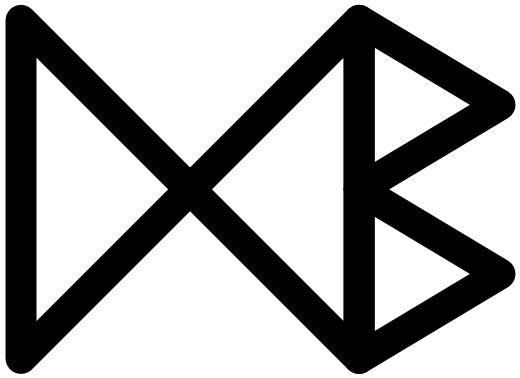

# Multidimensional Data Modeling for Location-Based Services

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

September 30, 2002

TR-2

A DB Technical Report

| | |
|---|---|
| Title | Multidimensional Data Modeling for Location-Based Services |
| |  |
| Author(s) | Christian S. Jensen, Augustas Kligys, Torben Bach Pedersen, Igor Timko |
| Publication History | |

For additional information, see the DB TECH REPORTS homepage: ⟨www.cs.auc.dk/DBTR⟩.



The DB TECH REPORTS icon is made from two letters in an early version of the Rune alphabet, which was used by the Vikings, among others. Runes have angular shapes and lack horizontal lines because the primary storage medium was wood, although they may also be found on jewelry, tools, and weapons. Runes were perceived as having magic, hidden powers. The first letter in the logo is "Dagaz," the rune for day or daylight and the phonetic equivalent of "d." Its meanings include happiness, activity, and satisfaction. The second letter is "Berkano," which is associated with the birch tree. Its divinatory meanings include health, new beginnings, growth, plenty, and clearance. It is associated with Idun, goddess of Spring, and with fertility. It is the phonetic equivalent of "b."

**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 of their users in multidimensional databases, and their delivery of content in response to user requests may be based on the issuing of complex, multidimensional queries.

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.

# 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, geo-positioning is becoming increasingly available and accurate.

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. Example objects of interest here include consumers using WAP-enabled mobile-phone terminals and personal digital assistants, tourists carrying on-line 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 [5]. 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 functionality. The services capture the requests, including their geographical origins, they receive in multidimensional databases, also called data warehouses [1]. Querying these databases enable 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. In addition, the accumulated data is used for delayed modification of the services provided, and for longer-term strategic decision making.

This scenario entails the capture of spatial data in a multidimensional database, which poses new data modeling challenges. For example, an appropriate data model should support non-normalized, i.e., non-onto, non-covering, or non-strict, dimension hierarchies [13] 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 for spatial data. For example, a street that extends from a city into a rural area is only partially contained in the city. Thus, partial containment relationships between hierarchy members, i.e., location entities such as streets and cities, must be supported by the conceptual 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 pre-aggregation techniques [17] should be applicable to databases conforming to the data model.

This paper first analyzes the mobile e-service application domain, formulating requirements to a conceptual model. It then presents a new multidimensional data model with an accompanying algebraic query language that arguably meets the requirements. For example, the model supports non-normalized 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. Perhaps most notably, partial containment leads to 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 pre-aggregation, i.e., pre-computation of select aggregate results that can be reused to obtain other aggregates, is a technique that is essential in 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 pre-aggregation techniques in an implementation of the model.

The present paper is a revised and substantially extended version of an earlier conference paper [6]. In particular, the contents of Sections 3.4, 4, 5, 6, and Appendix A are entirely new. To the knowledge of the authors, 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 [13, 18]. 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 [11, 13]. The model that we extend was chosen because it is formally defined and because it compares favorably to fourteen related data models [13]. The paper's algorithms for the normalization of partial containment dimension hierarchies extend algorithms presented by Pedersen et al. [10, 12] for use with the model being extended.

Pedersen and Tryfona [14] propose a different approach to the modeling of spatial data. The authors 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 in on how to support practical pre-aggregation with such overlapping facts. The conceptual model underlying that work is the model being extended here. Ferri et al. [4] propose a method to couple a multidimensional data model with a Geographical Information System (GIS) to get the combined power of these technologies.

The area of "imperfect" data has received a great deal of attention in general and specialized database contexts [3]. Within multidimensional databases, work has been done on irregular multidimensional data [2, 8, 13, 16] and the associated summarizability problems [9, 13, 15]. However, none of these works consider partial containment dimension hierarchies.

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

## 2 Usage Scenario and Requirements

This section introduces a prototypical usage scenario for a multidimensional database in the context of a location-based service, and it uses this scenario to illustrate important requirements to a multidimensional data model. The scenario is also used for exemplification throughout the paper. We initially describe the

usage scenario.

## 2.1 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 Figure 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 diagram uses its naming convention to distinguish among two different types of binary relationships among 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.

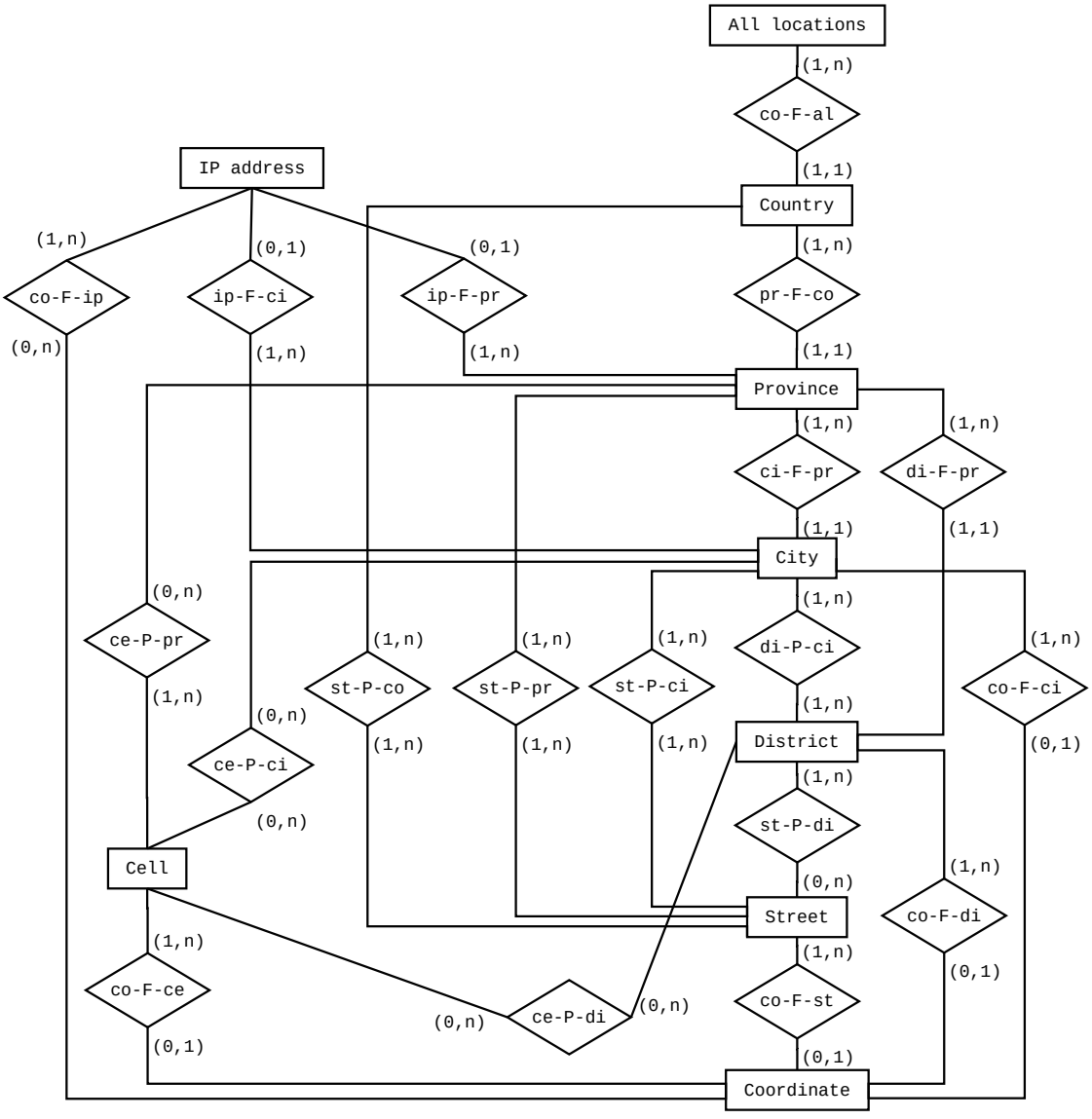


Figure 1: Location ER Diagram

For example, consider the relationship type co-F-st between entity types Coordinate and Street, and consider st-P-di, which relates Street and District. The meaning is that a coordinate is either fully contained or uncontained in a street, which may in turn additionally be contained only partially in a district.

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

The existence of a partial containment relationship type between entity types in the case study also implies the existence of a full containment relationship type between these 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 less objects will satisfy this relationship.

In a multidimensional data model, user requests are 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, Hour, etc. The USER dimension captures aspects of the users issuing the request with categories such as Spoken Language, Personal Interest, Actual Age, Main Occupation, etc. The LOCATION dimension captures the, possibly changing, locations of the users when the requests were issued. 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.

In Section 3, we illustrate how the Location ER diagram can be mapped to a LOCATION dimension.

## 2.2 Data Model Requirements

We discuss next the requirements for a multidimensional data model that contends with our usage scenario. While they are all highly relevant to our context, most of the requirements are more general and were formulated earlier. We describe the requirements only briefly and refer to the literature for further detail [13]. Other requirements are given elsewhere [7].

1. *Explicit and multiple hierarchies in dimensions* Dimension values are assigned to 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. 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 would get a more precise result if streets are directly rolled up to countries than if streets are rolled up to countries through districts, cities, and provinces.

2. *Partial containment* We have seen that two spatial values may not only be either disjoint or have one be 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 it will 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. *Non-normalized hierarchies* Situations occur naturally where a hierarchy value has more than one parent, where a value has no relationship to any value in the category immediately above it in the

dimension hierarchy, and where a value has no relationship to any value in any category below it. For example, a street 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 or at the level of a mobile phone cell.

