Extending the Multidimensional Data Model to Handle Complex Data

Svetlana Mansmann
Databases and Information Systems Group at the University of Konstanz, Germany
svetlana.mansmann@uni-konstanz.de

Marc H. Scholl
Computer Science at the University of Konstanz, Germany
Marc.Scholl@uni-konstanz.de

Data Warehousing and OLAP (On-Line Analytical Processing) have turned into the key technology for comprehensive data analysis. Originally developed for the needs of decision support in business, data warehouses have proven to be an adequate solution for a variety of non-business applications and domains, such as government, research, and medicine. Analytical power of the OLAP technology comes from its underlying multidimensional data model, which allows users to see data from different perspectives. However, this model displays a number of deficiencies when applied to non-conventional scenarios and analysis tasks.

This paper presents an attempt to systematically summarize various extensions of the original multidimensional data model that have been proposed by researchers and practitioners in the recent years. Presented concepts are arranged into a formal classification consisting of fact types, factual and fact-dimensional relationships, and dimension types, supplied with explanatory examples from real-world usage scenarios. Both the static elements of the model, such as types of fact and dimension hierarchy schemes, and dynamic features, such as support for advanced operators and derived elements. We also propose a semantically rich graphical notation called X-DFM that extends the popular Dimensional Fact Model by refining and modifying the set of constructs as to make it coherent with the formal model. An evaluation of our framework against a set of common modeling requirements summarizes the contribution.

Categories and Subject Descriptors: Software & Applications [**databases**]:

General Terms:
Additional Key Words and Phrases: Data Warehousing, On-Line Analytical Processing, Multidimensional Data Model

Journal of Computing Science and Engineering, Vol. 1, No. 2, December 2007, Pages 125-160.

1. INTRODUCTION

OLAP (*On-line Analytical Processing*) [Codd et al. 1993] emerged in the 90s as a new technology for providing all the key people in the enterprise with access to whatever level of information they need for decision making. OLAP is employed on top of data warehouse systems. Data warehouse provides a separate database that integrates the data extracted from various operative systems and external sources and rearranges it into multidimensional views to enable simple but powerful aggregation. Applicability of OLAP is by no means restricted to business scenarios. Its universality bears on the concept of data "analyzability": the data should be homogenized, integrated, and preprocessed to enable efficient and goal-oriented analysis [Bauer 2004]. The need for this kind of analysis is encountered in virtually any application domain dealing with large data volumes accumulated over time. In the last years, deployment of data warehouses has reached out for a multitude of non-business domains and rather unconventional applications, such as government, academia, life sciences, bio-informatics, education, research, medicine, etc.

Even though data warehousing is an established and widely adopted practice in modern information technology platform, there exist numerous open research issues in this area [Hümmer et al. 2002]. Many of those issues arise due to the attempts to apply the business performance oriented OLAP techniques to non-conventional application scenarios. The causes of deficiencies and failures are manifold, from the underlying conceptual model to frontend "bottle-necks".

The universality of OLAP should not be taken for absolute, but should rather be considered in the context of quantitative analysis, based on aggregating large data volumes and applying data mining algorithms for extracting additional knowledge. Other types of analysis may require domain-specific models and approaches adequately capturing the semantics of the respective domain.

1.1 Contribution and Outline

Challenged by the limitations of the conventional OLAP approach, data warehouse researchers dedicate tremendous efforts to extending its flexibility and adaptability to novel application domains and analytical tasks. In the context of our research, the term "complex data" refers to data domains that cannot be adequately captured by the standard model. The standard model requires the data to be available in form of rigidly structured facts consisting of numeric measures as the focus of analysis and their descriptive dimensions as the context of the analysis. Examples of data scenarios violating this model are non-balanced or ragged dimension hierarchies [Jagadish et al. 1999; Niemi et al. 2001; Malinowski and Zimányi 2006; Mansmann and Scholl 2007], many-to-many mappings between facts and dimensions [Pedersen et al. 2001; Song et al. 2001], absence of pre-defined measures [Park et al. 2005; Mansmann et al. 2007b], and inadequacy of standard aggregation functions and operators [Ravat et al. 2007; Jensen et al. 2002]. The aim of this work is to collect and systematically classify a wide palette of extensions to the multidimensional data model proposed in the recent years. We also clarify some ambivalent and contradicting definitions found in the literature and propose a consistent terminological framework.

For modeling the illustrative examples that accompany formal concepts through-out the paper, we adopt the popular Dimensional Fact Model [Golfarelli et al. 1998] and extend it whenever the original notation is unable to adequately capture the semantics. We call the resulting extended conceptual model X-DFM.

