuration are represented in Figure 5.7 and Figure 5.8.

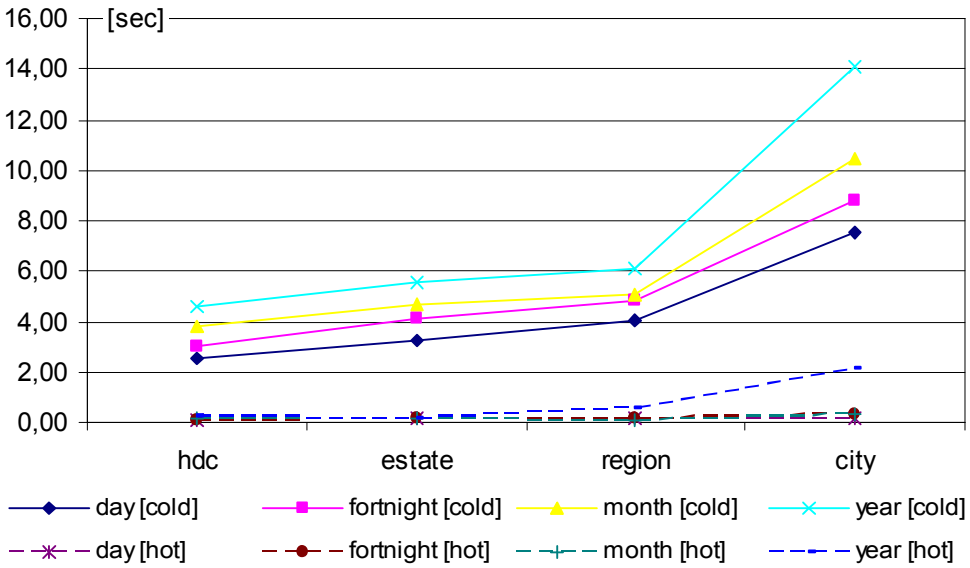


Fig. 5.7 The query evaluation time for the HQT_1 queries for the database two on the primary hardware configuration.

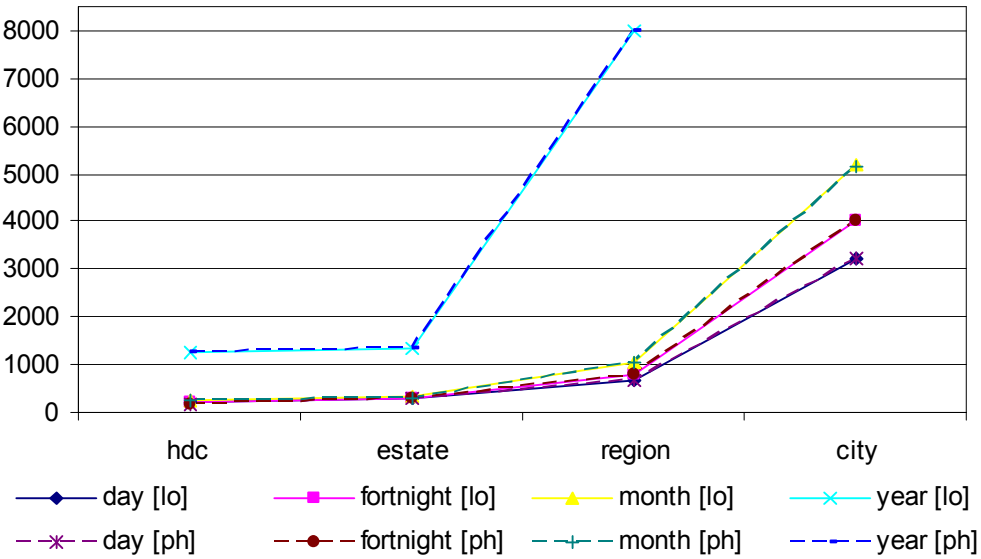


Fig. 5.8 The number of I/O operations (hot cache) for the HQT_1 queries for the database two on the primary hardware configuration.

The results for the HQT_1 queries for the database two on the secondary hardware configuration are represented in Figure 5.9 and Figure 5.10.

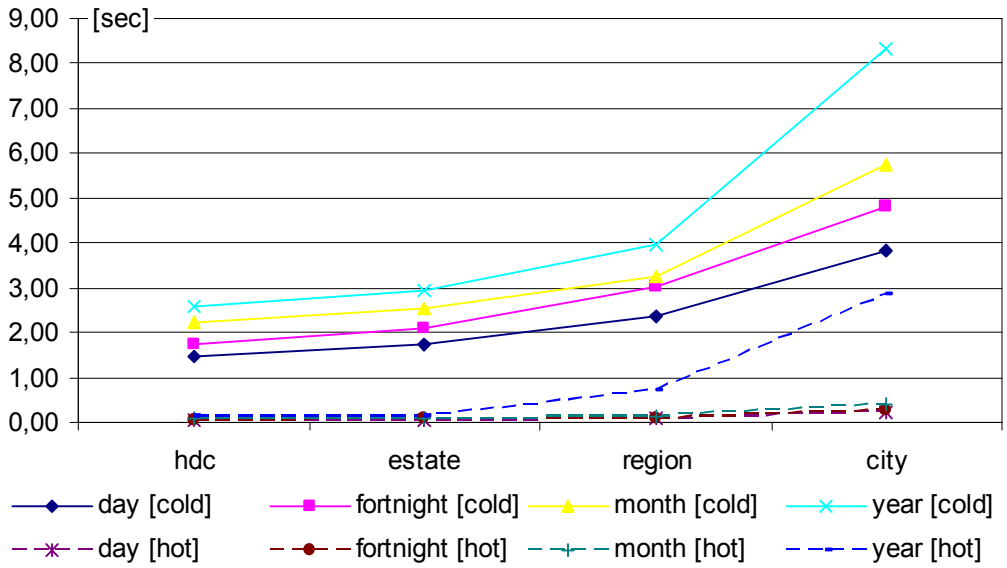


Fig. 5.9 The query evaluation time for the HQT_1 queries for the database two on the secondary hardware configuration.

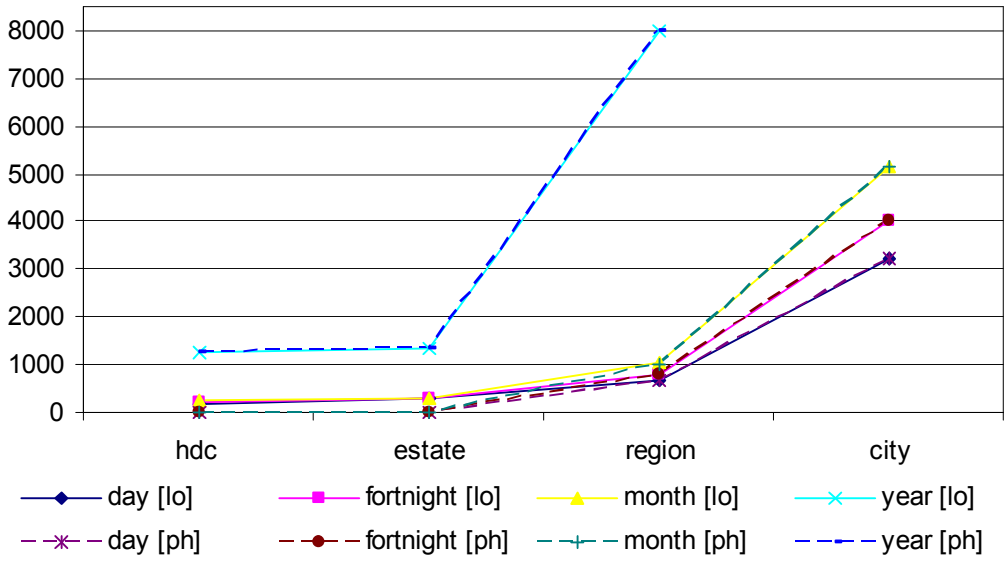


Fig. 5.10 The number of I/O operations (hot cache) for the HQT_1 queries for the database two on the secondary hardware configuration.

5.2.1.2 Results for HQT_2 queries

The results for the HQT_2 queries for the database one on the primary hardware configuration are represented in Figure 5.11 and Figure 5.12.

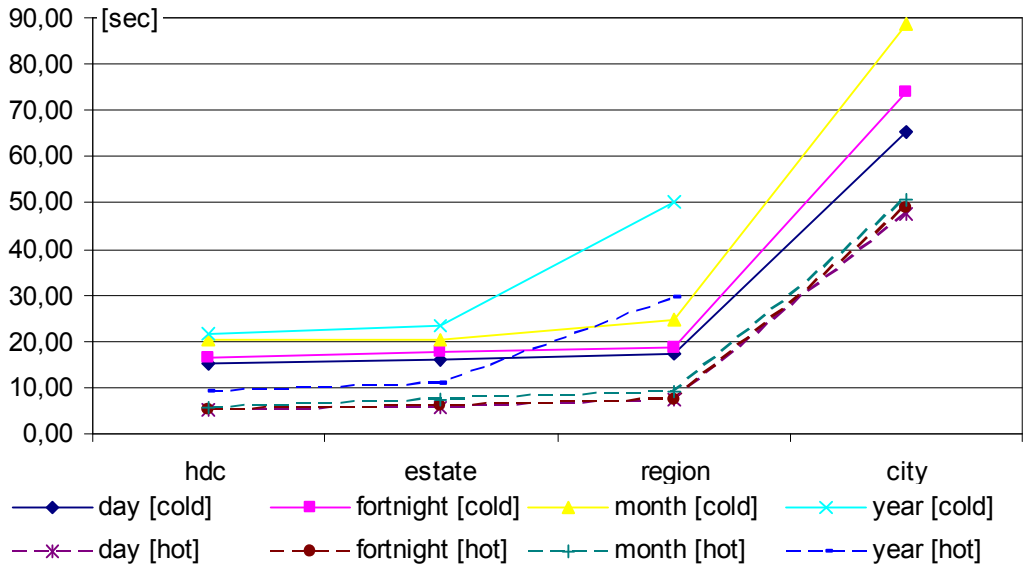


Fig. 5.11 The query evaluation time for the HQT_2 queries for the database one on the primary hardware configuration.

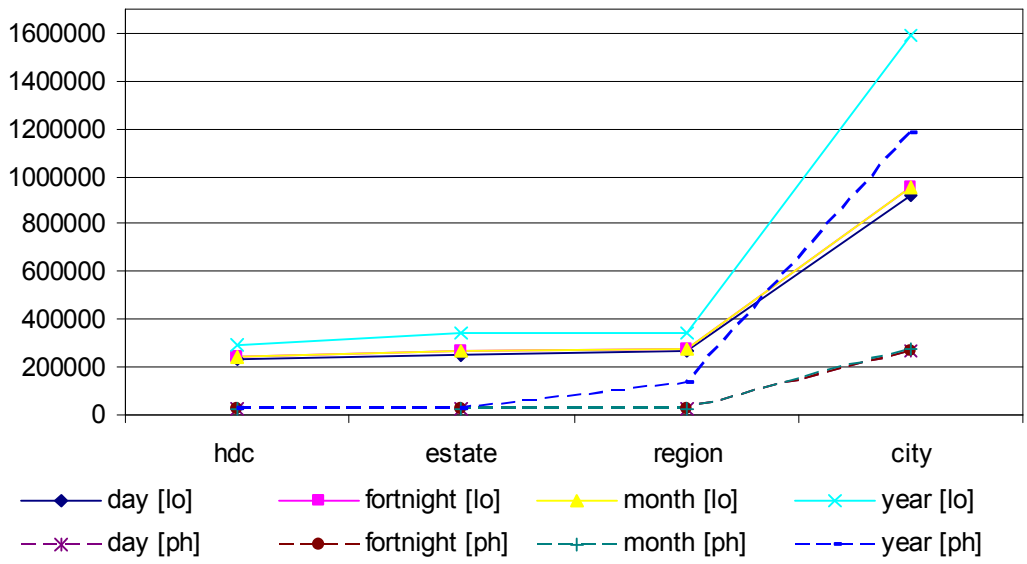


Fig. 5.12 The number of I/O operations (hot cache) for the HQT_1 queries for the database one on the primary hardware configuration.

The results for the HQT_2 queries for the database one on the secondary hardware configuration are represented in Figure 5.13 and Figure 5.14.

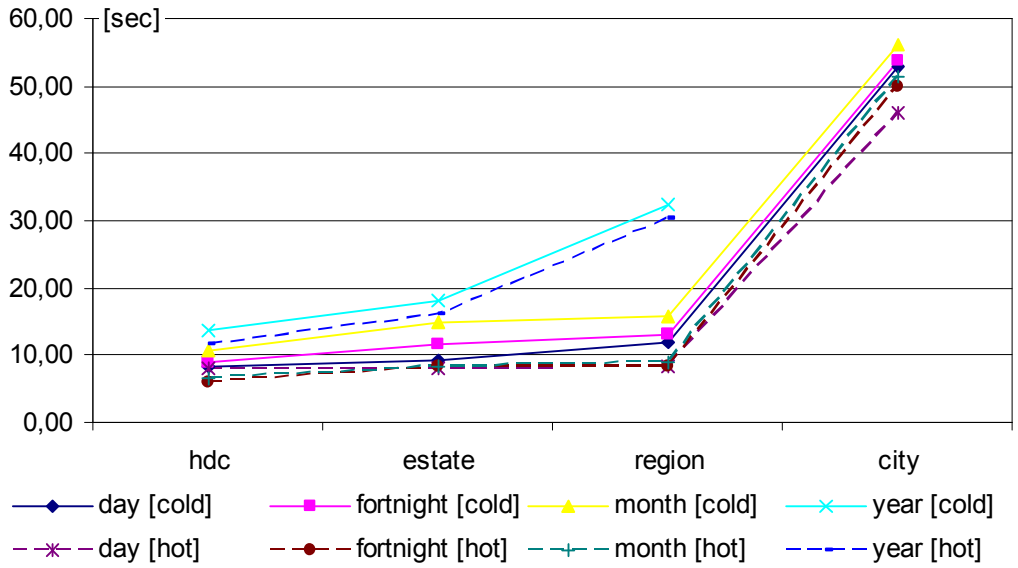


Fig. 5.13 The query evaluation time for the HQT_2 queries for the database one on the secondary hardware configuration

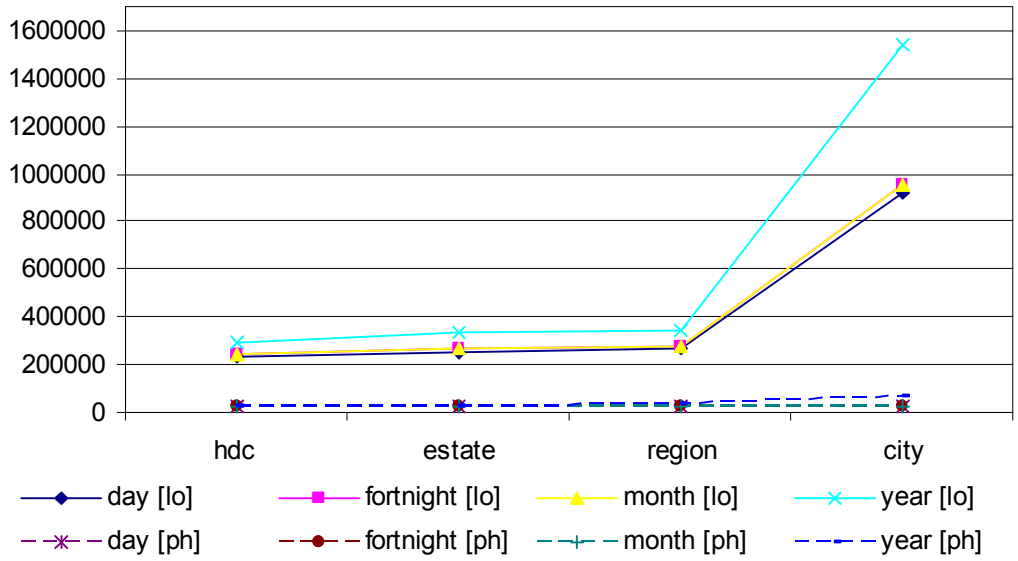


Fig. 5.14 The number of I/O operations (hot cache) for the HQT_2 queries for the database one on the secondary hardware configuration

The results for the HQT_2 queries for database two on the primary hardware configuration are represented in Figure 5.15 and Figure 5.16.

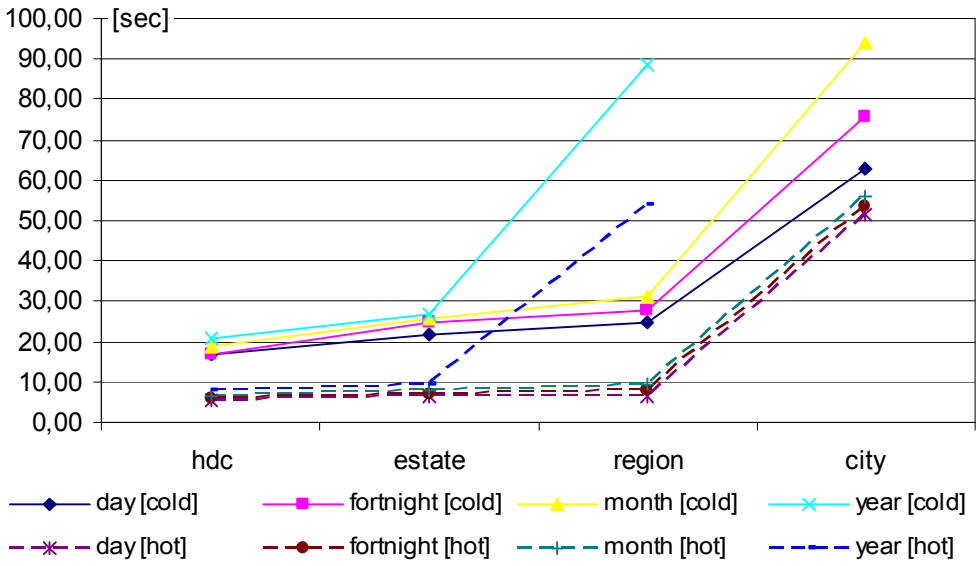


Fig. 5.15 The query evaluation time for the HQT_2 queries for the database two on the primary hardware configuration.