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. This would be useful, e.g., in a situation where a request may be 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 on an aggregation path may become imprecise, in which case the result of a query may also be imprecise. In order to reduce the error in a result, it is important to handle 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 nor any other existing model.

# 3 Data Model

This section briefly describes the relevant aspects of the existing multidimensional data model [13], the focus being to extend it to support partial containment. The section also presents properties of the model, considers the model's fulfillment of the requirements, and discusses the use of the model for the design of dimensions.

## 3.1 Definition of the Data Model

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 $(\mathcal{C}_\mathcal{T}, \sqsubseteq_\mathcal{T}, \top_\mathcal{T}, \bot_\mathcal{T})$, where $\mathcal{C}_\mathcal{T} = \{\mathcal{C}_j, j = 1, \ldots, k\}$ are *category types* of the dimension type $\mathcal{T}$, $\sqsubseteq_\mathcal{T}$ is a partial order on the set $\mathcal{C}_\mathcal{T}$, and $\top_\mathcal{T}$ and $\bot_\mathcal{T}$ are the top and bottom elements of the order, respectively. 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 $\mathcal{C}_j$.

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

1. $\forall(\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}\ ((\mathcal{C}_i \neq \mathcal{C}_j) \wedge (\mathcal{C}_i \sqsubseteq^P_\mathcal{T} \mathcal{C}_j) \Rightarrow (\mathcal{C}_j \not\sqsubseteq^P_\mathcal{T} \mathcal{C}_i))$ (anti-symmetry)

2. $\forall(\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T} \times \mathcal{C}_\mathcal{T}\ (((\mathcal{C}_i \sqsubseteq^P_\mathcal{T} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubseteq^P_\mathcal{T} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubseteq^P_\mathcal{T} \mathcal{C}_k))$ (transitivity)

Relations $\sqsubseteq_\mathcal{T}$ and $\sqsubseteq^P_\mathcal{T}$ are related as follows.

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

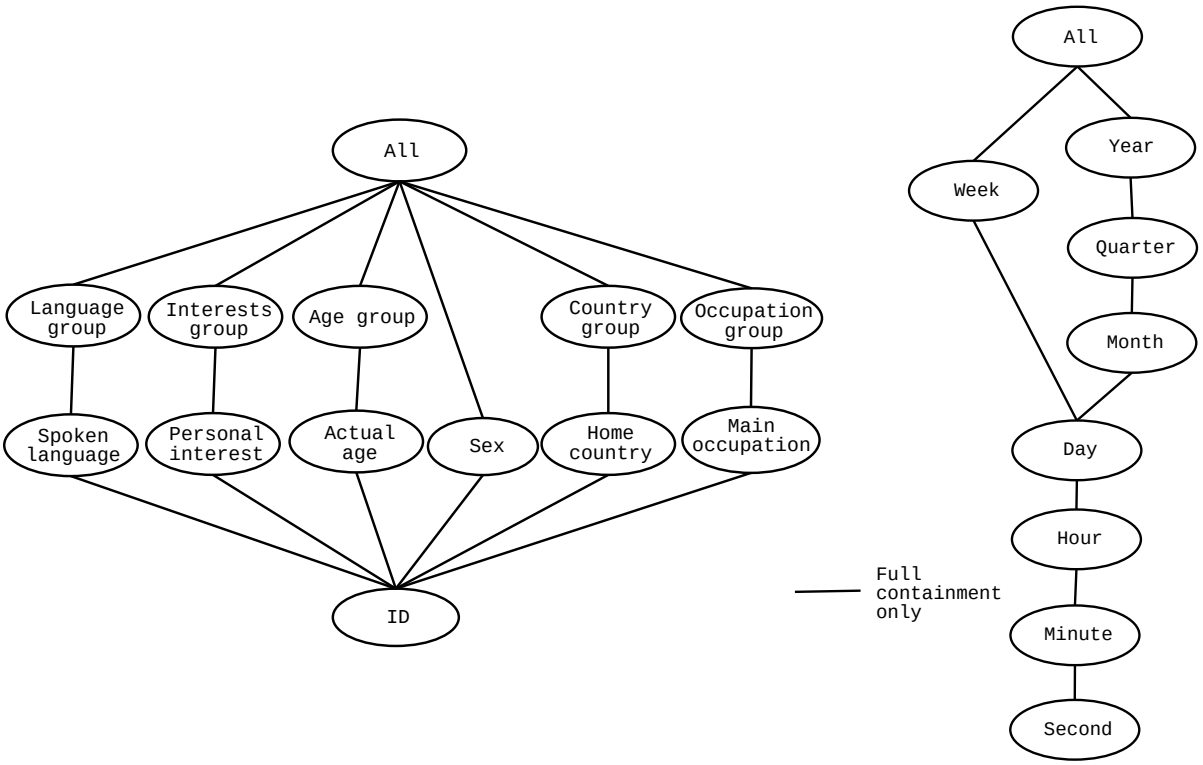


Figure 2: USER and TIME Dimensions

1. $((\mathcal{C}_i \sqsubseteq^P_{\mathcal{T}} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubseteq_{\mathcal{T}} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubseteq^P_{\mathcal{T}} \mathcal{C}_k))$
2. $((\mathcal{C}_i \sqsubseteq_{\mathcal{T}} \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubseteq^P_{\mathcal{T}} \mathcal{C}_k)) \Rightarrow (\mathcal{C}_i \sqsubseteq^P_{\mathcal{T}} \mathcal{C}_k))$

After the extension, a dimension type $\mathcal{T}$ is a five-tuple: $(\mathcal{C}_{\mathcal{T}}, \sqsubseteq_{\mathcal{T}}, \sqsubseteq^P_{\mathcal{T}}, \top_{\mathcal{T}}, \bot_{\mathcal{T}})$. We use the notation $\sqsubseteq^{(P)}_{\mathcal{T}}$ to indicate the union of the two orders $\sqsubseteq_{\mathcal{T}}$ and $\sqsubseteq^P_{\mathcal{T}}$. The functions $Anc^P$ and $Desc^P$ provide ancestors and descendants based on the $\sqsubseteq^P_{\mathcal{T}}$ relation. Similarly to above, the notation $Anc^{(P)}$ ($Desc^{(P)}$) provides ancestors (descendants) based on both relations.

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

Next, a fact schema defines the structure of some domain (in our case study, the domain of a mobile e-service) at a high level of abstraction. The fact schema states that facts are entities of a particular type (in our case, all the facts are requests of a mobile e-service), and it states that heterogeneous entities that characterize facts (cities, age groups, streets, years, IP addresses, personal interests, coordinates, minutes, job categories, etc.) are organized into dimensions, e.g., LOCATION and TIME dimensions. The definition of a dimension type refines the structure of the domain. We see that in a dimension, each entity type has a corresponding category type (e.g., *Coordinate*, *City*, etc.) and the types are organized into multiple containment hierarchies that reflect containment hierarchies of the domain (e.g., *Coordinate* $<$ *Cell* $<$ *Province* $<$ *Country* $<\top$ and *Coordinate* $<$ *IP address* $<$ *Province* $<$ *Country* $<\top$).

**Example 1** In Figures 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

mean relationships between category types. Note that application requirements generally affect the design of dimensions. For example, the LOCATION dimension models only selected aspects of the miniworld captured in the ER diagram in Figure 1.

In Section 2, we identify three dimensions, so we have a 3-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 $\mathcal{T}_{loc} = \{\mathcal{C}_{loc}, \sqsubseteq_{\mathcal{T}_{loc}}, \sqsubseteq^P_{\mathcal{T}_{loc}}, \mathcal{C}_{all}, \mathcal{C}_{coordinate}\}$. The relations on set $\mathcal{C}_{loc} = \{\mathcal{C}_{coordinate}, \mathcal{C}_{street}, \mathcal{C}_{district}, \mathcal{C}_{city}, \mathcal{C}_{province}, \mathcal{C}_{country}, \mathcal{C}_{ipaddress}, \mathcal{C}_{cell}, \mathcal{C}_{all}\}$ are given as follows: if there exists a relationship type of full (partial) containment variety between entity types, then the corresponding category types are related by $\sqsubseteq_{\mathcal{T}_{loc}}$ ($\sqsubseteq^P_{\mathcal{T}_{loc}}$) (e.g., $\mathcal{C}_{street} \sqsubseteq^P_{\mathcal{T}_{loc}} \mathcal{C}_{district}$ and $\mathcal{C}_{coordinate} \sqsubseteq_{\mathcal{T}_{loc}} \mathcal{C}_{province}$).

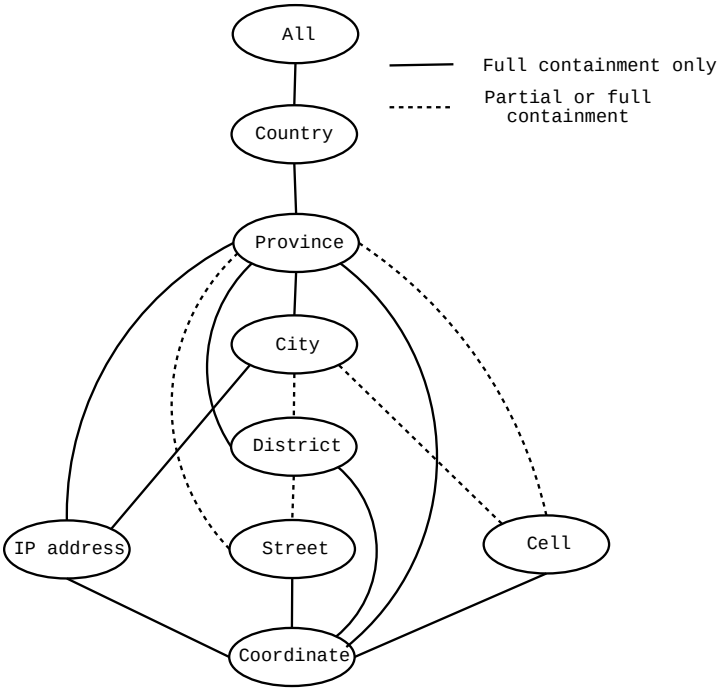


Figure 3: LOCATION Dimension

After defining the schemas, or *intensions*, of the data model, we proceed to define the *extensions* of the data model, starting out again with the prototypical data model.