In a multidimensional scheme, the entire data is shaped into cubes consisting of facts and dimensions. This fundamental observation coined the structure of our classification framework and, consequently, the structure of this paper. Section 2 sets the stage by providing an overview of the OLAP fundamentals. Related work in the field of overcoming the limits of the classical OLAP approach is described in Section 3. In Section 4 we formulate the modeling requirements of comprehensive data analysis. Section 5 contains a formalized presentation of our proposed multidimensional model. In Section 6 we proceed by providing a categorization of facts types and their relationships, followed by definition of various dimension schemes and dimension hierarchy types as well as relations within and across dimension hierarchies in Section 7. In Section 8 we evaluate the concepts and approaches proposed in this work against a set of commonly stated multidimensional properties. In the concluding Section 9 our contribution is summarized and directions for future research are identified.

2. OLAP FUNDAMENTALS

2.1 Multidimensional Data Model

OLAP technology draws its analytical power from the underlying *multidimensional data model*. The data is shaped into *cubes* of uniformly structured *facts*, consisting of analytical values, normally of numeric type, referred to as *measures*, uniquely determined by descriptive values drawn from a set of *dimensions* [Pedersen and Jensen 2001]. Each dimension forms an axis of a cube, with dimension members as coordinates of the cube's cells storing the respective measure values. Figure 1 shows a strongly simplified example of a 3-dimensional data cube that stores student enrollment numbers (measure NumPersons) determined by dimensions Country, Degree, and Semester. In real-world applications, data cubes may have arbitrarily many dimensions, and are therefore denoted *hypercubes*.

The values within a dimension are further organized into *classification hierarchies* to support additional aggregation levels. Attributes whereupon the hierarchy is defined are called *dimension levels*, or *categories*. Dimension levels along with their partial order are referred to as the dimension's *intension*, or *schema*, whereas the hierarchy of its members forms the dimension's *extension*, or *instance*. The hierarchical property on which the hierarchy is based is called the *analysis criterion*. Multiple hierarchies may be defined within a dimension, based on the same or to different analysis criteria. Hierarchies defined upon the same criterion are called *multiple alternative*, with time dimension as a classical example, as date values within a query may be summarized by week or by month, but not by any combination of the two. Hierarchies based on various criteria are called *parallel*, with a corresponding example of Degree dimension depicted in Figure 2: one classification is based on the attribute Degree Type while the other draws upon Subject. In contrast to multiple alternatives, parallel hierarchies can be explored in combination, as their aggregation paths are not related to each other.

In addition to the analysis criterion itself, dimension categories may include non-

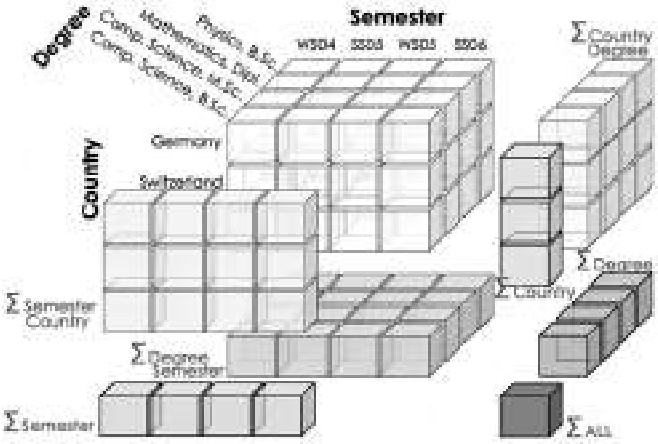


Figure 1. A sample 3-dimensional cube (fragment) storing student enrollment numbers.

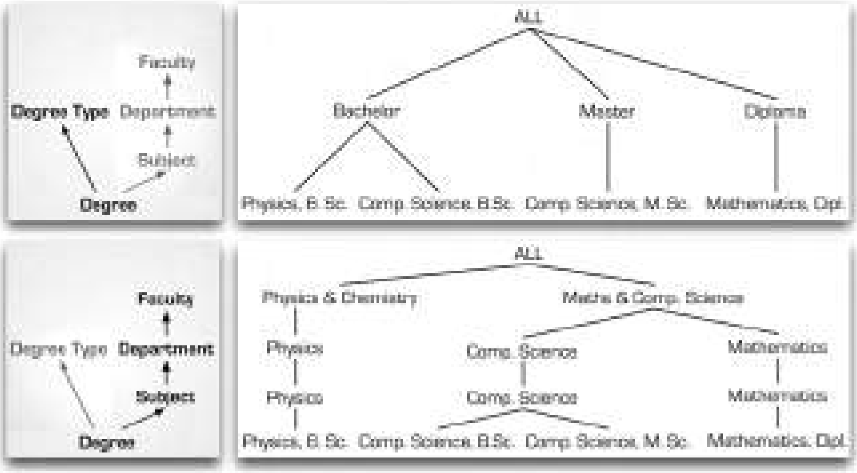


Figure 2. Dimension Degree with multiple hierarchies: schema (left) and instances (right).

hierarchical characteristics, or *properties*. In the example of Degree hierarchy, Department level may have properties such as Dean, Location, and Foundation Date.