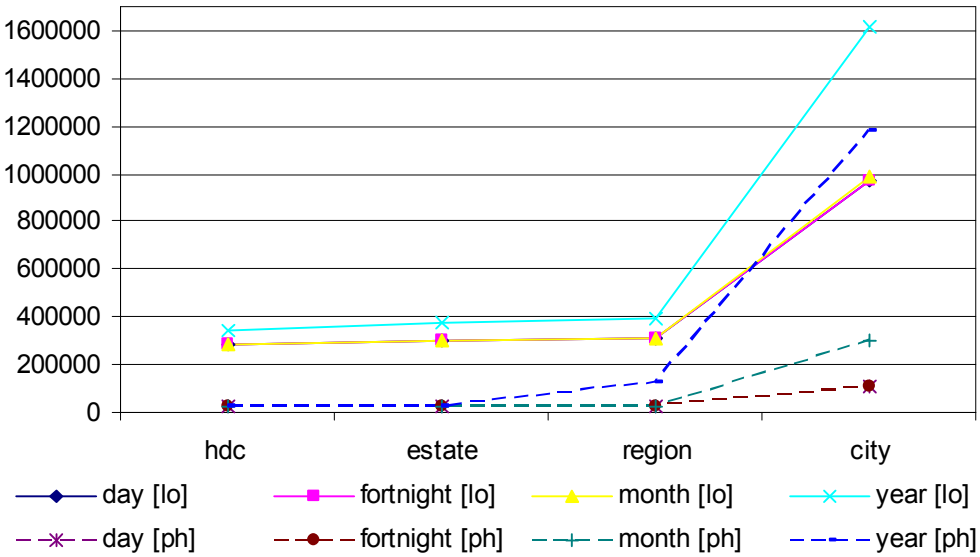


Fig. 5.16 The number of I/O operations (hot cache) for the HQT_1 queries for the database two on the primary hardware configuration.

The results for the HQT_2 queries for the database two on the secondary hardware configuration are represented in Figure 5.17 and Figure 5.18.

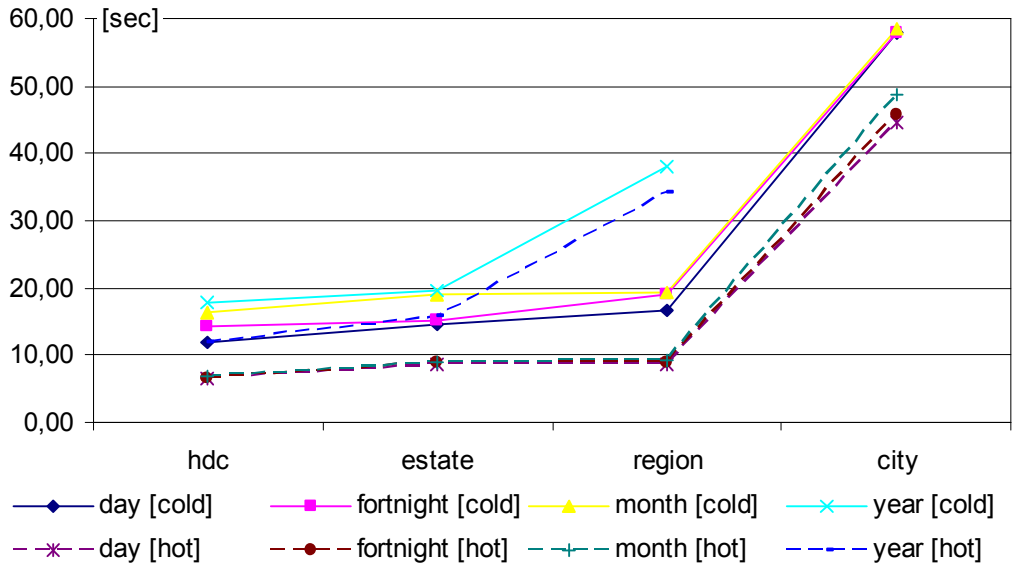


Fig. 5.17 The query evaluation time for the HQT_2 queries for the database two on the secondary hardware configuration.

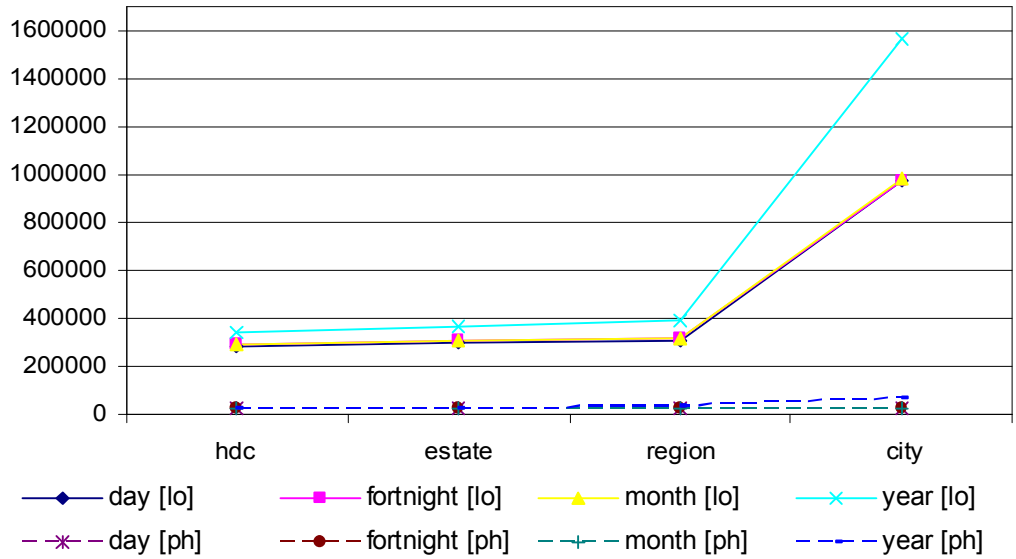


Fig. 5.18 The number of I/O operations (hot cache) for the HQT_2 queries for the database one on the secondary hardware configuration.

5.2.1.3 Conclusions

The results obtained for the HQT_1 query template confirm the potential of the data model for SA. The results show the efficient processing of multidimensional hierarchical clustering queries with high selectivity restrictions. Since the query response time is proportional to the size of the result set, the overall performance depends mainly on selectivity, if the selectivity drops (along any dimension) the performance also decreases.

Despite the limited memory, the query evaluation times in Figure 5.3 show the influence of the exact data access and OS caching for high selectivity restrictions. The conclusion can be drawn from the results represented in Figure 5.4 where the number of logical and physical I/O operations reported by the DBMS is practically the same. The query evaluation times in Figure 5.5 obtained for the large memory configuration (both for DBMS and OS) demonstrate even better performance, which results from better cache utilisation. The results in Figure 5.6 show the DBMS cache saturation for low selectivity restrictions (time $\geq$ month and space $\geq$ region) for which the numbers of logical and physical I/Os are very similar (cf. Figure 5.4). The results obtained for the HQT_1 queries for the database two demonstrate analogous characteristics thus confirming scalability with respect to the database size. The property is a consequence of the multidimensional hierarchical clustering and the exact data access which precisely targets the relevant tuples of the result set regardless of the overall dataset size.

In addition to the hierarchical restrictions, the Building dimension is also restricted on a feature attribute in the HQT_2 queries. Ideally, the extra restriction should reduce the number of I/O operations on the fact table and consequently lower the response time. Although the response time is still proportional to the result set size, the overall execution time is significantly higher than the one obtained for the HQT_1 queries. The reason for the higher execution time stems from the large number of query boxes on the fact table induced by the feature restriction. The results for the HQT_2 queries on the database one on the primary hardware configuration represented in Figure 5.11 demonstrate the evaluation time by an order of magnitude higher than the corresponding times for the HQT_1 queries (Figure 5.3). The non-efficient execution of the HQT_2 queries is confirmed by the numbers of logical and physical I/O operations (Figure 5.12) which are of three orders of magnitude higher than the one of the HQT_1 queries. The results for the HQT_2 queries are slightly improved (from 50% to 20%) for the large memory

configuration which makes the execution of the query template entirely CPU bound. As a result, the evaluation time for the HQT_2 queries for the large memory configuration are nearly the same for the cold and hot cache (Figure 5.13). The comparison of the physical I/Os between the small (Figure 5.12) and large (Figure 5.14) memory configuration confirms DBMS caching which makes CPU the main performance bottleneck of the HQT_2 query processing. Similar characteristic can be observer for the database two, both for the small (Figure 5.15 and 5.16) and large memory configuration (Figure 5.17 and 5.18). The operator tree for a sample HQT_2 query is represented below:

```
(N0:sel { 3 "column_0" "column_1" "column_2" }              --#tup: 1
 (N1:proj                                                  --#tup: 1
  (N2:group { count[*] sum[1] sum[2] }                     --#tup: 1
   (N3:rel { "heat_consumption" proj 2(7 5) } --
                #pages(index/leaf/nohit):168730/484/443   --#tup: 50
    (N4:times { keyaccess }                                --#tup: 23122
     (N5:times { keyaccess }                               --#tup: 23122
      (N6:times { keyaccess }                              --#tup: 23122
       (N7:ivmk { betw }                                   --#tup: 23122
        (N8:cs2ival { "building" 14 }                      --#tup: 23122
         (N9:sort { +1 }                                   --#tup: 28870
          (N10:proj                                        --#tup: 28870
           (N11:restr                                      --#tup: 28870
            (N12:rel { "building" proj 2(8 14)}            --#tup: 144945
            (N13:eq { }
             (N14:attr { N11[1] } )
             (N15:const { 'wood' char(4) } )))
           (N16:build
            (N17:attr { N10[2] } ))))))
      (N18:ivmk { betw }                                   --#tup: 1
       (N19:sort { +1 }                                    --#tup: 1
        (N20:proj                                          --#tup: 1
         (N21:index { "@@sys_surrHX_3" proj 2(4 5)
                      user_tablename="customer" singletup} --#tup: 1
          (N22:ivmk { eq }                                 --#tup: 1
           (N23:const { 'companies' char(9) } )))
         (N24:build
          (N25:bitor
           (N26:subrg { false }
            (N27:attr { N20[1] } )
            (N28:const { 1 integer } )
            (N29:const { 4 integer } ))
            (N30:const { 0b00000000000000000000000000 bits2(24) } ))
          (N31:bitor
           (N32:subrg { false }
            (N33:attr { N20[1] } )
            (N34:const { 1 integer } )
            (N35:const { 4 integer } ))
            (N36:const { 0b000011111111111111111111 bits2(24) } )))))))
     (N37:ivmk { betw }                                    --#tup: 1
      (N38:cs2ival { "hdc" 11 }                            --#tup: 1
       (N39:sort { +1 }                                    --#tup: 884
        (N40:proj                                          --#tup: 884
         (N41:restr                                        --#tup: 884
```

```
            (N42:index { "@@sys_surrHX_2" proj 3(2 5 6)
                                user_tablename="hdc" }        --#tup: 8128)
            (N43:eq { }
             (N44:attr { N41[1] } )
             (N45:const { 'Srodmiescie' char(11) } )))
           (N46:build
            (N47:attr { N40[2] } )))))))
      (N48:ivmk { betw }                                      --#tup: 1
       (N49:cs2ival { "day" 10 }                              --#tup: 1
        (N50:sort { +1 }                                      --#tup: 1
         (N51:proj                                            --#tup: 1
          (N52:restr                                          --#tup: 1
           (N53:mat { "day" }                                 --#tup: 365
            (N54:index { "@@sys_surrHX_5" proj 2(6 6)
                         user_tablename="day" }               --#tup: 365
             (N55:ivmk { eq }                                 --#tup: 1
              (N56:const { 2001 integer } ))))
           (N57:and
            (N58:eq { }
             (N59:attr { N52[6] } )
             (N60:const { 1  integer } ))
            (N61:eq { }
             (N62:attr { N52[4] } )
             (N63:const { 1  integer } ))))
          (N64:build
           (N65:attr { N51[10] } )))))))))
  (N66:build
   (N67:attr { N1[3] } )
   (N68:attr { N1[2] } )
   (N69:attr { N1[1] } ))))
{nosort}
```

The analysis of the query's operator tree confirms the large number of query boxes generated for the feature restriction "*building.technology = wood*". While the I/O operations for multi -query boxes has already been optimised in the DBMS, the predicate evaluation is still a bottleneck. For each tuple retrieved from disk, 2 * 4 * #query boxes predicates have to be evaluated in the worst case (lower and upper range bound for each dimension). This extensive CPU load leads to the slower execution time as compared to queries with only one (or a few) query boxes.

5.2.2 Queries with spatial predicates

The second category of tests concerns queries which include spatial predicates restricting coordinates and geometries. The following analysis enables the performance assesment of data access for attributes whose values correlate the proposed physical data organisation. The polygons used for the spatial restrictions are represented in Figure 5.19.

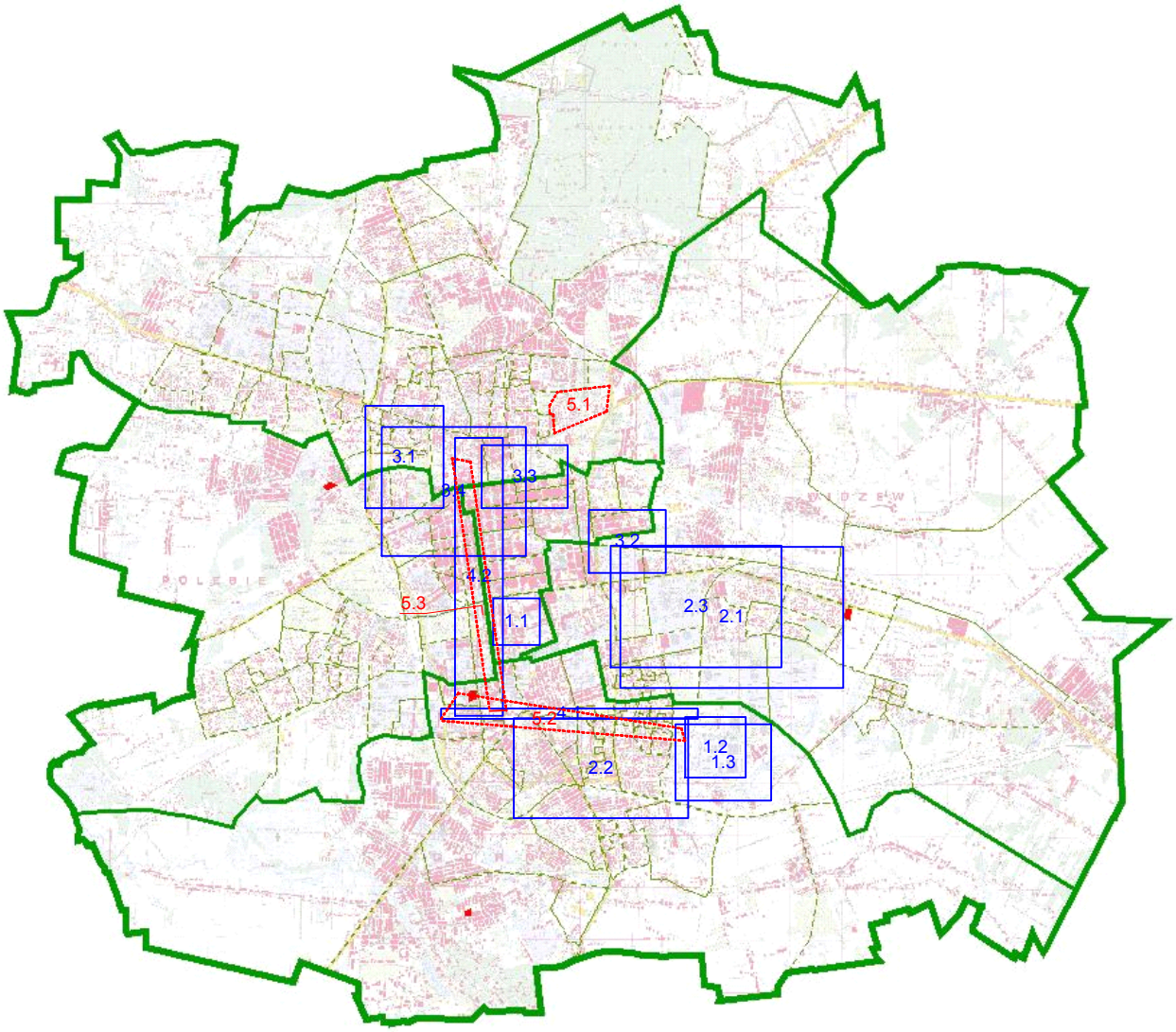


Fig. 5.19 The polygon restrictions used for the spatial predicates.