A *dimension* of type $\mathcal{T}$ is a two-tuple $D = (C_D, \sqsubseteq)$, 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 $\sqsubseteq$ is a partial order on $\bigcup_j C_j$ (from now on we simply write $Dim$ instead of $\bigcup_j C_j$). The definition of the partial order is: given a pair of values $(e_i, e_j)$, $e_i \sqsubseteq e_j$ means that $e_i$ is fully contained in $e_j$. The category of type $\bot_{\mathcal{T}}$ contains values with the smallest size. The category of type $\top_{\mathcal{T}}$ (the largest value size) has exactly one value, denoted $\top$, containing all values in the dimension. It is assumed that the partial order on category types and the functions $Anc$ and $Desc$ work directly on categories, with the order given by the corresponding category types.

We extend the definition of a dimension by generalizing the existing partial order $\sqsubseteq$ on dimension values, which is capable only of expressing *full containment* hierarchies and so is not powerful enough for our needs. Specifically, we replace $\sqsubseteq$ by a relation $P \subseteq Dim \times Dim \times [0; 1]$. In a triple $(e_i, e_j, p) \in P$, we refer to the value $p$ as the *degree of containment*.

After we have introduced this extension, a dimension is a two-tuple $D = (C_D, P)$. We write as follows.

1. $(e_i, e_j, 1) \in P$ or simply $e_i \sqsubseteq_1 e_j$, if we *guarantee* that the dimension value $e_i$ is *fully* contained in the dimension value $e_j$;

2. $(e_i, e_j, p) \in P$ or simply $e_i \sqsubseteq_p e_j$, if we *guarantee* that the dimension value $e_i$ is *partially* contained in the dimension value $e_j$ and the smallest possible size of the contained part is $(p \cdot 100)$ % $(0 < p < 1)$;

3. $(e_i, e_j, 0) \in P$ or simply $e_i \sqsubseteq_0 e_j$, if it is *possible* that the dimension value $e_i$ is *partially* contained in the dimension value $e_j$;

4. $\forall p \in [0;1]((e_i, e_j, p) \notin P)$ or simply $e_i \not\sqsubseteq e_j$ if we *guarantee* that the dimension value $e_i$ is *neither fully nor partially* contained in the dimension value $e_j$.

Note that the value $\top$ of type $\top_{\mathcal{T}}$ (the largest value size) is required to *fully* contain all other values. Notice that if $e_i \in C_i$ and $e_j \in C_j$ ($Type(C_i) = \mathcal{C}_i$ and $Type(C_j) = \mathcal{C}_j$), then the following must hold: $\forall p \in [0;1)((e_i \sqsubseteq_p e_j) \Rightarrow (\mathcal{C}_i \sqsubseteq_{\mathcal{T}}^{P} \mathcal{C}_j))$ and $(e_i \sqsubseteq_1 e_j) \Rightarrow (\mathcal{C}_i \sqsubseteq_{\mathcal{T}} \mathcal{C}_j)$.

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 $(\mathcal{C}_i \sqsubseteq_{\mathcal{T}} \mathcal{C}_j \sqsubseteq_{\mathcal{T}} \mathcal{C}_k)$. The basic properties of the new relation are as follows.

1. *reflexivity:*
   $\forall e \in Dim\ (e \sqsubseteq_1 e)$

2. *transitivity of full containment (1-to-1 transitivity):*
   $\forall (e_i, e_j, e_k) \in C_i \times C_j \times C_k\ (((e_i \sqsubseteq_1 e_j) \wedge (e_j \sqsubseteq_1 e_k)) \Rightarrow (e_i \sqsubseteq_1 e_k))$

When 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 are guaranteed to hold.

3. *transitivity of partial containment:* Assume that $(\mathcal{C}_i \sqsubseteq_{\mathcal{T}}^{(P)} \mathcal{C}_j \sqsubseteq_{\mathcal{T}}^{(P)} \mathcal{C}_k)$. Then the following holds. $\forall (e_i, e_j, e_k) \in C_i \times C_j \times C_k$ :

   (a) *p-to-1 transitivity:* $\forall p \in [0;1)((e_i \sqsubseteq_p e_j) \wedge (e_j \sqsubseteq_1 e_k) \Rightarrow (e_i \sqsubseteq_p e_k))$
   Obviously, $e_k$ may contain the part of $e_i$, which is not in $e_j$, but generally we know nothing about that. We infer what we can guarantee: what is contained in $e_j$ is *also* contained in $e_k$.
   For example, if 20% of *Street1* is in *District1*, but *District1* lies fully within *Province2*, then we say that 20% of the street *Street1* is in *Province2*. The result can be imprecise, but we acknowledge that some part of *Street1* lies in *Province2* and indicate the guaranteed percentage.

   (b) *1-to-p transitivity:* $\forall p \in [0;1)((e_i \sqsubseteq_1 e_j) \wedge (e_j \sqsubseteq_p e_k) \Rightarrow (e_i \sqsubseteq_0 e_k))$
   If $e_i$ is fully contained in $e_j$ and $e_j$ is partially contained in $e_k$, then we can only guarantee that $e_i$ *may* be contained in $e_k$.
   For example, if the coordinate $Coord1$ is in *Cell1* and 80% of *Cell1* is contained in *Province1*, then we say that the coordinate $Coord1$ *may* be in the province *Province1*.

   (c) *p-to-p transitivity:* $\forall (p_i, p_j) \in [0;1) \times [0;1)((e_i \sqsubseteq_{p_i} e_j) \wedge (e_j \sqsubseteq_{p_j} e_k) \Rightarrow (e_i \sqsubseteq_0 e_k))$
   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 guarantee that $e_i$ *may* be contained in $e_k$.
   For example, if 30% of *Street1* is in *District2*, but 90% of *District2* is in *City1*, then we say that *Street1 may* be in *City1*.

A dimension contains dimension values that describes entities of some domain. For example, we could have as values the coordinates $Coord1$, $Coord2$, the streets $Street1$, $Street2$, the cities $City1$, $City2$, the district $District1$, the province $Province1$, etc. The values each belong to precisely one category (e.g., $Street1$ belongs to the *Street* category), and they are related to other values via a dimension hierarchy. Specifically, one value is fully (partially) contained in another if the domain entities they represent are

related according to full (partial) containment. For example, $Coord2$ may be fully contained in the province $Province1$, and $Street2$ may be partially contained in $Province1$. We can build multiple hierarchies on the dimension values, with some values being of the finest granularity and thus not containing any other values (coordinates) and with one value containing everything ($\top$).

Notice that if there are no partial containment relationships in a domain then we could still use the extended model. In that case, we would just use notation $e_i \sqsubseteq_1 e_j$ and $e_i \not\sqsubseteq e_j$ (guarantee of full and no containment, respectively) and would never use notation $e_i \sqsubseteq_p e_j$ and $e_i \sqsubseteq_0 e_j$ (guarantee and possibility of partial containment, respectively).

**Example 2** Our LOCATION dimension is given by $D_{loc} = (C_{loc}, P_{loc})$, where $C_{loc} = \{Coordinate, Street, District, City, Province, Country, IPAddress, Cell, \top\}$ (one category for each node in Figure 3). If 20% of the street $Street1$ is in district $District1$, we could write $Street1 \sqsubseteq_{0.2} District1$. Next, we write $Coord2 \sqsubseteq_1 Province1$ for the coordinate $Coord2$, which lies in province $Province1$. As for the properties of the new relation, it is natural that, for example, $Province1$ is fully contained in itself ($Province1 \sqsubseteq_1 Province1$).

We term a relationship between dimension values $e_i \sqsubseteq_p e_j$ *direct* if it is given directly in the relation $P$ (without using transitivity); otherwise, the relationship is *indirect*. For example, the relationship $Street1 \sqsubseteq_{0.2} District1$ is direct, and if also $District1 \sqsubseteq Province2$ then $Street1 \sqsubseteq_{0.2} Province2$ (p-to-1 transitivity) is an indirect relationship.

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 $R \subseteq F \times Dim$. 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 aspect: we deal with *p-characterization* as a consequence of introducing partial containment. We say that a fact $f \in F$ is *0-characterized by dimension value* $e_k$, written $f \rightsquigarrow_0 e_k$, if $\exists e_i \in Dim\ (((f, e_i) \in R) \wedge (e_i \sqsubseteq_p e_k) \wedge (p < 1))$. In addition, we will refer to characterization as introduced in the prototypical model as *1-characterization*, written $f \rightsquigarrow_1 e_k$, if $\exists e_i \in Dim\ (((f, e_i) \in R) \wedge (e_i \sqsubseteq_1 e_k))$.

A fact-dimension relation links facts and corresponding dimension values, e.g., a request could be issued from $Street1$. 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 state that a fact is *known for sure* to be characterized by a dimension value, and we use 0-characterization to say that a fact *may* be characterized by a dimension value.

**Example 3** In our case, 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 $R_{loc}$. If request $A$ was issued from the street $Street1$, we write $(A, Street1) \in R_{loc}$ and $A \rightsquigarrow_0 District1$. If request $B$ was issued from the coordinate $Coord2$, $B \rightsquigarrow_1 Province1$ holds.

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.

**Example 4** In our case, we can define the multidimensional object $M_{case} = (\mathcal{S}_{case}, F_{request}, D_{case}, R_{case})$.

### 3.2 Model Properties

We proceed to define important model 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 $\forall e_j \in C_j\ (\exists (e_i, p) \in C_i \times [0;1]\ (e_i \sqsubseteq_p e_j))$; 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 5** The mapping from *Province* to *Country* is onto because each country is partitioned into provinces. However, the mapping from *IP Address* 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 $\mathcal{C}_i \sqsubseteq_{\mathcal{T}}^{(P)} \mathcal{C}_j \sqsubseteq_{\mathcal{T}}^{(P)} \mathcal{C}_k$, we say that the mapping from $C_j$ to $C_k$ is *covering with respect to* $C_i$ if $\forall (e_i, p) \in C_i \times [0;1]\ (\forall e_k \in C_k\ ((e_i \sqsubseteq_p e_k) \Rightarrow \exists (e_j, p_i, p_j) \in C_j \times [0;1] \times [0;1]\ ((e_i \sqsubseteq_{p_i} e_j) \wedge (e_j \sqsubseteq_{p_j} e_k))))$; otherwise, it is *non-covering*. 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 *non-covering*.

