The VLDB Journal (2004) 13: 1–21 / Digital Object Identifier (DOI) 10.1007/s00778-003-0091-3

Multidimensional data modeling for location-based services

Christian S. Jensen, Augustas Kligys, Torben Bach Pedersen, Igor Timko

Aalborg University, Department of Computer Science, Fredrik Bajers Vej 7E, 9220 Aalborg Øst, Denmark;
e-mail: {csj,augustas,tbp,timko}@cs.auc.dk

Edited by J. Veijalainen. Received: 28 September 2002 / Accepted: 5 April 2003
Published online: August 12, 2003 – © Springer-Verlag 2003

Abstract. With the recent and continuing advances in areas such as wireless communications and positioning technologies, mobile, location-based services are becoming possible.

Such services deliver location-dependent content to their users. More specifically, these services may capture the movements and requests of their users in multidimensional databases, i.e., data warehouses, and content delivery may be based on the results of complex queries on these data warehouses. Such queries aggregate detailed data in order to find useful patterns, e.g., in the interaction of a particular user with the services.

The application of multidimensional technology in this context poses a range of new challenges. The specific challenge addressed here concerns the provision of an appropriate multidimensional data model. In particular, the paper extends an existing multidimensional data model and algebraic query language to accommodate spatial values that exhibit partial containment relationships instead of the total containment relationships normally assumed in multidimensional data models. Partial containment introduces imprecision in aggregation paths. The paper proposes a method for evaluating the imprecision of such paths. The paper also offers transformations of dimension hierarchies with partial containment relationships to simple hierarchies, to which existing precomputation techniques are applicable.

Keywords: Location-based services – Multidimensional data – Data modeling – Partial containment

1 Introduction

Several trends in hardware technologies combine to enable the deployment of mobile, location-based e-services. These trends include continued advances in the miniaturization of electronics technologies, in display devices, and in wireless communications. Other trends include the improved performance of general computing technologies and the general improvement in the performance/price ratio of electronics. Perhaps most importantly, geopositioning is becoming increasingly available and accurate.

Correspondence to: I. Timko

It is expected that the coming years will witness very large quantities of wirelessly Internet-worked objects that are location-enabled and capable of movement to varying degrees. Examples of objects of interest here include consumers using Internet-enabled mobile-phone terminals and personal digital assistants, tourists carrying online and position-aware "cameras" and "wrist watches," vehicles with computing and navigation equipment, etc.

These developments pave the way to a range of qualitatively new types of Internet-based services [12]. These types of services—which either make little sense or are of limited interest in the traditional context of fixed-location, desktop computing—include the following: traffic coordination, management, and way-finding, location-aware advertising, integrated information services, e.g., tourist services, safety-related services, and location-based games that merge virtual and physical spaces.

A single generic scenario may be envisioned for these location-based services. Moving service users disclose their positional information to services, which in turn use this and other information to provide specific content and functionality. The services capture the requests they receive, including their geographical origins, in a multidimensional database, i.e., a *data warehouse*. We note that the privacy of service users is a concern and that legislation is available that regulates this aspect (e.g., [8]). We are aware that some service providers require each customer to enter into an explicit agreement with the provider that covers the provider's possible use of the customer's location data.

Querying the resulting data warehouse enables the services to analyze their interactions with the users, thus allowing the services to customize their interactions with the users. As a result, each user receives a service customized to the user's specific preferences and needs and current situation. For example, the query "show the number of requests per district for user X" provides valuable information about the geographical behavior of user X. In addition, the accumulated data are used by the service providers for delayed modification of the services provided and for longer-term strategic decision making. For example, the query "show the number of requests per city per quarter for the last year" gives information about the changes in service use for different cities over time.

A data warehouse [1,15,25] is a large repository that organizes data specifically for analytical purposes by employing a multidimensional view of data. Multidimensional models view a central data element for the given domain, e.g., a service request, as a *fact* (also termed a *cell*), which is uniquely defined by a combination of *dimension values*, each of which stems from one of a number of hierarchically organized *dimensions*. Typical dimensions are the *location* from which the request originates, the profile of the *user* that has issued the request, and the *time* of the request. Dimensions are organized as *hierarchies* of *levels*, also termed *categories*. For example, the time dimension may have Day, Week, Month, Quarter, and Year levels.

The multidimensional view is particularly well suited for complex data analyses, which include data aggregation [25], i.e., the counting of facts that are characterized by specific values from the dimensions. Typical operations on multidimensional data warehouses use the dimension hierarchies to dynamically change the level of detail in order to gain an understanding of a particular phenomenon.

If more detail is desired, e.g., to understand why the number of requests dropped sharply in Q4 2002, a “drill down” is performed, where numbers of requests per month are used in place of numbers of requests per quarter. If the opposite is true, i.e., less detail is desired in order to get a better overview, a “roll up” is performed. This means that it is crucial for multidimensional data warehouses to have well-designed dimension hierarchies that capture the useful levels of detail. We assume this kind of data analysis in our scenario.

The scenario is realistic. For example, the Danish company Euman A/S [7] has developed and deployed a service delivery system capable of providing location-based services. Although the current object-relational database underlying the system is not optimized for complex data analyses, the database contains data, e.g., data on geo-referenced transportation infrastructure, that can be used to implement a multidimensional data warehouse. This in turn enables complex multidimensional analyses of the interactions among the services delivered by the system and the users.

The scenario entails the capture of spatial data in a multidimensional data warehouse. This poses new challenges. For example, an appropriate data model should support irregular, so-called nonnormalized, dimension hierarchies [26] where the hierarchies are not balanced trees. Next, while dimension values in conventional multidimensional data models either are disjoint or exhibit total containment relationships, partial containment is prevalent in spatial data. For example, a roadway that extends from a city into a rural area is only partially contained in the city. Thus, partial containments between dimension values, i.e., location values such as roadways and cities, must be supported by the data model. The inclusion of advanced modeling facilities in a data model should not preclude the provision of an efficient implementation of the data model. In a multidimensional context, this implies that conventional preaggregation techniques [32] should remain applicable.

This paper first analyzes the mobile e-service application domain, formulating requirements to a data model. It then presents a new multidimensional data model with an accompanying algebraic query language that arguably meets the requirements. Notably, the model supports nonnormalized hierarchies and partial containment. Partial containment, together with its transitivity property, is the key new aspect of the model, and the paper treats this topic in detail.

Partial containment introduces additional imprecision in aggregation paths. Because it is important to be able to evaluate the imprecision of a path (e.g., for choosing the most precise one), the paper offers a path imprecision evaluation method. Practical preaggregation, i.e., precomputation of select aggregate results that can be reused to obtain other aggregates, is a technique that is essential for efficiently implementing any multidimensional data model, including the one proposed here. We thus propose algorithms for making its dimension hierarchies onto, covering, and aggregation strict. This enables the application of standard preaggregation techniques in an implementation of the model.

This paper is a revised and substantially extended version of an earlier conference paper [13]. In particular, the contents of Sects. 4.7, 5, 6, 7, and the appendix are entirely new.

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 describes key requirements of a multidimensional data model for location-based services, and Sect. 4 then presents a data model that aims to satisfy those requirements. Section 5 completes the description of the model by defining its algebraic query language. Section 6 presents the method for evaluating the imprecision of an aggregation path. Section 7 provides an overview of the algorithms for normalizing dimension hierarchies. Section 8 concludes and points to future work. The appendix provides the details of the normalization algorithms. The paper can be read and understood without reading the appendix.

2 Related work

In the domain of spatial data modeling, most related scientific and industrial work is dedicated to object-relational extensions of SQL. In particular, Egenhofer [6] proposed a spatial model and query language that compared favorably to several related languages. A spatiotemporal model and a query language were formally defined by Güting et al. [10]. Dedicated designs of spatial relational algebras with formal semantics were also proposed by Scholl and Voisard [31] and Gorgano et al. [2]. As for industrial standards, the Open GIS Consortium [20] adopted a specification [19] for implementation of a spatial SQL extension, and Oracle Spatial [17] conforms to this specification.

In essence, these works develop means of analyzing spatial data, given, among other things, varying relationships between spatial objects, e.g., overlapping, containment, etc. However, we believe that the object-relational view of data does not fully support complex data aggregation. In part, this is due to the lack of hierarchies. We therefore develop a multidimensional data model and algebra that are capable of capturing an advanced kind of relationship between spatial objects, i.e., partial containment relationships.

Multidimensional data warehouses [1,15,25] are generally accepted as the most powerful platform for data analysis in terms of expressive power and performance. Expressive power is achieved mainly by using the multidimensional concepts of dimensions and hierarchies. Good performance is achieved primarily by using *preaggregation*, i.e., storing precomputed results of aggregate queries and using these to answer new queries more efficiently. However, current multidimensional database technology does not support the complex structures needed to handle complex spatial information.

To the authors' knowledge, no other existing multidimensional data model offers built-in support for partial containment hierarchies. This deficiency is also suggested by surveys of multidimensional data models [26, 33]. However, rather than proposing an entirely new multidimensional data model and query language, the proposed model and query language extend a previously proposed multidimensional model and algebra [23,26]. The model that we extend was chosen because it is formally defined and because it compares favorably to 14 related data models [26]. The paper's algorithms for the normalization of partial containment dimension hierarchies extend algorithms presented by Pedersen et al. [22,24] for use with the model being extended.

Pedersen and Tryfona [27] propose a slightly different approach to the multidimensional modeling of spatial data. They ignore partial containment relationships among hierarchy values and instead consider spatial facts, i.e., values characterized by hierarchy values, that are two-dimensional regions. Their focus is on how to support practical preaggregation with such overlapping facts. The conceptual model underlying their work is the model being extended here.

Ferri et al. [9] propose a method to couple a multidimensional data model with a Geographical Information System (GIS) to combine the strengths of these technologies. Modern GISs such as ArcInfo [14] and MapInfo [3] provide some support for complex geostatistical and spatial analysis. Currently, the systems neither directly support multidimensional data modeling nor use preaggregation. Incorporation of these features into the systems would enable complex data aggregation queries and consequently enhance analytical capabilities of the systems. As a result, it would be possible to use the systems in our scenario of customizing location-based services to users' needs.

The area of "imperfect" data has received a great deal of attention in general as well as specialized database contexts [5]. Within multidimensional databases, work has been done on irregular multidimensional data [4,15,26,29] and the associated summarizability problems [16,26,28]. However, none of these works consider partial containment dimension hierarchies.

In the industrial domain, linear referencing [30] has been used quite widely for the positioning of business data, e.g., user locations and other points of interest, located along linear elements (e.g., roadways) in transportation infrastructures. For example, Oracle Spatial [17] offers support for linear referencing. In addition, a generic data model [18] has been recommended for the capture of different aspects of entire transportation infrastructures and related business data. By applying the multidimensional view on linearly referenced data, we would enable complex aggregation queries on this kind of business data. In order to achieve this, it is necessary that a multidimensional model provide support for nonnormalized partial containment hierarchies. For this reason, we believe that our model, which supports hierarchies of this type, could serve as a basis for complex analysis of linearly referenced data.

Finally, the World Wide Web Consortium [35] has recently published a draft specification [34] of an XML-based language for describing location information. In our scenario, that language could facilitate data exchange.

3 Usage scenario and requirements

We introduce a prototypical usage scenario for a multidimensional database in the context of location-based services, and we use this scenario to illustrate important requirements to a multidimensional data model. The scenario is also used for exemplification throughout the paper.

3.1 Usage scenario

In our prototypical usage scenario, a user issues a service request that is characterized by a combination of values including values that capture the time and date of the request, the profile of the user, and the location from which the request originates.

The ER diagram in Fig. 1 describes location values that may be used for capturing the origins of service requests as well as location values that may prove useful in analyses of service requests that involve the origins of the requests. The meanings of most of the entity types found in the diagram follow from the names of the types, though there are some exceptions, namely, the entity type IP Address represents fixed IP addresses, e.g., those of office or home desktop computers, the type Cell represents wireless network cells, the type District represents city districts, and the type Roadway represents all types of roads.

The schema is meant to illustrate certain problematic properties of locations, such as partial containment hierarchies, while still maintaining simplicity. The schema is not meant to capture all aspects of locations. For example, generic international locations or locations in oceans are not handled. For information on how to model international locations, we refer to the literature [15].

The diagram uses its naming convention to distinguish between two different types of binary relationships between entities, namely, full and partial containment relationships among the spatial extents of the related entities. In the diagram, an "F" in a relationship name indicates a total, or *full*, containment relationship type, and a "P" indicates that only *partial* containment may be assumed.

For example, consider the relationship type co-F-ro between entity types Coordinate and Roadway, and consider ro-P-di, which relates Roadway and District. The meaning is that a coordinate is either fully contained or not contained in a roadway, which, in turn, may be fully or (only) partially contained in a district.

Note that all the relationship types in the diagram are stored relationship types. For example, with the relationship types co-F-ro and ro-P-di present in the diagram, the relationship type co-F-di may seem redundant. However, this third relationship type captures nonredundant information. For example, some coordinates are not contained in any roadways, but are still contained in districts.

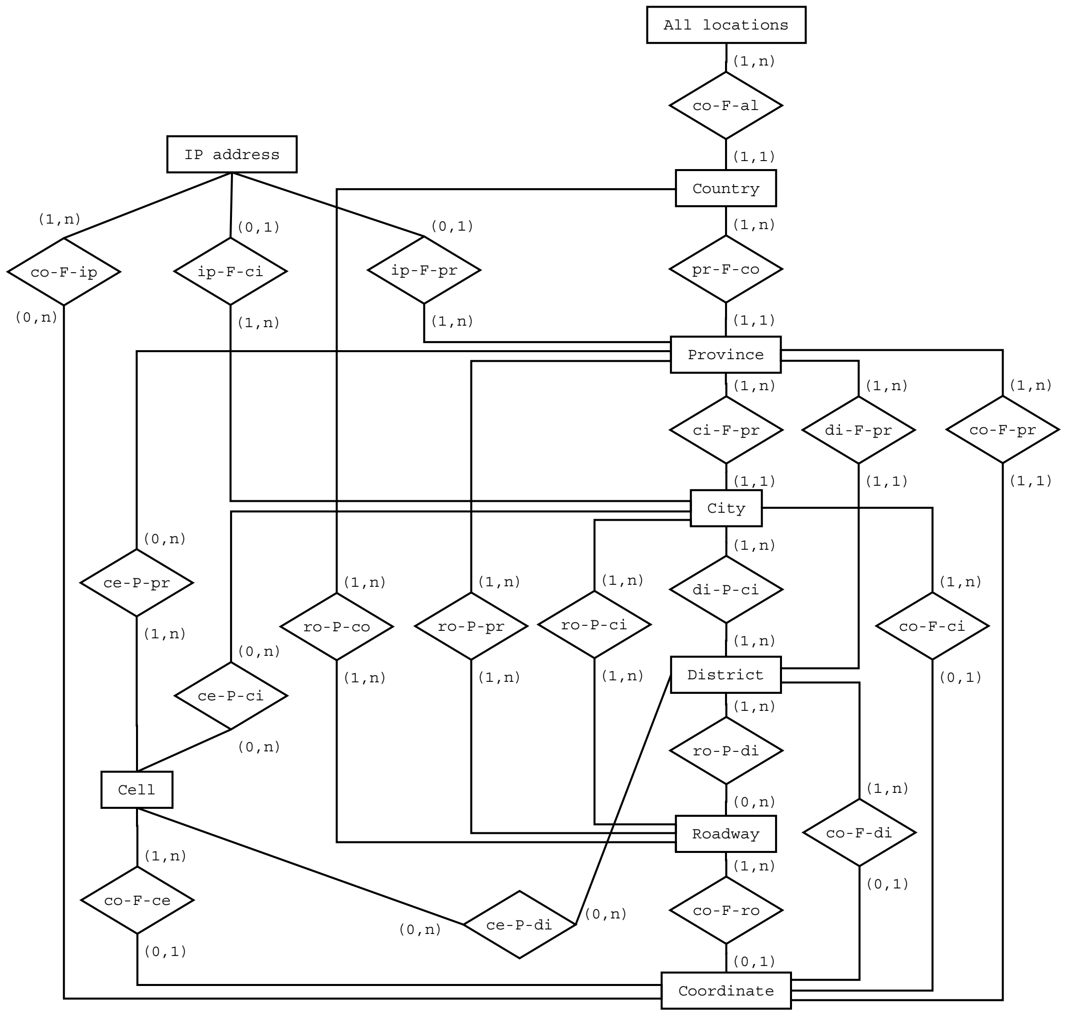


Fig. 1. Location ER diagram

The existence of a partial containment relationship type between two entity types in the case study also implies the existence of a full containment relationship type between these two entity types, as full containment is a special case of partial containment. The intuition is that if objects of one type may be partially contained in objects of another type, then some objects of the former type may also be fully contained in objects of the latter type, although fewer objects will satisfy this relationship.