The spatial query temple (SQT) including spatial predicates is as follows:

```
SELECT SUM(ch_energy), SUM(hw_energy), COUNT(*)
FROM  heat_consumption h, hdc, day, building, customer
WHERE h.hdc_number = hdc.hdc_number AND h.day_number = day.day_number
AND   h.building_id = building.building_id AND h.customer_id =
      customer.customer_id
AND   <hierarchical and spatial restrictions>;
```

5.2.2.1 Results for SQT queries

The results for the SQT queries for the database one on the primary hardware configuration are represented in Figure 5.20 and Figure 5.21.

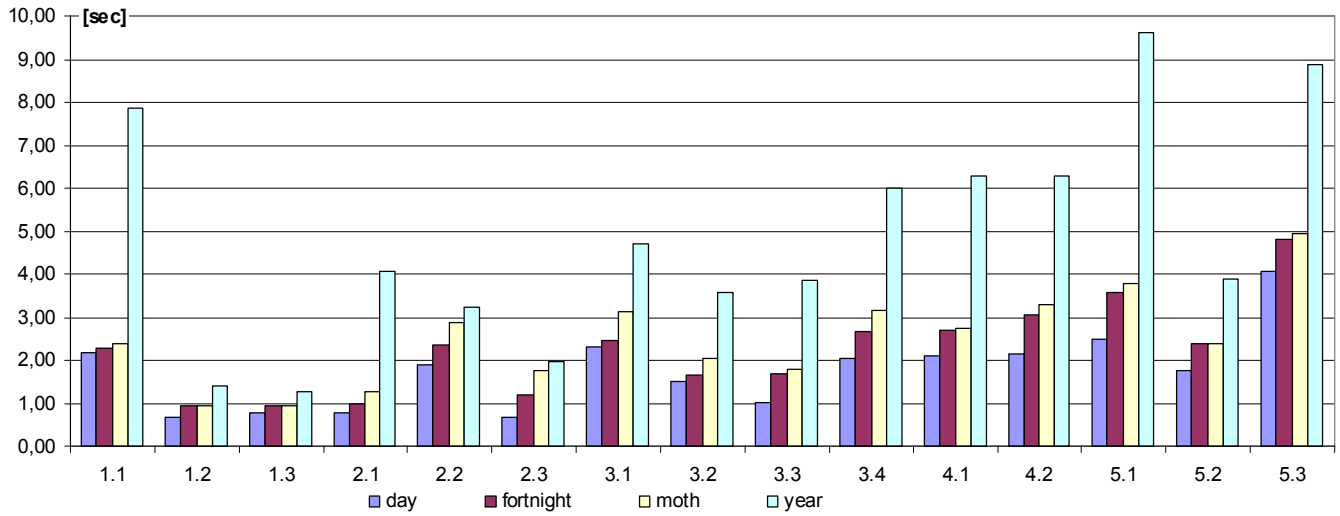


Fig. 5.20 The query evaluation time for the SQT queries for the database one on the primary hardware configuration (cold cache).

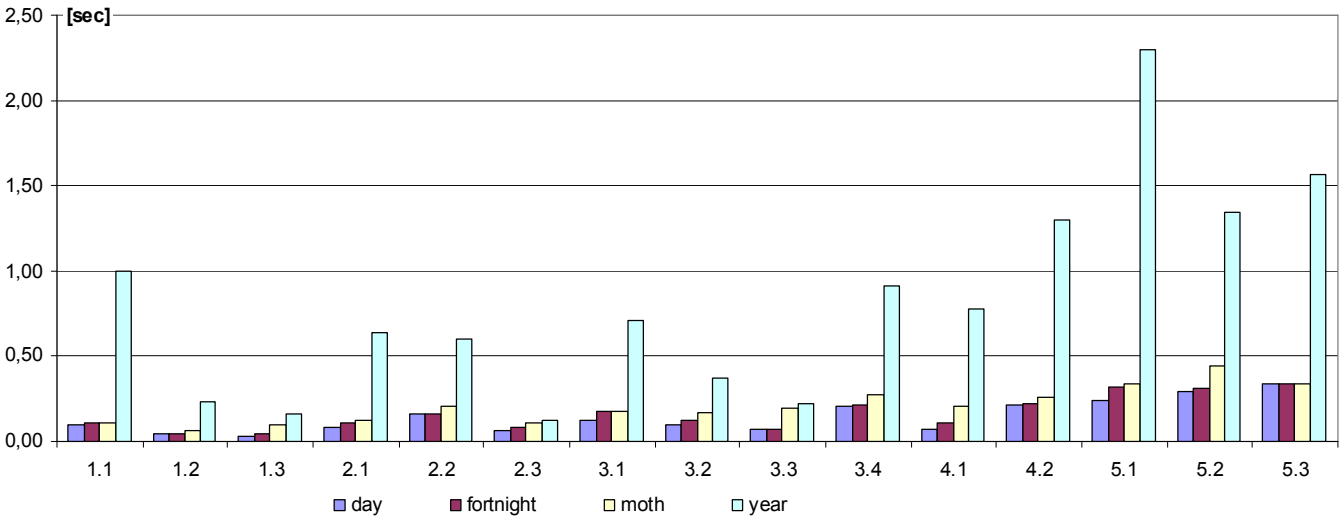


Fig. 5.21 The query evaluation time for the SQT queries for the database one on the primary hardware configuration (hot cache).

The results for the SQT queries for the database one on the secondary hardware configuration are represented in Figure 5.22 and Figure 5.23.

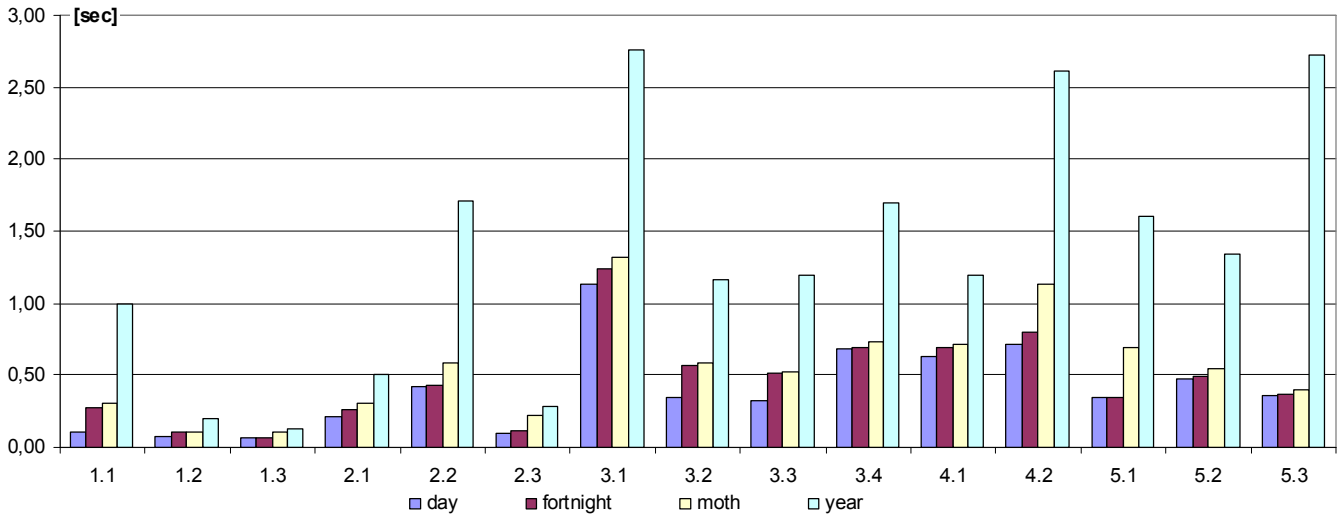


Fig. 5.22 The query evaluation time for the SQT queries for the database one on the secondary hardware configuration (cold cache).

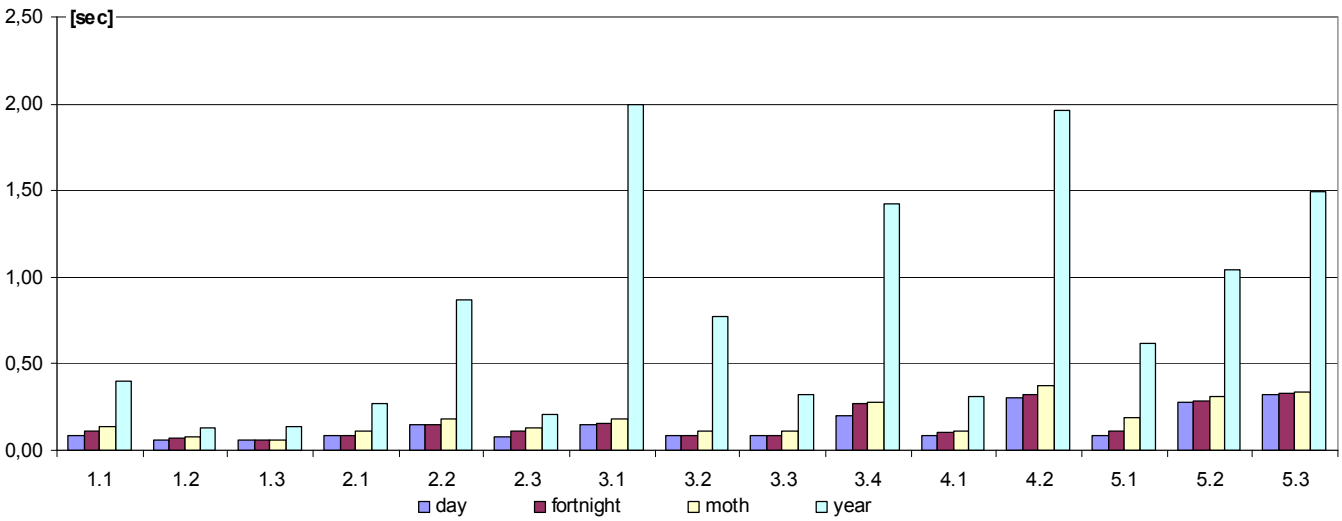


Fig. 5.23 The query evaluation time for the SQT queries for the database one on the secondary hardware configuration (cold cache).

The results for the SQT queries for the database two on the primary hardware configuration are represented in Figure 5.24 and Figure 5.25.

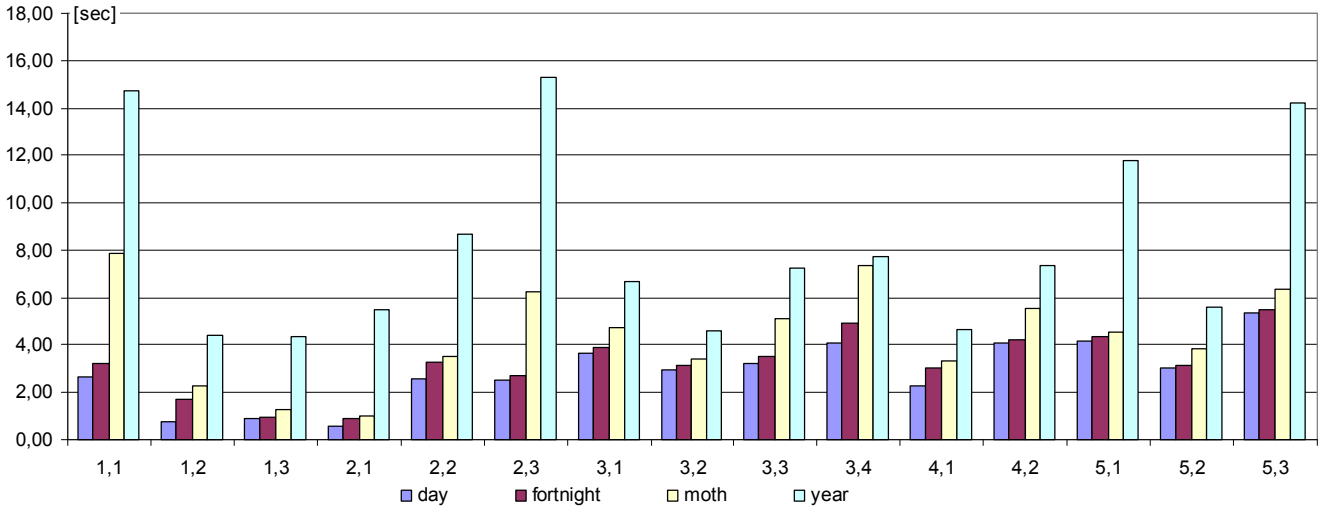


Fig. 5.24 The query evaluation time for the SQT queries for the database two on the primary hardware configuration (cold cache).

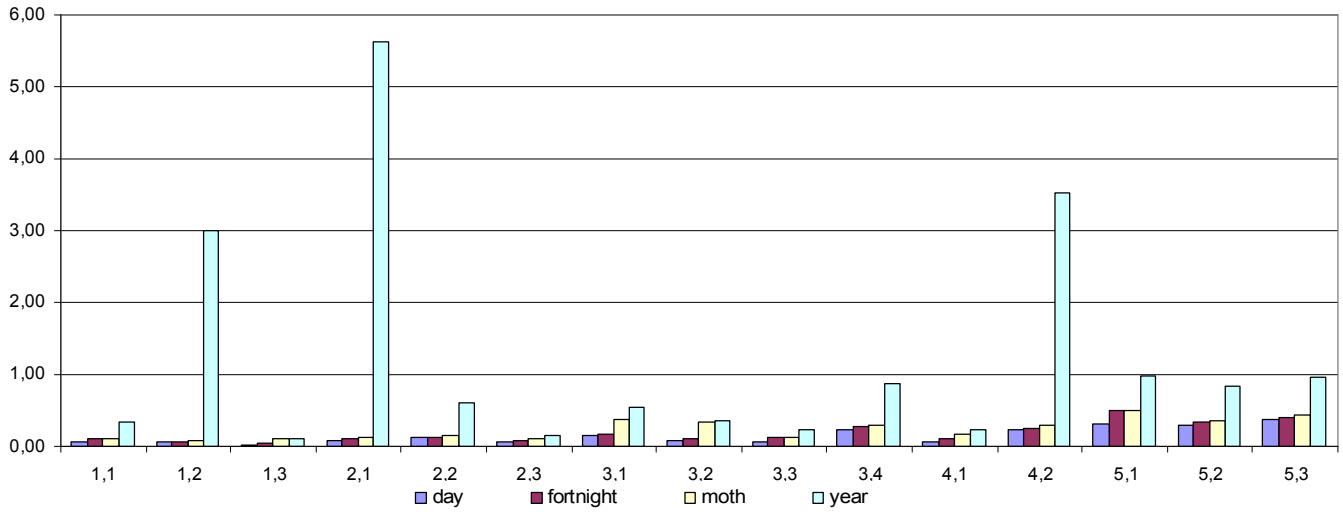


Fig. 5.25 The query evaluation time for the SQT queries for the database two on the primary hardware configuration (hot cache).

The results for the SQT queries for the database two on the secondary hardware configuration are represented in Figure 5.27 and Figure 5.28.

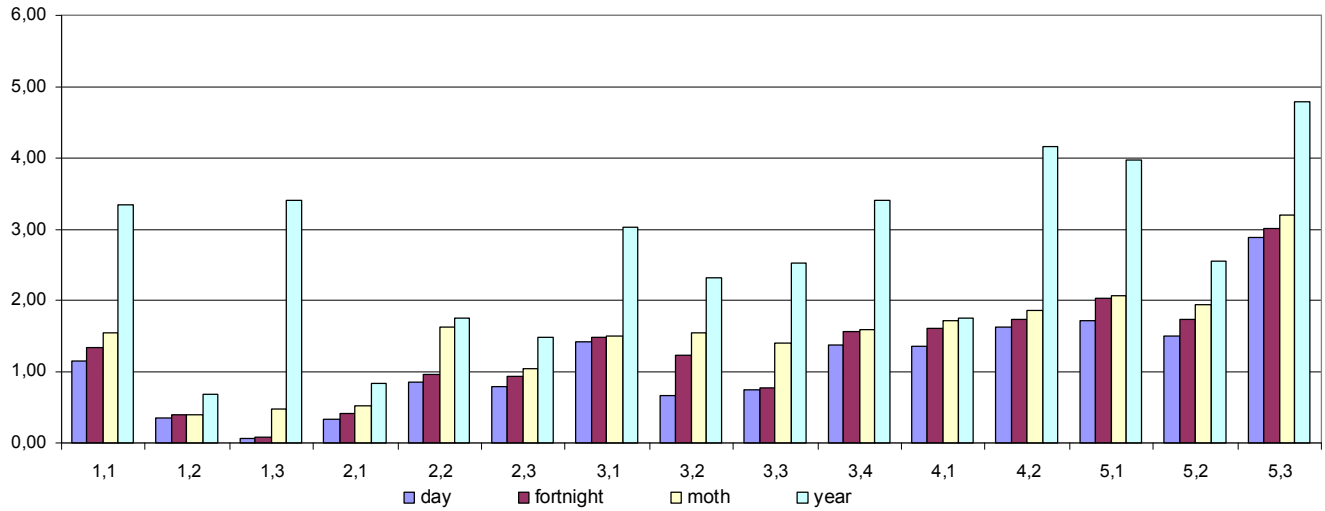


Fig. 5.27 The query evaluation time for the SQT queries for the database two on the secondary hardware configuration (cold cache).