**Example 6** Consider the categories *Street*, *Province*, and *Country*. Each street going through some country also goes through a province. So, the mapping from *Province* to *Country* is covering with respect to *Street*. Consider the categories *Coordinate*, *Street*, and *District*. Some coordinates do not lie on any street, so we map them directly to districts. This means that the mapping from *Street* to *District* is non-covering with respect to *Coordinate*. Thus, the hierarchy of the dimension $D_{loc}$ is non-covering. 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 $\forall (e_i, p_{i_1}, p_{i_2}) \in C_i \times [0;1] \times [0;1]\ (\forall (e_{j_1}, e_{j_2}) \in C_j \times C_j\ ((e_i \sqsubseteq_{p_{i_1}} e_{j_1}) \wedge (e_i \sqsubseteq_{p_{i_2}} e_{j_2}) \Rightarrow ((e_{j_1} = e_{j_2}) \wedge (p_{i_1} = p_{i_2}))))$; otherwise, it is *non-strict*. If in a dimension all the mappings are strict, we say that the dimension hierarchy is *strict*; otherwise, it is *non-strict*.

**Example 7** The mapping from *IP Address* to *Province* is strict because an address uniquely identifies a province. But the mapping from *Cell* to *Province* is non-strict because a cell may be shared by provinces. Thus, the hierarchy of the dimension $D_{loc}$ is non-strict. 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 a mapping from $C_i$ to $C_j$ exists, i.e., $C_j \in Anc^{(P)}(C_i)$, that is non-strict, then $Anc^{(P)}(C_j) = \emptyset$; otherwise, it is *aggregation non-strict*.

**Example 8** Consider the categories *Cell* and *Province*. As the mapping from *Cell* to *Province* is non-strict and $Anc^{(P)}(Province) \neq \emptyset$, the hierarchy of the dimension $D_{loc}$ is aggregation non-strict. 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 *non-normalized*. We say that a multidimensional object is *normalized*, if all the 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 *non-normalized*.

**Example 9** 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 non-normalized because it is non-onto, non-covering, and aggregation non-strict. Therefore, the multidimensional object $M_{case}$ is non-normalized.

### 3.3 Meeting the Requirements

We now examine whether the requirements stated in Section 2.2 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 non-normalized hierarchies. Non-strict 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. Non-covering 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 is a basis for supporting imprecision in the data [13].

### 3.4 Building Dimension Schemas

By allowing partial containment relationships in dimensions, the model presented here generalizes existing multidimensional models and offers new means of modeling dimensions. In this section, we explore pertinent implications for the design of dimensions using the new, generalized model. The insights presented here should be taken into account in a complete methodology for dimensional database design.

Section 2.1 describes a prototypical usage scenario for a multidimensional database in the context of a location-based service. In particular, Figure 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 Figure 3.

The ER diagram in Figure 1 captures information about containment relationships among various location entity types. Transitive relationships among the types are not shown, and there are no explicit descriptions of hierarchies. When designing a dimension, we build explicit hierarchies based on this diagram that enable the capture of data and the relevant analyses of data. For example, as a reflection of relationship types st-P-di and di-P-ci in the ER diagram, our dimension hierarchy will have a category *Street* that is below a category *City*. 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. In our case, the ER diagram happens to have a corresponding entity type.

When building the LOCATION dimension, we obtain multiple hierarchies. The use of these is caused in part by the new 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 miniworld and ER diagram. For example, the groupings of the coordinates of service requests by mobile cells and by an administrative unit such as streets are exclusive. Therefore, the *Cell* category does not fit anywhere in the (main) hierarchy, $\mathit{Coordinate} < \mathit{Street} < \mathit{District} < \mathit{City} < \mathit{Province} < \mathit{Country} < \top$. It is instead part of separate hierarchies, e.g., $\mathit{Coordinate} < \mathit{Cell} < \mathit{Province} < \mathit{Country} < \top$, which skip the *Street* category. In general, building these kinds of hierarchies translates into inserting categories and corresponding relationships in the LOCATION dimension. The other cases where it is necessary to build an additional hierarchy are discussed next.

An additional relationship may be inserted to "mend" non-covering hierarchies. To illustrate, recall from Example 6 that it is possible for a coordinate to not lie on any street, while it does lie in some district. The consequence is that we cannot map all coordinates to their corresponding districts via the *Street* entity

type in Figure 1 or the *Street* category in Figure 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 $\top$.

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

First, in the real world, each street goes through a province that is part of some country, meaning that the relationship between the *Province* and the *Country* categories is covering with respect to the *Street* category. This means that we are able to relate streets to corresponding countries through values from the *Province* category—we do not depend on a direct relationship between the *Street* 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.

If high-precision partial containment relationships are important, we insert direct relationships. For example, had it been important that streets roll-up to countries as precisely as possible, we would add a direct relationship between *Street* and *Country*. This illustrates a trade-off: if one wants high fidelity, this comes at the cost of increased the size and complexity of a dimension. The higher precision we want, the more direct relationships are needed.

Another aspect of dimension design, created by the introduction of partial containment, 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 the relationships between categories are not always obvious, consider the relationship between the *District* and *City* categories. In the miniworld, 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 partial solution to the 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*. The solution is partial because it does not contend well with large cities and city districts that contain church districts.

Another possible solution 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., $City \sqsubseteq^{(P)}_{\mathcal{T}_{loc}} District$ and $District \sqsubseteq^{(P)}_{\mathcal{T}_{loc}} City$. With these relationships present, we are able 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., $City1 \sqsubseteq_{0.6} District1$ and $District1 \sqsubseteq_{0.2} City1$.

However, the anti-symmetry 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 anti-symmetry, in our case, by p-to-p transitivity it may be inferred that $District1 \sqsubseteq_0 District1$, which has a lower degree of containment than what we obtain automatically by the reflexivity of the order on dimension values, i.e., $District1 \sqsubseteq_1 District1$. In addition, in case of relationships between the same dimension value, the latter degree is logically and practically the most suitable.

# 4 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 $D_{M_1} = \{D_i^1, i = 1, \ldots, n\}$, $D_{M_2} = \{D_i^2, i = 1, \ldots, n\}$, $R_{M_1} = \{R_i^1, i = 1, \ldots, n\}$, and $R_{M_2} = \{R_i^2, i = 1, \ldots, n\}$. 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 10** In order to illustrate the workings of the operators, we construct two example MO's denoted $M^1_{case}$ and $M^2_{case}$ . In Figure 4(a), the schema of the LOCATION dimension of the MO's is depicted, which is a simplified version of that of $M_{case}$. In Figures 4(b) and 4(c), 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 Figures 4(b) and 4(c) represent the fact-dimension relationships in the LOCATION dimensions. Note that facts may map directly to dimension values in non-bottom categories. The TIME and USER dimensions of the MO's are identical to the corresponding dimensions of $M_{case}$.

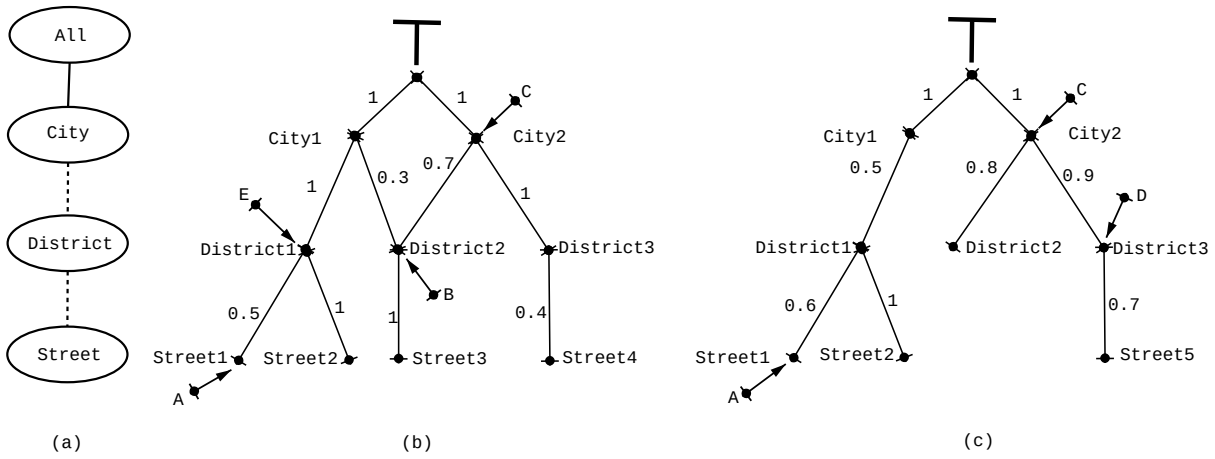


Figure 4: Schema (a) and LOCATION dimensions of $M^1_{case}$ (b) and $M^2_{case}$ (c)

## 4.1 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 [13]. Given a predicate $q : Dim_1 \times \ldots \times Dim_n \rightarrowtail \{true, false\}$, the *selection* operator for the prototypical model, $\sigma$, 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, $\sigma_{ext}$, 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 $F' = \{f \in F \mid \exists(e_1, \ldots, e_n) \in Dim_1 \times \ldots \times Dim_n\ (\exists(p_1, \ldots, p_n) \in [0;1] \times \ldots \times [0;1]\ ((q_{ext}(e_1, \ldots, e_n, p_1, \ldots, p_n)) \wedge (f \rightsquigarrow_{p_1} e_1) \wedge \ldots \wedge (f \rightsquigarrow_{p_n} e_n)))\}$.

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 11** Suppose we want to select requests from $M^1_{case}$ that were *surely* issued from $District2$. 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: $(e_{loc} = District2) \wedge ((e_{userage} \in [21;30]) \wedge (e_{time} = [07\backslash 14\backslash 2001]) \wedge (p_{loc} = 1) \wedge (p_{userage} = 1) \wedge (p_{time} = 1)$. Notice that $B \rightsquigarrow_1 District2$, so request $B$ will be in the result.