In a multidimensional schema, user requests will be modeled as facts and the values that *characterize* the user requests are organized into dimensions. For our scenario, we will have three dimensions. The TIME dimension captures the time of the user requests and has categories (levels) such as *Second*, *Minute*, and *Hour*. The USER dimension captures aspects of the users issuing the requests. It has categories such as *Spoken Language*, *Personal Interest*, *Actual Age*, and *Main Occupation*. The LOCATION dimension captures the (possibly changing) locations of the users when the users issue requests. Entity types in the Location ER diagram are then represented as categories in the hierarchy of categories that makes up the LOCATION dimension, and relationship types in the Location ER diagram *may* be represented as relationships among categories in the LOCATION dimension. It is normal practice in multidimensional modeling to include only some of the relationships found in the source data, the primary driver being to obtain hierarchies useful for roll-up/drill-down operations [15].

In Sect. 4, we illustrate how the Location ER diagram is mapped to the LOCATION dimension. In particular, the issues involved in deciding how the LOCATION dimension should be modeled are discussed in detail in Sect. 4.7.

As noted above, the presented usage scenario is based on a real-world location-based service delivery system [7]. The database of this system contains the data necessary for implementing a multidimensional data warehouse. For example, the available data on geo-referenced transportation infrastructure can be used to build a LOCATION dimension. Although the available data do not directly capture containment relationships between spatial entities, they can be inferred from the data.

3.2 Data model requirements

Next we discuss the requirements for a multidimensional data model that contends with our usage scenario. While the requirements are all highly relevant to our context, most of them are more general and were formulated earlier. We describe the requirements only briefly and refer to the literature for further detail [26]. Other requirements are given elsewhere [11].

1. *Explicit and multiple hierarchies in dimensions* Dimension values are partitioned into categories of values, and categories are related via containment relationships. For example, coordinates belong to a *Coordinate* category, and *Coordinate* is contained in *Country*, meaning that coordinates are contained in countries. *Explicit* hierarchies are highly useful in data analysis as they are used for aggregating data to the right level of detail in exploratory analyses that use roll-up/drill-down operations [25]. Support for *multiple* hierarchies means that multiple aggregation paths are possible. These are important for a number of reasons. The key reason is that multiple hierarchies exist naturally in much data. Another reason is that these enable better handling of the imprecision in queries caused by partial containment in dimension structures. For example, in the LOCATION dimension, we obtain a more precise result if roadways are rolled up to countries directly than if roadways are rolled up to countries through districts, cities, and provinces.
2. *Partial containment* We have seen that two spatial values may be not only either disjoint or have one contained in the other; they may overlap. A multidimensional data model should provide built-in support for dimensions with partial containment relationships. This will increase the modeling power of the model and enable new kinds of queries. Specifically, we will be able to perform aggregation of data along hierarchies with partial containment (e.g., districts would, though approximately, roll up to cities).
3. *Nonnormalized hierarchies* Situations occur naturally where a hierarchy value has more than one parent, a value has no relationship to any value in the category immediately above it in the dimension hierarchy, or a value has no relationship to any value in any category below it. For example, a roadway value may be related to several district parent values, and a city value may have no cell child values.
4. *Different levels of granularity* In our scenario, user requests are characterized by values drawn from the dimensions. Support for different levels of granularity enables a request to refer to other values than those in the category at the lowest level of a dimension hierarchy. For example, the position of the user may be known at the level of a coordinate (precise) or at the level of a mobile phone cell (imprecise).
5. *Many-to-many relationships between facts and dimensions* This requirement implies that a fact may be related to more than one value in a dimension. For example, this is useful in a situation where a request is related to more than one service user.
6. *Handling of imprecision* When facts are characterized by dimension values from different levels, imprecision in the data occurs. In addition, partial containment introduces imprecision. Both types of imprecision may lead to imprecise aggregate query results. In the first case, a result may be imprecise because data for a query is missing. In the second case, the transitive relationships between members of aggregation paths may become imprecise (see Sect. 6 for details), rendering the results of queries imprecise. This calls for means of handling imprecision.

We base our proposal for a new model on an existing data model that satisfies Requirements 1, 3, 4, and 5. Moreover, Requirement 6 is partially satisfied by the algebra associated with the preexisting model. However, Requirement 2 (partial containment) is not met by this or any other existing model.

4 Data model

This section extends the existing multidimensional data model [26] to support partial containment. The section also presents properties of the new model, considers its fulfillment of the requirements, and discusses how to use it when designing dimensions.

4.1 Data model definition: dimension schemas

An *n-dimensional fact schema* is a two-tuple $\mathcal{S} = (\mathcal{F}, \mathcal{D})$, where $\mathcal{F}$ is a *fact type* and $\mathcal{D} = \{\mathcal{T}_i, i = 1, \ldots, n\}$ is a set of *dimension types*. A dimension type $\mathcal{T}$ is a four-tuple $(\mathcal{C}_\mathcal{T}, \sqsubset_\mathcal{T}, \top_\mathcal{T}, \bot_\mathcal{T})$, where $\mathcal{C}_\mathcal{T} = \{\mathcal{C}_j, j = 1, \ldots, k\}$ are *category types* of the dimension type $\mathcal{T}$, and $\sqsubset_\mathcal{T}$ is a partial order on the set $\mathcal{C}_\mathcal{T}$. Next, $\top_\mathcal{T}$ is the top element of the order, meaning that $\forall \mathcal{C} \in \mathcal{C}_\mathcal{T} \setminus \{\top_\mathcal{T}\}$ $(\mathcal{C} \sqsubset_\mathcal{T} \top_\mathcal{T})$. Symbol $\bot_\mathcal{T}$ is the bottom element; the precise meaning of this will be described shortly. A function $Anc : \mathcal{C}_\mathcal{T} \rightarrowtail 2^{\mathcal{C}_\mathcal{T}}$ is defined that returns the set of immediate ancestors of a category type $\mathcal{C}_j$. Function $Desc : \mathcal{C}_\mathcal{T} \rightarrowtail 2^{\mathcal{C}_\mathcal{T}}$ returns the set of immediate descendants of $\mathcal{C}_j$. The relation $\sqsubset_\mathcal{T}$ captures the *full* containment relationships between category types.

We extend the definition of a dimension type by introducing an additional relation $\sqsubset^P_\mathcal{T} \subseteq \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}$. This new relation captures the *partial* containment relationships between category types. The properties of the new relation, which are properties of a partial order, are as follows:

1. $\forall \mathcal{C} \in \mathcal{C}_\mathcal{T} (\mathcal{C} \not\sqsubset^P_\mathcal{T} \mathcal{C})$ (antireflexivity)
2. $\forall (\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}$
   $((\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_j) \Rightarrow (\mathcal{C}_j \not\sqsubset^P_\mathcal{T} \mathcal{C}_i))$ (antisymmetry)
3. $\forall (\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}$
   $(((\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubset^P_\mathcal{T} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_k))$ (P-to-P transitivity)

Relations $\sqsubset_\mathcal{T}$ and $\sqsubset^P_\mathcal{T}$ are related as follows:

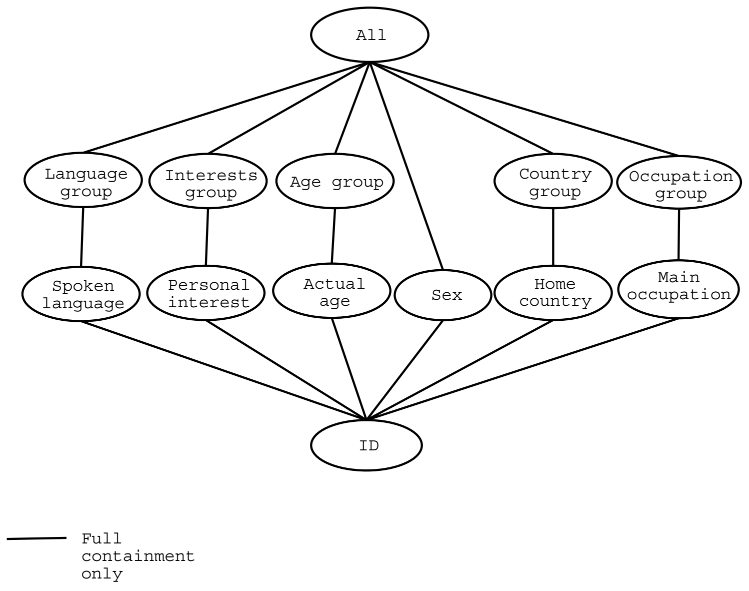
$\forall (\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}$

1. $((\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubset_\mathcal{T} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_k))$ (P-to-F transitivity)
2. $((\mathcal{C}_i \sqsubset_\mathcal{T} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubset^P_\mathcal{T} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubset^P_\mathcal{T} \mathcal{C}_k))$ (F-to-P transitivity)

After the extension, a dimension type $\mathcal{T}$ is a five-tuple:

$$(\mathcal{C}_\mathcal{T}, \sqsubset_\mathcal{T}, \sqsubset^P_\mathcal{T}, \top_\mathcal{T}, \bot_\mathcal{T})$$

We use the notation $\sqsubset^{(P)}_\mathcal{T}$ to indicate the union of the two orders $\sqsubset_\mathcal{T}$ and $\sqsubset^P_\mathcal{T}$. With this notation in place we can define the meaning of $\bot_\mathcal{T}$ being the bottom element: $\forall \mathcal{C} \in \mathcal{C}_\mathcal{T} \setminus \{\bot_\mathcal{T}\}$ $(\bot_\mathcal{T} \sqsubset^{(P)}_\mathcal{T} \mathcal{C})$. The functions Anc^P and $Desc^P$ provide ancestors and descendants based on the $\sqsubset^P_\mathcal{T}$ relation, and functions $Anc^{(P)}$ and $Desc^{(P)}$ provide ancestors and descendants based on both relations.



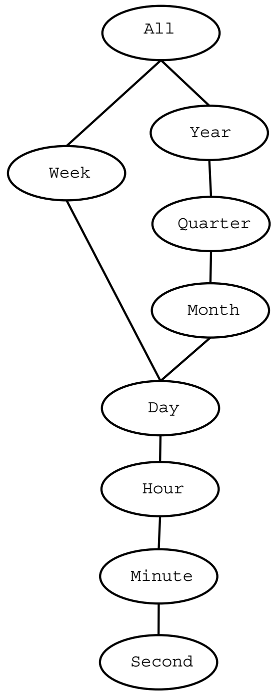


Fig. 2. USER and TIME dimensions

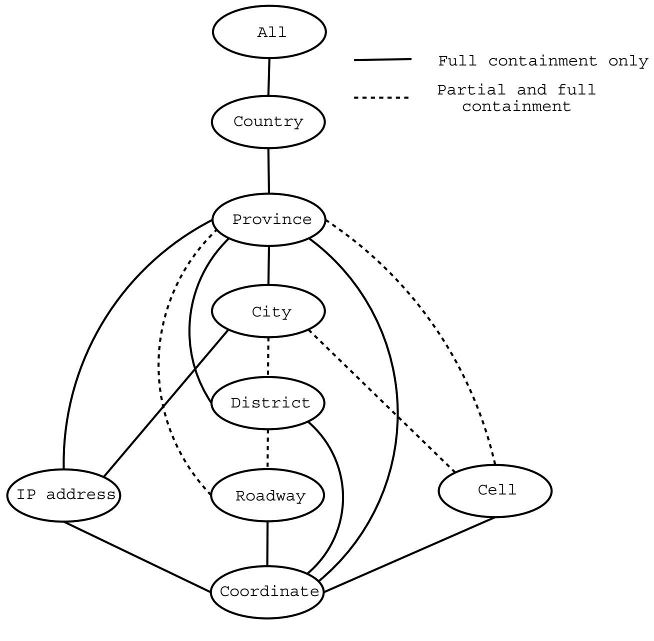


Fig. 3. LOCATION dimension

We use a fact schema to define the structure of a multidimensional data warehouse. The schema is generally capable of capturing some subset of the structure of some domain (in our scenario, the domain of a mobile e-service) at some level of abstraction. The fact schema defines facts as entities of a particular type (in our scenario, all the facts are service requests). Heterogeneous entities that characterize facts (cities, age groups, roadways, years, IP addresses, personal interests, coordinates, minutes, job categories, etc.) are organized into multiple dimensions, e.g., LOCATION and TIME dimensions. In a dimension, each type of entity has a corresponding category type (e.g., *Coordinate*, *City*, etc.). The types are organized into multiple partial and full containment hierarchies, along which the facts will be aggregated.

While these hierarchies reflect some containment hierarchies of the domain (e.g., *Coordinate* $<$ *Cell* $<$ *Province* $<$ *Country* $< \top$ and *Coordinate* $<$ *IP address* $<$ *Province* $<$ *Country* $< \top$), application requirements also impact the design of dimensions. With the new model fully defined, Sect. 4.7 covers dimensional database design based on the model. In essence, if we eventually wish to aggregate facts characterized by entities of type $\mathcal{C}_i$ (in our model, by *dimension values*, as defined later) with respect to entities of type $\mathcal{C}_k$, then we relate these two types. Specifically, if entities of types $\mathcal{C}_i$ and $\mathcal{C}_k$ exhibit full (partial) containment relationships, we introduce the relationship $\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_k$ ($\mathcal{C}_i \sqsubset_{\mathcal{T}}^{P} \mathcal{C}_k$).

The use of relations $\sqsubset_{\mathcal{T}}^{P}$ and $\sqsubset_{\mathcal{T}}$ for building hierarchies of category types is the motivation behind defining them as partial orders. This ensures that the resulting data warehouse schema supports data aggregation. First, for a pair of category types $(\mathcal{C}_i, \mathcal{C}_j)$, the antisymmetry properties ensure that either $\mathcal{C}_i$ is placed higher in the hierarchy than $\mathcal{C}_j$, or vice versa—both variants are not allowed at the same time. This uniquely indicates the direction of aggregation from bottom (category type $\perp_{\mathcal{T}}$) to top (category type $\top_{\mathcal{T}}$). Second, the transitivity enables "comparison" of category types, i.e., if $\mathcal{C}_i$ is lower than $\mathcal{C}_j$ and $\mathcal{C}_j$ is lower than $\mathcal{C}_k$, then $\mathcal{C}_i$ is lower than $\mathcal{C}_k$. This defines possible aggregation paths.

Example 1. In Figs. 2 and 3, we present the result of applying the model to our domain. The figures depict the USER, TIME, and LOCATION dimensions. Nodes denote category types and links between nodes imply relationships between category types.

Since Sect. 3 identifies three dimensions, we have a three-dimensional fact schema $\mathcal{S}_{case} = (\mathcal{F}_{case}, \mathcal{D}_{case})$, where $\mathcal{F}_{case} = Request$ and the set of dimension types is $\mathcal{D}_{case} = \{\mathcal{T}_{loc}, \mathcal{T}_{user}, \mathcal{T}_{time}\}$. The dimension type of the LOCATION dimension is $\mathcal{T}_{loc} = \{\mathcal{C}_{loc}, \sqsubset_{\mathcal{T}_{loc}}, \sqsubset_{\mathcal{T}_{loc}}^{P}, \mathcal{C}_{all}, \mathcal{C}_{coordinate}\}$. As a rule, the relations on set $\mathcal{C}_{loc} = \{\mathcal{C}_{coordinate}, \mathcal{C}_{roadway}, \mathcal{C}_{district}, \mathcal{C}_{city}, \mathcal{C}_{province}, \mathcal{C}_{country}, \mathcal{C}_{ipaddress}, \mathcal{C}_{cell}, \mathcal{C}_{all}\}$ are given as follows: if there exists a relationship type of the full (partial) containment variety between entity types, the corresponding category types are related by $\sqsubset_{\mathcal{T}_{loc}}$ ($\sqsubset^{P}_{\mathcal{T}_{loc}}$) (e.g., $\mathcal{C}_{roadway} \sqsubset^{P}_{\mathcal{T}_{loc}} \mathcal{C}_{district}$ and $\mathcal{C}_{coordinate} \sqsubset_{\mathcal{T}_{loc}} \mathcal{C}_{province}$). However, as noted, application requirements, e.g., support for certain roll-up and drill-down operations, also influence the design of dimensions. Thus, the LOCATION dimension models only selected aspects of the miniworld captured in the ER diagram in Fig. 1 (see Sect. 4.7 for details).