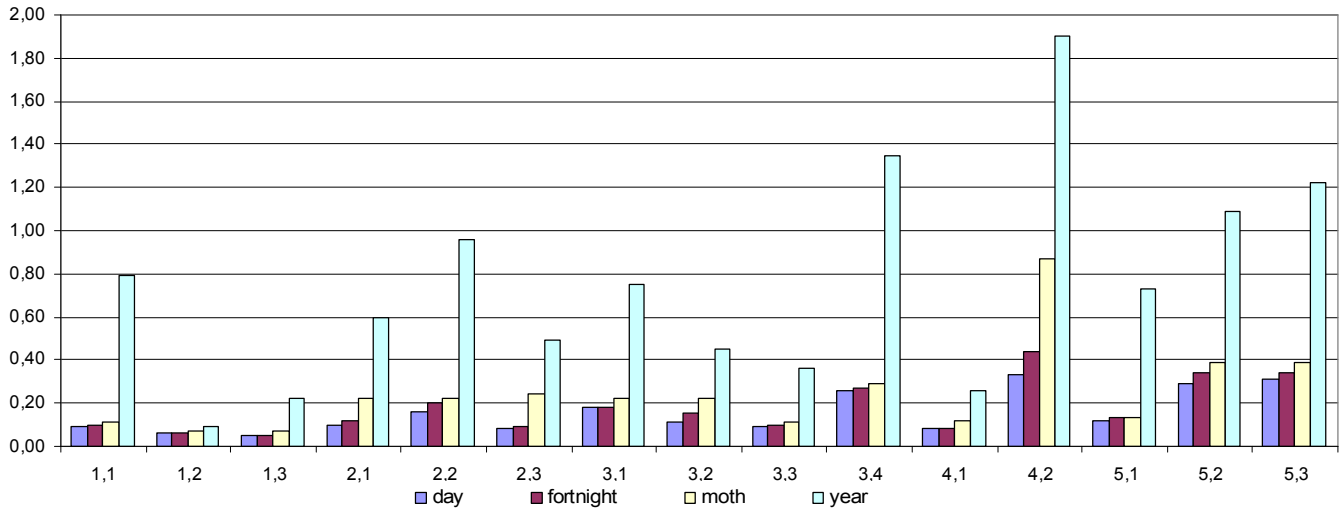


Fig. 5.28 The query evaluation time for the SQT queries for the database two on the secondary hardware configuration (hot cache).

5.2.2.2 Conclusions

The results represented in Section 5.2.2.1 confirm the high efficiency of spatial queries evaluated on top of the data model for SA. The efficient spatial query processing is a consequence of the correlation between the proximity-preserving data organisation and spatial coordinates and geometries which are relevant for the spatial queries. As a result, the spatial query processing is similar to the one of the hierarchical queries. The performance analysis also confirms the scalability of the spatial query processing for various selectivities along all dimensions, hardware configurations and database sizes.

5.2.3 Complex queries

The following queries are exemplify complex query patterns typical of spatial data warehousing. The patterns simulate static reporting which differs from the interactive ad-hoc analysis in scope (multi-fact table, such as drill-across) and character (static, predefined or parameterised). The difference impacts the structure of the query patterns implementing the static data analysis. The performance assessments presented in the Section evaluates the following query templates:

"Total heat consumption before and after renovation" (THCR) template

```
SELECT br.building_id, d.year, d.month_in_year,
      SUM (CASE WHEN hc.day_number >= br.day_number + br.duration THEN
            hc.ch_energy ELSE 0 END) /
           (CASE WHEN
            SUM(CASE WHEN hc.day_number >= br.day_number + br.duration
                  THEN 1 ELSE 0 END) > 0 THEN
            SUM(CASE WHEN hc.day_number >= br.day_number + br.duration
                  THEN  1 ELSE 0 END) ELSE 1 END) AS Consumption_After,
      SUM (CASE WHEN hc.day_number < br.day_number THEN hc.ch_energy
            ELSE  0 END) /
           (CASE WHEN SUM(CASE WHEN hc.day_number < br.day_number THEN 1
            ELSE 0 END) > 0 THEN
            SUM(CASE WHEN hc.day_number < br.day_number THEN 1 ELSE 0
                  END) ELSE 1 END) AS Consumption_Before
FROM building_renovation br, heat_consumption hc, hdc, day d
WHERE hc.building_id = br.building_id
      AND hc.hdc_number = hdc.hdc_number
      AND br.day_number = d.day_number
      AND br.day_number IN
            (SELECT MAX(day_number) FROM building_renovation GROUP BY
            building_id)
      AND <hierarchical restrictions>
GROUP BY br.building_id, d.year, d.month_in_year;
```

"Total loss for a given Heat Source in specified time" (TLHS) template

```
SELECT hp.heat_source_number, hp.day_number,
      hp.energy AS Production, SUM(hc.ch_energy) AS Consumption
FROM  supply_catchment sc, day d, heat_consumption hc, heat_production
      hp, heat_source hs, fortnight f
WHERE sc.hdc_number = hc.hdc_number
      AND hp.heat_source_number = hs.heat_source_number
      AND sc.heat_source_number = hs.heat_source_number
      AND hp.day_number = d.day_number
      AND hc.day_number = d.day_number
      AND f.fortnight_number = d.fortnight_number
      AND f.season_number = sc.season_number
      AND <hierarchical restrictions>
GROUP BY hp.heat_source_number, hp.day_number, hp.energy;
```

"The list of buildings connected to a given Heat Source" (LBHS) template

```
SELECT DISTINCT hs.heat_source_number, b.building_id, b.name
FROM  heat_source hs, supply_catchment sc, hdc, power_demand pd, building
      b, season s
WHERE hs.heat_source_number = sc.heat_source_number
      AND sc.hdc_number = hdc. hdc_number
      AND sc.season_number = s.season_number
      AND hdc.hdc_number = pd.hdc_number
      AND pd.building_id = b.building_id
      AND <hierarchical restrictions>;
```

"List of customers with decreasing heat demands" (LCDHD) template

```
SELECT pd1.customer_id, pd1.season_number, SUM(pd2.ch_power -
      pd1.ch_power)
FROM  power_demand pd1, power_demand pd2, hdc, season s, customer c
WHERE pd1.hdc_number = pd2.hdc_number
      AND pd1.building_id = pd2.building_id
      AND pd1.customer_id = pd2.customer_id
      AND pd1.season_number = pd2.season_number - 1
      AND pd1.hdc_number = hdc.hdc_number
      AND pd1.customer_id = c.customer_id
      AND pd1.season_number = s.season_number
      AND <hierarchical restrictions>
GROUP BY pd1.customer_id, pd1.season_number
HAVING SUM(pd2.ch_power - pd1.ch_power) < -20;
```

"The total cost of outages for the selected HDCs" (TCO) template

```
SELECT cst_id, cst_name, SUM(dur), SUM(tot_cost)
FROM (
      SELECT c.customer_id cst_id, c.name cst_name, o.duration dur,
            SUM(o.duration * r.outage_rate) tot_cost
      FROM  ay d, outage o, hdc, power_demand pd, customer c, rate r
      WHERE d.day_number = o.day_number
            AND o.hdc_number = hdc.hdc_number
            AND hdc.hdc_number = pd.hdc_number
            AND pd.customer_id = c.customer_id
            AND pd.rate_id = r.rate_id
            dAND <hierarchical restrictions>
      GROUP BY c.customer_id, c.name, o.hdc_number, o.day_number,
      outage.duration
)
GROUP BY cst_id, cst_name;
```

“Total duration and cost of outages for the selected customers” (TDO) template

```
SELECT cst_id, cst_name, SUM(dur), SUM(tot_cost)
FROM (SELECT c.customer_id cst_id, c.name cst_name, o.duration dur,
            SUM(o.duration * r.outage_rate) tot_cost
      FROM  day d, outage o, hdc h, power_demand pd, customer c, rate r
      WHERE d.day_number = o.day_number
            AND o.hdc_number = hdc.hdc_number
            AND hdc.hdc_number = pd.hdc_number
            AND pd.customer_id = c.customer_id
            AND pd.rate_id = r.rate_id
            AND <hierarchical restrictions>
      GROUP BY c.customer_id, c.name, o.hdc_number, o.day_number,
      o.duration
)
GROUP BY cst_id, cst_name;
```

5.2.3.1 Results for complex queries

The results for the SQT queries for the database one and two on the primary and secondary hardware configuration are represented in Figure 5.29 and Figure 5.30.

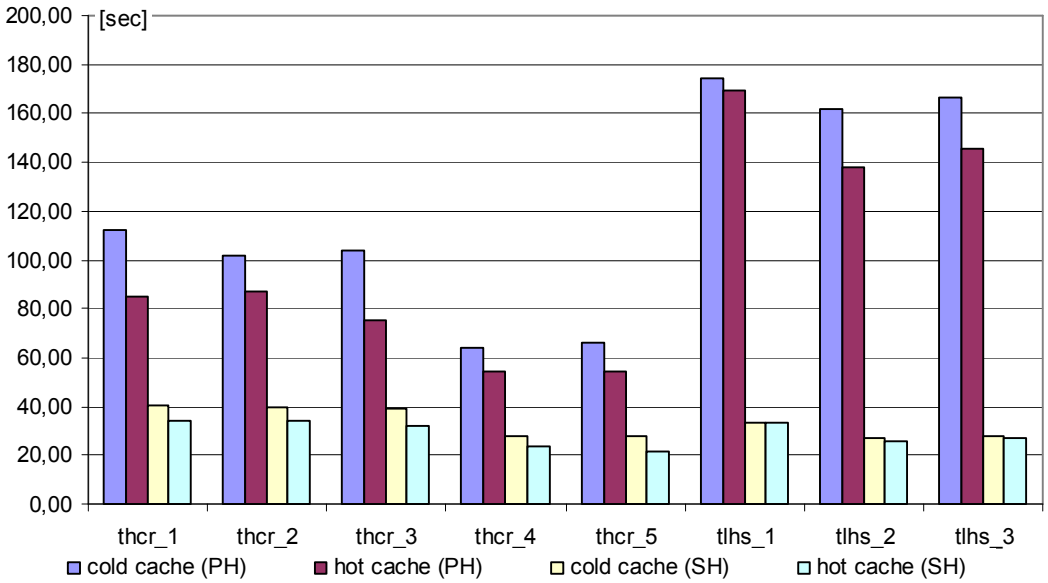


Fig. 5.29 The query evaluation time for the complex queries (set one) for the database one on the primary (PH) and secondary (SH) hardware configuration.

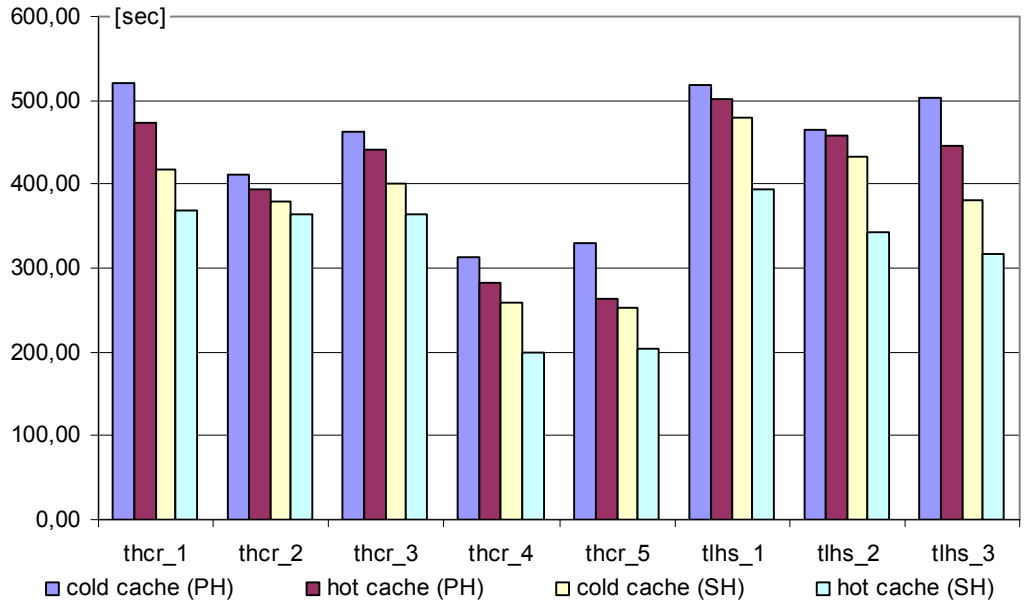


Fig. 5.30 The query evaluation time for the complex queries (set one) for the database two on the primary (PH) and secondary (SH) hardware configuration.

The results for the SQT queries for the database one and two on the primary and secondary hardware configuration are represented in Figure 5.31 and Figure 5.32.

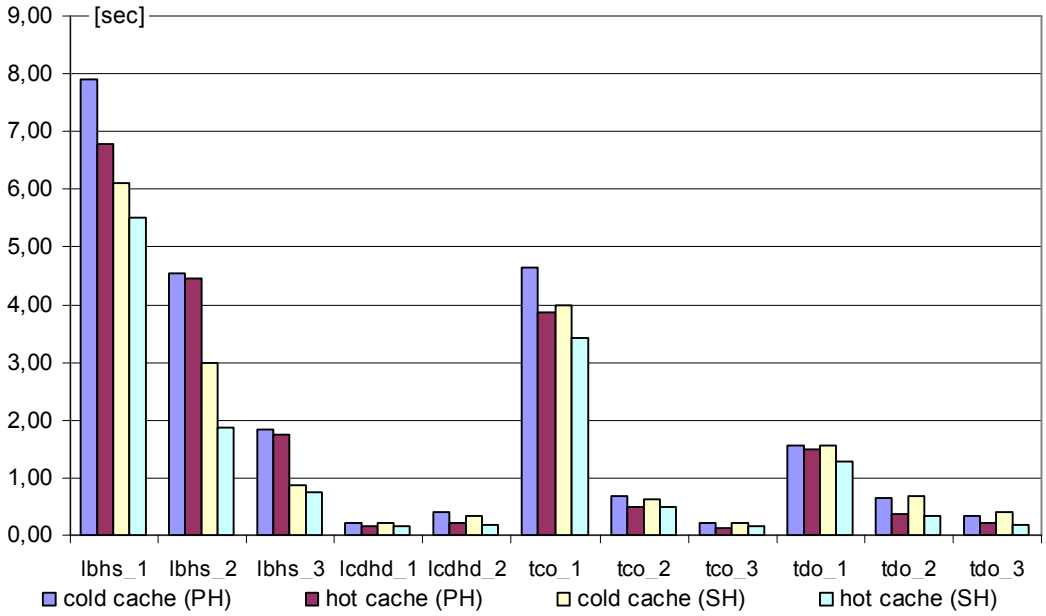


Fig. 5.31 The query evaluation time for the complex queries (set two) for the database one on the primary (PH) and secondary (SH) hardware configuration.

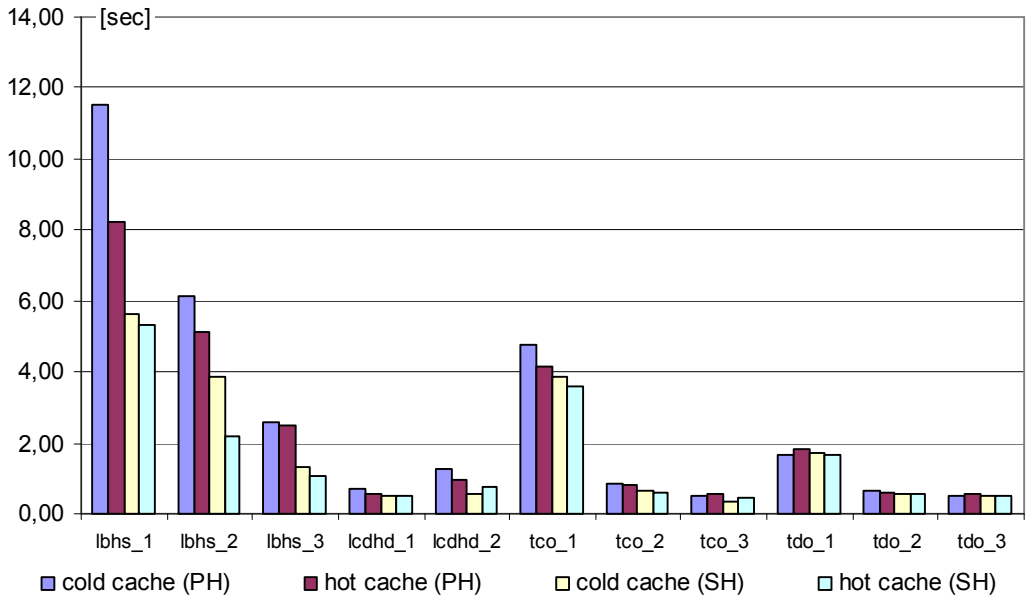


Fig. 5.32 The query evaluation time for the complex queries (set two) for the database two on the primary (PH) and secondary (SH) hardware configuration.