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

## 4.2 Union Operator

The union operator is used to take the union of two MOs. Consider two dimensions $D_1 = (C_{D_1}, \sqsubseteq_{D_1})$ and $D_2 = (C_{D_2}, \sqsubseteq_{D_2})$ 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: $D' = D_1 \bigcup^D D_2 = (C_{D'}, \sqsubseteq_{D'})$, where $C_{D'} = \{C^1_j \bigcup C^2_j, j = 1, \ldots, m\}$ and $\forall(e_1, e_2) \in (Dim_1 \cup Dim_2) \times (Dim_1 \cup Dim_2)\ ((e_1 \sqsubseteq_{D'} e_2) \Leftrightarrow ((e_1 \sqsubseteq_{D_1} e_2) \vee (e_1 \sqsubseteq_{D_2} e_2)))$. In what follows, we use notation $C'_j$ for $C^1_j \bigcup C^2_j$.

Consider two $n$-dimensional MO's with $\mathcal{S}_1 = \mathcal{S}_2$. The *union* operator on *MOs* 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 as follows.

1. $\forall(e_1, e_2) \in (Dim_1 \cup Dim_2) \times (Dim_1 \cup Dim_2)\ ((\exists p \in [0;1]((e_1, e_2, p) \in P_{D'}) \Leftrightarrow (\exists(p_1, p_2) \in [0;1] \times [0;1](((e_1, e_2, p_1) \in P_{D_1}) \vee ((e_1, e_2, p_2) \in P_{D_2}))))$;
2. $\forall(e_1, e_2, p) \in P_{D'}\ (((e_1, e_2) \in C'_i \times C'_j) \wedge (C'_j \in Anc^{(P)}(C'_i))) \Rightarrow (\exists(p_1, p_2) \in [0;1] \times [0;1]\ (((e_1, e_2, p_1) \in P_{D_1}) \vee ((e_1, e_2, p_2) \in P_{D_2})) \wedge (p = max(p_1, p_2))))$.

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.

In other words, given two MO's 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 transfered unchanged into the resulting dimension. However, if the values are directly related in both the 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.

**Example 13** The LOCATION dimension of an MO obtained by uniting $M^1_{case}$ and $M^2_{case}$ is depicted in Figure 5. Note that some degrees of containment for the indirect relationships, e.g., Street1 $\sqsubseteq_{0.3}$ City1, are not found in any of the original MOs, and can only be inferred using the transitivity rules.

### 4.3 Aggregate Formation Operator

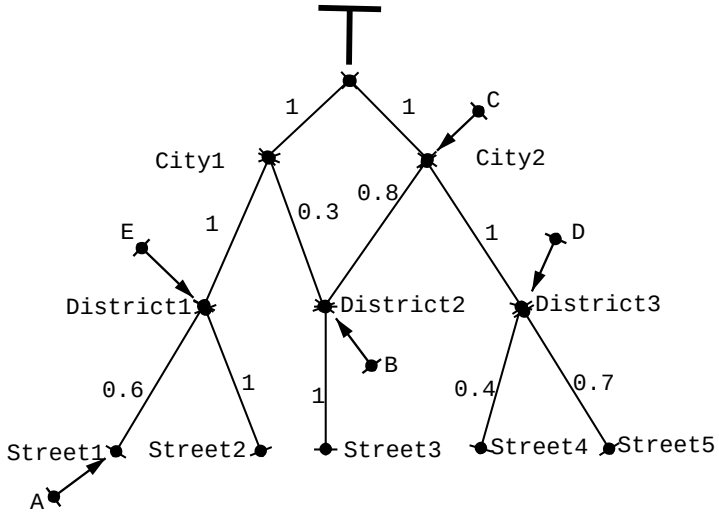


Figure 5: A union of $M^1_{case}$ and $M^2_{case}$

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 it.

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, $\alpha$, is defined as follows. $M' = \alpha[D_{n+1}, g, C_1, \ldots, C_n](M) = (\mathcal{S}', F', D'_{M'}, R'_{M'})$, where

$\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$, $\mathcal{F}' = 2^{\mathcal{F}}$, $\mathcal{D}' = \{\mathcal{T}'_i, i = 1, \ldots, n\} \bigcup \{\mathcal{T}_{n+1}\}$,
$\mathcal{T}'_i = (\mathcal{C}'_i, \sqsubseteq'_{\mathcal{T}_i}, \bot'_{\mathcal{T}_i}, \top'_{\mathcal{T}_i})$, $\mathcal{C}'_i = \{\mathcal{C}_{ij} \in \mathcal{T}_i \mid Type(C_i) \sqsubseteq_{\mathcal{T}_i} \mathcal{C}_{ij}\}$, $\sqsubseteq'_{\mathcal{T}_i} = \sqsubseteq_{\mathcal{T}_i|_{\mathcal{C}'_i}}$, $\bot'_{\mathcal{T}_i} = Type(C_i)$, $\top'_{\mathcal{T}_i} = \top_{\mathcal{T}_i}$,
$F' = \{Group(e_1, \ldots, e_n) \mid ((e_1, \ldots, e_n) \in C_1 \times \ldots \times C_n) \wedge \wedge (Group(e_1, \ldots, e_n) \neq \emptyset)\}$,
$D' = \{D'_i, i = 1, \ldots, n\} \bigcup \{D_{n+1}\}$, $D'_i = (C'_{D'_i}, \sqsubseteq'_{D'_i})$,
$C'_{D'_i} = \{C'_{ij} \in D_i \mid Type(C'_{ij}) \in \mathcal{C}'_i\}$, $\sqsubseteq'_{D'_i} = \sqsubseteq_{D_i|_{D'_i}}$,
$R'_{M'} = \{R'_i, i = 1, \ldots, n\} \bigcup \{R'_{n+1}\}$,
$R'_i = \{(f', e'_i) \mid \exists (e_1, \ldots, e_n) \in C_1 \times \ldots \times C_n ((f' = Group(e_1, \ldots, e_n)) \wedge (f' \in F') \wedge (e_i = e'_i)\}$, and
$R'_{n+1} = \bigcup_{(e_1, \ldots, e_n) \in C_1 \times \ldots \times C_n} \{(Group(e_1, \ldots, e_n), g(Group(e_1, \ldots, e_n))) \mid Group(e_1, \ldots, e_n) \neq \emptyset\}$.

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 that are 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 14** Consider the MO $M^1_{case}$ in Figure 4(b). 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: $\alpha[RESULT, COUNT, District, \top, \top](M^1_{case})$.

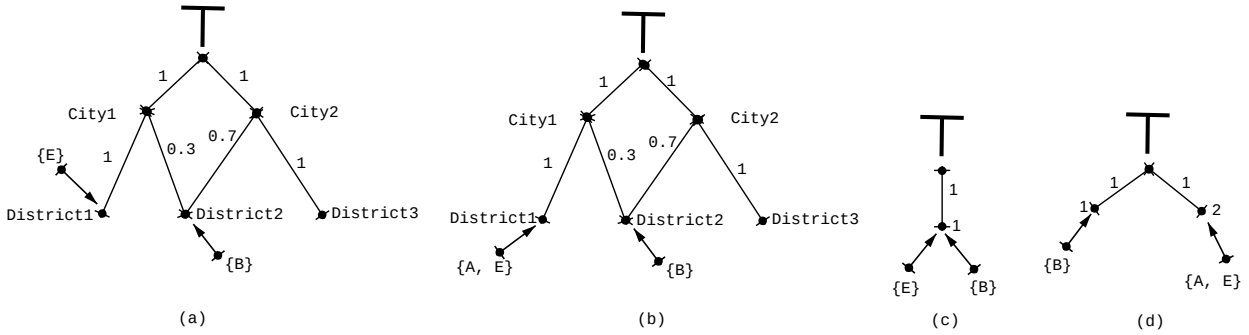


Figure 6: LOCATION and RESULT dimensions after the query $\alpha$, conservative (a, c) and liberal (b, d) groupings

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 into a group that are *known for sure* to belong to that group. We define the corresponding operator, $Group_c$, as follows: $Group_c(e_1, \ldots, e_n) = \{f \mid (f \in F) \wedge (f \rightsquigarrow_1 e_1) \wedge \ldots \wedge (f \rightsquigarrow_1 e_n)\}$. Since only precise data will be used in calculations and the remaining data will be 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 15** Assume the aggregation query from Example 14. 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 $District1$. This means that the conservative grouping of the requests by districts would yield the fact groups $\{E\}$ and $\{B\}$ mapped to the values $District1$ and $District2$ in the LOCATION dimension respectively as depicted in Figure 6(a). In this case, the count for both groups of requests would be 1, which can be seen from the result dimension in Figure 6(c).

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: $Group_l(e_1, \ldots, e_n) = \{f \mid (f \in F) \wedge (f \rightsquigarrow_{p_1} e_1) \wedge \ldots \wedge (f \rightsquigarrow_{p_n} e_n) \wedge (\forall i \in \{1, \ldots, n\}((p_i = 1) \vee (p_i = 0)))\}$. 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, is taken into consideration.

**Example 16** Assume the aggregation query from Example 14. The liberal grouping of the requests by districts would include the request $A$ into a group, i.e., we would get the fact groups $\{A, E\}$ and $\{B\}$ mapped to the values *District1* and *District2* as depicted in Figure 6(b). In this case, the count for the groups would be 2 and 1 respectively, which is shown in Figure 6(d).

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., $Group_w = Group_l$. We determine the weight of membership for a fact with the help of a function $Weight : Dim_1 \times \ldots \times Dim_n \times F \rightarrowtail \mathbb{R}$ by combining the fact's degrees of containment.

For example, the simplest way to combine the degrees is to take their product. Namely, if $f \in Group_w(e_1, \ldots, e_n)$ and $f \rightsquigarrow_{p_1} e_1, \ldots, f \rightsquigarrow_{p_n} e_n$, then $Weight_{simp}(e_1, \ldots, e_n, f) = \prod_{i=1}^{n} p_i$.

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 is imprecise with respect to just one dimension, i.e., $\forall f \in Group_w(e_1, \ldots, e_n)((\exists e_i \in \{e_1, \ldots, e_n\}(f \rightsquigarrow_0 e_i)) \Rightarrow (Weight_{simp}(e_1, \ldots, e_n, f) = 0))$.