We term a relationship between category types $\mathcal{C}_i \sqsubset^{(P)}_{\mathcal{T}} \mathcal{C}_j$ *direct* if it is given directly in the relation $\sqsubset^{(P)}_{\mathcal{T}}$ (without using transitivity); otherwise, the relationship is *indirect*. For example, the relationship $Roadway \sqsubset^{(P)}_{\mathcal{T}} District$ is direct, and if also $District \sqsubset^{(P)}_{\mathcal{T}} City$ then $Roadway \sqsubset^{(P)}_{\mathcal{T}} City$ is an indirect relationship.

4.2 Data model definition: dimension instances

After defining the schema level for dimensions in the data model, we proceed to define dimension instances, starting again from the prototypical data model.

Given a fact schema $\mathcal{S}$, a *dimension* of type $\mathcal{T} \in \mathcal{D}$ is a two-tuple $D = (C_D, \sqsubset)$, where $C_D = \{C_j, j = 1, \ldots, k\}$ is a set of *categories*. Each category C_j has a unique corresponding type $\mathcal{C}_j$ (a function $Type : C_D \rightarrowtail \bigcup_i \mathcal{C}_{\mathcal{T}_i}$ is defined and we write $Type(C_j) = \mathcal{C}_j$). A category C_j is a set of *dimension values* of type $\mathcal{C}_j$.

The relation $\sqsubset$ is a partial order on $\bigcup_j C_j$ (we henceforth simply write Dim instead of $\bigcup_j C_j$). The definition of the partial order is as follows. Given a pair of values $(e_i, e_j) \in C_i \times C_j$ such that $Type(C_i) \sqsubset_{\mathcal{T}} Type(C_j)$, $e_i \sqsubset e_j$ means that e_i is fully contained in e_j. Dimension values of the category of type $\perp_{\mathcal{T}}$, i.e., the "lowest" dimension values, are contained in values of other categories but do not contain anything themselves. The category of type $\top_{\mathcal{T}}$ has exactly one value, i.e., the "highest" value, denoted $\top$, containing all values in the dimension. Note that the partial order on category types and the functions Anc and $Desc$ imply a corresponding order and corresponding functions on categories.

We extend the definition of a dimension by generalizing the existing partial order $\sqsubset$ on dimension values, which is capable only of expressing *full containment* hierarchies. Specifically, we replace $\sqsubset$ by a relation $P \subseteq Dim \times Dim \times [0;1]$. In a triple $(e_i, e_j, d) \in P$, we refer to the value d as the *degree of containment*. With this extension, a dimension is a two-tuple $D = (C_D, P)$.

Example 2. Our LOCATION dimension is given by $D_{loc} = (C_{loc}, P_{loc})$, where $C_{loc} = \{Coordinate, Roadway, District, City, Province, Country, IPAddress, Cell, \top\}$ (one category for each node in Fig. 3).

As reflected in the name of the new relation (P stands for "partial"), triples in the relation P define *partial* containment relationships between dimension values. The definition of the relation is as follows. Given a pair of values $(e_i, e_j) \in C_i \times C_j$ such that $Type(C_i) \sqsubset^{(P)}_{\mathcal{T}} Type(C_j)$, we define $(e_i, e_j, d) \in P$, or simply $e_i \sqsubset_d e_j$, to mean that dimension value e_i is contained in dimension value e_j so that the size of the part of e_i contained in e_j is larger than or equal to d times the size of e_i.

Intuitively, we expect that in most cases when we record a relationship in our warehouse, $0 < d < 1$. To record this, it is required that $Type(C_i) \sqsubset^{P}_{\mathcal{T}} Type(C_j)$. In the special case where $d = 1$, which is defined only if $Type(C_i) \sqsubset_{\mathcal{T}} Type(C_j)$, we say that dimension value e_i is *fully* contained in dimension value e_j. When $d = 0$, which requires that $Type(C_i) \sqsubset^{P}_{\mathcal{T}} Type(C_j)$, e_i *may* be contained in e_j.

Finally, if $Type(C_i) \sqsubset^{(P)}_{\mathcal{T}} Type(C_j)$, we also define $e_i \not\sqsubset e_j$ to mean that dimension value e_i is not contained in dimension value e_j.

In the following, we assume that $e_i \in C_i$, $e_j \in C_j$ and $e_k \in C_k$. We also assume that $Type(C_i) = \mathcal{C}_i$, $Type(C_j) = \mathcal{C}_j$, $Type(C_k) = \mathcal{C}_k$ and that $(\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j \sqsubset_{\mathcal{T}} \mathcal{C}_k)$. The basic properties of the new relation are as follows.

1. *Containment in all:*
   $\forall e \in Dim \setminus \{\top\}(e \sqsubset_1 \top)$
   Naturally, each dimension value e is fully contained in the "highest" dimension value.
2. *Transitivity of full containment (f-to-f transitivity):*
   $\forall (e_i, e_j, e_k) \in C_i \times C_j \times C_k$
   $(((e_i \sqsubset_1 e_j) \wedge (e_j \sqsubset_1 e_k)) \Rightarrow (e_i \sqsubset_1 e_k))$
   If e_i is fully contained in e_j and e_j is fully contained in e_k, it is natural to infer that e_i is fully contained in e_k.

In defining the transitivity of partial containment, we employ a "safe" approach, where the idea is that we infer the relationships between dimension values with the maximum degrees of containment that hold.

3. *Transitivity of partial containment:*
   Assume that $(\mathcal{C}_i \sqsubset^{(P)}_{\mathcal{T}} \mathcal{C}_j \sqsubset^{(P)}_{\mathcal{T}} \mathcal{C}_k)$. Then the following hold.
   $\forall (e_i, e_j, e_k) \in C_i \times C_j \times C_k :$
   (a) *p-to-f transitivity:*
   $\forall d \in [0;1)((e_i \sqsubset_d e_j) \wedge (e_j \sqsubset_1 e_k) \Rightarrow (e_i \sqsubset_d e_k))$
   While e_k may contain the part of e_i that is not in e_j, the conditions of the property do not give us information on this. We infer only what we can guarantee: what is contained in e_j is *also* contained in e_k.
   (b) *f-to-p transitivity:*
   $\forall d \in [0;1)((e_i \sqsubset_1 e_j) \wedge (e_j \sqsubset_d e_k) \Rightarrow (e_i \sqsubset_0 e_k))$
   If e_i is fully contained in e_j and e_j is partially contained in e_k then we can only infer that at least no part of e_i is contained in e_k. In other words, we infer that e_i may be contained in e_k.
   (c) *p-to-p transitivity:*
   $\forall (d_i, d_j) \in [0;1) \times [0;1)$
   $((e_i \sqsubset_{d_i} e_j) \wedge (e_j \sqsubset_{d_j} e_k) \Rightarrow (e_i \sqsubset_0 e_k))$
   The reasoning follows the pattern from above: if e_i is partially contained in e_j and e_j is also partially contained in e_k, then we can only infer that e_i may be contained in e_k.

4.3 Building a dimension hierarchy

To convey the intuition behind the definition of a dimension, we consider the construction of the hierarchy in a dimension instance, the focus being on the relation P.

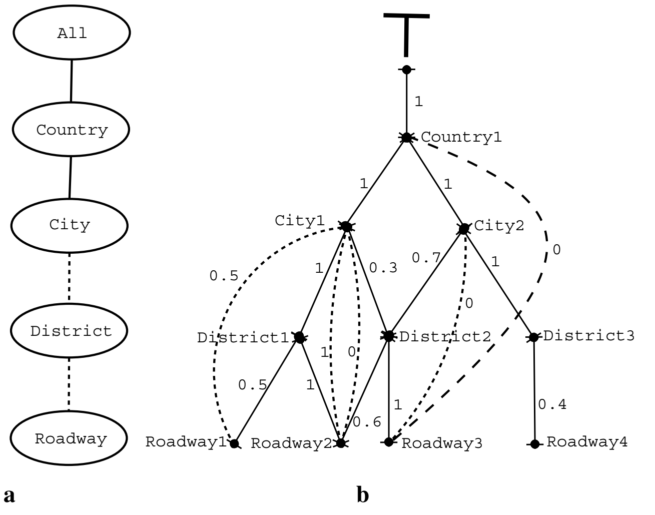


Fig. 4a,b. Schema **a** and instance **b** of a simplified LOCATION dimension

In the explanation of how we construct a dimension hierarchy, we term a relationship between dimension values $e_i \sqsubset_d e_j$ *direct* if it is given explicitly in the relation P (without using transitivity); otherwise, the relationship is *indirect*. In short, direct relationships are used to support data aggregation from one category to any category immediately above it. In turn, indirect relationships support data aggregation from one category to any higher category.

In order to exemplify the process of building dimension hierarchies, we introduce a dimension $D'_{loc} = (C_{D'_{loc}}, P'_{loc})$, which is a simplified version of the LOCATION dimension D_{loc}. Figure 4a depicts the schema of this dimension.

Based on our scenario, we assume that we have the following dimension values: coordinates $Coord1$ and $Coord2$, roadways $Roadway1$ and $Roadway2$, cities $City1$ and $City2$, district $District1$, province $Province1$, etc. Each of the values belongs to precisely one category (e.g., $Roadway1$ belongs to the $Roadway$ category), and they are related to other values via a dimension hierarchy given by partial order P'_{loc}.

Figure 4b then depicts an (incomplete) instance that corresponds to this schema. More specifically, the solid links between dimension values represent relationships that would be captured explicitly in relation P'_{loc}, i.e., direct relationships. The numbers next to the links denote containment degrees. The dotted and dashed links represent indirect, inferred relationships between dimension values.

We now explain how to build a relation P while exemplifying the process by building relation P'_{loc}. First, we populate P'_{loc} with special direct relationships between dimension values that hold for every domain. Specifically, for each dimension value, e.g., $Roadway1$, we add $Roadway1 \sqsubset_1 \top$ to the relation.

Second, we add other direct relationships, but now domain-specific. For example, if we know that $District1$ lies fully within $City1$ and that 50% of $Roadway1$ is in $District1$, then we add $District1 \sqsubset_1 City1$ and $Roadway1 \sqsubset_{0.5} Disctrict1$ to the relation. We do not introduce zero-degree containments in this step because we assume that all relationships that exist are known to us. If we were uncertain about some relationships, direct zero-degree containment relationships could result.

Third, by applying transitivity to the relationships that we have so far, we infer new, indirect relationships. While transitivity is initially applied to the direct relationships, it is applied repeatedly until no new relationships may be inferred. We proceed to consider some examples.

Using f-to-f transitivity, if $Roadway2 \sqsubset_1 District1$ and $District1 \sqsubset_1 City1$, we infer $Roadway2 \sqsubset_1 City1$. Thus, if we know that $Roadway2$ is fully contained in $District1$ and that $District1$ is fully contained in $City1$, then we infer that $Roadway2$ is fully contained in $City1$.

If $Roadway1 \sqsubset_{0.5} District1$ and $District1 \sqsubset_1 City1$, we may use p-to-f transitivity to infer $Roadway1 \sqsubset_{0.5} City1$. So if we know that 50% of $Roadway1$ is in $District1$ and that $District1$ lies fully within $City1$, we infer that 50% of the roadway $Roadway1$ is in $City1$. The result can be imprecise, but we acknowledge that some part of $Roadway1$ lies in $City1$ and indicate the guaranteed percentage.

As an example of using f-to-p transitivity, if $Roadway3 \sqsubset_1 District2$ and $District2 \sqsubset_{0.7} City2$, we infer $Roadway3 \sqsubset_0 City2$. This means that if we know that the roadway $Roadway3$ is in $District2$ and also that 70% of $District2$ is contained in $City2$, we can only infer that $Roadway3$ may be contained in $City2$.

We may also use p-to-p transitivity: if $Roadway2 \sqsubset_{0.6} District2$ and $District2 \sqsubset_{0.3} City1$, we infer $Roadway2 \sqsubset_0 City1$. In other words, we can only infer that $Roadway2$ may be contained in $City1$.

To summarize, in our model, we relate two values according to full, or partial with nonzero-degree, containment if it is given that the domain entities they represent exhibit the specific relationship. We relate two values according to partial containment with zero degree if we can *infer* that the entities they represent *may* be related according to partial containment. If it is given that two entities are unrelated or if we cannot infer their relation, we do not relate the corresponding values.

We note that if there are no partial containment relationships in a domain, we could still use the extended model. For category types, we then just use notation $\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j$ and $\mathcal{C}_i \not\sqsubset_{\mathcal{T}} \mathcal{C}_j$ (full and no containment, respectively) and never use notation $\mathcal{C}_i \sqsubset^P_{\mathcal{T}} \mathcal{C}_j$ (partial containment). For dimension values, we just use notation $e_i \sqsubset_1 e_j$ and $e_i \not\sqsubset e_j$ (full and no containment, respectively) and do not use notation $e_i \sqsubset_d e_j$, where $d \in [0; 1)$ (partial containment).

4.4 Data model definition: facts

For the formal definition of facts, we define $e_i \sqsubseteq_1 e_j \equiv (e_i \sqsubset_1 e_j) \vee (e_i = e_j)$ and $\mathcal{C}_i \sqsubseteq_{\mathcal{T}} \mathcal{C}_j \equiv (\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j) \vee (\mathcal{C}_i = \mathcal{C}_j)$.

Consider a set of *facts* F of type $\mathcal{F}$ and a dimension $D = (C_D, P)$. A *fact-dimension relation* R is defined as $R \subseteq F \times Dim$. In the prototypical model, a fact $f \in F$ is said to be *characterized by dimension value* e_k, written $f \rightsquigarrow e_k$, if $\exists e_i \in Dim\ ((f, e_i) \in R \wedge e_i \sqsubseteq e_k)$. It is required that $\forall f \in F(\exists e \in Dim\ ((f, e) \in R))$.

We extend this definition in only one respect: as a consequence of introducing partial containment, we need to use *p-characterization.* We say that a fact $f \in F$ is *0-characterized by dimension value* e_k, written $f \rightsquigarrow_0 e_k$, if $\exists e_i \in Dim$

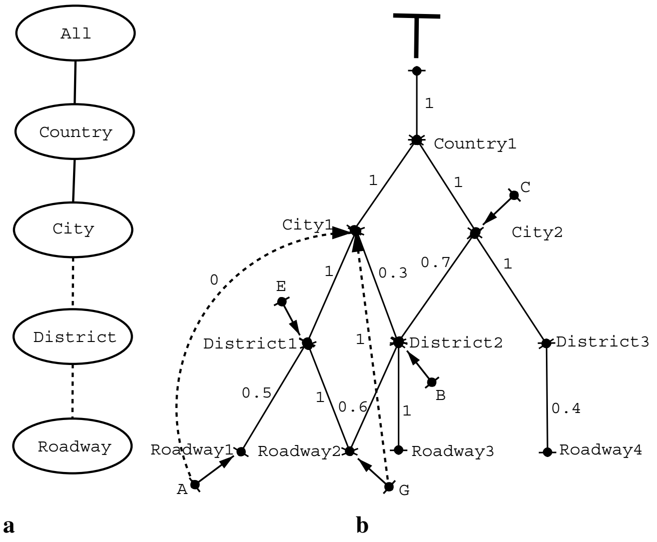


Fig. 5a,b. Relationships between facts and a simplified LOCATION dimension

$(((f, e_i) \in R) \wedge (e_i \sqsubset_d e_k) \wedge (d < 1))$. In addition, we will refer to the characterization from the prototypical model as *1-characterization*, written $f \leadsto_1 e_k$, if $\exists e_i \in Dim\ (((f, e_i) \in R) \wedge (e_i \sqsubseteq_1 e_k))$.

Example 3. In our case study, the set of facts of type $\mathcal{F}_{request}$ is $F_{request} = \{A, B, C, \ldots\}$. The fact-dimension relation between D_{loc} and $F_{request}$ could be denoted as R_{loc}.

A fact-dimension relation links facts and corresponding dimension values. Each fact is related to at least one dimension value in each dimension. Characterization is propagated up along a hierarchy of dimension values. We use 1-characterization to mean that a fact is *known for sure* to be characterized by a dimension value, and we use 0-characterization to mean that a fact *may* be characterized by a dimension value.