2.2 Implementation Issues

The most popular implementation of an OLAP architecture is a relational one (ROLAP). Data cubes are stored in relations of types *fact table* and *dimension table*. A *fact table* stores the fact entries and is composed of two types of columns – measures and dimensions – where each dimension column is a foreign key to the respective dimension table. The primary key of a fact table is usually a composite key made up of all its foreign keys. A *dimension table* is used for storing the members of each dimension along with its classification hierarchies.

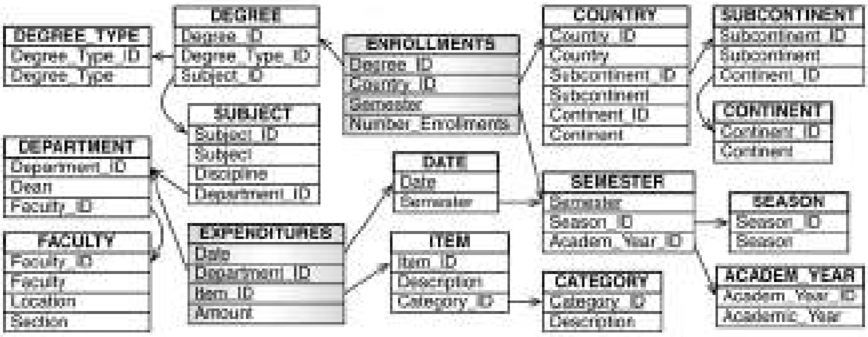


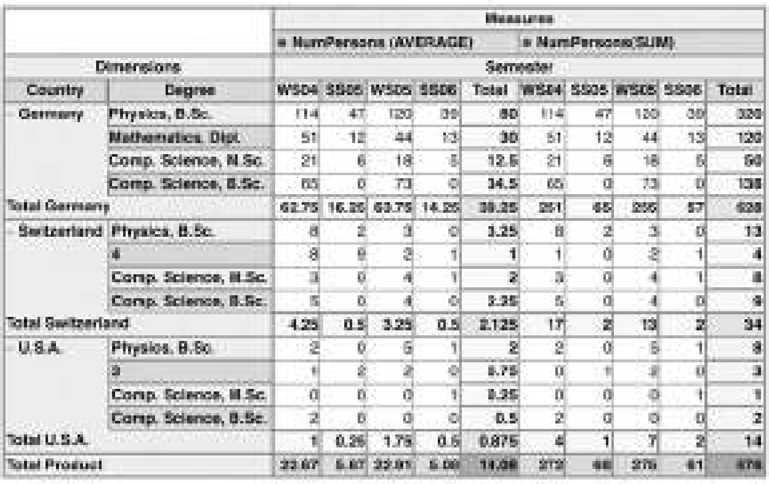
Figure 3. Galaxy schema of two fact tables with normalized dimension tables.

The two logical design options are star schema and snowflake schema [Kimball 1996], differing solely in the way they handle dimension hierarchies. *Star schema*, used in most data warehouses, places each dimension with all its hierarchies into exactly one de-normalized relation to facilitate navigation and improve query performance. *Snowflake schema* is a refinement of the star schema, in which each dimension hierarchy is decomposed into multiple tables, one per level, to avoid redundancy. Normalized storage is also advantageous for explicit sharing of dimensions and their parts among multiple data cubes. Multiple fact tables related via dimension sharing form a *galaxy*. This schema is very flexible and powerful, however, it comes at the expense of high design overhead because many variants of aggregation must be considered. Figure 3 shows an example of a galaxy constructed from the snowflake schemas of cubes ENROLLMENTS and EXPENDITURES. Even though the cubes do not fully share any dimension, they are related at non-bottom granularity via shared dimension levels Semester and Department.

2.3 OLAP Queries and Operators

A traditional interface for analyzing OLAP data is a *pivot table*, or *cross-tab*, which is a multidimensional spreadsheet produced by specifying one or more measures of interest and selecting dimensions to serve as vertical (and, optionally, horizontal) axes for summarizing the measures. The power of this presentation technique comes from its ability to summarize detailed data along various dimensions and arrange the aggregates computed at different granularity levels into a single view preserving the "part-of" relationships between the aggregates. Figure 4 exemplifies the idea of "unfolding" a 3-dimensional data cube from Figure 1 into a pivot table.