**Example 17** Assume the aggregation query from Example 14 and the corresponding liberal grouping of the facts from Example 16 . The function $Weight_{simp}$ would apply weights of membership to our facts as follows: $Weight_{simp}(District1, \top, \top, A) = 0 \times 1 \times 1 = 0$, $Weight_{simp}(District1, \top, \top, E) = 1 \times 1 \times 1 = 1$, $Weight_{simp}(District2, \top, \top, B) = 1 \times 1 \times 1 = 1$. Thus, the weighted grouping will, unlike the liberal grouping, show that $E$ is *certain* to be in *District1* while $A$ only *may* be in *District1*. 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 *District1* of $1.5$.

# 5 Imprecision In The Aggregation Path

With partial containment in the model and transitivity of partial containment defined, we face the problem of how to choose the 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 $C_j \in Anc^{(P)}(C_i)$. In our case, we could define the aggregation path $\alpha = \{Street, District, City, Province\}$.

Two aggregation paths $\alpha_i$ and $\alpha_j$ 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, $\alpha$ and $\alpha_{alt} = \{Street, Province\}$ are alternative aggregation paths. An aggregation path is *direct* if it consists of just two elements. Thus, $\alpha_{alt}$ is a direct aggregation path.

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

Given an aggregation path $\beta$, $e_i \in C_i$, and $e_j \in C_j$, we say that a direct relationship $e_i \sqsubseteq_p e_j$ is *on the path* $\beta$ if $C_j$ is successor of $C_i$ in $\beta$ (in our case, *Street1* $\sqsubseteq_{0.2}$ *District1* is a direct relationship on path $\alpha$).

Consider an aggregation path $\gamma = \{C_{first}, \ldots, C_{last}\}$ and its alternative direct aggregation path $\gamma_{alt}$. We build two sets $M$ and $M_{alt}$ for $\gamma$ and $\gamma_{alt}$, respectively. Both sets contain relationships $e_{first} \sqsubseteq_p e_{last}$, 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 on the path $\gamma$. Set $M_{alt}$ contains relationships on the path $\gamma_{alt}$.

We proceed to compare the aggregation paths, for which purpose, we introduce the concept of *imprecision level* of an aggregation path $\gamma$ (denoted $IL_\gamma$). The imprecision level $IL_\gamma$ is evaluated by the following

algorithm:

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

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.

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 with 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 non-covering mappings into consideration. Specifically, it ignores direct relationships that exists to accommodate non-covering hierarchies, e.g., relationships between dimension values in the *Coordinate* and *District* categories. If this is the 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 have to use the ignored relationships. However, it is not reasonable to use as an alternative direct aggregation path the one that is included to accommodate non-covering 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 pre-aggregation applied to the warehouse. Without pre-aggregation, there is no difference in speed between aggregation along a path and the alternative direct path. However, with pre-aggregation, we can speed up aggregation along a given path considerably by pre-computing some steps in it. Thus, having evaluated the imprecision level and aggregation speed (possibly with different pre-aggregation 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 $\epsilon = \{Coordinate, Street, District, City, Province, Country, \top\}$ because the path $\epsilon_{alt} = \{Coordinate, \top\}$ is absent. Instead, we could split $\epsilon$ into two subpaths, $\epsilon^1 = \{Coordinate, Street, District, City, Province\}$ and $\epsilon^2 = \{Province, Country, \top\}$. Then $\epsilon^1$ could be compared to $\epsilon^1_{alt}$ (supposing every coordinate is directly related to a province) to get the imprecision level of the subpath.

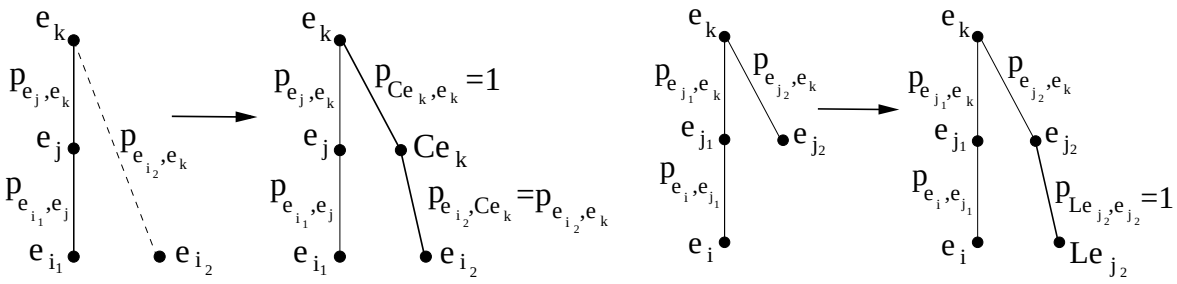


Figure 7: Transformations by the PMakeCovering and PMakeOnto Algorithms

# 6 Hierarchy Transformations

Using the data model described in Section 3, we are able to specify non-normalized dimension hierarchies. However, such hierarchies pose problems to practical pre-aggregation, as will be explained shortly. In this section, we briefly present algorithms that normalize hierarchies (the algorithms are described in details elsewhere [10, 12]) and extend the algorithms to accommodate partial containment. Due to space constraints the details are deferred to Appendix A.

If a hierarchy is non-covering, then some links between dimension values skip one or more levels So, at a "skipped" level some data is 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 street, but lies in district *District1*. The mapping from category *Street* to category *District* is then non-covering with respect to the category *Coordinate*, and the level of *Street* is "skipped." Thus, we cannot use aggregates at the *Street* 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 *IP Address* to category *City* is non-onto and *City1*, the dimension value of *City*, 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 non-strict, some dimension values have multiple parents. So, moving from a child level to a parent level, we may count the same data several times. This means that we are not able to use aggregation results at the child level to compute aggregates at the parent level. For example, suppose a street *Street1* crosses districts *District1* and *District2*. So, *Street1* has two parent dimension values, *District1* and *District2*, in the category *District*. Moving from the level of *Street* to the level of *District*, we count aggregates for *Street1* twice. Thus, we cannot use aggregates at the level of *Street* to calculate aggregate results at the level of *District*.

A suite of hierarchy transformation algorithms remove these problems and thus enable correct use of pre-aggregation for non-normalized 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 18** The workings of the algorithms is best illustrated by an example. For the PMakeCovering algorithm, consider the transformation to the left in Figure 7. Here, the direct link between $e_k$ (using the example of a non-covering 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 to the right in Figure 7. 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 Figure 8. Here, the multiple parents of value $e_i$ (corresponding to *Street1* in our example introducing a non-strict hierarchy) are fused into one new value $e = \{e_{j_1}, ..., e_{j_m}\}$ (in our example, $m = 2$ and the fused value is $\{$*District1, District2*$\}$), which is then linked to $e_i$ and the parents $e_{j_1}, ..., e_{j_m}$ (i.e., *District1* and *District2*). To keep the figure simple, we have not shown additional values in the $e_i$ category, nor the grandparents of $e_i$ which $e$ is also linked to. The degree of containment is computed to provide the highest possible degree that can still be guaranteed, see Appendix A for details.

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

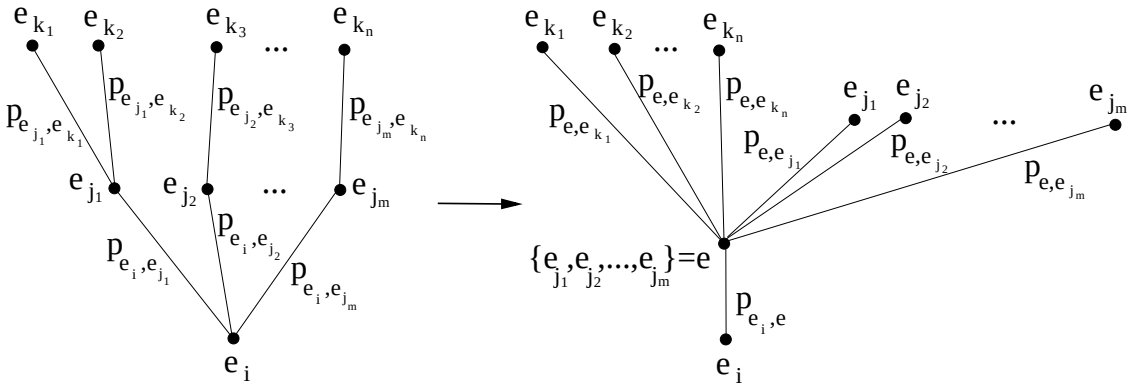


Figure 8: Transformations by the PMakeStrict Algorithm

# 7 Conclusions and Research Directions

Mobile e-services promises to become a significant application domain for multidimensional modeling of spatial data. 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 satisfies other important requirements (e.g., non-normalized dimension hierarchies) to support for 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 $((e_i \sqsubseteq_{p_i} e_j) \wedge (e_j \sqsubseteq_{p_j} e_k)) \Rightarrow (e_i \sqsubseteq_p e_k)$, then what is the value of $p$? We have provided a "safe" definition of partial containment, meaning that we deduce only those relationships that we can guarantee. But at the same time we try to maximize degrees of inferred relationships. We feel that this is the most basic and useful approach.

The algebra for the prototypical model, with extended selection, union, and aggregate formation operators, serves well as a formal basis for an end-user query language for our model. We have extended several elements of the algebraic operators to reflect support for partial containment in the model. 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 choices of which path to choose among several alternatives, the paper provides a means of evaluating the imprecisions of paths. This enables the choice of paths based on their precision and their associated speed of aggregation.

Enabling practical pre-aggregation requires that dimension hierarchies be onto, covering, and aggregation strict. We have extended existing hierarchy transformations to support partial containment. This extension was based on two requirements to hierarchy transformations. The first requires that full containment relationships between dimension values be preserved. The second requires that the degrees of containment in partial containment relationships should be as close to the original degrees as possible, but not higher (safe approach). Extending the existing transformations for making hierarchies onto and covering is straightforward, and we have retained all old degrees. Incorporating support for partial containment into the transformation that makes hierarchies aggregation strict is a more complex task. We have succeeded in preserving full containment at the expense of degrees for partial containment relationships (some degrees reduce to "zero").

It would be of interest to further study several aspects of the proposed data model. 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 [14] 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 [13] 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 [17]. 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.

## Acknowledgments

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.