Example 4. Figure 5 illustrates characterization of facts from $F_{request}$ by values from the simplified LOCATION dimension D'_{loc} in Fig. 4. Unnumbered, solid arrows denote fact-dimension relationships. Numbered, dotted arrows denote propagated characterizations of facts.

We link a fact to a dimension value if the domain object represented by that fact is characterized by the object represented by the dimension value. For example, if we know that request A has been issued from $Roadway1$, then we add the pair $(A, Roadway1)$ to the relation R'_{loc}.

We also infer characterizations of facts. This is analogous to inferring indirect relationships between dimension values using f-to-f and f-to-p transitivity. Specifically, we 1-characterize a fact by a dimension value if the dimension value that it is linked to is fully contained in that dimension value. So, if $(G, Roadway2) \in R'_{loc}$ and $Roadway2 \sqsubset City1$, then $G \leadsto_1 City1$. In this case, it is natural to say that fact G is *surely* characterized by $City1$. Next, we 0-characterize a fact by a dimension value if the dimension value that it is linked to is only partially contained in that dimension value. For example, if $(A, Roadway1) \in R'_{loc}$ and $Roadway1 \sqsubset_0 City1$, then $A \leadsto_0 City1$. In this case, it is natural to say that fact A *may* be characterized by $City1$.

Finally, a *multidimensional object* (MO) is a four-tuple $M = (\mathcal{S}, F, D_M, R_M)$, where $\mathcal{S} = (\mathcal{F}, \mathcal{D} = \{\mathcal{T}_i, i = 1, \ldots, n\})$ is a fact schema, F is a set of facts of type $\mathcal{F}$, $D_M = \{D_i, i = 1, \ldots, n\}$ is a set of dimensions, where dimension D_i is of type $\mathcal{T}_i$, and where $R_M = \{R_i, i = 1, \ldots, n\}$ is a set of fact-dimension relations such that $\forall i\ ((f, e) \in R_i \Rightarrow ((f \in F) \wedge \exists C \in C_{D_i}(e \in C)))$. A multidimensional object brings the different parts of the domain model together and completes the definition of the model.

Example 5. In our case, we can define the multidimensional object $M_{case} = (\mathcal{S}_{case}, F_{request}, D_{case}, R_{case})$.

4.5 Model properties

We proceed to define important properties of the data model. The definitions extend the ones given in the prototypical model with support for partial containment. In the definitions, we assume a multidimensional object $M = (\mathcal{S}, F, D_M, R_M)$ and a dimension $D \in D_M$. We also assume that $Type(C_i) = \mathcal{C}_i$, $Type(C_j) = \mathcal{C}_j$, and $Type(C_k) = \mathcal{C}_k$.

Definition 1. Given two distinct categories C_i and C_j, where $C_j \in Anc^{(P)}(C_i)$, we say that the mapping from C_i to C_j is *onto* if $\forall e_j \in C_j\ (\exists(e_i, d) \in C_i \times [0; 1]\ (e_i \sqsubset_d e_j))$; otherwise, it is *non-onto*. If all the mappings in a dimension are onto, we say that the dimension hierarchy is *onto*; otherwise, it is *non-onto*.

Example 6. The mapping from *Province* to *Country* is onto because each country is partitioned into provinces. However, the mapping from *IPAddress* to *City* is non-onto because some cities have no computers (IP addresses). Thus, the hierarchy of the dimension D_{loc} is non-onto. The hierarchy of the D_{time} dimension is onto.

Definition 2. Given three distinct categories C_i, C_j, and C_k, where $\mathcal{C}_i \sqsubset_{\mathcal{T}}^{(P)} \mathcal{C}_j \sqsubset_{\mathcal{T}}^{(P)} \mathcal{C}_k$, we say that the mapping from C_j to C_k is *covering with respect to* C_i if $\forall(e_i, d) \in C_i \times [0; 1]\ (\forall e_k \in C_k\ ((e_i \sqsubset_d e_k) \Rightarrow \exists(e_j, d_i, d_j) \in C_j \times [0; 1] \times [0; 1]\ ((e_i \sqsubset_{d_i} e_j) \wedge (e_j \sqsubset_{d_j} e_k))))$; otherwise, it is *noncovering*. If in a dimension all the mappings with respect to all the categories are covering, we say that the dimension hierarchy is *covering*, otherwise, it is *noncovering*.

Example 7. Consider the categories $Roadway$, $Province$, and $Country$ in D_{loc}. Each roadway going through some country also goes through a province. So the mapping from $Province$ to $Country$ is covering with respect to $Roadway$. Consider the categories $Coordinate$, $Roadway$, and $District$. Some coordinates do not lie on any roadway, so we map them directly to districts. This means that the mapping from $Roadway$ to $District$ is noncovering with respect to $Coordinate$. Thus, the hierarchy of the dimension D_{loc} is noncovering. In contrast, the hierarchy of the dimension D_{time} is covering.

Definition 3. Given two distinct categories C_i and C_j, where $C_j \in Anc^{(P)}(C_i)$, we say that the mapping from C_i to C_j is *strict* if $\forall(e_i, d_{i_1}, d_{i_2}) \in C_i \times [0; 1] \times [0; 1]\ (\forall(e_{j_1}, e_{j_2}) \in C_j \times C_j\ ((e_i \sqsubset_{d_{i_1}} e_{j_1}) \wedge (e_i \sqsubset_{d_{i_2}} e_{j_2}) \Rightarrow ((e_{j_1} = e_{j_2}) \wedge (d_{i_1} = d_{i_2}))))$; otherwise, it is *nonstrict*. If in a dimension all the mappings are strict, we say that the dimension hierarchy is *strict*; otherwise, it is *nonstrict*.

Example 8. The mapping from *IPAddress* to *Province* is strict because an address uniquely identifies a province. But the mapping from *Cell* to *Province* is nonstrict because a cell may be shared by provinces. Thus, the hierarchy of the dimension D_{loc} is nonstrict. The hierarchy of the dimension D_{time} is strict.

Definition 4. We say that a dimension hierarchy is *aggregation strict* if it is strict or the following holds: if $C_j \in Anc^{(P)}(C_i)$ and a mapping from C_i to C_j exists that is nonstrict then $Anc^{(P)}(C_j) = \emptyset$; otherwise, it is *aggregation nonstrict.*

Example 9. Consider the categories *Cell* and *Province*. As the mapping from *Cell* to *Province* is nonstrict and $Anc^{(P)}(Province) \neq \emptyset$, the hierarchy of the dimension D_{loc} is aggregation nonstrict. The hierarchy of dimension D_{time} is aggregation strict because it is strict.

Definition 5. We say that a dimension hierarchy is *normalized* if it is onto, covering, and aggregation strict; otherwise, it is *nonnormalized.* We say that a multidimensional object is *normalized* if all its dimensions D_i are normalized and $\forall R_i \in R_M(((f, e) \in R_i) \Rightarrow (e \in \perp_{D_i}))$; otherwise, it is *nonnormalized.*

Example 10. The hierarchy of the dimension D_{time} is normalized because it is onto, covering, and strict. But the hierarchy of the dimension D_{loc} is nonnormalized because it is non-onto, noncovering, and aggregation nonstrict. Therefore, the multidimensional object M_{case} is nonnormalized.

4.6 Meeting the requirements

We now examine whether the requirements stated in Sect. 3.2 have been met. Explicit and multiple hierarchies are supported with the help of the partially ordered dimension types. Partial containment is supported with the help of special relations on category types and dimension values. The relation on dimension values supports nonnormalized hierarchies. Nonstrict hierarchies are captured by allowing a dimension value in a category to be related to several values in an ancestor category. Non-onto hierarchies may be built: a dimension value in a category is allowed to have no children in a descendant category. Noncovering hierarchies are also supported because a value is not required to be related to another value in an immediate parent category, i.e., a link between dimension values may "skip" one or more levels.

Many-to-many relationships between facts and dimensions can be implemented by relating a fact to several dimension values in a dimension and relating a dimension value to several facts. This is allowed by the definition of fact-dimensional relationships. Different levels of granularity are handled: facts may map to dimension values from different categories. The combination of support in the data model for different levels of granularity of facts and partial containment of dimension values provides a basis for supporting imprecision in the data [26].

4.7 Designing dimension schemas

We turn our attention to the design decisions that go into the creation of multidimensional dimension schemas. We initially consider the design context, then offer five guidelines for dimension schema design.

4.7.1 Design context

The design of a multidimensional schema typically begins with the analysis of a single *business process*, e.g., users requesting services, and then determines the relevant facts and dimensions for this process.

It is important that the dimensions be rich on contextual information that can be used for characterizing the facts. Rich dimensions enable multiple aggregations of facts and enable roll-up and drill-down operations. Supplying this rich context typically requires data from several data sources.

Only information relevant to the analysis of the particular business process is captured in the multidimensional schema; much other information is omitted. For example, a multidimensional schema based on our scenario will leave out some of the relationships present in Fig. 1. We do not try to capture every aspect of the miniworld in one-multidimensional schema. A multidimensional model is not a replacement for the ER model or UML.

It is beyond the scope of this paper to describe a full data warehouse design process in detail; for this, we refer to the literature [15]. Rather, we consider the design issues that are particular to the data occuring in location-based services, most notably spatial data hierarchies with partial containment relationships. We summarize the discussions into a set of general guidelines for the design of dimension schemas with such data. The insights and guidelines presented here are thus part of some complete methodology for multidimensional database design, e.g., the one described by Kimball et al. [15].

4.7.2 Dimension design guidelines

Because the data model presented here allows partial containment relationships in dimensions, it generalizes existing models and offers new means of modeling dimensions. Below we explore pertinent implications of using the model for the design of dimensions.

Section 3.1 describes a prototypical usage scenario for a multidimensional database in the context of a location-based service. In particular, Fig. 1 depicts an ER diagram that presents various location values of relevance to location-based services. We proceed to consider the process of mapping the ER diagram to the LOCATION dimension shown in Fig. 3.

The ER diagram captures information about containment relationships among various location entity types. Transitive relationships are not shown, and there are no explicit descriptions of hierarchies. We must build explicit hierarchies based on this diagram that enable the capture of data and the relevant analyses.

For example, as a reflection of the relationship types ro-P-di and di-P-ci in the ER diagram—and because a roadway is typically contained in one city, so that we find it most informative to aggregate requests per roadway into requests per city—we will have a category *Roadway* that is below a category *City* in our dimension hierarchy. We identify the *Coordinate* category as the lowest category because its corresponding entity type is only contained in other types.

The highest category has a single value, denoted $\top$, that contains all other values. This value is very useful in analyses as we can easily express aggregation over a whole dimension. In our case, the ER diagram happens to have a corresponding entity type, but in many cases the $\top$ category is implicit and must be created specially for the multidimensional schema. We summarize this into the guideline "*(1) build explicit hierarchies with top and bottom categories*."

When building the LOCATION dimension, we obtain multiple hierarchies. The use of these is caused in part by the support for partial containment, so we explore this aspect in some detail. An obvious reason for introducing multiple hierarchies is that mutually exclusive hierarchies exist in the scenario. For example, the groupings of the coordinates of service requests by mobile cells and by an administrative unit such as roadways are exclusive, as one category cannot meaningfully be said to be contained in another. Therefore, the *Cell* category does not fit anywhere in the (main) hierarchy, *Coordinate* $<$ *Roadway* $<$ *District* $<$ *City* $<$ *Province* $<$ *Country* $< \top$. It is instead part of separate hierarchies, e.g., *Coordinate* $<$ *Cell* $<$ *Province* $<$ *Country* $< \top$, which skip the *Roadway* category. In general, building these kinds of hierarchies translates into inserting categories and corresponding relationships in the LOCATION dimension. We summarize this into the guideline "*(2) introduce an additional hierarchy if a category does not fit into the existing hierarchies*." The other cases where it is necessary to build additional hierarchies are discussed next.

An additional relationship may be inserted to "mend" a noncovering hierarchy. To illustrate, recall from Example 7 that it is possible for a coordinate to not lie on any roadway, while it does lie in some district. The consequence is that we cannot map all coordinates to their corresponding districts via the *Roadway* entity type in Fig. 1 or the *Roadway* category in Fig. 3. As we consider this mapping important for data analyses, we include a direct relationship from *Coordinate* to *District* in the LOCATION dimension. This relationship then creates a new path, or hierarchy, from *Coordinate* to $\top$. We summarize this as follows: "*(3) insert direct relationships to capture noncovering hierarchies*."

Next, note that there are some relationship types from the ER diagram that do not have corresponding relationships in the dimension. For example, the ro-P-co relationship type would yield a direct relationship between the *Roadway* and *Country* categories, which is absent from the dimension. The following reasoning went into this design decision.

First, in the real world, each roadway goes through a province that is part of some country, meaning that the relationship between *Province* and *Country* is covering with respect to *Roadway*. We are thus able to relate roadways to corresponding countries through values from the *Province* category – we do not depend on a direct relationship between the *Roadway* and the *Country* categories. Second, transitive partial containment relationships between dimension values are generally less precise than direct ones. However, in some situations, such as the one we are considering, maintaining a high precision of the degrees of containment in relationships between values of two categories may not be important. For example, a single roadway typically contributes only very little to the aggregate for a whole country, so the imprecision caused by rolling up through provinces is negligible and is preferred over creating a more complex schema.

If high-precision partial containment relationships are important, we insert direct relationships. For example, had it been important that roadways roll up to countries as precisely as possible, we would add a direct relationship between *Roadway* and *Country*. This illustrates a trade-off: if one wants high precision, this comes at the cost of increasing the size and complexity of a dimension. The higher precision we want, the more direct relationships are needed. We summarize as follows: "*(4) start with the relevant immediate parent-child relationships and insert direct, nonimmediate relationships if and only if high precision is desired*."

Another aspect of dimension design is how to determine which category should be below which other category. While this may not be obvious in the general case, it is most often easy to decide how to relate two dimension categories. This is the case when values from one category are inherently "smaller" than those of another category. For example, since provinces are parts of countries, there is a full containment relationship from *Province* to *Country*, not the other way around.

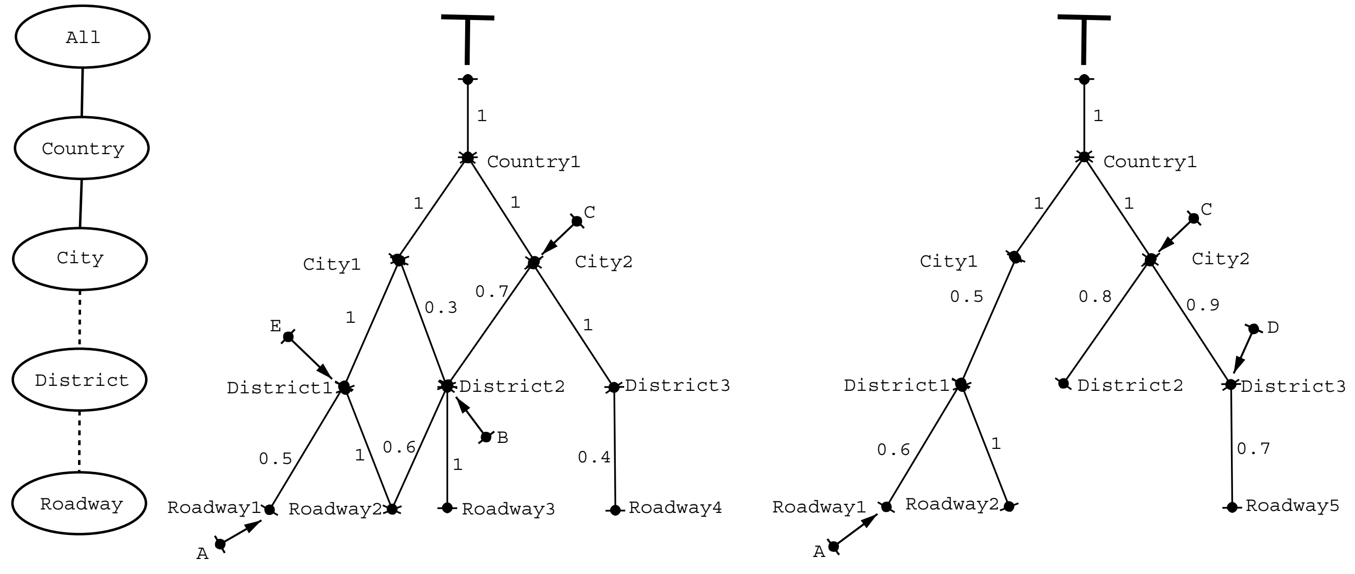
To illustrate that relationships between categories are not always obvious, consider the relationship between the *District* and *City* categories. In reality, districts exist that are contained in cities – they are termed city districts. The LOCATION dimension assumes this district type. However, there are also districts that contain cities, e.g., church districts may include several small "cities." In addition, dimension values from two different categories can be of the same size, e.g., cities and districts are not related by containment relationships but simply overlap.