End-users analyze multidimensional data interactively using advanced visual interfaces. Query specification is done implicitly by populating a selected presentation layout, such as the pivot table or a more sophisticated visualization technique, with data and incrementally refining the view. Any OLAP query follows the same scheme, i.e., it consists of the same query clauses, some of which are optional. In SQL, a query is structured as follows (each line corresponds to a single clause, with optional elements placed in square brackets):



		Measures									
		≡ NumPersons (AVERAGE)					≡ NumPersons(SUM)				
Dimensions		Semester									
Country	Degree	WS04	SS05	WS05	SS06	Total	WS04	SS05	WS05	SS06	Total
Germany	Physics, B.Sc.	114	47	120	39	80	114	47	120	39	320
	Mathematics, Dipl.	51	12	44	13	30	51	12	44	13	120
	Comp. Science, M.Sc.	21	6	18	5	12.5	21	6	18	5	50
	Comp. Science, B.Sc.	65	0	73	0	34.5	65	0	73	0	138
Total Germany		62.75	16.25	63.75	14.25	39.25	251	65	255	57	628
Switzerland	Physics, B.Sc.	8	2	3	0	3.25	8	2	3	0	13
	4	8	9	2	1	1	1	0	2	1	4
	Comp. Science, M.Sc.	3	0	4	1	2	3	0	4	1	8
	Comp. Science, B.Sc.	5	0	4	0	2.25	5	0	4	0	9
Total Switzerland		4.25	0.5	3.25	0.5	2.125	17	2	13	2	34
U.S.A.	Physics, B.Sc.	2	0	5	1	2	2	0	5	1	8
	3	1	2	2	0	0.75	0	1	2	0	3
	Comp. Science, M.Sc.	0	0	0	1	0.25	0	0	0	1	1
	Comp. Science, B.Sc.	2	0	0	0	0.5	2	0	0	0	2
Total U.S.A.		1	0.25	1.75	0.5	0.875	4	1	7	2	14
Total Product		22.67	5.67	22.91	5.08	14.08	272	68	275	61	676

Figure 4. A pivot table view of average and total number of student enrollments broken down vertically by Country and Degree and horizontally by Semester.

```
SELECT [ dimension_attribute, ] measure_list
FROM table_list
[ WHERE predicate_list ]
[ GROUP BY [ ROLLUP | CUBE ] dimension_list ]
[ HAVING measure_predicate_list ]
[ ORDER BY attribute_list [sort_direction] ]
```

Dimension, measure, table, and predicate lists consist of at least one element each. Thereby, the simplest possible query for instantiating a visualization with a single aggregate value is as follows:

```
SELECT some_measure FROM some_table
```

The above query is generated automatically from the action of dropping a measure from the navigation into the visualization. Further clauses serve for refining the initial query: *i*) **WHERE** and **HAVING** clauses allow to specify selection conditions on any attributes and aggregated measure fields, respectively, *ii*) **GROUP BY**, contains the dimensions to aggregate along, and *iii*) **ORDER BY** sorts the output.

To enable powerful real-time data analysis, OLAP tools provide a set of specialized query operations for manipulating multidimensional data. OLAP operators take a data cube as an input and produce a new cube as an output. These operations are defined at logical level and have to be implemented into a visual framework in form of navigation or interaction options.

Classical operators found in the literature are the following ones:

- *DRILL-DOWN* deepens the level of granularity along a dimension already in the **GROUP BY** clause of the query. This is by far the most commonly used operator as the data is typically analyzed starting from a few coarsely aggregated values with their subsequent refinement.
- *ROLL-UP* decreases the level of granularity along a dimension. The extreme roll-up case of aggregating across the entire dimension, thus resulting in decrementing

the cube's dimensionality, is also known as the *PROJECT* operation.

—*SLICE&DICE* selects a sub-cube by specifying selection conditions on multiple dimensions in the drill path.

—*RANKING* outputs the top/bottom n cube cells with respect to the aggregate's value.

—*PIVOT* changes dimensional orientation of the view, e.g., swaps columns and rows in a pivot table.

A number of extended drilling operators are provided by some vendors:

—*DRILL-THROUGH* shows the original fact entries behind the aggregates.

—*DRILL-WITHIN* drills down to a different classification hierarchy of the same dimension.

—*DRILL ANYWHERE* increases dimensionality by drilling down into a dimension not yet in the drill path.

—*DRILL-ACROSS* joins multiple related data cubes along their shared dimensions to combine or compare their measures.

Another group of operators perform filtering, i.e., reduction of the subset of interest, and are variants of *SLICE&DICE*:

—*SLICE* reduces dimensionality of the data set by filtering one of the dimensions in the drill path to a single value.

—*DICE* specifies the values to be excluded from a dimension in the drill path.

—*SELECT* reduces a dimension in the drill path to a set of values or to a certain value range.

—*FILTER* specifies selection conditions on dimensions outside of the drill path, thus resulting in changed aggregated values.

—*CONDITIONAL HIGHLIGHTING* marks the aggregates satisfying a specified condition in the context of the original data set.

A view reordering operator *SWITCH* allows users to manually rearrange the elements in the visual presentation.

Finally, there exist two advanced operators, defined in [Pourabbas and Rafanelli 2000], which enable dynamic manipulation of the cube's scheme:

—*PUSH* allows to specify a measure from an arbitrary dimension category.