5.2.3.2 Conclusions

The results for the **THCR query template** shown in Figure 5.29 indicate inefficient join processing between *Building_Renovation* and *Heat_Consumption* fact tables. The weak performance results from the full table scan and a sort before joining the two fact tables. Similarly, **the TLHS query template** contains a join path between *Supply_Catchment*, *Heat_Consumption* and *Heat_Production* fact tables which is also not handled efficiently. The rule-base optimiser is not able to recognise data distributions underlying the data schema and consequently creates the inefficient execution plans. The inefficiency concerns join processing beginning with the largest fact table and using the sort/merge algorithm. The following operator tree confirms the observation.

```
(N0:sel {4 "heat_source_number" "day_number" "Production" "Consumption" }
                                                        --#tup: 10
 (N1:group { [ 3 2 4 ] sum[1] }                         --#tup: 183
  (N2:sort { +3 +2 +4 }                                 --#tup: 978135
   (N3:mjoin { 6=2 4=3 5=1 proj 4(7 1 2 3) }            --#tup: 978135
    (N4:sort { +6 +4 +5 }                               --#tup: 978135
     (N5:mjoin { 1=2 proj 7(2 3 4 5 6 7 9) }            --#tup: 978135
      (N6:sort {  +1 }                                  --#tup: 183
       (N7:times { proj 6(1 3 4 5 6 7) }                --#tup: 183
        (N8:times { proj 6(1 2 3 4 5 6) }               --#tup: 183
         (N9:times { proj 5(2 1 4 5 6) }                --#tup: 732
          (N10:index { "@@sys_surrHX_5" proj 3(3 4 6)
                       user_tablename="day" }           --#tup: 183
           (N11:times { }                               --#tup: 1
            (N12:ivmk { eq }                            --#tup: 1
             (N13:const  { 2001 integer } ))
            (N14:ivmk { eq }                            --#tup: 1
             (N15:const { 'summer' char(6) } ))))
          (N16:rel { "heat_production" proj 3(1 2 3) } --#tup: 732
           (N17:ivmk { eq }                             --#tup: 1
            (N18:attr { N9[2] } ))))
         (N19:index { "@@sys_surrHX_7" proj 2(2 4)
                       user_tablename="heat_source" } --#tup: 183
          (N20:times { }                                --#tup: 1
           (N21:ivmk { eq }                             --#tup: 1
            (N22:const { 'local coal'  char(10) } ))
           (N23:ivmk { eq }                             --#tup: 1
            (N24:attr { N8[4] } )))))
        (N25:rel { "fortnight" proj 1(3) }              --#tup: 183
         (N26:ivmk { eq }                               --#tup: 1
          (N27:attr { N7[2] } )))))
       (N28:sort { +2 }                                 --#tup: 28793516
        (N29:rel { "heat_consumption" proj 3(2 4 5) } -
                  #pages(index/leaf/nohit):0/0/0        --#tup: 29285255
    ))))
     (N30:sort { +2 +3 +1 }                             --#tup: 972450
      (N31:rel { "supply_catchment" proj 3(1 2 3) } -
                #pages(index/leaf/nohit):0/0/0          --#tup: 975360
))))))
{nosort}
```

The results obtained for the other complex queries (LBHS, LCDHD, TCO, TDO query patterns) demonstrate better performance mainly because the join with the largest fact table (Heat_Consumption) is avoided. As a result, the inability of the query optimiser to create optimal execution plans does not significantly impact the overall performance. The different evaluation times for various query restrictions confirm the performance scalability with respect to selectivity. Similarly, the obtained results proves performance gains for the secondary hardware configuration which confirms the positive cache influence. However, retrieving a target result set for the complex queries is not as accurate as in the case of the star-queries. The conclusion can be drawn from the obtained results which indicates worse scalability with respect to the database size as compared to the hierarchical queries.

The current implementation of the data model for SA has focused so far only on single fact table data access. The response times obtained for the complex query patterns in the current implementation confirms problems with the query optimizer in the case when multiple fact tables are combined within one query.

Chapter 6

Conclusions and future work

6.1 Conclusions

Spatial analysis (SA) is a discipline focusing on the interpretation and evaluation of results obtained in the process of modelling and analysing spatial phenomena. Extracting useful and interesting patterns from spatial datasets is, however, more difficult than extracting corresponding patterns from numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial interdependencies. The complexity entails sophisticated requirements on data organization and management in order to effectively support SA. The thesis meets the requirements by proposing the dedicated data model which integrates analytical and spatial data while providing efficient data access for query processing. In particular, the properties of the data model for SA, defined in Section 1.4, are addressed in the thesis as follows:

I. **The data model should include both spatial and analytical data** – On the conceptual and logical levels, the data model for SA integrates both spatial and analytical data into a single schema. The physical schema of the data model allows for data partitioning in order to further improve data access, i.e. complex spatial attributes are physically stored in a dedicated partition.

II. **The data model should support various representations of spatiality** – The data model for SA supports various representations of spatiality because different data types are used for modelling spatial objects. In order to improve manageability and accessibility, metadata describing interrelationships between the representations must be defined.

III. **The data model should represent multidimensional character of SA** – In order to reflect the multidimensional character of spatial phenomena, the data model for SA is based on the models of multidimensional databases. In particular, the logical schema of the data model adapts the data warehousing schemata which are the relational representations of a multidimensional data cube.

IV. **The data model should represent hierarchical character of SA** – Since the data model for SA is designed for the relational data representation, no explicit support of hierarchies is possible without metadata. Recognising the importance of hierarchies in SA, the proposed data model exploits the concept in the dedicated physical data organisation. Various spatial hierarchies are considered in the thesis in order to cover a wide range of application domains.

V. **The data model should treat symmetrically all properties of the analysed phenomena** – Since the data model for SA is based on multidimensional data models, the properties of modelled phenomena are represented as independent dimensions, e.g. time and spatial dimension.

VI. **The data model should be flexible to organise data on the physical level with respect to arbitrary semantics** – The main idea underlying the data model for SA consists in proximity-preserving data organisation. The exact clustering depends, however, on arbitrary modelling decisions which are related to the properties of spatial phenomena or the character of a given SA method.

VII. **The data model should support locality-preserving data organisation on physical level** – The main performance improvement assumption underlying the data model for SA is locality- (distance, proximity) preserving hierarchical data organisation. The clustering is obtained by applying the space-filling curve (Z-curve) and multidimensional indexing techniques (UB-tree and MHC/HI).

VIII. **The data model should support data partitioning** – The relational representation of the data model for SA enables applying the conventional partitioning methods, e.g. list or range partitioning. The thesis proposes two additional partitioning methods which exploit the hierarchical data clustering that underlies the physical layer of the data model for SA.

IX. **The data model should provide means to integrate other performance enhancements for SA** – Since the data model for SA is designed for the relational data representation, any data access or query processing techniques can be integrated into the DBMS to further improve the performance of SA.

The issues related to query processing and optimization exploiting the properties of the data model for SA are addressed in the thesis as follows:

X. **Spatial queries, e.g. range queries, nearest neighbour queries, spatial joins** – The data model for SA adapts the R-tree index for spatial data management. The index structure provides the efficient data access and evaluation of complex spatial queries.

XI. **Aggregations at different levels of a spatial hierarchy** – Since the logical schema of the data model for SA is based on the relational data representation, relational query languages provide grouping and aggregation operations for the attributes of a spatial hierarchy. The proposed data model supports the operations with the hierarchical data clustering and dedicated query processing algorithms.

XII. **Multidimensional analytical queries** – The data model for SA supports analytical queries in scope feasible in the relational data representation. The physical schema of the proposed data model and the dedicated data access algorithms are specifically designed for the evaluation of the query patterns.

Besides the performance analysis of the data model for SA (see Section 5.2), the adequacy of the proximity-preserving data organisation supporting SA was also verified by the author in two research engagements. The paper by Okoniewski et al. [OGG01] represents an algorithm for the computation of multidimensional quantitative association rules. The author contribution in the research concerned the application of the UB-tree index to improve the performance of multidimensional range query processing. The second research engagement concerned the analysis of spatial clustering algorithms with respect to various data indexing profiles. The results of the research, demonstrating considerable performance gains for the proximity-preserving data organisation, were described in a Master's Thesis supervised by the author [BW03].

6.2 Future work

Although locality is the central organizing principle of geo-space, the property is not required to be a function of Euclidean space [Mil04]. Geographic phenomena appear not to be consistent with the Tobler's First Law (see Section 1.1) may in fact be following non-Euclidean nearness relations, e.g. geographic diffusion processes such as disease propagation, movement and interaction at the urban, regional, and national scales etc. Since the data model for SA is based on the Euclidian geo-representation, some research is required to provide efficient database support for the non-Euclidian geo-spaces.

The concept of locality in geography can be expanded to space and time. In recent years, numerous theories and techniques for analyzing space-time behaviour at multiple scales have been proposed. Although the data model for SA has not been originally designed for complex spatio-temporal data analysis, the multidimensional hierarchical clustering of independent dimensions constitutes a foundation for efficient spatio-temporal data management. Research effort concerning the incorporation of temporal database facilities into the proposed data model is, however, required to create a complete solution.

Implementing various SA techniques as well as performance testing of the techniques on the data model for SA can reveal other promising research directions. The adaptation of the existing performance improvement methods benefiting from the properties of the proposed data model also appears to be worth further investigation. For example, the multidimensional hierarchical clustering can be used in the computation of materialized views by deriving aggregates efficiently from raw data. The approach allows to avoid the materialization of many aggregation levels and consequently reduces the view maintenance problem for summary tables to a large extent.

Annex I

System implementing the data model for SA

I.1 Introduction

The presentation of the data model for SA in Chapter IV assumes there exists one DBMS for management of spatial data and capable of the multidimensional hierarchical clustering Although such assumption seems rational and sufficient for theoretical divagations, the performance analysis of the proposed data model requires the implementation of such DBMS or the integration of systems with complementary functionalities. The relative immaturity of the research in the area as well as the complexity of requirements result in a lack of one DBMS providing the complete database support for SA. As a result, most database systems for SA separate analytical and spatial data management by creating loosely coupled architecture in the form of a heterogeneous database system [FCT01, RSV02, Raf03, FTSSR04]. Similarly, in order to implement the proposed data model for SA, a heterogeneous database system was developed using the Transbase database engine and Smallworld GIS [GGDW01, GW03]. The design of the system was based on the reference architecture which integrates different database management systems into one logical data management platform [SL90]. In the system, the Transbase DBMS was responsible for the multidimensional hierarchical clustering and the processing of analytical queries whereas SmallWarld GIS provided the map-driven user interface, spatial data management and the processing of complex spatial predicates. The architectural diagram of the system as well as the description of the system's components are presented in Section I.2.

I.2 Architecture of the system

Figure I.1 represents the architecture of the system implementing the data model for SA.

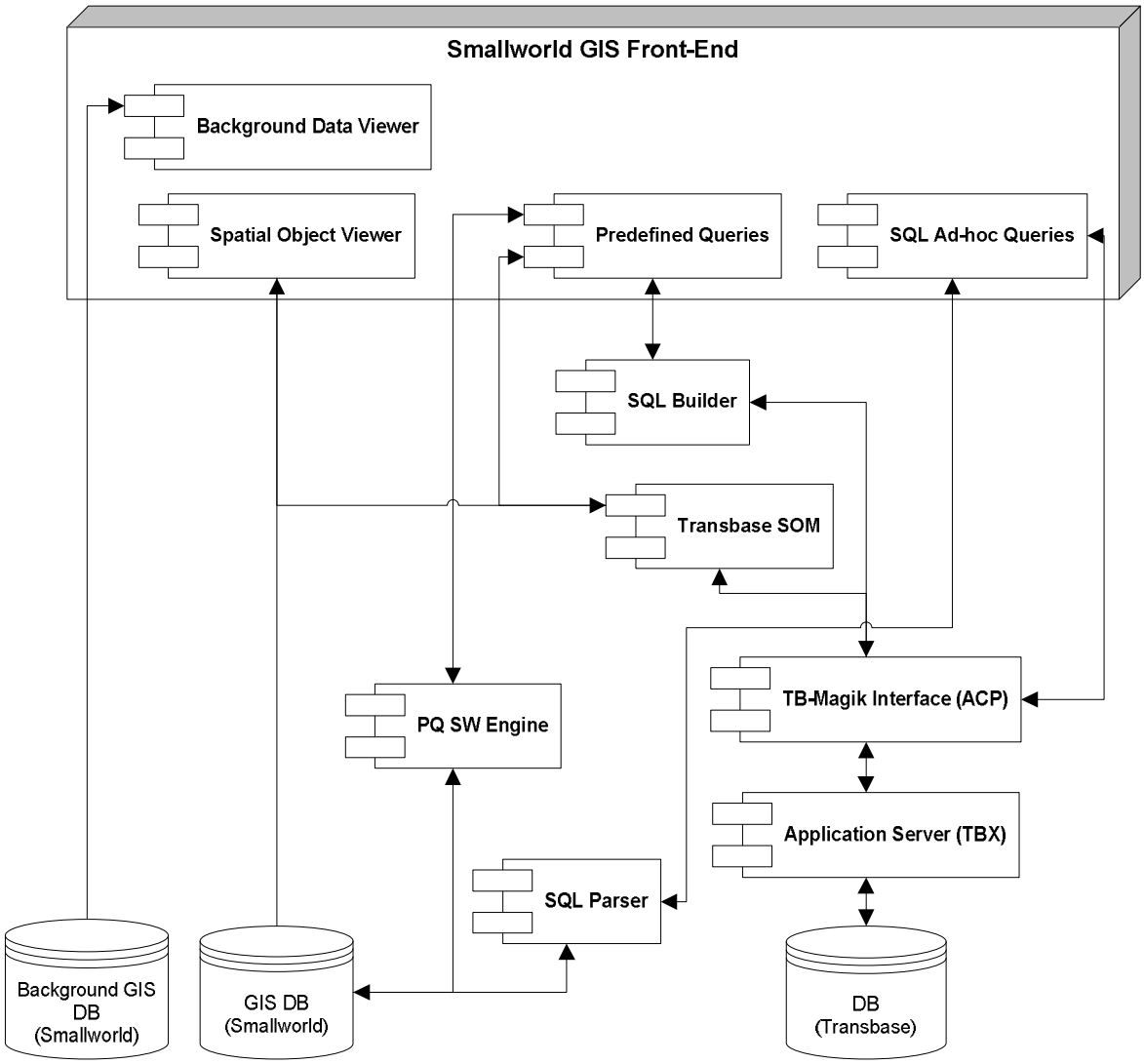


Fig. I.1 The architecture of the system implementing the data model for SA.

The roles of the system's components are as follows:

1. **Background Data Viewer** displays raster map images in the background of the system's presentation layer,
2. **Spatial Object Viewer** is the system's presentation layer which displays spatial objects on a map interface,

3. **Predefined Queries** is a front-end tool for the parameterisation of predefined queries (the queries used in the performance analysis – see Section 5.2) and the presentation of results,
4. **SQL Ad-Hoc Queries** is a front-end tool for ad-hoc queries specified in the SQL query language,
5. **PQ SW Engine** (Predefined Queries Smallworld Engine) is a back-end tool used by the Predefined Queries component to execute queries, collect and deliver results,
6. **SQL Parser translates SQL** queries for Smallworld Magik query interface,
7. **SQL Builder** creates SQL statements based on the parameters from Predefined Queries component,
8. **Transbase SOM** provides interoperability between the relational (Transbase) and object-oriented (Smallworld) representation of spatial objects,
9. **TB Magik Interface** and TBX Application Server provide communication facilities between the Smallworld and Transbase environments.

Annex II

Case study of the data model for SA

II.1 Overview of application domain

An application domain for the data model proposed in the thesis is a business and technical activity of a municipal heat production and distribution company. The choice of the application domain is a consequence of the author's participation in EDITH project (see Section 1.3), during which some of the concepts presented in the thesis were originally created. The scope of the project included among others the development of a GIS-oriented data warehouse which was based on real-life data and simulated realistic data processing [GW03]. An overview of the thematic scope of the application domain is presented below.

A **Heat Source** (a coal power plant, or a simple local gas plant) supplies, through a network of pipes, a number of **Heat Distribution Centres (HDC)**, each located usually in a **Building** which it supplies. It is, however, possible that a single HDC supplies a number of small buildings or a large building is supplied by a number (usually not more than 2) HDCs. Due to some regulation reasons, the **Supply Catchments** (the assignment of HDC's to heat sources) changes every **Season**. There are two heating seasons in a year: winter and summer, beginning and ending at the beginning/end or middle of a month. The heat distribution company provides two services: central heating (CH – heating the air in the building) and hot water (HW – heating water supplied by a water distribution company). A **Customer** orders power, measured in MW and defined for a given season and a given building and HDC. On the other hand, the conglomerate 'customer-building-HDC' consumes energy whose amount, in the ideal case, should correspond to the power ordered. The customer pays to the company according to the **Rate** assigned to him and, of course, the amount of energy consumed. There are three important events which have to be registered: **Building Renovations** (e.g. replacing windows, insulating buildings' walls)

that influence the amount of energy consumed, **HDC Renovations** (e.g. replacing old equipment) that influence the amount of energy produced, and **Outages** (HDC is off-line due to emergency or planned activities) that result in a non-sales period and a fine paid to the customer by the distribution company.