One approach to addressing this problem is to divide the common *District* category into several categories, one for each district type, thus introducing, e.g., *Church District* and *City District* categories. The category *City District* is then placed below the *City* category, and *Church District* is placed above *City*. This approach does not contend well with large cities and city districts that contain church districts.

Another approach is to allow districts of all types to belong to the unique *District* category, with a pair of symmetric direct relationships between the *City* and *District* categories, i.e., *City* $\sqsubset^{(P)}_{\mathcal{T}_{loc}}$ *District* and *District* $\sqsubset^{(P)}_{\mathcal{T}_{loc}}$ *City*. These relationships enable us to capture the desired relationships between district and city dimension values. For example, if city $City1$ and church district $District1$ overlap, we may include two relationships with the appropriate degrees of containment, e.g., $City1 \sqsubset_{0.6} District1$ and $District1 \sqsubset_{0.2} City1$.

However, the antisymmetry property of the order on categories does not allow symmetric, direct relationships. This restriction aims to avoid inappropriate transitive relationships between dimension values. For example, without antisymmetry, in our case, by p-to-p transitivity it may be inferred that $District1 \sqsubset_0 District1$, which is undefined. Also, performing this kind of roll up makes little sense.

We summarize this last discussion into the last guideline: "*(5) choose the hierarchical relationship between two cate-*



a b c

Fig. 6a–c. Schema **a** and LOCATION dimensions of **b** M^1_{case} and **c** M^2_{case}

gories based on the most common case and such that aggregation makes the most sense."

5 The algebra

In this section, we present an algebra for the extended data model. It is based on the algebra for the prototypical model. We redefine the operators (selection, union, and aggregate formation) that need to be extended in order to support partial containment. The operators that can be taken directly from the prototypical algebra without modification include projection, rename, difference, and identity-based join, as well as derived operators such as value-based join, duplicate removal, SQL-like aggregation, star join, drill down, and roll up.

For unary operators, we assume a single n-dimensional MO $M = \{\mathcal{S}, F, D_M, R_M\}$, where $D_M = \{D_i, i = 1, \ldots, n\}$ and $R_M = \{R_i, i = 1, \ldots, n\}$. For binary operators we assume two n-dimensional MO's $M_1 = (\mathcal{S}_1, F_1, D_{M_1}, R_{M_1})$ and $M_2 = (\mathcal{S}_2, F_2, D_{M_2}, R_{M_2})$, where $D_{M_1} = \{D^1_i, i = 1, \ldots, n\}$, $D_{M_2} = \{D^2_i, i = 1, \ldots, n\}$, $R_{M_1} = \{R^1_i, i = 1, \ldots, n\}$, and $R_{M_2} = \{R^2_i, i = 1, \ldots, n\}$. Given a dimension D_i with the set of categories $C_{D_i} = \{C_j, j = 1, \ldots, k\}$, we use the notation Dim_i for $\bigcup_j C_j$.

Example 11. In order to illustrate the workings of the operators, we construct two example MOs denoted by M^1_{case} and M^2_{case} . Figure 6a depicts the schema of the LOCATION dimension of the MOs, which is a simplified version of that of M_{case}. In Figs. 6b and 6c, the structure of the LOCATION dimensions of M^1_{case} and M^2_{case}, respectively, is presented, with numbers near the links denoting the degrees of containment. The arrows in Figs. 6b and 6c represent the fact-dimension relationships in the LOCATION dimensions. Note that facts may map directly to dimension values in nonbottom categories. The TIME and USER dimensions of the MOs are identical to the corresponding dimensions of M_{case}.

5.1 Selection operator

The selection operator is used to select a subset of the facts in an MO based on a predicate. We first restate the definition from [26]. Given a predicate $q : Dim_1 \times \ldots \times Dim_n \rightarrowtail \{true, false\}$, the *selection* operator for the prototypical model, σ, is defined as: $\sigma[q](M) = M' = (\mathcal{S}', F', D'_{M'}, R'_{M'})$, where $\mathcal{S}' = \mathcal{S}$,
$F' = \{f \in F \mid \exists(e_1, \ldots, e_n) \in Dim_1 \times \ldots \times Dim_n;$
$((q(e_1, \ldots, e_n)) \wedge (f \rightsquigarrow e_1) \wedge \ldots \wedge (f \rightsquigarrow e_n))\}$,
$D'_{M'} = D_M$,
$R'_{M'} = \{R'_i, i = 1, \ldots, n\}$,
and $R'_i = \{(f', e) \in R_i \mid f' \in F'\}$.

The selection operator for the extended model, σ_{ext}, uses the new $2n$-ary predicate $q_{ext} : Dim_1 \times \ldots \times Dim_n \times [0;1] \times \ldots \times [0;1] \rightarrowtail \{true, false\}$. The resulting set of facts is $F' = \{f \in F \mid \exists(e_1, \ldots, e_n) \in Dim_1 \times \ldots \times Dim_n \; (\exists(d_1, \ldots, d_n) \in [0;1] \times \ldots \times [0;1] \; ((q_{ext}(e_1, \ldots, e_n, d_1, \ldots, d_n)) \wedge (f \rightsquigarrow_{d_1} e_1) \wedge \ldots \wedge (f \rightsquigarrow_{d_n} e_n)))\}$.

We thus restrict the set of facts to those that are characterized by dimension values where q_{ext} evaluates to *true*. In addition, we restrict the fact-dimension relations accordingly, while the dimensions and the fact schema stay the same. The operator supports partial containment by letting the value of the predicate depend on degrees of containment. This allows us to formulate queries that select either facts that are *surely* characterized by a dimension value, or facts that *may* be characterized by a value, or both.

Example 12. Suppose we want to select requests from M^1_{case} that were *surely* issued from *District*2. In addition, the time of the requests we are interested in is July 14, 2001, and the users associated with the requests must be 21–30 years old. The predicate for the query is: $(e_{loc} = District2) \wedge ((e_{userage} \in [21;30]) \wedge (e_{time} = [07\backslash 14\backslash 2001]) \wedge (d_{loc} = 1) \wedge (d_{userage} = 1) \wedge (d_{time} = 1)$. Notice that $B \rightsquigarrow_1 District2$, so request B will be in the result.

Example 13. Suppose we want to select requests from M^2_{case} that *may* have been issued from $City1$. Moreover, we take only nighttime requests (i.e., from 10 p.m. to 6 a.m.) into consideration. The predicate for this query may be given as follows: $(e_{loc} = City1) \wedge (e_{time} \in \{10\text{ p.m.}, \ldots, 12\text{ p.m.}, \ldots, 1\text{ a.m.}, \ldots, 6\text{ a.m.}\}) \wedge (d_{loc} = 0) \wedge (d_{time} = 1)$. Notice that $A \rightsquigarrow_0 City1$, so the request A will be in the result.

5.2 Union operator

The union operator is used to take the union of two MOs. Consider two dimensions $D_1 = (C_{D_1}, \sqsubset_{D_1})$ and $D_2 = (C_{D_2}, \sqsubset_{D_2})$ of the same type $\mathcal{T}$, where $C_{D_1} = \{C^1_j, j = 1, \ldots, m\}$ and $C_{D_2} = \{C^2_j, j = 1, \ldots, m\}$. The *union* operator on *dimensions* for the prototypical model, $\bigcup^D$, is defined as follows: $D' = D_1 \bigcup^D D_2 = (C_{D'}, \sqsubset_{D'})$, where $C_{D'} = \{C^1_j \bigcup C^2_j, j = 1, \ldots, m\}$ and $\forall (e_1, e_2) \in (Dim_1 \cup Dim_2) \times (Dim_1 \cup Dim_2)$ $((e_1 \sqsubset_{D'} e_2) \Leftrightarrow ((e_1 \sqsubset_{D_1} e_2) \vee (e_1 \sqsubset_{D_2} e_2)))$. In what follows, we use notation C'_j for $C^1_j \bigcup C^2_j$.

Consider two n-dimensional MOs with $\mathcal{S}_1 = \mathcal{S}_2$. The *union* operator on *MO*s for the prototypical model, $\bigcup$, is defined as: $M' = M_1 \bigcup M_2 = (\mathcal{S}', F', D'_{M'}, R'_{M'})$, where $\mathcal{S}' = \mathcal{S}_1$, $F' = F_1 \bigcup F_2$, $D'_{M'} = \{D^1_i \bigcup^D D^2_i, i = 1, \ldots, n\}$, $R'_{M'} = \{R^1_i \bigcup R^2_i, i = 1, \ldots, n\}$.

We proceed to first define an extended dimension union operator (denoted $\bigcup^D_{ext}$). Consider two dimensions $D_1 = (C_{D_1}, P_{D_1})$ and $D_2 = (C_{D_2}, P_{D_2})$ of the same type $\mathcal{T}$. We modify the condition for the partial order in the resulting dimension. Specifically, we require the following.

1. $\forall (e_1, e_2) \in (Dim_1 \cup Dim_2) \times (Dim_1 \cup Dim_2)$ $((\exists d \in [0;1]((e_1, e_2, d) \in P_{D'}) \Leftrightarrow$ $(\exists (d_1, d_2) \in [0;1] \times [0;1]$ $(((e_1, e_2, d_1) \in P_{D_1}) \vee ((e_1, e_2, d_2) \in P_{D_2}))))$
2. $\forall (e_1, e_2, d) \in P_{D'}$ $(((e_1, e_2) \in C'_i \times C'_j) \wedge (C'_j \in Anc^{(P)}(C'_i))) \Rightarrow$ $(\exists (d_1, d_2) \in [0;1] \times [0;1]$ $(((e_1, e_2, d_1) \in P_{D_1}) \vee ((e_1, e_2, d_2) \in P_{D_2})) \wedge (d = max(d_1, d_2))))$

Only the degrees of containment for the *direct* relationships are found using these rules. The indirect relationships between values in the resulting dimension are inferred using our transitivity rules.

Stated less formally, given two MOs with common fact schemas, the union operator for the extended model takes the set union of facts and the fact-dimension relations. Dimensions are combined with the help of the $\bigcup^D_{ext}$ operator. Specifically, given two dimensions of the same type, we perform set union on corresponding categories and build a new relation on dimension values: there exists a relationship between two dimension values if there exists a relationship between the values in the first dimension, in the second dimension, or in both. The degree of containment for a resulting relationship is determined in a natural way, namely, if two values are directly related in one of the two dimensions only, then their degree is transferred unchanged into the resulting dimension. However, if the values are directly related in both dimensions with two different degrees, then without breaking the principle of the safe approach we can return the *maximum* of the two as the new degree.

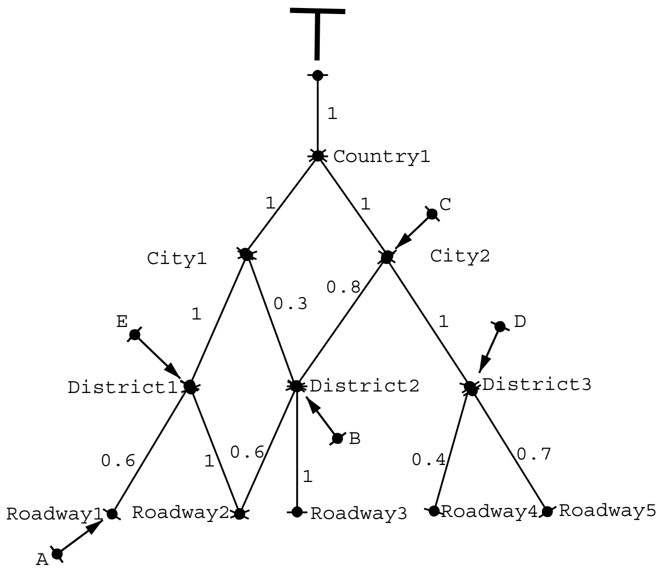


Fig. 7. Union of M^1_{case} and M^2_{case}

Example 14. The LOCATION dimension of an MO obtained by uniting M^1_{case} and M^2_{case} is depicted in Fig. 7. Note that some degrees of containment for the indirect relationships, e.g., $Roadway1 \sqsubset_{0.6} City1$, are not found in any of the original MOs and can only be inferred using the transitivity rules.

5.3 Aggregate formation operator

The aggregate formation operator is used when applying aggregate functions to an MO. We first restate the definition from the prototypical model. We assume a family of aggregation functions G that "look up" the required data for the facts in the relevant fact-dimension relation, e.g., $COUNT_i$ finds its data in the fact-dimension relation R_i and counts them.

In addition, the operator $Group : Dim_1 \times \ldots \times Dim_n \rightarrowtail 2^F$ is defined. The operator groups the facts characterized by the same dimension values, i.e., $Group(e_1, \ldots, e_n) = \{f \mid (f \in F) \wedge (f \rightsquigarrow e_1) \wedge \ldots \wedge (f \rightsquigarrow e_n)\}$.

Given a new (result) dimension D_{n+1} of a new (result) type $\mathcal{T}_{n+1}$, an aggregation function $g : 2^F \rightarrowtail Dim_{n+1}$ and a set of grouping categories $\{C_i \in D_i, i = 1, \ldots n\}$, the aggregate formation operator for the prototypical model, α, is defined as follows: $M' = \alpha[D_{n+1}, g, C_1, \ldots, C_n](M) = (\mathcal{S}', F', D'_{M'}, R'_{M'})$, where

$$\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$$
$$\mathcal{F}' = 2^{\mathcal{F}},$$
$$\mathcal{D}' = \{\mathcal{T}'_i, i = 1, \ldots, n\} \bigcup \{\mathcal{T}_{n+1}\}$$
$$\mathcal{T}'_i = (\mathcal{C}'_i, \sqsubset'_{\mathcal{T}_i}, \bot'_{\mathcal{T}_i}, \top'_{\mathcal{T}_i})$$
$$\mathcal{C}'_i = \{\mathcal{C}_{ij} \in \mathcal{T}_i \mid Type(C_i) \sqsubseteq_{\mathcal{T}_i} \mathcal{C}_{ij}\}$$
$$\sqsubset'_{\mathcal{T}_i} = \sqsubset_{\mathcal{T}_i|_{\mathcal{C}'_i}},$$
$$\bot'_{\mathcal{T}_i} = Type(C_i),$$
$$\top'_{\mathcal{T}_i} = \top_{\mathcal{T}_i}$$
$$F' = \{Group(e_1, \ldots, e_n) \mid ((e_1, \ldots, e_n) \in C_1 \times \ldots \times$$

$C_n) \wedge (\mathit{Group}(e_1,\ldots,e_n) \neq \emptyset)\}$
$D' = \{D'_i, i=1,\ldots,n\} \bigcup \{D_{n+1}\}$,
$D'_i = (C'_{D'_i}, \sqsubset'_{D'_i})$,
$C'_{D'_i} = \{C'_{ij} \in D_i \mid \mathit{Type}(C'_{ij}) \in \mathcal{C}'_i\}$
$\sqsubset'_{D'_i} = \sqsubset_{D_i}|_{D'_i}$
$R'_{M'} = \{R'_i, i=1,\ldots,n\} \bigcup \{R'_{n+1}\}$
$R'_i = \{(f', e'_i) \mid \exists (e_1,\ldots,e_n) \in C_1 \times \ldots \times C_n((f' = \mathit{Group}(e_1,\ldots,e_n)) \wedge (f' \in F') \wedge (e_i = e'_i)\}$
and
$R'_{n+1} = \bigcup_{(e_1,\ldots,e_n)\in C_1\times\ldots\times C_n} \{(\mathit{Group}(e_1,\ldots,e_n), g(\mathit{Group}(e_1,\ldots,e_n))) \mid \mathit{Group}(e_1,\ldots,e_n) \neq \emptyset\}$

Thus, for every combination of dimension values $(e_1,\ldots,e_n)$ in the given grouping categories, the aggregation function g is applied to the set of facts characterized by $(e_1,\ldots,e_n)$, and the result is placed in the new dimension. The new facts are of type *sets of the argument fact type*, and the argument dimension types are restricted to the category types that are greater than or equal to the types of the grouping categories. The dimension type for the result is added to the set of dimension types.