## References


[1] Blaha M (ed.) (2001) Special Section: Data Warehouses. IEEE Computer Magazine 34(12):38–79

[2] Dyreson CE (1996) Information Retrieval from an Incomplete Data Cube. In Proceedings of the Twenty-second International Conference on Very Large Databases, pp. 532–543

[3] 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 Academic Publishers, pp. 413–458

[4] Ferri F, Pourabbas E, Rafanelli M, Ricci FL (2000) Extending Geographic Databases for a Query Language to Support Queries Involving Statistical Data. In Proceedings of the Twelfth International Conference on Scientific and Statistical Databases, pp. 220–230

[5] Jensen CS (2002) Research Challenges in Location-Enabled M-Services. In Proceedings of the Third International Conference on Mobile Data Management, pp. 3–7

[6] Jensen CS, Kligys A, Pedersen TB, Timko I (2002) Multidimensional Data Modeling for Location-Based Services. In Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, to appear

[7] Jensen CS, Pedersen TB (2001) Mobile E-Services and Their Challenges to Data Warehousing. Datenbank Rundbrief 27:8–16

[8] Kimball R, Reeves L, Ross M, Thornthwaite W (1998) The Data Warehouse Lifecycle Toolkit, Wiley

[9] Lenz H, Shoshani A (1997) Summarizability in OLAP and Statistical Databases. In Proceedings of the Ninth International Conference on Scientific and Statistical Databases, pp. 39–48

[10] Pedersen TB, Jensen CS, Dyreson CE (1999) Extending Practical Pre-Aggregation in On-Line Analytical Processing. In Proceedings of the Twentyfifth International Conference on Very Large Databases, pp. 663–674

[11] Pedersen TB (2000) Aspects of Data Modeling and Query Processing for Complex Multidimensional Data. Ph.D. Thesis, Aalborg University

[12] Pedersen TB, Jensen CS, Dyreson CE (2000) The TreeScape System: Reuse of Pre-Computed Aggregates over Irregular OLAP Hierarchies. In Proceedings of the Twentysixth International Conference on Very Large Databases, pp. 595–598

[13] Pedersen TB, Jensen CS, Dyreson CE (2001) A Foundation for Capturing and Querying Complex Multidimensional Data. In Information Systems 26(5):383–423, Special Issue: Data Warehousing

[14] Pedersen TB, Tryfona N (2001) Pre-Aggregation In Spatial Data Warehouses. In Proceedings of the Seventh International Symposium on Advances in Spatial and Temporal Databases, pp. 460–478

[15] Rafanelli M, Shoshani A (1990) STORM: A Statistical Object Representation Model. In Proceedings of the Fifth Conference on Statistical and Scientific Database Management, pp. 14–29

[16] Rafanelli M (ed.) (2002) Multidimensional databases: Problems and Solutions. Idea Group Publishing

[17] Thomsen E, Spofford G, Chase D (1999) Microsoft OLAP Solutions. Wiley Computer Publishing

[18] Vassiliadis P, Sellis TK (1999) A Survey of Logical Models for OLAP Databases. SIGMOD Record 28(4):64–69

# A Hierarchy Transformation Algorithms

We now describe the hierarchy transformation approach in detail.

## A.1 Pre-Aggregation and Non-Normalized Hierarchies

To *pre-aggregate* 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, results of *full pre-aggregation*, when all combinations of aggregates are pre-computed, may require too much storage space, thus making full pre-aggregation impractical. Instead, in modern data warehouse systems *practical pre-aggregation* is used, which means that only select combinations of aggregates are stored and reused later for computing other aggregates.

Furthermore, it is important that we could pre-aggregate values at any combination of dimension levels and reuse the pre-aggregated values to compute higher level aggregate results. If this requirement is met, we say that our data is *summarizable*. In [14], it is mentioned that data captured by the model that we present in Section 3 (though without our extension for handling partial containment) is summarizable, if a multidimensional object for the data is normalized. So, the problems with practical pre-aggregation occur, if in a multidimensional object facts map to dimension values at different levels or dimension hierarchies are either non-onto, non-covering, or non-strict.

Generally, pre-aggregation procedure performed at the lowest level does not take into consideration fact-dimension relationships at higher levels. So, if all the facts do not map to dimension values at a lowest level, then some data is 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 a street $Street1$. So, the corresponding fact is mapped to the value $Street1$, which belongs to the category *Street*—not the lowest one. Data for the request at the lowest level is missing. Thus, we are not able to use aggregates at the level of lowest category *Coordinate* to calculate aggregate results at the level of *Street*.

With the extension for capturing partial containment introduced into the model, generally data in a normalized multidimensional object may become non-summarizable. The proposed transitivity property infers only guaranteed relationships between dimension levels, which means that in a real-world situation we would miss some data. Anyway, normalization of dimension hierarchies is a first step towards achieving summarizability.

## A.2 Hierarchy Transformation Algorithms

We present the pseudocoded version of 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 $C_j$ ($C_j \in Anc^{(P)}(C_i)$). A column $X$ in a table contains degrees of containment. In Figures 7 and 8, we illustrate the effect 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 get into details on how mappings are transformed (this is done in [10] and [11]). Instead, we explain the extension and how new hierarchies meet the formulated requirements.

## A.3 Making Hierarchies Covering

To the left in Figure 7, 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 $C_j \in Anc^{(P)}(C_i)$. Finally, we assume a degree of containment $p \in [0; 1]$.

Before the transformation we have: $e_{i_2} \sqsubseteq_{p_{e_{i_2},e_k}} e_k$ We may retain the same relationship between the values after the transformation, if for the new degrees of partial containment we let $p_{e_{i_2},Ce_k} = p_{e_{i_2},e_k}$ and $p_{Ce_k,e_k} = 1$. By using 1-to-1 or p-to-1 transitivity (depending on the values of degrees), we get: $((e_{i_2} \sqsubseteq_{p_{e_{i_2},Ce_k}} Ce_k) \wedge (Ce_k \sqsubseteq_{p_{Ce_k,e_k}} e_k)) \Rightarrow (e_{i_2} \sqsubseteq_{p_{e_{i_2},e_k}} e_k)$. So, the relationship $e_{i_2} \sqsubseteq_{p_{e_{i_2},e_k}} e_k$ is retained by the algorithm. Obviously, the first requirements is met: if before the transformation $p_{e_{i_2},e_k} = 1$, so is after the transformation. The second requirement is also met: the degree $p_{e_{i_2},e_k}$ 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. Namely, we state that $Type(C_i) \sqsubseteq^P_{\mathcal{T}} Type(C_j)$, if $Type(C_i) \sqsubseteq^P_{\mathcal{T}} Type(C_k)$, and $Type(C_i) \sqsubseteq_{\mathcal{T}} Type(C_j)$, if $Type(C_i) \sqsubseteq_{\mathcal{T}} Type(C_k)$. By doing so, we allow arbitrary value of $p_{e_{i_2},Ce_k}$. In addition, we must add the relationship $Type(C_j) \sqsubseteq_{\mathcal{T}} Type(C_k)$, if it was not present before the transformation, to allow $p_{Ce_k,e_k} = 1$.

(1) **procedure** PMakeCovering($C$)
(2) **for each** $P \in Anc^{(P)}(C)$ **do**
(3) **begin**
(4) **for each** $H \in Anc^{(P)}(C)$ **where** $Type(P) \sqsubseteq^{(P)}_{\mathcal{T}} Type(H) \wedge Type(P) \neq Type(H)$ **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) $R_{C,P,X} \leftarrow R_{C,P,X} \cup \{(c, Mark(h), p_{c,h}) \mid (c,h) \in L \wedge (c,h,p_{c,h}) \in R_{C,H,X}\}$
(11) **end**
(12) PMakeCovering($P$)
(13) **end**

## A.4 Making Hierarchies Onto

To the right in Figure 7, 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 $C_j \in Anc^(P)(C_i)$. Finally, we assume that a degree of containment $p \in [0;1]$.

The transformation does not require altering the present degrees of containment. We let the completely new degree $p_{Le_{j_2},e_{j_2}} = 1$ (it conforms with the logic: a "placeholder" $Le_{j_2}$ for a value $e_{j_2}$ is fully contained in that value). Obviously, 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., $Type(C_i) \sqsubseteq_{\mathcal{T}} Type(C_j)$.

(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 Figure 8, we assume that $e_i \in C_i$, $\forall l \in \{1,\ldots,m\}(e_{j_l} \in C_j)$, $\forall l \in \{1,\ldots,n\}(e_{k_l} \in C_k)$, and $e = \{e_{j_1}, e_{j_2}, \ldots, e_{j_m}\} \in C_j^f$ (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 $C_j^f \in Anc^{(P)}(C_i)$. Finally, we assume a degree of containment $p \in [0;1]$.

Before the transformation we have the set of relationships $L = \{e_i \sqsubseteq_{p_{e_i,e_{j_l}}} e_{j_l}, l = 1,\ldots,m\}$. 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 property ($\forall l \in \{1,\ldots,m\}(((e_i \sqsubseteq_{p_{e_i,e}} e) \wedge (e \sqsubseteq_{p_{e,e_{j_l}}} e_{j_l})) \Rightarrow (e_i \sqsubseteq_{p_{e_i,e_{j_l}}} e_{j_l}))$). The common initial condition for each relationship in the set $L$ is $e_i \sqsubseteq_{p_{e_i,e}} e$. In general, presence of the common condition does not allow us to retain all the old degrees of containment in the set. We deal with the situation in a following way: if the set $L$ has any full containment relationships, we retain them (using 1-to-1 transitivity, the first step to this is to let $p_{e_i,e} = 1$) and otherwise we perform approximation, which conforms to the "safe" approach, and retain the minimal degree in the set (using p-to-1 transitivity, the first step to this is to let $p_{e_i,e} = min(\{p_{e_i,e_{j_l}}, l = 1,\ldots,m\})$. After having assigned a value to $p_{e_i,e}$, we perform further assignment. If we retain the full containment, then for completion we let $\forall l \in \{1,\ldots,m\}(((p_{e_i,e_{j_l}} = 1) \Rightarrow (p_{e,e_{j_l}} = 1)) \wedge ((p_{e_i,e_{j_l}} \neq 1) \Rightarrow (p_{e,e_{j_l}} = 0)))$. But if we retain the minimal degree in the set, then for completion we must let $\forall l \in \{1,\ldots,m\}(p_{e,e_{j_l}} = 1)$, and by p-to-1 transitivity we get the required result: $\forall l \in \{1,\ldots,m\}(e_i \sqsubseteq_{min(\{p_{e_i,e_{j_l}}, l=1,\ldots,m\})} e_{j_l})$. Notice that in the former case for some relationships by 1-to-1 transitivity we get $e_i \sqsubseteq_1 e_{j_l}$, but for some relationships we cannot avoid using 1-to-p transitivity and we only get $e_i \sqsubseteq_0 e_{j_l}$. This means that we preserve the full containment indeed, but miss the guaranteed degrees that were less than one.