—*PULL* is the converse of *PUSH* that allows to convert a measure into a dimension.

OLAP operations differ in their complexity: some result in a new database query (e.g., *DRILL-DOWN*) or generation of new metadata (e.g., *PUSH* and *PULL*), others can be computed in-memory from the original query result (e.g., *ROLL-UP* and *DICE*) or require simple rearrangement on the display (e.g., *SWITCH*).

3. RELATED WORK

Deficiencies of the original multidimensional data model and proposals of extended models have become an active data warehousing research issue in the last decade. The necessity to develop novel concepts was emphasized [Zurek and Sinnwell 1999] and a series of extensions have been proposed in the literature. Most of the proposals were coined by a set of requirements drawn from specific application scenarios and, thus, do not claim to be ultimate or universal. Disclosure of novel applications continues to impose new modeling challenges and will undoubtedly continue to encourage further contributions.

[Pedersen et al. 2001] formulated 11 requirements of comprehensive data analysis and evaluated 14 state-of-the-art data models for data warehousing from both the research community and commercial systems against those requirements. As none of the models appeared to provide more than 6 of the 11 features, the authors proposed their own extended model for capturing and querying complex multidimensional data. Evaluation criteria specified in [Pedersen et al. 2001] are by no means universal as those were drawn from a specific case study. Nevertheless, the proposed extended model, supporting such features as non-summarizable hierarchies, many-to-many relationships between facts and dimensions, handling temporal changes and imprecision, is one of the most powerful among existing models. A similar attempt to classify and evaluate the state of the art in the multidimensional modeling is presented in [Abelló et al. 2001]. However, the authors used two orthogonal sets of classification criteria, namely, the kind of constructs/concepts they provide and the design phase at which they are employed. Another assessment of conceptual models is provided in [Luján-Mora et al. 2006], in which six prominent multidimensional models are evaluated against an exhaustive set of requirements regarding facts, dimensions, measures, operators, etc. The model is capable of handling advanced concepts, such as derived measures, many-to-many mappings, measure additivity properties, and multiple dimension hierarchies.

[Trujillo et al. 2001] propose an O-O multidimensional modeling (OOMD) approach that provides a theoretical foundation for the use of object-oriented features in data warehousing and OLAP applications. This approach introduces a set of minimal constraints and extensions to the UML for representing multidimensional modeling properties for these applications. In [Luján-Mora et al. 2002], the authors propose to use UML package diagrams for facilitating the data warehouse design. The proposed approach benefits from the package grouping mechanism of UML to group classes into higher-level units and create different levels of abstraction. Furthermore, a UML extension based on the self-extensibility mechanisms of UML by means of package stereotypes is provided.

Major research efforts in the field of multidimensional modeling are focused on handling complex dimensions [Niemi et al. 2001; Pedersen et al. 2001; Hurtado and Mendelzon 2002; Malinowski and Zimányi 2006; Mansmann and Scholl 2007]. It is but comprehensible: traditional models enforce homogeneity, completeness, strictness, and balancedness in dimension hierarchies, which appears to be a too rigid setting for many real-world scenarios. This rigidness comes from the requirement of *summarizability* for all dimensional hierarchies. The concept of summarizability, coined in [Rafanelli and Shoshani 1990] and further explored by other authors [Lenz and Shoshani 1997; Hurtado and Mendelzon 2001], requires distributive aggregate functions and dimension hierarchy values, or informally, that 1) facts map directly to the lowest-level dimension values and to only one value per dimension, and 2) dimensional hierarchies are balanced trees [Lenz and Shoshani 1997]. In practice, summarizability guarantees correct aggregation and optimized performance, as any aggregate view is obtainable from a set of pre-computed views defined at lower aggregation levels. However, data hierarchies in many real-world applications are not summarizable and, therefore, inadequate as OLAP dimensions. In a survey on open issues in multidimensional modeling [Hümmer et al. 2002] identified unbalanced and irregular hierarchies and missing data as the most pressing challenges of dimensional modeling.

[Hurtado and Mendelzon 2001] proposed integrity constraints for inferring summarizability in heterogeneous dimensions and defined a formal framework for constraint-conform hierarchy modeling [Hurtado and Mendelzon 2002]. An approach to modeling dimension hierarchies with no enforcement of balancedness or homogeneity along with the corresponding SQL extensions called SQL($\mathcal{H}$) is described in [Jagadish et al. 1999]. [Niemi et al. 2001] analyzed unbalanced and ragged data trees and demonstrated how dependency information can assist in designing summarizable hierarchies. [Lehner et al. 1998] relaxed the condition of summarizability to enable modeling of generalization hierarchies by defining a generalized multidimensional normal form (GMNF) as a yardstick for the quality of multidimensional schemata. [Lechtenbörger and Vossen 2003] pointed out the methodological deficiency in deriving multidimensional schema from the relational one and extend the framework of normal forms proposed in [Lehner et al. 1998] to provide more guidance in the data warehouse design process. A remarkable contribution to the conceptual design was made by Malinowski and Zimányi who presented a comprehensive classification of dimensional hierarchies including those not addressed by current OLAP systems [Malinowski and Zimányi 2004] and formalized their conceptual model and its relational mapping [Malinowski and Zimányi 2006].