The new set of facts consists of sets of the original facts, where original facts in a set share a combination of characterizing dimension values. The argument dimensions are restricted to the remaining category types, and the result dimension is added. The fact-dimension relations for the argument dimensions now link sets of facts directly to their corresponding combination of dimension values, and the fact-dimension relation for the result dimension links sets of facts to the function results for these sets.

Example 15. Consider the MO M^1_{case} in Fig. 6b. Suppose we want to count the number of requests issued from different districts regardless of issue time and user information. The aggregate formation operator for the query would look as follows: $\alpha[\mathit{RESULT}, \mathit{COUNT}, \mathit{District}, \top, \top](M^1_{case})$.

In the extended model, we have introduced p-characterization of facts, which allows us to capture imprecision in the data. This sort of imprecision must be accommodated by the aggregate formation operator. Specifically, this imprecision in data may be handled by grouping facts in different ways, i.e., by using alternative grouping operators. There are three ways of handling this, namely, by means of *conservative*, *liberal*, and *weighted* fact groupings.

In the *conservative grouping*, we include only those facts in a group that are *known for sure* to belong to that group. We define the corresponding operator, Group_c, as follows: $\mathit{Group}_c(e_1,\ldots,e_n) = \{f \mid (f \in F) \wedge (f \rightsquigarrow_1 e_1) \wedge \ldots \wedge (f \rightsquigarrow_1 e_n)\}$. Since only precise data will be used in calculations and the remaining data discarded, this kind of grouping is useful for computing a "lower bound" for a query result, in the sense that the query result contains as little data as possible.

Example 16. Assume the aggregation query from Example 15. The requests B and E are guaranteed to have been issued from certain districts, while for the request A we only know that it may have been issued from the district $\mathit{District}1$. This means that the conservative grouping of the requests by districts would yield the fact groups $\{E\}$ and $\{B\}$ mapped to the values $\mathit{District}1$ and $\mathit{District}2$ in the LOCATION dimension, respectively, as depicted in Fig. 8a. In this case, the count for both groups of requests would be 1, which can be seen from the result dimension in Fig. 8c.

In the *liberal grouping*, a group is formed from the facts that are known to belong to the group as well as from those facts that *might* belong to that group. We define the corresponding operator, Group_l, as: $\mathit{Group}_l(e_1,\ldots,e_n) = \{f \mid (f \in F) \wedge (f \rightsquigarrow_{d_1} e_1) \wedge \ldots \wedge (f \rightsquigarrow_{d_n} e_n) \wedge (\forall i \in \{1,\ldots,n\}((d_i = 1) \vee (d_i = 0)))\}$. Liberal grouping can be used for computing an "upper bound" for a query result, in the sense that the query result contains as much data as possible, because all the data, both precise and imprecise, are taken into consideration.

Example 17. Assume the aggregation query from Example 15. The liberal grouping of the requests by districts would include the request A in a group, i.e., we would get the fact groups $\{A, E\}$ and $\{B\}$ mapped to the values $\mathit{District}1$ and $\mathit{District}2$ as depicted in Fig. 8b. In this case, the count for the groups would be 2 and 1, respectively, which is shown in Fig. 8d.

Finally, in the *weighted grouping*, we gather both kinds of facts but apply a *weight of membership* to each fact in a group. For this kind of grouping we use the liberal grouping operator, i.e., $\mathit{Group}_w = \mathit{Group}_l$. We determine the weight of membership for a fact with the help of a function $\mathit{Weight} : \mathit{Dim}_1 \times \ldots \times \mathit{Dim}_n \times F \rightarrowtail \mathbb{R}$ by combining the fact's degrees of containment.

For example, the simplest way to combine the degrees is to take their product, namely, if $f \in \mathit{Group}_w(e_1,\ldots,e_n)$ and $f \rightsquigarrow_{d_1} e_1, \ldots, f \rightsquigarrow_{d_n} e_n$, then $\mathit{Weight}_{simp}(e_1,\ldots,e_n, f) = \prod_{i=1}^{n} d_i$.

Since weighted data will be used in calculations, with this kind of grouping an "average" for a query result could be computed. However, the choice of weighting function is crucial; an inadequate function may introduce additional imprecision into the query result. For example, Weight_{simp} may sometimes be inadequate because it applies zero weights even for data that are imprecise with respect to just one dimension, i.e., $\forall f \in \mathit{Group}_w(e_1,\ldots,e_n)((\exists e_i \in \{e_1,\ldots,e_n\} (f \rightsquigarrow_0 e_i)) \Rightarrow (\mathit{Weight}_{simp}(e_1,\ldots,e_n, f) = 0))$.

Example 18. Assume the aggregation query from Example 15 and the corresponding liberal grouping of the facts from Example 17. The function Weight_{simp} would apply weights of membership to our facts as follows: $\mathit{Weight}_{simp}(\mathit{District}1, \top, \top, A) = 0 \times 1 \times 1 = 0$, $\mathit{Weight}_{simp}(\mathit{District}1, \top, \top, E) = 1 \times 1 \times 1 = 1$, $\mathit{Weight}_{simp}(\mathit{District}2, \top, \top, B) = 1 \times 1 \times 1 = 1$. Thus, the weighted grouping will, unlike the liberal grouping, show that E is *certain* to be in $\mathit{District}1$ while A only *may* be in $\mathit{District}1$. This can then be exploited by the aggregation function, e.g., by counting the facts with their *expected degree of membership* in the group, which is 1 for E and 0.5 for A, resulting in a count for $\mathit{District}1$ of 1.5.

6 Imprecision in aggregation paths

With partial containment in the model and transitivity of partial containment defined, we face the problem of how to choose the

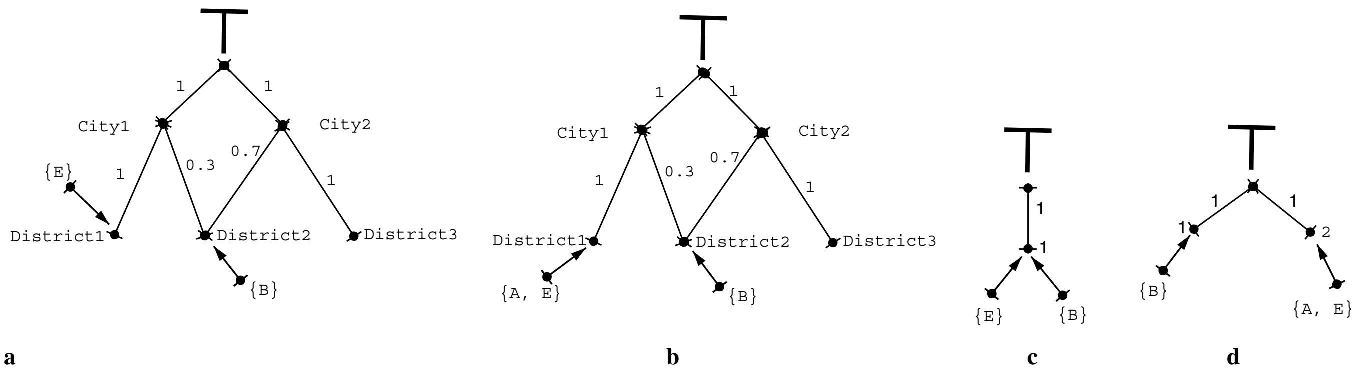


Fig. 8a–d. LOCATION and RESULT dimensions after the query α **a** and **c** describe conservative grouping. **b** and **d** describe liberal grouping

most precise data aggregation path while processing a query, or more generally how to evaluate the level of imprecision of a path. We address these aspects next.

Given a dimension $D = (C_D, P)$, an *aggregation path* of D is a sequence of distinct categories such that for any element of the sequence C_i and its successor C_j, the following holds: $C_j \in Anc^{(P)}(C_i)$. In our case, we could define the aggregation path $\alpha = \{Roadway, District, City, Province\}$.

Two aggregation paths α_i and α_j are *alternative aggregation paths* if $C^i_{first} = C^j_{first}$ and $C^i_{last} = C^j_{last}$, where C^i_{first} and C^j_{first} are their first elements and C^i_{last} and C^j_{last} are their last elements. So, α and $\alpha_{alt} = \{Roadway, Province\}$ are alternative aggregation paths. An aggregation path is *direct* if it consists of just two elements. Thus, α_{alt} is a direct aggregation path.

Recall that we term a relationship between dimension values $e_i \sqsubset_d e_j$ direct if it is given directly in the relation P (without using transitivity); otherwise, the relationship is indirect.

Given an aggregation path β, $e_i \in C_i$, and $e_j \in C_j$, we say that a direct relationship $e_i \sqsubset_d e_j$ is *in the path* β if C_j is the successor of C_i in β (in our case, $Roadway1 \sqsubset_{0.2} District1$ is a direct relationship in path α).

Consider an aggregation path $\gamma = \{C_{first}, \ldots, C_{last}\}$ and its alternative direct aggregation path γ_{alt}. We build two sets M and M_{alt} for γ and γ_{alt}, respectively. Both sets contain relationships $e_{first} \sqsubset_d e_{last}$, where $e_{first} \in C_{first}$ and $e_{last} \in C_{last}$. Set M contains indirect relationships that are deduced using the transitivity property on the relationships in the path γ. Set M_{alt} contains relationships in the path γ_{alt}.

We proceed to compare the aggregation paths, for which purpose we introduce the concept of *imprecision level* of an aggregation path γ (denoted IL_γ). The imprecision level IL_γ is evaluated by the following algorithm:

```
(1) procedure EvaluateImprecision
(2)   $IL_\gamma \leftarrow 0$
(3)   for each $(e_{first} \sqsubset_{d_{M_{alt}}} e_{last}) \in M_{alt}$
(4)   where $e_{first} \in C_{first} \wedge e_{last} \in C_{last}$ do
(5)     if $(e_{first} \sqsubset_{d_M} e_{last}) \in M$
(6)       then $M \leftarrow M \setminus \{(e_{first} \sqsubset_{d_M} e_{last})\}$
(7)            $IL_\gamma \leftarrow IL_\gamma + |d_{M_{alt}} - d_M|$
(8)       else $IL_\gamma \leftarrow IL_\gamma + d_{M_{alt}}$
(9)   for each $(e_{first} \sqsubset_{d_M} e_{last}) \in M$ do $IL_\gamma \leftarrow IL_\gamma + d_M$
```

The algorithm distinguishes among three cases. First, if the link between two values exists in both paths (lines 6–7), the difference between the containment degrees is added to the running imprecision level total. Second, if the link exists only in the alternative path (line 8), the alternative containment degree is added. Third, for the remaining links, i.e., links existing only in the original paths, the original containment degrees are added. Note that if there is more than one link between two values in a given aggregation path, e.g., $e_{first} \sqsubset_{p_1} e_{last}$ and $e_{first} \sqsubset_{p_2} e_{last}$, then the link with the maximum degree, i.e., $e_{first} \sqsubset_{max(p_1,p_2)} e_{last}$, appears in the algorithm.

The algorithm gives a high imprecision-level value if there is a large difference between the degrees of containment in the original and those in the alternative path. Note that the algorithm is meant only for *choosing* the best path among several alternatives, i.e., the algorithm does not provide an absolute estimate of the imprecision. This definition of imprecision level conforms to the intuitive understanding of imprecision, i.e., the greater the value of the imprecision level of a path, the more imprecise results from aggregating data along the path become.

The method does not take noncovering mappings into consideration. Specifically, it ignores direct relationships that exist to accommodate noncovering hierarchies, e.g., relationships between dimension values in the *Coordinate* and *District* categories. In this case, we introduce additional imprecision, and the result of the algorithm could lead to the choice of a wrong candidate for the most precise aggregation path. Thus, we must use the ignored relationships. However, it is not reasonable to use as an alternative direct aggregation path the one that is included to accommodate noncovering hierarchies because this introduces excessive imprecision in the structure. For example, if not every coordinate is directly related to a province, $\delta_{alt} = \{Coordinate, Province\}$ must *not* be used.

The proposed method is helpful in choosing aggregation paths while managing the trade-off between the speed of aggregation and the amount of preaggregation applied to the warehouse. Without preaggregation, there is no difference in speed between aggregation along a path and the alternative direct path. However, with preaggregation we can speed up aggregation along a given path considerably by precomputing some steps in it. Thus, having evaluated the imprecision level and aggregation speed (possibly with different preaggregation variants) of several alternative aggregation paths, we could choose the path with the highest speed (if we are interested in speed), the path with the lowest imprecision level (if we are interested in precision), or a path that balances aggregation speed and imprecision.

The applicability of the method is limited by the fact that in many cases we do not have an alternative direct aggregation path. In this situation, we can split the path according to the available paths and evaluate the level of imprecision of its subpaths. For example, in our case, we cannot use the method directly for the path $\epsilon = \{Coordinate, Roadway, District, City, Province, Country, \top\}$ because the path $\epsilon_{alt} = \{Coordinate, \top\}$ is absent. Instead, we could split ϵ into two subpaths, $\epsilon^1 = \{Coordinate, Roadway, District, City, Province\}$ and $\epsilon^2 = \{Province, Country, \top\}$. Then ϵ^1 could be compared to ϵ^1_{alt} (supposing every coordinate is directly related to a province) to get the imprecision level of the subpath.

7 Hierarchy transformations

Using the data model described in Sect. 4, we are able to specify nonnormalized dimension hierarchies. However, such hierarchies pose problems to practical preaggregation, as will be explained shortly. In this section, we briefly present algorithms that normalize hierarchies (the algorithms are described in detail elsewhere [22, 24]) and extend the algorithms to accommodate partial containment. Due to space constraints, the details are deferred to the appendix.

If a hierarchy is noncovering, then some links between dimension values skip one or more levels. Thus, at a "skipped" level some data are missing, and we cannot use aggregation results at a "skipped" level for computing aggregates at higher levels. For example, suppose a coordinate $Coord1$ does not lie on any roadway, but in district $District1$. The mapping from category $Roadway$ to category $District$ is then noncovering with respect to the category $Coordinate$, and the level of $Roadway$ is "skipped." Thus, we cannot use aggregates at the $Roadway$ level to calculate aggregate results at the $District$ level.

If a dimension hierarchy is non-onto, we are not able to reach some dimension values moving from bottom to top. Again, this means that we cannot reuse aggregates at the lowest level to compute aggregate results for the levels with "unreachable" values. For example, suppose there are no computers (IP addresses) in city *City1*. The mapping from category $IPAddress$ to category $City$ is non-onto and $City1$, the dimension value of $City$ category, is "unreachable." Thus, we are unable to use aggregates at the $Coordinate$ level to calculate aggregate results at the $City$ level.

Finally, if a hierarchy is nonstrict, some dimension values have multiple parents. Thus, moving from a child level to a parent level, we may count the same data several times. This means that we are unable to use aggregation results at the parent level to compute aggregates at the grandparent level. For example, suppose a roadway $Roadway1$ crosses districts $District1$ and $District2$, so that $Roadway1$ has two parent dimension values, $District1$ and $District2$, in category $District$. Moving from the level of $Roadway$ to the level of $District$, we count aggregates for $Roadway1$ twice. Thus, we cannot use aggregates at the level of $District$ to calculate aggregate results at the level of $City$.

A suite of hierarchy transformation algorithms remove these problems and thus enable correct use of preaggregation for nonnormalized hierarchies transparently to the user. The algorithms normalize dimension hierarchies in three steps: hierarchies are first made covering, then onto, and finally aggregation strict.

Hierarchies are made covering by introducing intermediate "placeholder" values in the levels that are skipped by the original links between dimension values and by linking these values appropriately to the relevant existing dimension values (which may be placeholder values). Similarly, a hierarchy is made onto by "padding" the hierarchy downwards by inserting "dummy" child values for dimension values with no children. Aggregation strictness is achieved by "fusing" the set of parent values of a lower-level dimension value into one fused value and linking this new value to the appropriate child, parent, and grandparent values before continuing the process upwards in the hierarchy.

The transformations include the rearranging of relationships between dimension values, but they do not take the partial containment relationships and their degrees of containment into account in the process.