II.2 Overview of schemata

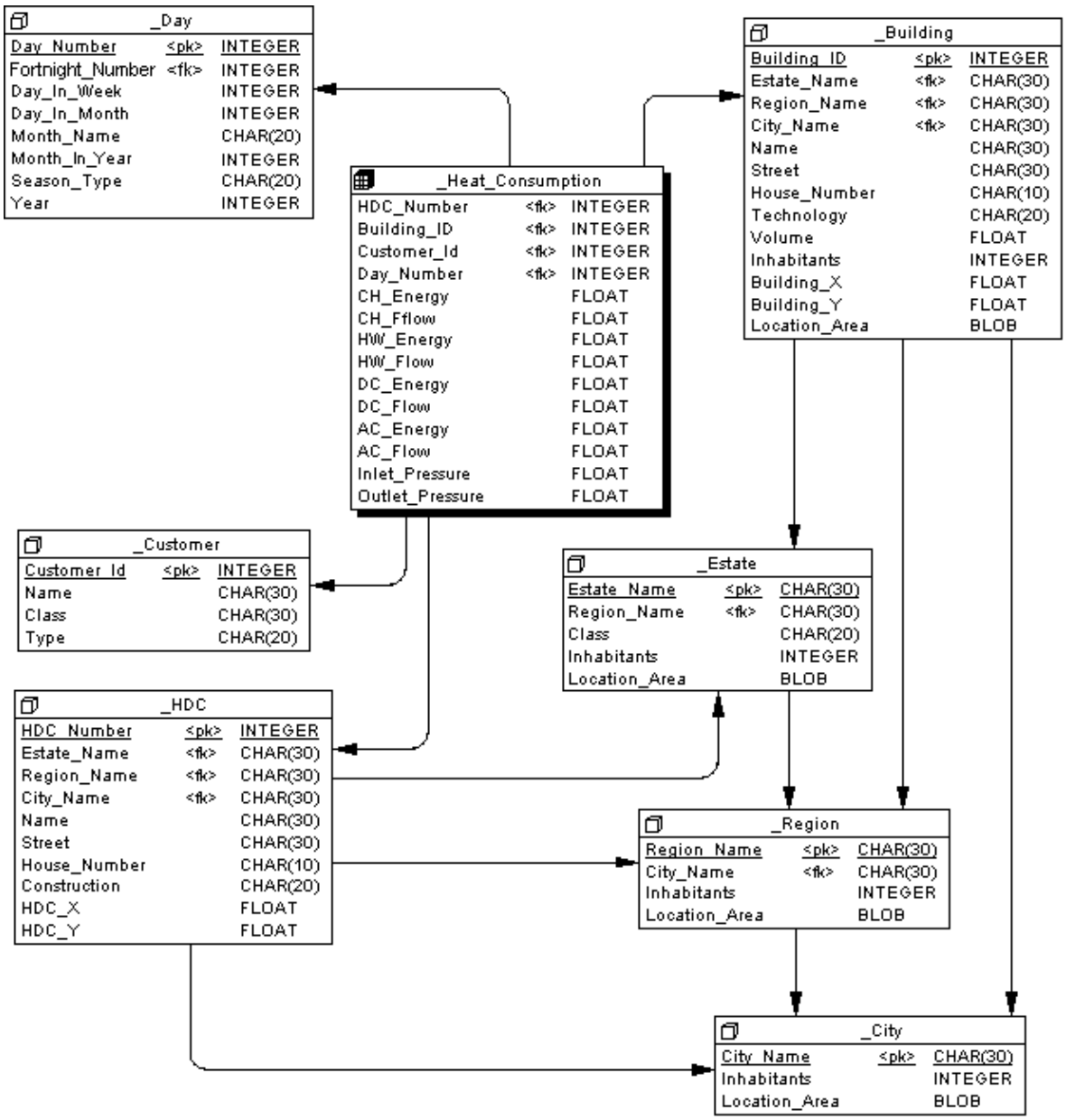


Fig. II.1 The **Heat Consumption** schema.

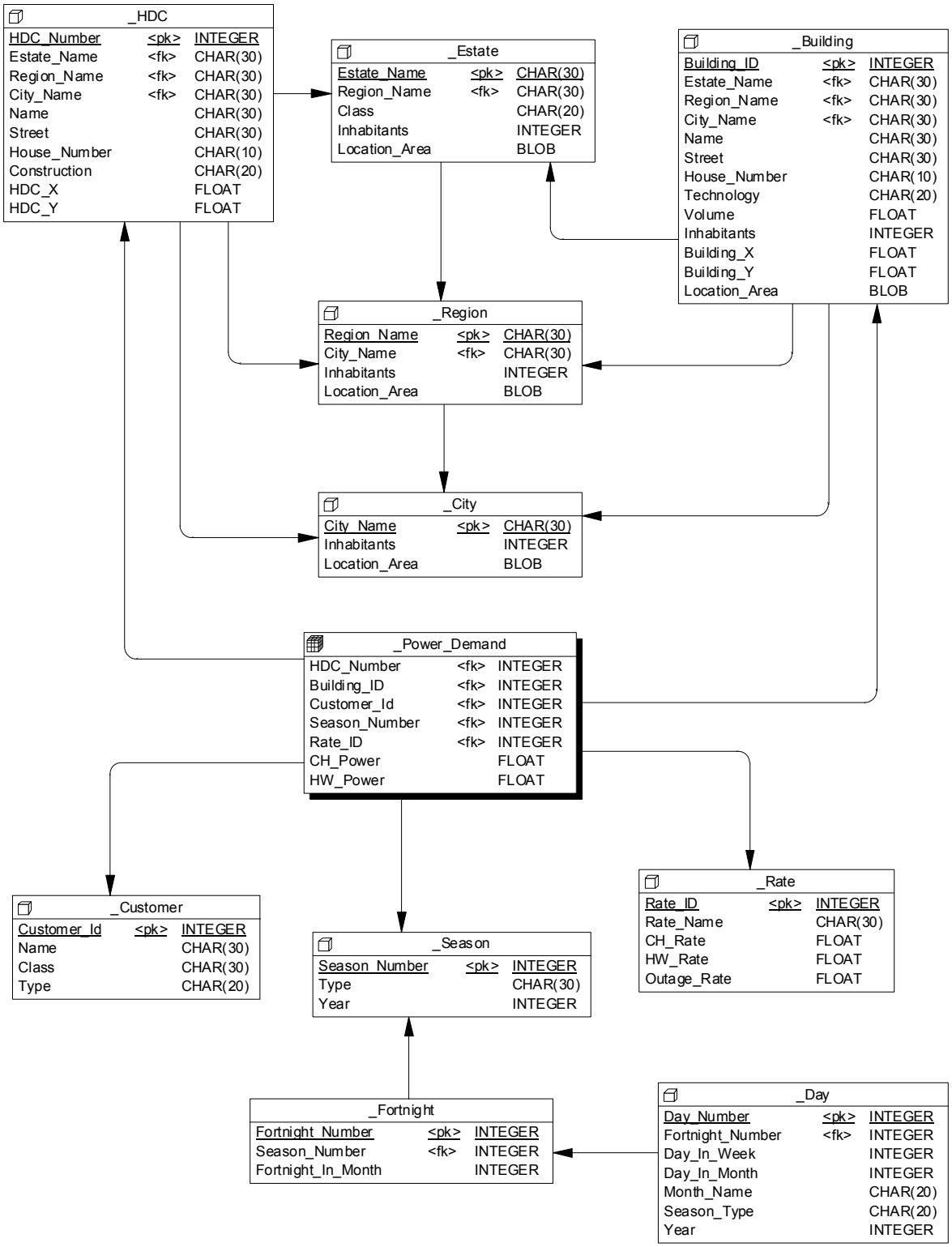


Fig. II.2 The **Power Demand** schema.

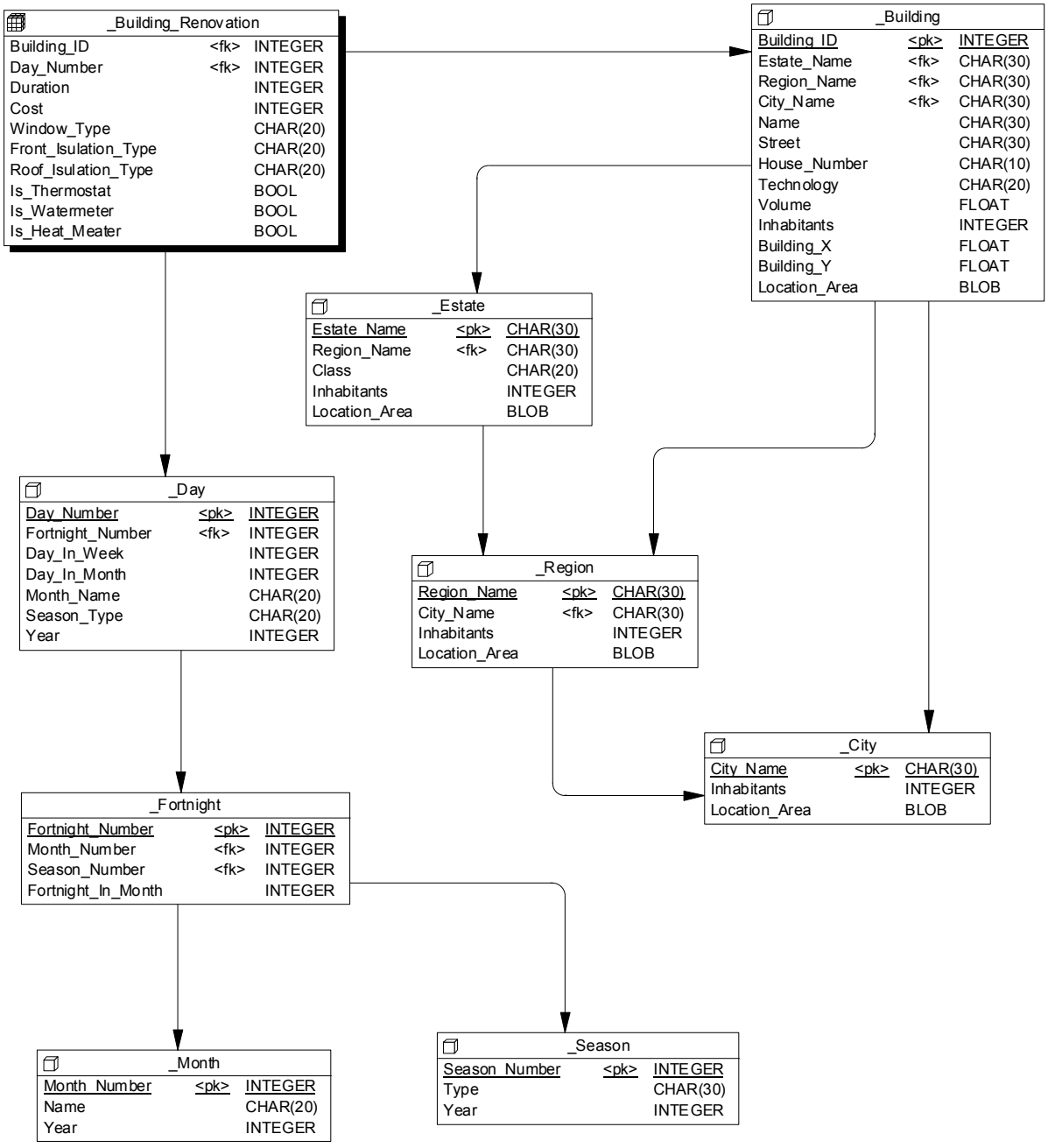


Fig. II.3 The **Building Renovation** schema.

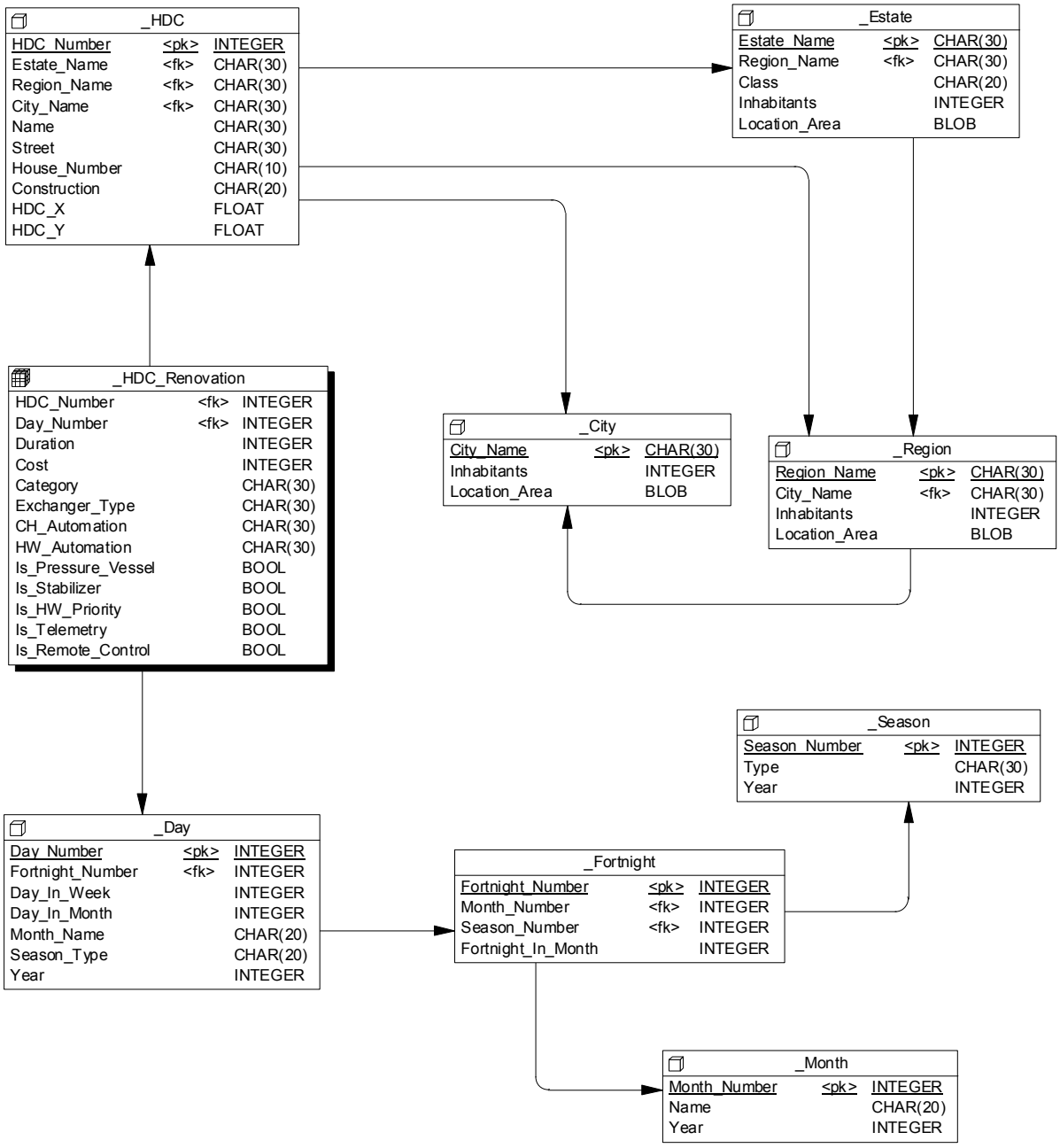


Fig. II.4 The **HDC Renovation** schema.

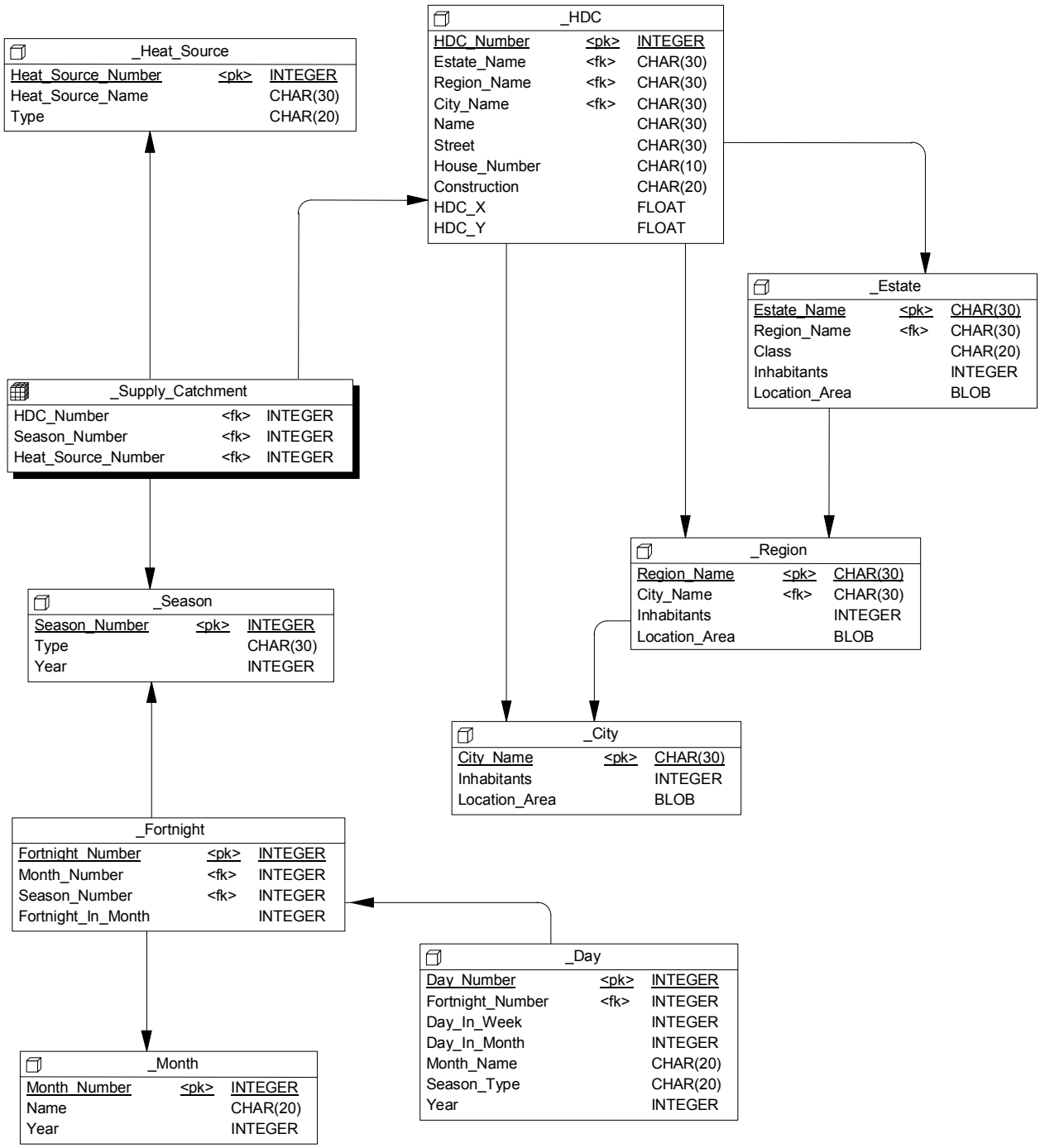


Fig. II.5 The **Supply Catchments** schema.