Another set of relationships to deal with is $N = \{e_i \sqsubseteq_{p_{e_i,e_{k_l}}} e_{k_l}, l = 1,\ldots,n\}$ (the set is defined for a hierarchy before the transformation). We do it taking into consideration the fact that the actions for the set $L$ have already been performed. Again, we would like to have these relationships between the values with the same degrees after the transformation. And again in the transformed hierarchy the relationships can be inferred only by using transitivity property ($\forall l \in \{1,\ldots,n\}(((e_i \sqsubseteq_{p_{e_i,e}} e) \wedge (e \sqsubseteq_{p_{e,e_{k_l}}} e_{k_l})) \Rightarrow (e_i \sqsubseteq_{p_{e_i,e_{k_l}}} e_{k_l}))$). Notice that the common initial condition for each relationship in the set is already given after we have dealt with the set $L$. This, of course, limits our capabilities. But anyway we deal with the situation according to the logic applied in the case of the set $L$: if the set $N$ has any full containment relationships, we retain them and otherwise we perform approximation which conforms with the "safe" approach. We can preserve full containment for sure, because if the set $N$ has any full containment relationships, the degree in the first condition $p_{e_i,e}$ is equal to one. If this is the case, we split the set $N$ into $N_1$ and $N_p$ ($N = N_1 \bigcup N_p$ and $N_1 \bigcap N_p = \emptyset$). Elements of the set $N_1$ are full containment relationships, while elements of the set $N_p$ are partial containment relationships. We let $\forall l \in \{1,\ldots,n\}(((e_{k_l} \in N_1) \Rightarrow (p_{e,e_{k_l}} = 1)) \wedge ((e_{k_l} \in N_p) \Rightarrow (p_{e,e_{k_l}} = 0)))$. By doing so, we really retain full containment where it is present, but miss the guaranteed degrees that were less than one (analogy with the case of the set $L$). If $p_{e_i,e}$ is not equal to one, the set $N$ does not have any full containment relationships. It means that we must perform approximation. We let $\forall l \in \{1,\ldots,n\}((\exists e_{j_{l'}}(e_{j_{l'}} \sqsubseteq p_{e_{j_{l'}},e_{k_l}} e_{k_l})) \wedge ((p_{e_{j_{l'}},e_{k_l}} = 1) \Rightarrow (p_{e,e_{k_l}} = 1)) \wedge (p_{e_{j_{l'}},e_{k_l}} \neq 1) \Rightarrow (p_{e,e_{k_l}} = 0))$. So, for some relationships in the set $N$ by p-to-p transitivity we get $e_i \sqsubseteq_0 e_{k_l}$ (this is true before the transformation as well) and for some relationships in $L$ by p-to-1 transitivity we get $e_i \sqsubseteq_{p_{e_i,e_{k_l}}} e_{k_l}$ (the degree is less than that before the transformation, but still above zero).

We see that the first and the second requirements to a procedure of recording new degrees of containment are met.

The drawback of the algorithm is that in some cases it substitutes zero degrees for non-zero ones. For

example, if $n = l = 2$, then there are four degrees to work with and sixteen variants for the set of degree values (each degree is allowed to be one or non-one). Therefore, there are sixty-four relationships. Twelve of them turn their degree from non-zero to zero.

In a transformed dimension, we modify relationships between category types as needed. If we need partial containment for the algorithm to work as described, we allow partial containment between dimension value of a corresponding categories. Otherwise, we only allow full containment. Thus, 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. $(\mathcal{C}_i \sqsubseteq_\mathcal{T} \mathcal{C}_j \sqsubseteq_\mathcal{T} \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubseteq_\mathcal{T} \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubseteq_\mathcal{T} \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubseteq_\mathcal{T} \mathcal{C}_k))$
2. $(\mathcal{C}_i \sqsubseteq_\mathcal{T}^P \mathcal{C}_j \sqsubseteq_\mathcal{T} \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubseteq_\mathcal{T}^P \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubseteq_\mathcal{T}^P \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubseteq_\mathcal{T} \mathcal{C}_k))$
3. $(\mathcal{C}_i \sqsubseteq_\mathcal{T} \mathcal{C}_j \sqsubseteq_\mathcal{T}^P \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubseteq_\mathcal{T} \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubseteq_\mathcal{T} \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubseteq_\mathcal{T}^P \mathcal{C}_k))$
4. $(\mathcal{C}_i \sqsubseteq_\mathcal{T}^P \mathcal{C}_j \sqsubseteq_\mathcal{T}^P \mathcal{C}_k) \Rightarrow ((\mathcal{C}_i \sqsubseteq_\mathcal{T}^P \mathcal{C}_j^f) \wedge (\mathcal{C}_j^f \sqsubseteq_\mathcal{T}^P \mathcal{C}_j) \wedge \wedge(\mathcal{C}_j^f \sqsubseteq_\mathcal{T}^P \mathcal{C}_k))$

Notice that the pseudocode contains a new (compared to the variant in [10] and [11]) function $onPath : [0;1] \rightarrowtail [0;1]$. The function works as follows: if $((e_i \sqsubseteq_{p_i} e_j) \wedge (e_j \sqsubseteq_{p_j} e_k)$, then $onPath(p_j) = p_i$.

(1) **procedure** PMakeStrict($C$)
(2) **for each** $P \in Anc^{(P)}(C)$ **do**
(3) **begin**
(4) **if** $(\exists e_1 \in C(\exists e_2, e_3 \in P(\exists p_{1,2}, p_{1,3}(e_1 \sqsubseteq_{p_{1,2}} e_2 \wedge e_1 \sqsubseteq_{p_{1,3}} e_3 \wedge e_2 \neq e_3)))) \ \wedge \ Anc^{(P)}(P) \neq \emptyset$
(5) **then begin**
(6) $N \leftarrow CreateCategory(2^P)$
(7) **if** $\exists p_{1,2}\ (p_{1,2} = 1) \ \wedge \ (e_1, e_2, p_{1,2}) \in R_{C,P,X}$
(8) **then** $R_{C,N,X} \leftarrow \{(e_1, Fuse(\{e_2 |\ (e_1, e_2, p_{1,2}) \in R_{C,P,X}\}), 1)\}$
(9) **else** $R_{C,N,X} \leftarrow \{(e_1, Fuse(\{e_2 \mid (e_1, e_2, p_{1,2}) \in R_{C,P,X}\}), min(\{p_{1,2} \mid (e_1, e_2, p_{1,2}) \in R_{C,P,X}\}))\}$
(10) $N \leftarrow \Pi_N(R_{C,N,X})$
(11) $R_{N,P,X} \leftarrow \{(e_0, e_2, \emptyset) \mid e_0 \in N \wedge e_2 \in Unfuse(e_0)\}$
(12) **for each** $e_2 \in Unfuse(e_0)$ where $e_0 \in N$ **do**
(13) **begin**
(14) **if** $p_{1,0} = 1 \ \wedge \ p_{1,2} \neq 1 \ \wedge \ (e_1, e_0, p_{1,0}) \in R_{C,N,X} \wedge (e_1, e_2, p_{1,2}) \in R_{C,P,X}$
(15) **then** $R_{N,P,X} \leftarrow R_{N,P,X} \cup \{(e_0, e_2, 0)\} \ \setminus \ \{(e_0, e_2, \emptyset)\}$
(16) **else** $R_{N,P,X} \leftarrow R_{N,P,X} \cup \{(e_0, e_2, 1)\} \ \setminus \ \{(e_0, e_2, \emptyset)\}$
(17) **end**
(18) $Anc^{(P)}(C) \leftarrow Anc^{(P)}(C) \cup \{N\} \setminus \{P\}$
(19) $Anc^{(P)}(N) \leftarrow \{P\}$
(20) **for each** $G \in Anc^{(P)}(P)$ **do**
(21) **begin**
(22) $L \leftarrow \Pi_G(\Pi_{N,P}(R_{N,P,X}) \bowtie \Pi_{P,G}(R_{P,G,X}))$
(23) $R_{N,G,X} \leftarrow \{(e_0, e_3, \emptyset) \mid e_0 \in N \wedge \ e_3 \in L\}$
(24) **for each** $e_3 \in L$ **do**
(25) **begin**
(26) **if** $p_{1,0} = 1 \ \wedge \ (e_1, e_0, p_{1,0}) \in R_{C,N,X} \ \wedge \ e_0 \in N$
(27) **then**

(28) **if** $p_{2,3} = 1 \wedge p_{1,2} = 1 \wedge p_{1,2} \in onPath(p_{2,3}) \wedge e_2 \in P$
(29) **then** $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 1)\} \setminus \{(e_0, e_3, \emptyset)\}$
(30) **else** $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 0)\} \setminus \{(e_0, e_3, \emptyset)\}$
(31) **else if** $p_{2,3} = 1 \wedge e_2 \in P$
(32) **then** $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 1)\} \setminus \{(e_0, e_3, \emptyset)\}$
(33) **else** $R_{N,G,X} \leftarrow R_{N,G,X} \cup \{(e_0, e_3, 0)\} \setminus \{(e_0, e_3, \emptyset)\}$
(34) **end**
(35) $Anc^{(P)}(N) \leftarrow Anc^{(P)}(N) \cup \{G\}$
(36) $Anc^{(P)}(P) \leftarrow Anc^{(P)}(P) \setminus \{G\}$
(37) **end**
(38) PMakeStrict($N$)
(39) **end**
(40) **else** PMakeStrict($P$)
(41) **end**