When extending the transformations, we must assign the proper degrees of containment with the relationships created during the transformations. We pose two requirements to the extended algorithms. First, full containment relationships between dimension values must be preserved. The reason is that if we guarantee that a dimension value is fully contained in another one, then this is the best we can get for the values. It would be unreasonable to eliminate or degrade such relationships. Second, as we follow the safe approach to the transitivity of containment, a new degree must be less than or equal to the corresponding old one.

Example 19. The workings of the algorithms is best illustrated by an example. For the PMakeCovering algorithm, consider the transformation in Fig. 9a. Here the direct link between e_k (using the example of a noncovering hierarchy in the beginning of this section, e_k may correspond to $District1$) and e_{i_2} (corresponding to $Coord1$) is transformed by inserting a new placeholder value Ce_k (in our example, $CDistrict1$) in the intermediate category. This new value is then linked to e_k and e_{i_2}. The degree of containment of the first link is naturally 1, while the degree of the second link is inherited from the original direct link. For the PMakeOnto algorithm, consider the transformation in Fig. 9b. Here a new dummy value Le_{j_2} (or $LCity1$ in our example of a non-onto hierarchy) is inserted below e_{j_2} ($City1$) with a natural containment degree of 1. For the PMakeStrict algorithm, consider the transformation in Fig. 10. Here the multiple parents of value e_i (corresponding to $Roadway1$ in our example introducing a nonstrict hierarchy) are fused into one new value $e = \{e_{j_1}, \ldots, e_{j_m}\}$ (in our example, $m = 2$ and the fused value is $\{District1, District2\}$), which is then linked to e_i and the parents $e_{j_1}, \ldots, e_{j_m}$ (i.e., $District1$ and $District2$). To keep the figure simple, we have not shown additional values in the e_i category. The degree of containment is computed to provide the highest possible degree that can still be guaranteed. See the appendix for details.

The transformation algorithms described in the above example enable the use of practical preaggregation while still preserving the information about partial containment.

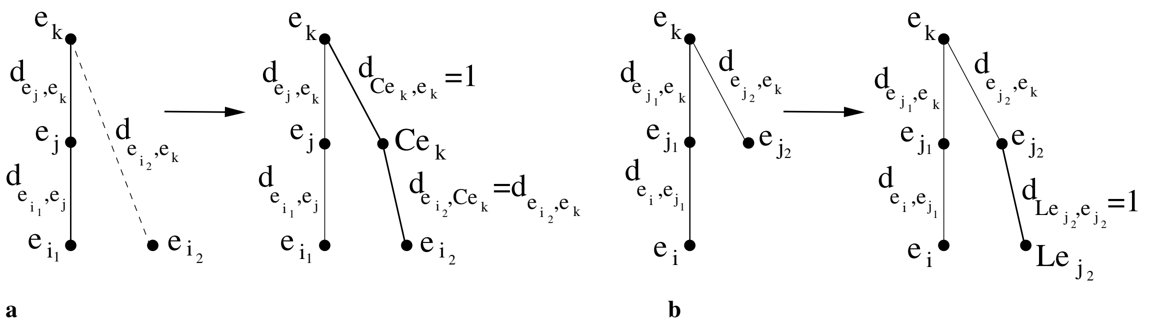


Fig. 9a,b. Transformations by the **a** PMakeCovering and **b** PMakeOnto algorithms

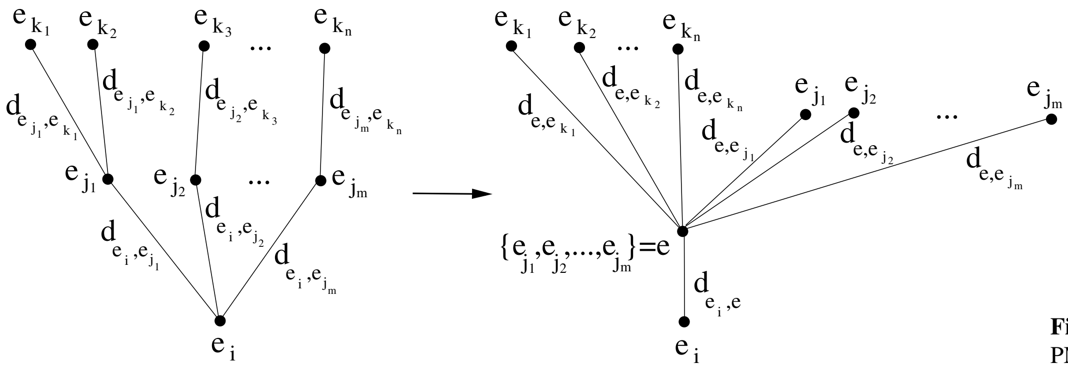


Fig. 10. Transformations by the PMakeStrict algorithm

8 Conclusions and research directions

Mobile, location-based e-services promise to become a significant application domain for multidimensional modeling of spatial data. We envision that such services will store data about their interaction with the users in multidimensional data warehouses, which will be subsequently used to analyze the data in order to improve the services. This paper aims to provide better support for this application domain in multidimensional data warehouses, the focus being on data modeling and aggregation. This domain poses a number of interesting requirements to a multidimensional data model. One of them, partial containment among dimension values, is not supported by existing data models. This paper extends an existing, so-called prototypical, model that already satisfies other important requirements to also support partial containment.

A key problem faced is to define transitivity of partial containment. The essence of the problem is which degree of containment to assign to an inferred relationship: if $((e_i \sqsubset_{d_i} e_j) \wedge (e_j \sqsubset_{d_j} e_k)) \Rightarrow (e_i \sqsubset_d e_k)$, what is the value of d? We have provided a "safe" definition of partial containment, meaning that we deduce only those relationships that must hold. But at the same time, we try to maximize the containment degrees in inferred relationships. We feel that this is the most basic and useful approach.

The paper also defines an algebra – with extended selection, union, and aggregate formation operators – that we believe serves well as a formal basis for an end-user query language for our model. Several elements of the algebraic operators reflect the support for partial containment. Specifically, the predicate for selection depends on degrees of containment as well as on dimension values. In addition, the union operator on dimension assigns proper degrees of containment to resulting dimension values. Finally, the aggregate formation operator uses different grouping strategies.

The presence of partial containment introduces imprecision in dimension hierarchies. In particular, one aggregation path from one level in a hierarchy to a higher level may be more precise than an alternative path. In order to make informed decisions on which path to choose among several alternatives, the paper provides a means of evaluating the imprecisions of paths. This allows the selection of paths based on their precisions and associated speeds of aggregation.

Enabling practical preaggregation requires that dimension hierarchies be onto, covering, and aggregation strict. The paper extends existing hierarchy transformations to support partial containment. This extension satisfies the requirements that full containment relationships between dimension values be preserved and that the degrees of containment in partial containment relationships be as close to the original degrees as possible, but not higher (safe approach). It is relatively straightforward to provide transformations for making hierarchies onto and covering and at the same time retain old degrees. Incorporating support for partial containment into the transformation that makes hierarchies aggregation strict is a more complex task. We preserve full containment at the expense of degrees for partial containment relationships (some degrees reduce to "zero").

It would be of interest to study several aspects of the proposed data model further. The paper adopts a "safe" approach to inferring partial containment relationships among dimension values. Other approaches could be explored where "probable" partial relationships are inferred. Such approaches could be used together with the safe approach. In addition, it is possible to use more information when safely inferring partial containment relationships.

Next, with the partial containment, summarizability [27] is generally not guaranteed, even if a multidimensional object is normalized. It is of interest to determine which conditions are sufficient for summarizability with partial containment.

Moreover, the extended aggregate formation operator does not handle the imprecision introduced by mapping of facts to dimension values of different granularities. It seems that the approach suggested in the literature [26] to handling this sort of imprecision can be extended to also contend with partial containment.

Furthermore, it is very relevant to devise a prototype implementation of the model using an existing OLAP system [32]. Data models of existing systems do not meet all the formulated requirements. Therefore, they do not provide direct support for all the elements of our model. This raises issues related to model-to-model transformations.

Finally, it is important to evaluate the applicability of the model. This could be achieved by populating a data warehouse that implements the model with the real-world spatial data and then performing experiments with the warehouse. Appropriate data on geo-referenced transportation infrastructures are currently available from a database of a location-based service delivery system developed and deployed by the Danish company Euman A/S [7].

Acknowledgements. This research was supported in part by grants from the Danish National Centre for IT Research, the Nordic Academy for Advanced Study, and the Nykredit Corporation.

We are thankful to the anonymous reviewers and the associate editor for comments that helped improve the presentation.

A Hierarchy transformation algorithms

The appendix describes the hierarchy transformation approach in detail.

A.1 Preaggregation and nonnormalized hierarchies

To *preaggregate* data means to store results of aggregate queries. It is done in order to decrease the query response time of a data warehouse system. However, the results of *full preaggregation*, when all combinations of aggregates are precomputed, may require too much storage space, making full preaggregation impractical. Modern data warehouse systems instead use *practical preaggregation*, which means that only select combinations of aggregates are stored and reused later for computing other aggregates.

Furthermore, it is important that we can preaggregate values at any combination of dimension levels and reuse the preaggregated values to compute higher level aggregate results. If this requirement is met, we say that our data is *summarizable*. In [27], it is mentioned that data captured by the model that we present in Sect. 4 (though without our extension for handling partial containment) is summarizable, if the multidimensional object for the data is normalized. So the problems with practical preaggregation occur if, in a multidimensional object, facts map to dimension values at different levels or dimension hierarchies are either non-onto, noncovering, or nonstrict.

Generally, preaggregation at the lowest level of a dimension does not take into consideration fact-dimension relationships at higher levels. So if all facts do not map to dimension values at the lowest level, some data are missing. This means that we cannot reuse aggregates at the lowest level to compute higher-level aggregate results. For example, suppose a user has issued a request from the roadway $Roadway1$. The corresponding fact is then mapped to the value $Roadway1$, which belongs to the category $Roadway$, not the lowest one. Data for the request are missing at the lowest level. We are thus unable to use aggregates at the level of the lowest category $Coordinate$ to calculate aggregate results at the level of $Roadway$.

With the extension for capturing partial containment introduced into the model, generally data in a normalized multidimensional object may become nonsummarizable. The proposed transitivity property infers only guaranteed (conservative) relationships between dimension levels, which means that real-world relationships exist that will be missed. In spite of this, normalization of dimension hierarchies is a first step toward achieving summarizability.

A.2 Hierarchy transformation algorithms

We present pseudocode for the algorithms. The input to each algorithm is a set of tables $R_{C_i,C_j,X}$ that specifies the relationships between dimension values in categories C_i and C_j ($C_j \in Anc^{(P)}(C_i)$). Column X in the table captures degrees of containment. In Figs. 9a, 9b, and 10, we illustrate the effects of the algorithms that make hierarchies covering, onto, and aggregation strict. On the left, a fragment of a dimension hierarchy is depicted, while on the right, the same fragment after the transformation is presented. When analyzing the algorithms, we do not cover in detail how the mappings are transformed (this is done elsewhere [22,23]). Instead, we explain the extension and how the transformed hierarchies meet the formulated requirements.

A.3 Making hierarchies covering

In Fig. 9a, we assume that $e_{i_1} \in C_i, e_{i_2} \in C_i, e_j \in C_j, Ce_k \in C_j$, and $e_k \in C_k$. We also assume that $C_k \in Anc^{(P)}(C_j)$, $C_k \in Anc^{(P)}(C_i)$, and $C_j \in Anc^{(P)}(C_i)$.

Before the transformation we have: $e_{i_2} \sqsubset_{d_{e_{i_2},e_k}} e_k$. We may retain the same relationship between the values after the transformation, if for the new degrees of partial containment we let $d_{e_{i_2},Ce_k} = d_{e_{i_2},e_k}$ and $d_{Ce_k,e_k} = 1$. By using f-to-f or p-to-f transitivity (depending on the values of the degrees), we get: $((e_{i_2} \sqsubset_{d_{e_{i_2},Ce_k}} Ce_k) \wedge (Ce_k \sqsubset_{d_{Ce_k,e_k}} e_k)) \Rightarrow (e_{i_2} \sqsubset_{d_{e_{i_2},e_k}} e_k)$. Thus, the relationship $e_{i_2} \sqsubset_{d_{e_{i_2},e_k}} e_k$ is retained by the algorithm.

Obviously, the first requirement is met: if before the transformation $d_{e_{i_2},e_k} = 1$, this also holds after the transformation. The second requirement is also met: the degree $d_{e_{i_2},e_k}$ never increases after the transformation (in fact, it remains the same).

Notice that after the transformation, altering the relationships between C_i, C_j, and C_k may be required. That is, we state that $Type(C_i) \sqsubset_{\mathcal{T}}^{P} Type(C_j)$ if $Type(C_i) \sqsubset_{\mathcal{T}}^{P} Type(C_k)$, and $Type(C_i) \sqsubset_{\mathcal{T}} Type(C_j)$ if $Type(C_i) \sqsubset_{\mathcal{T}} Type(C_k)$. By doing this we allow an arbitrary value of $d_{e_{i_2},Ce_k}$. In addition, we must add the relationship $Type(C_j) \sqsubset_{\mathcal{T}} Type(C_k)$, if it was not present before the transformation, to allow $d_{Ce_k,e_k} = 1$.

```
(1) procedure PMakeCovering($C$)
(2)  for each $P \in Anc^{(P)}(C)$ do
(3)  begin
(4)     for each $H \in Anc^{(P)}(C)$ where
           $Type(P) \sqsubset_{\mathcal{T}}^{(P)} Type(H)$ do
(5)     begin
(6)        $L \leftarrow \Pi_{C,H}(R_{C,H,X}) \setminus \Pi_{C,H}(\Pi_{C,P}(R_{C,P,X})$
                                         $\bowtie \Pi_{P,H}(R_{P,H,X}))$
(7)        $N \leftarrow \Pi_H(L)$
(8)        $P \leftarrow P \cup \{Mark(h) \mid h \in N\}$
(9)        $R_{P,H,X} \leftarrow R_{P,H,X} \cup \{(Mark(h), h, 1) \mid h \in N\}$
(10)       $R_{C,P,X} \leftarrow R_{C,P,X} \cup \{(c, Mark(h), d_{c,h}) \mid (c,h) \in L$
                                    $\wedge (c, h, d_{c,h}) \in R_{C,H,X}\}$
(11)    end
(12)    PMakeCovering($P$)
(13) end
```

A.4 Making hierarchies onto

In Fig. 9b, we assume that $e_i \in C_i$, $Le_{j_2} \in C_i$, $e_{j_1} \in C_j$, $e_{j_2} \in C_j$, and $e_k \in C_k$. We also assume that $C_k \in Anc^{(P)}(C_j)$ and $C_j \in Anc^{(P)}(C_i)$.

The transformation does not require altering the present degrees of containment. We let the completely new degree $d_{Le_{j_2},e_{j_2}} = 1$ (it conforms to the logic that a "placeholder" Le_{j_2} for a value e_{j_2} is fully contained in that value). Since no degrees are altered, the first and second requirements are met. Finally, we add the needed relationship between the categories C_i and C_j, i.e., $Type(C_i) \sqsubset_{\mathcal{T}} Type(C_j)$.

```
(1) procedure PMakeOnto($P$)
(2)  for each $C \in Desc^{(P)}(P)$ do
(3)  begin
(4)     $N \leftarrow P \setminus \Pi_P(R_{C,P,X})$
(5)     $C \leftarrow C \cup \{Mark(n) \mid n \in N\}$
(6)     $R_{C,P,X} \leftarrow R_{C,P,X} \cup \{(Mark(n), n, 1) \mid n \in N\}$
(7)     PMakeOnto($C$)
(8)  end
```

A.5 Making hierarchies aggregation strict

In Fig. 10, we assume that $e_i \in C_i$, $\forall l \in \{1,\dots,m\}(e_{j_l} \in C_j)$, $\forall l \in \{1,\dots,n\}(e_{k_l} \in C_k)$ and $e = \{e_{j_1}, e_{j_2}, \dots, e_{j_m}\} \in C_j^f$ (a category containing "fused" dimension values of C_j). We also assume that before the transformation $C_k \in Anc^{(P)}(C_j)$ and $C_j \in Anc^{(P)}(C_i)$, but after the transformation $C_k \in Anc^{(P)}(C_j^f)$, $C_j \in Anc^{(P)}(C_j^f)$, and $C_j^f \in Anc^{(P)}(C_i)$.