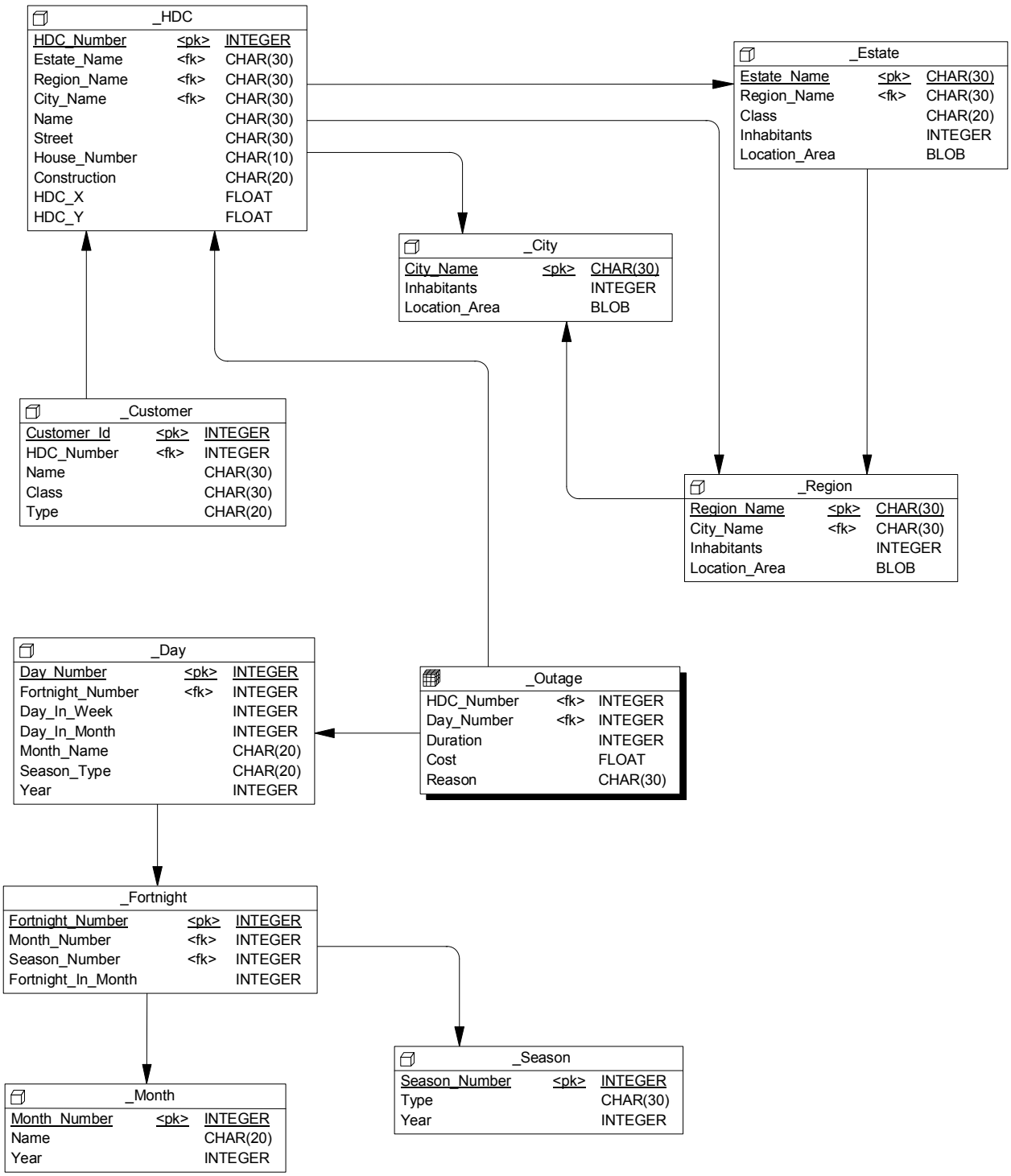


Fig. II.6 The **Outage** schema.

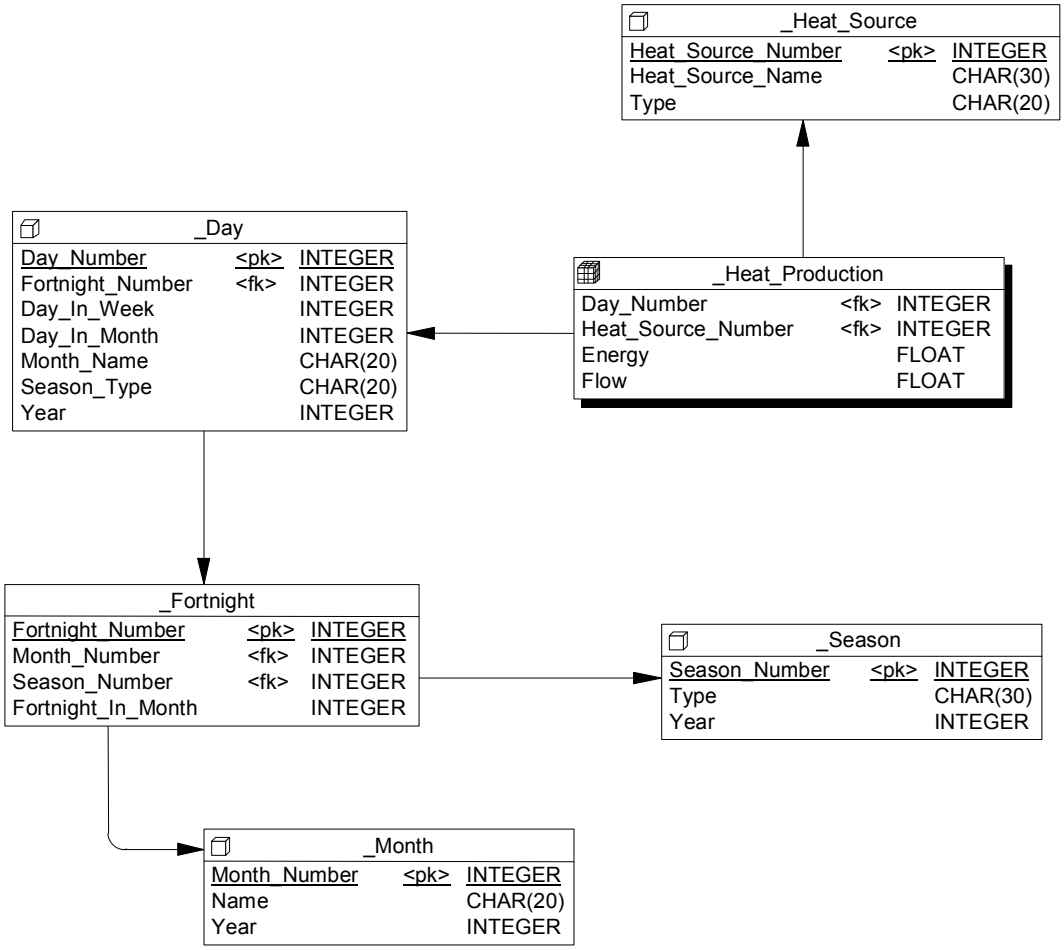


Fig. II.7 The **Heat Production** schema.

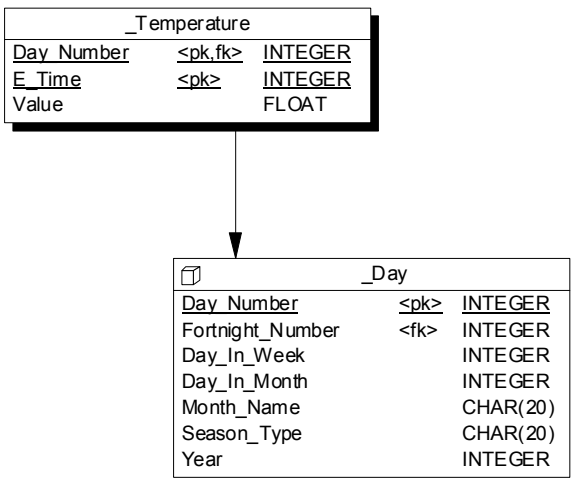


Fig. II.8 The **Temperature** schema.

Details of the physical design

The following hierarchies of the data model were defined and used for the definitions of compound surrogates:

```
Building    SURROGATE cs_bld_spat COMPOUND (
            city_name SIBLINGS 1,
            region_name SIBLINGS 20,
            estate_name SIBLINGS 20,
            building_id SIBLINGS 1000 )

HDC         SURROGATE cs_hdc_spat COMPOUND (
            city_name SIBLINGS 1,
            region_name SIBLINGS 20,
            estate_name SIBLINGS 20,
            hdc_number SIBLINGS 1000 )

Season      SURROGATE cs_season COMPOUND (
            year SIBLINGS 30,
            type SIBLINGS 5,
            season_number SIBLINGS 1 )

Day         SURROGATE cs_day_adv COMPOUND (
            year SIBLINGS 30,
            season_type SIBLINGS 5,
            month_in_year SIBLINGS 12,
            fortnight_number SIBLINGS 6,
            day_number SIBLINGS 16 )

Customer    SURROGATE cs_cust COMPOUND (
            class SIBLINGS 10,
            type SIBLINGS 50,
            customer_id SIBLINGS 10000 )
```

The following compound surrogates were used for the multidimensional hierarchical clustering of the fact tables:

```
Heat_Consumption        HCKEY IS cs_bld, cs_hdc, cs_cust, cs_day
Power_Demand            HCKEY IS cs_bld, cs_hdc, cs_cust, cs_season;
Building_Renovation     HCKEY IS cs_bld, cs_day;
HDC_Renovation          HCKEY IS cs_hdc, cs_day;
Outage                  HCKEY IS cs_hdc, cs_day;
Supply_Catchment        HCKEY IS cs_hdc, cs_season;
```

The following secondary indexes were created to improve the execution of queries referencing spatial coordinates.

```
Building    CREATE INDEX xy_bld_idx ON building (building_x, building_y,
            building_id) HCKEY NOT UNIQUE IS building_x, building_y;
HDC         CREATE INDEX xy_hdc_idx ON hdc (hdc_x, hdc_y, hdc_number)
            HCKEY NOT UNIQUE IS hdc_x, hdc_y;
```

Bibliography

[AADGNRS96] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi, *On the computation of multidimensional aggregates*, In Proceedings of the VLDB Conference, 506-521, India, 1996

[AE99] P. K. Agerwal, J, Ericson, *Geometric range searching and its relatives*, In B. Chazelle, J. E. Goodman, R. Pollack (eds.), Advances in Discrete and Computational Geometry, Contemporary Mathematics, Vol. 23, 1-59, 1999

[AGS97] R. Agrawal, A. Gupta. S. Sarawagi, *Modeling multidimensional databases*, In Proceedings of the ICDC Conference, 232-243, U.K., 1997

[AS96] R. Agrawal, K. Shim, *Developing Tightly-Coupled Data Mining Applications on a Relational Database System*, KDD, 287-290, 1996

[AT00] C. H. Ang, T. C. Tan, *Bitmap R-trees*, Informatica, Vol. 24, No. 2, 2000

[BH85] R. J. Bennett, R. P. Haining, *Spatial structure and spatial interaction: modelling approaches to the statistical analysis of geographical data*, Journal of Royal Statistical Society, Vol. 148, 1-36, 1985

[BK94] T. Brinkhoff, H.-P. Kriegel, *The impact of global clustering on spatial database systems*, In Proceedings of the VLDB Conference, 168-179, 1994

[BKS93] T. Brinkhoff, H. P. Kriegel, B. Seeger, *Efficient processing of spatial joins using R-trees*, In Proceedings of the SIGMOD Conference, 237-246, 1993

[BKSS94] T. Brinkhoff, H.-P. Kriegel, R. Schneider, B. Seeger, *Multi-step processing of spatial joins*, In Proceedings of the SIGMOD Conference, 197-208, 1994

[BLPCL97] Y. Bédard, S. Larrivée, M. J. Proulx, P. Y. Caron, F. Létourneau, *Geospatial data warehousing: positionnement technologique et stratégique*, Rapport pour le Centre de recherche pour la défense de Valcartier, 1997

[BMH00] Y. Bédard, T. Merret, J. Han, *Fundamentals of spatial data warehousing for geographic knowledge discovery*, In H. Miller and J. Han (eds.), Geographic Data Mining and Knowledge Discovery, Taylor & Francis, New York, 2000

[BS97] A. Berson, S. J. Smith, *Data warehousing, data mining, and OLAP*, McGraw-Hill, New York, 1997

[BT79] H. Beguin, J. F. Thisse, *An axiomatic approach to geographical space, Geographical Analysis*, Vol. 11, 325–41, 1979

[BW03] P. Biedacha, M. Wierzchowski, *Analysis of clustering algorithms for large spatial datasets*, Master Thesis, Institute of Computer Science, Warsaw University of Technology, 2003

[BZY96] C. Bailey-Kellogg, F. Zhao, K. Yip, *Spatial aggregation: language and applications*, In Proceedings of the AAI Conference, 517-522, USA, 1996

[Car98] P. Y. Caron, *Étude du potentiel OLAP pour supporter l'analyse spatio-temporelle*, Diplome de Université Laval, 1998

[Car98a] A. Car, *Hierarchical spatial reasoning: a geocomputational method*, Journal of Geocomputation, Vol. 18, 1998

[CD97] S. Chaudhuri, U. Dayal, *An overview of data warehousing and OLAP technology*, ACM SIGMOD Record, Vol. 26, 65-74, 1997

[CF94] A. Car, *Hierarchical spatial reasoning*, In Proceedings of the International Geocomputation Conference, England, 1998

[CHY96] M. S. Chen, J. Han, P.S. Yu, *Data mining: a overview from a database perspective*, IEEE Transactions on Knowledge and Data Engineering, 866-883, 1996

[CO73] A. D. Cliff, J. K. Ord, *Spatial autocorrelation*, Pion, London, 1973

[Cod93] E. F. Codd, *Providing OLAP (On-line Analytical Processing) to user-analysts: an IT mandate*, Technical Report, E. F. Codd and Associates, 1993

[Com79] D. Comer, *The ubiquitous B-Tree*, ACM Computing Surveys, Vol. 11, 1979

[Cre93] N. A. Cressie, *Statistics for spatial data*, John Wiley & Sons, New York, 1993

[Dem99] M. N. DeMers, *Fundamentals of geographic information systems*, John Wiley & Sons, New York, 1999

[Den80] P. J. Denning, *Working sets past and present*, IEEE Transactions on Software Engineering, 64–84, 1980

[DGGKMMORTW00] W. Daszczuk, P. Gawrysiak, T. Gerszberg, M. Kryszkiewicz, J. Mieścicki, M. Muraszkiewicz, M. Okoniewski, H. Rybiński, T.Traczyk, Z. Walczak, *Data mining for technical operation of telecommunications companies: a case study*, In Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, Vol. 3, 64-69, USA, 2000

[EGKS99] M. Ester, S. Gundlach, H. P. Kriegel, J. Sander, *Database primitives for spatial data mining*, In Proceedings of International Conference on Databases in Office, Engineering and Science, Germany, 1999

[EKS97] M. Ester, H. P. Kriegel, J. Sander, *Spatial data mining: a database approach*, In M. Scholl and A, Voisard (eds.), Advances in Spatial Databases, Lecture Notes in Computer Science 1262, Springer, 47-66, 1997

[EFKS00] M. Ester, A. Frommelt, H. P. Kriegel, J. Sander, *Spatial data mining: database primitives, algorithms and efficient DBMS Support,* Journal of Data Mining and Knowledge Discovery, Kluwer Academic Publish, Vol. 4, 193-216, 2000

[EKS01] M. Ester, H. P. Kriegel, J. Sander, *Algorithms and applications for spatial data mining*, In H. J. Miller and J. Han (eds.), Geographic Data Mining and Knowledge Discovery, Taylor and Francis, New York, 2000

[EKS99] M. Ester, H. P. Kriegel, J. Sander, *Knowledge discovery in spatial databases*, Invited paper at German Conference on Artificial Intelligence, Lecture Notes in Computer Science, Vol. 1701, 61-74, Germany, 1999

[EN00] R. Elmasri, S. Navathe, *Fundaments of database systems (second edition)*, Addison-Wesley, New York, 2000