To the best of our knowledge, most of the extensions proposed in the above contributions have not been implemented by any existing data warehouse systems. In a previous work [Vinnik and Mansmann 2006], we presented a prototypical analysis interface capable of supporting a subset of irregular dimension hierarchies and allowing interactive data exploration using hierarchical visualization techniques. A more recent work [Mansmann and Scholl 2007] builds upon the classification framework of [Malinowski and Zimányi 2006] and extends it by providing a more formal and comprehensive categorization of dimension hierarchy types. All enumerated classes are inspected for summarizability and a two-phase transformation algorithm for deriving a logical schema is proposed. As a proof of concept, all introduced model extensions were implemented in a visual interface with a schema-based dimensional navigation structure for exploring data cubes along complex dimension hierarchies.

A few works in the multidimensional modeling are concentrated on the challenges other than complex dimensions. These other issues address non-conventional requirements concerning facts, measures, and fact-dimensional mappings. The extended model of [Pedersen et al. 2001] accounts for such features as symmetric treatment of dimensions and measures, many-to-many relationships between facts and dimensions, aggregation semantics awareness, and variable granularity of facts. [Song et al. 2001] analyzed existing approaches to handling many-to-many relationships between facts and dimension and identified 4 additional approaches tailored towards various analysis requirements. [Abelló et al. 2001] clarify some concepts related to multidimensionality in general and fact modeling in particular. The authors demonstrate the convertibility of fact and dimension roles in multi-fact multidimensional schemes.

[Ravat et al. 2007] demonstrated how OLAP can be applied for analyzing semi-structured data, such as XML documents. As this data type is non-additive and non-numeric, the whole analysis framework needs to be adapted. The authors proposed a conceptual model based on a “factless multi-dimension” representation and define a set adapted multidimensional operations and aggregation functions relevant for this type of analysis. A novel framework for multidimensional analysis of XML documents, denoted XML-OLAP, in which the multidimensional data is actually stored in the XML format, is presented in [Park et al. 2005]. XML cubes contain either numeric or text data and are queried using a new multidimensional expression language XML-MDX, which supports conventional OLAP operations as well as specialized text mining operators.

In [Mansmann et al. 2007b] we applied the data warehousing approach to business process analysis. The requirement to store the original process execution data rather than pre-defined performance measurements helped us identify new types of fact structures, factual and fact-dimensional relationships, and aggregation behaviors. Besides, absence of explicitly specified measures in the scheme raises the issue of enabling dynamic measure specification at query time.

4. REQUIREMENTS OF COMPREHENSIVE DATA ANALYSIS

An abundance of multidimensional models proposed in recent years is a result of specifying different sets of requirements a model has to meet. In this section, we integrate the requirements and properties proposed by various authors with respect to comprehensive multidimensional analysis over complex data into a unified framework. This framework serves as a reference for specifying an extended data model for OLAP. The requirements can be subdivided into two major classes: 1) *static* properties dealing with the structuring of the multidimensional data space, and 2) *dynamic* properties dealing with the supported analysis tasks.

In the literature [Pedersen et al. 2001; Luján-Mora et al. 2006; Abelló et al. 2001] including our previous works [Mansmann and Scholl 2006; 2007; Mansmann et al. 2007b], the following major static properties of multidimensional modeling have been identified:

(1) *Explicit separation of the cube structure and its data instances.* The structure of a data cube is modeled as a fact-dimensional scheme. The actual content is crucial for refining the scheme as to identify irregular hierarchies, partial containment, etc.

(2) *Facts with no measures.* Some applications rely on storing the original data, i.e., without pre-defined measure attributes. According to one of Kimball’s laws, any many-to-many relationship should be modeled as fact [Kimball 1996]. Therefore, it is necessary to allow storage of any type of many-to-many relations in form of facts.

(3) *Complex measures.* The model should support structured and derived measures as well as specification of measure’s additivity, i.e., aggregation semantics.

(4) *Complex facts.* The model should be capable of handling deviating patterns within the facts, such as heterogeneity, variable granularity, and missing values.

(5) *Multi-fact structures.* It should be possible to model application scenarios comprising multiple related fact types. Inter-fact relationships result in case of sharing common dimensions, consequently, it should be possible to model dimension sharing.

(6) *Fully and partially shared dimensions.* Facts may be compatible to each other at non-bottom granularity levels. This happens when they have a pair of partially shared dimensions, i.e., whose hierarchy schemes converge at a category, non-bottom for at least on of them. To recognize partial sharing, it is imperative to explicitly specify the overlap between related fact schemes.