Before the transformation we have the set of relationships $L = \{e_i \sqsubset_{d_{e_i,e_{j_l}}} e_{j_l}, l = 1,\dots,m\}$. We would like to have these relationships between the values with the same degrees after the transformation. But notice that in the transformed hierarchy, the relationships can be inferred only with the help of transitivity: $\forall l \in \{1,\dots,m\}(((e_i \sqsubset_{d_{e_i,e}} e) \wedge (e \sqsubset_{d_{e,e_{j_l}}} e_{j_l})) \Rightarrow (e_i \sqsubset_{d_{e_i,e_{j_l}}} e_{j_l}))$. The common initial condition for each relationship in set L is: $e_i \sqsubset_{d_{e_i,e}} e$.

In general, the common condition does not allow us to retain all the old degrees of containment in the set. We deal with the situation as follows: if set L has any full containment relationships, we retain them (using f-to-f transitivity, the first step of which is to let $d_{e_i,e} = 1$); otherwise, we perform an approximation, which conforms to the "safe" approach, and retain the minimal degree in the set (using p-to-f transitivity, the first step of which is to let $d_{e_i,e} = min(\{d_{e_i,e_{j_l}}, l = 1,\dots,m\})$. After having assigned a value to $d_{e_i,e}$, we perform further assignments. If we retain the full containment then for completion, we infer relationships as follows: $\forall l \in \{1,\dots,m\}(((d_{e_i,e_{j_l}} = 1) \Rightarrow (d_{e,e_{j_l}} = 1)) \wedge ((d_{e_i,e_{j_l}} \neq 1) \Rightarrow (d_{e,e_{j_l}} = 0)))$. If we retain the minimal degree in the set then for completion, we must let $\forall l \in \{1,\dots,m\}$ $(d_{e,e_{j_l}} = 1)$. By p-to-f transitivity we then get the required result: $\forall l \in \{1,\dots,m\}(e_i \sqsubset_{min(\{d_{e_i,e_{j_l}}, l=1,\dots,m\})} e_{j_l})$. Notice that in the former case, for some relationships we get $e_i \sqsubset_1 e_{j_l}$ (by f-to-f transitivity), but for some relationships we cannot avoid using f-to-p transitivity, meaning that we get $e_i \sqsubset_0 e_{j_l}$. This means that we do preserve full containment but miss the guaranteed degrees that were less than 1.

We also need to contend with the set of relationships $N = \{e_i \sqsubset_{d_{e_i,e_{k_l}}} e_{k_l}, l = 1,\dots,n\}$ (defined for the hierarchy prior to the transformation). In doing so, we take into consideration that actions for set L have already been performed. Again, we would like for the transformation to retain these relationships with the same degrees. And again the relationships in the transformed hierarchy can be inferred only by transitivity ($\forall l \in \{1,\dots,n\}(((e_i \sqsubset_{d_{e_i,e}} e) \wedge (e \sqsubset_{d_{e,e_{k_l}}} e_{k_l})) \Rightarrow (e_i \sqsubset_{d_{e_i,e_{k_l}}} e_{k_l})))$.

Notice that the common initial condition for each relationship in set N is already given after we have dealt with the set L. Although this of course limits our capabilities, we follow the same logic that was applied in the case of set L: if set N has any full containment relationships, we retain them; otherwise, we apply a conservative approximation.

We can preserve full containment because any full containment relationships in set N have degree $d_{e_i,e}$ equal to 1 in the first condition. In this case, we partition set N into N_1 and N_p such that elements of set N_1 are full containment relationships, while elements of set N_p are partial containment relationships. We infer relationships as follows: $\forall l \in \{1,\dots,n\}(((e_{k_l} \in N_1) \Rightarrow (d_{e,e_{k_l}} = 1)) \wedge ((e_{k_l} \in N_p) \Rightarrow (d_{e,e_{k_l}} = 0)))$. By doing so, we retain full containment where it is present but miss guaranteed degrees that are less than 1 (analogously with the case of set L).

If $d_{e_i,e}$ is not equal to 1, the set N does not have any full containment relationships. We then perform approximation as follows: $\forall l \in \{1,\ldots,n\}((\exists e_{j_{l'}}(e_{j_{l'}} \sqsubset_{d_{e_{j_{l'}},e_{k_l}}} e_{k_l})) \wedge ((d_{e_{j_{l'}},e_{k_l}} = 1) \Rightarrow (d_{e,e_{k_l}} = 1)) \wedge (d_{e_{j_{l'}},e_{k_l}} \neq 1) \Rightarrow (d_{e,e_{k_l}} = 0))$. Thus, for some relationships in the set N, by p-to-p transitivity we get $e_i \sqsubset_0 e_{k_l}$ (this is true before the transformation as well) and for some relationships in L by p-to-f transitivity we get $e_i \sqsubset_{d_{e_i,e_{k_l}}} e_{k_l}$ (the degree is less than that before the transformation, but still greater than zero).

We see that the first and the second requirements to the procedure of inferring new degrees of containment are met. The drawback of the algorithm is that in some cases it substitutes zero degrees for nonzero ones. For example, if $n = l = 2$, then there are 4 degrees to work with and 16 variants for the set of degree values (each degree is allowed to be one or nonone). Therefore, there are 64 relationships. Twelve of them turn their degree from nonzero to zero.

In the transformed dimension, we modify the relationships between category types as needed. If we need partial containment for the algorithm to work, we introduce such relationships. Otherwise, we only allow full containment. Specifically, we perform the modification of the relationships between category types according to the rules presented in the list below. We assume that $Type(C_i) = \mathcal{C}_i$, $Type(C_j) = \mathcal{C}_j$, $Type(C_k) = \mathcal{C}_k$, and $Type(C_j^f) = \mathcal{C}_j^f$.

1. $(\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j \sqsubset_{\mathcal{T}} \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubset_{\mathcal{T}} \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubset_{\mathcal{T}} \mathcal{C}_k))$
2. $(\mathcal{C}_i \sqsubset_{\mathcal{T}}^P \mathcal{C}_j \sqsubset_{\mathcal{T}} \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubset_{\mathcal{T}}^P \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubset_{\mathcal{T}}^P \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubset_{\mathcal{T}} \mathcal{C}_k))$
3. $(\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j \sqsubset_{\mathcal{T}}^P \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubset_{\mathcal{T}} \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubset_{\mathcal{T}} \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubset_{\mathcal{T}}^P \mathcal{C}_k))$
4. $(\mathcal{C}_i \sqsubset_{\mathcal{T}}^P \mathcal{C}_j \sqsubset_{\mathcal{T}}^P \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubset_{\mathcal{T}}^P \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubset_{\mathcal{T}}^P \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubset_{\mathcal{T}}^P \mathcal{C}_k))$

Notice that the pseudocode contains a new (compared to the variant in [22] and [23]) function $onPath : [0; 1] \rightarrowtail [0; 1]$. The function works as follows: if $((e_i \sqsubset_{d_i} e_j) \wedge (e_j \sqsubset_{d_j} e_k)$ then $onPath(d_j) = d_i$.

```
procedure PMakeStrict($C$)
 for each $P \in Anc^{(P)}(C)$ do
 begin
    if $(\exists e_1 \in C(\exists e_2, e_3 \in P(\exists d_{1,2}, d_{1,3}(e_1 \sqsubset_{d_{1,2}} e_2$
                 $\wedge\, e_1 \sqsubset_{d_{1,3}} e_3 \wedge e_2 \neq e_3)))) \;\wedge\; Anc^{(P)}(P) \neq \emptyset$
    then begin
     $N \leftarrow CreateCategory(2^P)$
     if $\exists d_{1,2}\ (d_{1,2} = 1) \;\wedge\; (e_1, e_2, d_{1,2}) \in R_{C,P,X}$
     then $R_{C,N,X} \leftarrow \{(e_1, Fuse(\{e_2|$
                              $(e_1, e_2, d_{1,2}) \in R_{C,P,X}\}), 1)\}$
    else $R_{C,N,X} \leftarrow \{(e_1, Fuse(\{e_2 \mid (e_1, e_2, d_{1,2}) \in R_{C,P,X}\}),$
                     $min(\{d_{1,2} \mid (e_1, e_2, d_{1,2}) \in R_{C,P,X}\}))\}$
      $N \leftarrow \Pi_N(R_{C,N,X})$
      $R_{N,P,X} \leftarrow \{(e_0, e_2, \emptyset) \mid e_0 \in N \wedge e_2 \in Unfuse(e_0)\}$
      for each $e_2 \in Unfuse(e_0)$ where $e_0 \in N$ do
      begin
      if $d_{1,0} = 1 \;\wedge\; d_{1,2} \neq 1 \;\wedge\; (e_1, e_0, d_{1,0}) \in R_{C,N,X}$
         $\wedge\, (e_1, e_2, d_{1,2}) \in R_{C,P,X}$
      then $R_{N,P,X} \leftarrow R_{N,P,X} \cup \{(e_0, e_2, 0)\} \;\setminus\; \{(e_0, e_2, \emptyset)\}$
      else $R_{N,P,X} \leftarrow R_{N,P,X} \cup \{(e_0, e_2, 1)\} \;\setminus\; \{(e_0, e_2, \emptyset)\}$
      end
      $Anc^{(P)}(C) \leftarrow Anc^{(P)}(C) \cup \{N\} \setminus \{P\}$
      $Anc^{(P)}(N) \leftarrow \{P\}$
      for each $G \in Anc^{(P)}(P)$ do
      begin
      $L \leftarrow \Pi_G(\Pi_{N,P}(R_{N,P,X}) \bowtie \Pi_{P,G}(R_{P,G,X}))$
      $R_{N,G,X} \leftarrow \{(e_0, e_3, \emptyset) \mid e_0 \in N \;\wedge\; e_3 \in L\}$
      for each $e_3 \in L$ do
      begin
          if $d_{1,0} = 1 \;\wedge\; (e_1, e_0, d_{1,0}) \in R_{C,N,X} \;\wedge\; e_0 \in N$
          then
           if $d_{2,3} = 1 \wedge d_{1,2} = 1 \wedge d_{1,2} \in onPath(d_{2,3}) \wedge e_2 \in P$
           then $R_{N,G,X} \leftarrow R_{N,G,X} \;\cup \{(e_0, e_3, 1)\} \setminus \{(e_0, e_3, \emptyset)\}$
           else $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 0)\} \setminus \{(e_0, e_3, \emptyset)\}$
          else if $d_{2,3} = 1 \;\wedge\; e_2 \in P$
           then $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 1)\} \setminus \{(e_0, e_3, \emptyset)\}$
           else $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 0)\} \setminus \{(e_0, e_3, \emptyset)\}$
      end
      $Anc^{(P)}(N) \leftarrow Anc^{(P)}(N) \cup \{G\}$
      $Anc^{(P)}(P) \leftarrow Anc^{(P)}(P) \setminus \{G\}$
      end
      PMakeStrict($N$)
    end
    else PMakeStrict($P$)
 end
```

References

1. Blaha M (ed) (2001) Special section: data warehouses. IEEE Comput Mag 34(12):38–79
2. Gargano M, Nardelli E, Talamo M (1991) Abstract data types for the logical modeling of complex objects. Inform Sys 16(6):565–583
3. Daniel L, Loree P, Whitener A (2001) Inside MapInfo Professional. OnWord Press, Santa Fe
4. Dyreson CE (1996) Information retrieval from an incomplete data cube. In: Proc. 22nd international conference on very large databases, Bombay, 3–6 September 1996, pp 532–543
5. Dyreson CE (1997) A bibliography on uncertainty management in information systems. In: Motro A, Smets P (eds) Uncertainty management in information systems: from needs to solutions. Kluwer, Amsterdam, pp 413–458
6. Egenhofer MJ (1994) Spatial SQL: a query and presentation language. IEEE Transactions on knowledge and data engineering, 6(1):86–95
7. Euman (2003) http://www.euman.com. Current as of March 14, 2003
8. European Parliament and Council of the European Union (2002) Directive 2002/58/EC—Directive on Privacy and Electronic Communications. Available at: http://europa.eu.int/information_society/topics/telecoms/regulatory/new_rf/documents/l_20120020731en00370047.pdf. Current as of March 31, 2003
9. Ferri F, Pourabbas E, Rafanelli M, Ricci FL (2000) Extending geographic databases for a query language to support queries involving statistical data. In: Proc. 12th international conference on scientific and statistical database management, 26–28 July 2000, Berlin, pp 220–230
10. Güting RH, Böhlen MH, Erwig M, Jensen CS, Lorentzos NA, Schneider M, Vazirgiannis M (2000) A foundation for representing and querying moving objects. ACM Transactions on database systems, 25(1):1–42
11. Jensen CS, Pedersen TB (2001) Mobile e-services and their challenges to data warehousing. Datenbank Rundbrief 27:8–16
12. Jensen CS (2002) Research challenges in location-enabled m-services. In: Proc. 3rd international conference on mobile data management, 8–11 January 2002, Singapore, pp 3–7
13. Jensen CS, Kligys A, Pedersen TB, Timko I (2002) Multidimensional data modeling for location-based services. In: Proc. 10th ACM international symposium on advances in geographic information systems, 8–9 November 2002, McLean, VA, pp 55–61
14. Johnston K, Ver Hoef JM, Krivoruchko K (2001) Using ArcGIS Geostatistical Analyst. ESRI Press, Redlands, CA
15. Kimball R, Reeves L, Ross M, Thornthwaite W (1998) The data warehouse lifecycle toolkit. Wiley, New York
16. Lenz H, Shoshani A (1997) Summarizability in OLAP and statistical databases. In: Proc. 9th international conference on scientific and statistical database management, 11–13 August 1997, Olympia, WA, pp 39–48
17. Murray C (2002) Oracle spatial user guide and reference, Release 9.2. Oracle Corporation
18. National Cooperative Highway Research Program (1997) A generic data model for linear referencing systems. Transportation Research Board
19. Open GIS Consortium (1999) Open GIS simple features specification for SQL (Revision 1.1). Available at: http://www.opengis.org/techno/specs/99-049.pdf. Current as of March 31, 2003
20. Open GIS Consortium (2003) http://www.opengis.org. Current as of March 14, 2003
21. Open Mobile Alliance (2003) http://www.openmobilealliance.org. Current as of March 14, 2003
22. Pedersen TB, Jensen CS, Dyreson CE (1999) Extending practical pre-aggregation in on-line analytical processing. In: Proc. 25th international conference on very large databases, 7–10 September 1999, Edinburgh, pp 663–674
23. Pedersen TB (2000) Aspects of data modeling and query processing for complex multidimensional data. Ph.D. Thesis, Aalborg University, Aalborg Øst, Denmark
24. Pedersen TB, Jensen CS, Dyreson CE (2000) The TreeScape System: reuse of pre-computed aggregates over irregular OLAP hierarchies. In: Proc. 26th international conference on very large databases, Cairo, 10–14 September 2000, pp 595–598
25. Pedersen TB, Jensen CS (2001) Multidimensional database technology. IEEE Comput 34(12):40–46
26. Pedersen TB, Jensen CS, Dyreson CE (2001) A foundation for capturing and querying complex multidimensional data. Inform Sys 26(5):383–423
27. Pedersen TB, Tryfona N (2001) Pre-aggregation in spatial data warehouses. In: Proc. 7th international symposium on advances in spatial and temporal databases, Redondo Beach, CA, 12–15 July 2001, pp 460-478
28. Rafanelli M, Shoshani A (1990) STORM: a statistical object representation model. In: Proc. 5th conference on statistical and scientific database management, Charlotte, NC, 3–5 April 1990, pp 14–29
29. Rafanelli M (ed) (2002) Multidimensional databases: problems and solutions. Idea Group Publishing, Hershey, PA
30. Scarponcini P (2002) Generalized model for linear referencing in transportation. GeoInformatica 6(1):35–55
31. Scholl M, Voisard A (1989) Thematic Map Modeling. In: Abstracts of the 1st symposium on design and implementation of large spatial databases, Santa Barbara, 17–18 July 1989, pp 167–190
32. Thomsen E, Spofford G, Chase D (1999) Microsoft OLAP solutions. Wiley, New York
33. Vassiliadis P, Sellis TK (1999) A survey of logical models for OLAP databases. SIGMOD Record 28(4):64–69
34. World Wide Web Consortium (1999) POIX: Point Of Interest eXchange language specification. Available at: http://www.w3.org/TR/poix/. Current as of 31 March 2003
35. World Wide Web Consortium (2003) http://www.w3.org. Current as of 14 March 2003