[EP96] J. F. Elder, D. Pregibon, *A statistical perspective on knowledge discovery in databases*, In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.), Advances in Knowledge Discovery and Data Mining, The MIT Press, 83-115, 1996

[FB93] C. Faloutsos, P. Bhagwat, *Declustering using fractals*, In Proceedings of the International Conference on Parallel and Distributed Computing, 18-25, USA, 1993

[FCT01] A. C. Ferreira, M. L. Campos, A. K. Tanaka, *An architecture for spatial and dimensional analysis integration*, In Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, Orlando, 2001

[Fis93] M. M. Fischer, *Expert systems and artificial neural networks for spatial analysis and modelling: essential components for knowledge based geographical information systems*, Geographical Systems, Vol. 1(3), 221-235, 1993

[Fot83] A. S. Fotheringham, *A new set of spatial-interaction models: the theory of competing destinations*, Environment and Planning, Vol. 15, 15–36, 1983

[FR89] C. Faloutsos, S. Roseman, *Fractals for secondary key retrieval*, In Proceedings of the POS Conference, 247-252, USA, 1989

[FR92] C. Faloutsos, Y. Rong, *Dot: a spatial access methods using fractals*, In Proceedings of the ICDE Conference, 152-159, Japan, 1992

[FSU96] M. Fischer, H. J. Scholten, D. Unwind (eds.), *Spatial analytical perspectives on GIS*, GISDATA series No. 4, Taylor & Francis, New York, 1996

[FTSSR04] N. Fidalgo, V. C. Times, J. Silva, F. F. Souza, *GeoDWFrame: A framework for guiding the design of geographical dimensional schemas*, In Proceedings of the Data Warehousing and Knowledge Discovery Conference, 2004

[GBLP96] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, *Data cube: a relational operator generalizing group-by, crosstab, and sub-totals*, Journal of Data Mining and Knowledge Discovery, 29-53, 1996

[GG98] V. Gaede, O. Guenther, *Multidimensional access methods*, ACM Computing Surveys, Vol. 30(2), 170-231, 1998

[GGDW01] L. Gancarz, T. Gulczynski, P. Dobosz, S. Wdowiak, *Demonstrator of the GIS application on MHC/HI*, D8f deliverable of the Edith project, 2001

[GDW01] L. Gancarz, P. Dobosz, S. Wdowiak, *[GIS/DW] Measurement Results*, D9b-d deliverable of the Edith project, 2001

[GHQ95] A. Gupta, V. Harinarayan, D. Quass, *Aggregate-query processing in data warehousing environments*, In Proceedings of the VLDB Conference, Switzerland, 1995

[GJ03] M. F. Goodchild, D. G. Janelle, *Spatially integrated social science*, Oxford University Press, 2003

[GL96] C. Gotsman, M. Lindenbaum, *On the metric properties of discrete space-filling curves*, IEEE Transactions on Image Processing, Vol. 5, 794-797, 1996

[GMPS96] C. Glymour, D. Madigan, D. Preigbon, P. Smyth, *Statistical inference and data mining*, Communications of ACM, Vol. 39(11), 35-41, 1996

[GMZ04] A. Getis, J. Mur, H. Zoller (eds.), *Spatial econometrics and spatial statistics*, Palgrave Macmillan, New York, 2004

[Gon99] M. L. Gonzales, *Spatial OLAP: conquering geography*, database two Magazine, 1999

[GO92] A. Getis, J. K. Ord, *The analysis of spatial association by use of distance statistics*, Geographical Analysis, Vol. 24, 189–206, 1992

[GW03] L. Gancarz, S. Wdowiak, *GIS Warehouse – Federated DBMS environment for spatial data warehousing*, The First Symposium on Databases, Data Warehousing and Knowledge Discovery, (Baden-Baden'03) Scientific Publishers OWT, Poznań, 2003

[HA03] J. E. Harmon, S. J. Anderson, *The design and implementation of geographic information systems*, John Willey & Sons, New York, 2003

[Han98] D. Hand, *Data mining: statistics and more*, The American Statistician, Vol. 52(2), 112-118, 1998

[Han03] R. P. Hanning, *Spatial data analysis: theory and practice*, Cambridge University Press, 2003

[HF94] J. Han, Y. Fu, *Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases*, In Proceedings of the AAAI'94 Workshop of Knowledge Discovery in Databases (KDD'94), 157-168, USA, 1994

[HFP95] J. Hellerstein, J. Naughton, A. Pfeffer, *Generalized search trees for database systems*, In Proceedings of the VLDB Conference, 562–573, Switzerland, 1995

[HK01] J. W. Han, M. Kamber, *Data mining: concepts and techniques*, Morgan Kaufmann Publishers, New York, 2001

[HKS97] J. Han, K. Koperski, N. Stefanovic, *GeoMiner: A system prototype for spatial data mining*, In Proceedings of the SIGMOD Conference, 553-556, USA, 1997

[HKT00] J. Han, M. Kamber, A. K. H. Tung, *Spatial clustering methods in data mining: a survey*, H. Miller and J. Han (eds.), Geographic Data Mining and Knowledge Discovery, Taylor & Francis, New York, 2000

[HM02] C. Hurtado, A. Mendelzon, *Reasoning about summarizability in heterogeneous multidimensional schemas*, In Proceedings of the ICDT Conference, UK, 2001

[HRU96] V. Harinarayan, A. Rajaraman, J. Ullman, *Implementing data cubes efficiently*, In Proceedings of the SIGMOD Conference, Canada, 1996

[HSK98] J. Han, N. Stefanovic, K. Koperski, *Selective materialization: an efficient method for spatial data cube construction*, In Proceedings of the PAKDD Conference, 144–158, 1998

[HZ98] X. Huang, F. Zhao, *Finding structures in weather maps*, Technical Report OSU-CISRC-3/98-TR11, Department of Computer and Information Science, Ohio State University, 1998

[IM96] T. Imielinski, H. Mannila, *A database perspective on knowledge discovery*, Communications of ACM, Vol. 99 (11), 1996

[Imo96] W. H. Immon, *Building the data warehouse*, John Wiley & Sons, New York, 1996

[Jag91] H. V. Jagadish, *Linear clustering of objects with multiple attributes*, In Proceedings of the SIGMOD Conference, 332-342, USA, 1991

[KF94] I. Kamel, C. Faloutsos, *Hilbert R-tree - an improved R-tree using fractals*, In Proceedings of the VLDB Conference, 500-509, Chile, 1994

[Kim01] R. Kimball, *Spatially enabling your data warehouse*, Intelligent Enterprise Magazine, January 2001

[Kim96] R. Kimball, *Data warehouse toolkit*, John Wiley & Sons, New York, 1996

[Kim99] W. Kim, *I/0 problems in preparing data for data warehousing and data mining*, Journal of Object-Oriented Programming, SIGS Publications, Vol. 9, 13-7, 1999

[KH95] K. Koperski, J. Han, *Discovery of spatial association rules in geographic information databases*, In Proceedings of the SSD Conference, 47-66, 1995

[KHA96] K. Koperski, J. Han, J. Adhikary, *Spatial data mining: progress and challenges survey paper*, In Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Canada, 1996

[KHS98] K. Koperski, J. Han, N. Stefanovic, *An efficient two-step method for classification of spatial data*, In Proceedings of the International Symposium on Spatial Data Handling, Canada, 1998

[KMP03] Z. Królikowski, M. Morzy, J. Perek, *Set-oriented indexes for data mining queries*, In Proceedings of the ICEIS Conference , 316-323, 2003

[KN95] E. Knorr, R. T. Ng, *Applying computational geometry concepts to discovering spatial aggregate proximity relationships*, Technical Report, University of British Columbia, 1995

[KTSPMRFEB02] N. Karayannidis, A. Tsois, T. K. Sellis, R. Pieringer, V. Markl, R. Ramsak, R. Fenk, K. Elhardt, R. Bayer, *Processing star queries on hierarchically-clustered fact tables,* In Proceedings of the VLDB Conference, 730-741, Hong Kong, 2002

[Kub01] P. Kuba, *Data structures for spatial data mining*, Technical report FIMU-RS-2001-05, Masaryk University, 2001

[Kum94] A. Kumar, *A study of spatial clustering techniques*, In Proceedings of the DEXA Conference, Vol. 856, 57-71, 1994

[LAW98] W. Lehner, J. Albrecht, H. Wedekind, *Normal forms for multidimensional databases*, In Proceedings of the SSDBM Conference, 63-72, 1998

[LBMS02] B. List, R. Bruckner, K. Machaczek, J. Shiefer, *A comparison of data warehouse development methodologies. Case study of the process warehouse*, In Proceedings of the DEXA Conference, 2002

[LG92] T. K. Ling, C. H. Goh, *Logical database design with inclusion dependencies*, In Proceedings of the ICDE Conference, 642-649, USA, 1992

[LHO93] W. Lu, J. Han, B. C. Ooi, *Discovery of general knowledge in large spatial databases*, Workshop on Geographic Information Systems, 275-289, Singapore, 1993

[LS97] H.-J. Lenz, A. Shoshani, *Summarizability in OLAP and Statistical Data Bases*, In Proceedings of the SSDBM Conference, 132-143, 1997

[MAK02] M. F. Mokbel, W. G. Aref, I. Kamel, *Performance of multi-dimensional space-filling curves*, In Proceedings of the International Symposium on Advances in Geographic Information Systems, 149-154, USA, 2002

[Mar99] V. Markl, *MISTRAL: Processing relational queries using a multidimensional access technique*, Ph.D. Thesis, Technische Universität München, 1999

[MB97] V. Markl, R. Bayer, *The UB-tree: performance on multidimensional range searching*, Technical Report TUM-I9814, TU Munchen, 1997

[MD86] G. Mitchison, R. Durbin, *Optimal numberings of an NxN arrays*, Journal on Discrete and Algebraic Methods, Vol 7(4), 571–582, 1986

[MK01] A. M. MacEachren, M. J. Kraak, *Research challenges in geovisualization*, Cartography Geographic Information Science, Vol. 28(l), 2001

[MH01] H. J. Miller, J. Han (eds.), *Geographic data mining and knowledge discovery*, Taylor and Francis, New York, 2001

[Mil04] H. J. Miller, *Tobler's First Law and spatial analysis*, Annals of the Association of American Geographers, Vol. 94 (2), 284-289, 2004

[MJFS99] B. Moon, H. V. Jagadish, C. Faloutsos, J. H. Saltz, *Analysis of the clustering properties of the Hilbert space-filling curve*, Technical Report 99-10, University of Arizona, 1999

[MMK04] M. Morzy, T. Morzy, Z. Królikowski, *Incremental association rule mining using materialized data mining views*, In Proceeding of the ADVIS Conference, 77-87, 2004

[MMNM03] M. Morzy, T. Morzy, A. Nanopoulos, Y. Manolopoulos, *Hierarchical Bitmap index: an efficient and scalable indexing technique for set-valued attributes*, In Proceedings of the ADBIS Conference, 236-252, 2003

[MMWZ05] M. Morzy, T. Morzy, M. Wojciechowski, M. Zakrzewicz, *Incremental data mining using concurrent online refresh of materialized data mining views*, In Proceedings of the DaWaK Conference, 295-304, 2005

[MW03] H. J. Miller, E. A. Wentz, *Representation and spatial analysis in geographic information systems*, Annals of the Association of American Geographers, Vol. 93(3), 574-594, 2003

[MZ04] E. Malinowski, E. Zimanyi, *Representing spatiality in a conceptual multidimensional model*, In Proceedings of the GIS Conference, 12-21, 2004

[OGG01] M. Okoniewski, Ł. Gancarz, P. Gawrysiak, *Mining multi-dimensional quantitative associations*, in Lecture Notes in Computer Science 2543 Springer, 2001

[Ope92] S. Openshaw, *Learning to live with errors in spatial databases*, In M. Goodchild, S. Gopal (eds.), The Accuracy of Spatial Databases, Taylor and Francis, New York, 263-276, 1992

[PF91] G. Piatetsky-Shapiro, W. J. Frawley (eds.), *Knowledge discovery in databases*, AAAI/MIT Press, 1991

[PE97] D. Papadias, M. J. Egenhofer, *Algorithms for hierarchical spatial reasoning*, GeoInformatica, Vol. 1(3), 251-273, 1997

[PERMFBKTS03] R. Pieringer, K. Elhardt, F. Ramsak, V. Markl, R. Fenk, R. Bayer, N. Karayannidis, A. Tsois, T. K. Sellis, *Combining hierarchy encoding and pre-grouping: intelligent grouping in star join processing*, In Proceedings of the ICDE Conference, 329-340, India, 2003

[PKK92] A. Perez, S. Kamata, E. Kawaguchi, *Peano scanning of arbitrary size images*, In Proceedings of the Conference on Pattern Recognition, 565–568, 1992

[PKZT01] D. Papdias, P. Kalnis, J. Zhang, Y. Tao, *Efficient OLAP operations in spatial data warehouse*, In Proceedings of the SSTD Conference, 2001

[PZ04] S. Prasher, X. Zhou, *Multiresolution amalgamation: dynamic spatial data cube generation*, In Proceedings of the Australasian Database Conference, 103-111, 2004

[Raf03] M. Rafanelli (ed.), *Multidimensional databases: problems and solutions*, Idea Group Publishing, 2003

[RBM01] S. Rivest, Y. Bédard, P. Marchand, *Towards better support for spatial decision-making: defining the characteristics of spatial on-line analytical processing (SOLAP)*, Geomatica, Vol. 55(4), 539-555, 2001

[RBPN03] S. Rivest, Y. Bedard, M. J. Proulx, N. Nadeau, *SOLAP: A new type of interface to support spatio-temporal multidimensional data exploration and analysis*, International Society of Photogrammetry and Remote Sensing, Joint Workshop of WG II/5, IV/1 and IV/2 on “Spatial, Temporal and Multidimensional data modelling and Analysis”, Canada, 2003

[RMFZEB00] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, R. Bayer, *Integrating the UB-tree into a database system kernel*, In Proceedings of the VLDB Conference, 263-272, Egipt, 2000

[PMRB01] R. Pieringer, V. Markl, F. Ramsak, R. Bayer, *HINTA: A linearization algorithm for physical clustering of complex OLAP hierarchies*, In Proceedings of the DMDW Conference, 11, Switzerland, 2001

[Rot91] D. Rotem, *Spatial join indices*, In Proceedings of the ICDE Conference, 500-509, 1991

[RS90] M. Rafanelli, A. Shoshani, *STORM: A statistical object representation model*, In Proceedings of the SSDBM Conference, 14–29, USA, 1990

[RSV02] P. Rigaux, M. School, A. Voisard, *Spatial databases with application to GIS*, Morgan Kaufmann Publishers, New York, 2002

[Sag94] H. Sagan, *Space-filling curves*, Springer-Verlag, New York, 1994

[SH01] S. Shekhar, Y. Huang, *Discovering spatial co-location patterns: a summary of results*, In Proceeding of the SSTD Conference, 236-256, 2001

[SHK00] N. Stefanovic, J. Han, K. Koperski, *Object-based selective materialization for efficient implementation of spatial data cubes*, IEEE Transactions on Knowledge and Data Engineering, Vol. 12(6), 2000

[SHWL01] S. Shekhar, Y. Huang, W. Wu, C.T. Lu, *What's special about spatial data mining: three case studies*, In V. Kumar, R. Grossman, C. Kamath, R. Namburu (eds.), Data Mining for Scientific and Engineering Applications, Kluwer Academic Publishers, New York, 2001

[SL91] B. Salzberg, D. B. Lomet, *Spatial database access methods*, SIGMOD Record, Vol. 20(3), 1991

[SL90] A. P. Sheth, J. A. Larson, *Federated database systems for managing distributed, heterogeneous, and autonomous databases*, ACM Computing Surveys, vol. 22 (3), 183-236, 1990

[Ste97] N. Stefanovic, *Design and implementation of on-line analytical processing of spatial data*, Master Thesis, Simon Foster University, Vancouver, 1997

[SU03] D. O'Sullivan, D. Unwin, *Geographic information analysis*, John Wiley & Sons, New York, 2003

[SZHV03] S. Shekhar, P. Zhang, Y. Huang, R. Vatsavai, *Trends in spatial data mining*, In H. Kargupta, A. Joshi, K. Sivakumar, Y. Yesha (eds.), Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT Press, 2003

[Tho97] E. Thomsen, *The OLAP solutions: building multidimensional information systems*, John Wiley & Sons, New York, 1997

[Tob70] W. R. Tobler, *A computer movie simulating urban growth in the Detroit region*, Economic Geography, Vol. 46, 234-60, 1970

[Vit01] J. S. Vitter, *External memory algorithms and data structures: dealing with massive data*, ACM Computing Surveys, Vol. 33(2), 209–271, 2001

[Voo91] D. Voorhies, *Space-filling curves and a measure of coherence*, In J. Arvo, Graphics Gems II, Academic Press, 26–30, 1991

[WC04] L. A. Waller, C. A. Gotway, *Applied spatial statistics for public health data*, John Wiley & Sons, New York, 2004

[YZ96] K. M. Yip, F. Zhao, *Spatial aggregation: theory and applications*, Journal of Artificial Intelligence Research, Vol. 5, 1996

[XZJ01] J. Xiao, Y. Zhang, X. Jia, *Clustering non-uniform-sized spatial objects to reduce I/O cost for spatial-join processing*, The Computer Journal, Vol. 44, 384-397, 2001