(7) *Multiple roles of dimension categories.* In multi-fact schemes, the same dimension or its category may be used in multiple roles (e.g., time dimension may be used as start time and end time characteristic of a fact). Therefore, it should be possible to specify multiple roles of the same category.

(8) *Many-to-many fact-dimensional relationships.* Many-to-many mappings between facts and dimensions are common in practice and, therefore, should be manageable by the model.

(9) *Explicit hierarchies in dimensions.* Dimension hierarchies should be presented explicitly by the schema that distinguishes between dimension level attributes and property attributes belonging to a particular level.

(10) *Multiple hierarchies.* A single dimension can have multiple aggregation paths that may or may not converge at some upper level.

(11) *Complete hierarchies.* In a complete hierarchy, all child-level members fully roll-up to the same parent level and the extension of the latter consists of those child members only [Luján-Mora et al. 2006]. The model should provide constructs to specify the completeness, i.e., non-expandability, of a hierarchy.

(12) *Distinction between alternative and parallel hierarchies.* Multiple alternative hierarchies refer to the same analysis criterion and thus may not be used in combination as grouping conditions within a query. Parallel hierarchies are based on various criteria and may be used in combination.

(13) *Complex dimensions.* To support complex dimensions, the model should be able to capture the causes of complexity, such as non-covering, non-onto, and non-strict mappings, heterogeneity, etc.

(14) *Partial roll-up behaviors.* A "rolls-up-to" relationship between a fact and a dimension or between dimension categories may be full (each member participates in the relationship) or partial (members are allowed not to participate in the relationship). Partial containment may be a result of optionality, heterogeneity or specialization. The model should distinguish between various kinds of roll-up relationships.

(15) *Totally ordered hierarchies.* A dimension hierarchy is normally defined in term of partial ordering (parent-child relationships within pairs of members). However, in some hierarchies, members of the same hierarchy level may have to be ordered to enable sorting according to this ordering.

As for dynamic properties supported by the multidimensional model, we propose the following ones:

(1) *Symmetric treatment of facts and dimensions.* In a connected multi-fact scheme, a fact may act as a dimension of another fact, or a dimension may turn into a fact of a specific query.

(2) *Symmetric treatment of measure and dimension attributes.* Any attribute within a fact scheme may be turned into a measure of an analytical query.

(3) *Measure used as dimension.* Some queries may need to use the measure as a dimension of another measure within the same fact scheme.

(4) *Drill-across.* Drill-across operator enables combinations of multiple related data cubes in order to explore their measures in parallel or derive new measures.

(5) *Dynamic measure derivation.* Measures, not originally included into the scheme, can be added at query time by specifying their derivation formulae.

(6) *Dynamic dimension derivation.* Dimensions, derivable from the existing dimensions, but not originally included into the scheme, can be added at query time by specifying their derivation formula.

(7) *Dynamic hierarchy derivation.* Users should be able to arrange dimensional values into ad-hoc hierarchies of user-defined categories.

(8) *Resolution of many-to-many mappings.* In the presence of non-strict hierarchies, users should be prompted to resolve multi-parent relationships to ensure correct aggregation.

5. CONCEPTUAL MODEL: PRESENTATION AND FORMALIZATION

The aim of the conceptual model is to capture relevant data and relationships in the application domain in a semantically rich and implementation-independent fashion. Two major components of the semantic multidimensional model are the formalization and the graphical notation. Most of the existing models focus either of this components, but not both. Formal models tend to adopt some existing notation (e.g., ER, UML or their variants) or do not employ any.

In our opinion, the graphical notation should be fully aligned with the formal model in order to correctly capture its semantics. Therefore, we opt for a popular Dimensional Fact Model (DFM)proposed in [Golfarelli et al. 1998]. DFM is based on a pragmatic scientific approach, in which the graphical framework emanates from the formal conceptual framework. Besides, in the abundance of notations proposed in the literature, DFM stands out for its simplicity, elegance, and expressiveness for representing the concepts introduced in this paper.

In part of the formalization, our model adopts and modifies the notation used in our previous works [Mansmann and Scholl 2007; Mansmann et al. 2007a]. In its basics, the formalization relies on that of [Pedersen et al. 2001], [Jensen et al. 2002], and [Golfarelli et al. 1998] since those models have the necessary flexibility for handling complex dimensions. However, modifications become necessary as we address new requirements.

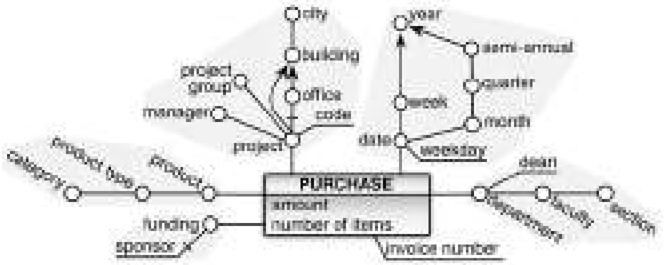


Figure 5. A sample 5-dimensional fact schema in the original DFM notation

5.1 X-DFM as the Graphical Modeling Notation

We require the graphical model to be as discriminative as the formal one, i.e., it should not map different concepts to the same graphical element. Semantical richness is preferred over minimality to ensure that the graphical model itself is sufficient for extracting all the necessary metadata.

We undertake numerous extensions of the original DFM and denote the resulting extended graphical notation X-DFM (eXtended Dimensional Fact Model) to distinguish it from the original DFM. The only purpose of the introduced extensions is to ensure the coherence between the formalization and the graphical notation.

DFM models data cubes in form of fact schemes. A fact scheme is a structured quasi-tree, which is a directed, acyclic, weakly connected graph, in which multiple directed paths may converge on the same vertex [Golfarelli et al. 1998], i.e., which allows multi-parent relationships between its vertices. Figure 5 demonstrates the usage of DFM at an example of modeling purchases within a university[1]. The resulting 5-dimensional fact scheme PURCHASE describes the purchase records in terms of measures amount and number of items characterized by dimensions funding, product, project, date, and unit.

DFM represents dimension level attributes by labeled circles while property attributes are terminal nodes represented by labeled lines, as shown in Figure 5. An arc connecting a pair of nodes represents a many-to-one relationship, also denoted "*rolls-up-to*" relationship, between them. DFM uses undirected arcs to represent directed edges as the direction can be unambiguously derived from the position of the fact node: any edge between C and C' is outgoing if C' has a longer distance to the fact node than C, and vice versa. The only ambiguity arises in the presence of multiple alternative or parallel hierarchies and exclusive partial roll-up behaviors, in which cases distinct paths meet in the same upper dimension level. To resolve such cycles, DFM uses directed arcs, as in the case with week and semi-annual both rolling up to year in the scheme depicted in Figure 5.

Optional relationships between any types of nodes are marked with a dash across the corresponding arc. Such relationship exists between project and office as only internal projects have office assignment. Similarly, an optional relationship my exist between a fact and a dimension and between a category and its property.

DFM allows fact nodes to have property, or non-dimensional, attributes, as is the

[1] Shaded bubbles enclosing each dimension are used solely as visual aids and are not part of DFM.

case with invoice number in Figure 5. However, that contradicts our fact scheme definition. Therefore, fact properties should be modeled as non-hierarchical dimensions. Kimball describes such properties as *degenerate* dimensions, i.e., consisting of a single key attribute [Kimball 1996].

With respect to the requirements specified in the previous section, DFM displays a number of deficiencies, such as the following ones:

—Facts are allowed to have non-dimension attributes. However, by definition, facts are composed solely of measures and dimensions.

—There is no construct for modeling many-to-many and one-to-one relationships between elements.

—Directed (i.e., many-to-one) relationships between the nodes are shown by non-directed edges. This seems to undermine the intuitiveness of the resulting scheme, especially when directed and undirected edges are used as alternative notations for the same concept.

—There is no distinction between hierarchy and non-hierarchy relationships between attributes: a "rolls-up-to" relationship between a pair of dimension categories does not visually differ from an association with a non-dimension property.

—There is no distinction between optional properties and partial rollup behavior.

—There is no construct for modeling heterogeneous rolls-up-to relationships.

—The scheme does not show the abstract top-level dimension categories that serve as root nodes of their hierarchies.

—DFM does not distinguish between multiple alternative and parallel hierarchies. However, the distinction is crucial for automatic recognition of valid aggregation paths. Multiple alternative hierarchies like the ones given by week and month offer alternative, i.e., mutually exclusive, aggregation paths for date. Parallel hierarchies, such as the ones given by manager and project group are defined on independent project characteristics and can thus be used as aggregation axes in arbitrary order. Parallel hierarchies behave like various dimensions of a fact.

—Measures inside a fact node are presented as plain text. However, each attribute is a node of the scheme and should be visually identifiable as such.

—There is no concept for presenting derived elements (facts, dimensions, measures).

We propose to resolve the above listed issues by applying the following logic:

—Non-dimension attributes of a fact should be modeled as degenerate dimensions.

—The box of the fact node should hold any attributes which exist only inside the fact entry. These attributes can be of type measure or a degenerate dimension.

—A dimension attribute with one-to-one relationship to the fact (i.e., fulfilling the primary key property w.r.t. the fact) is double-underlined.

—Measure attributes may be considered as a special kind of dimensions residing inside the fact. This assumption provides a basis for handling the requirements of interchangeability of measure and dimension roles. Measure's label is supplied with a black-colored circle.

—Different types of edges should be used for modeling different types or relationships. Edge types corresponding to various relationships (association, generalization, containment, etc.) can be adopted from the UML.

—Undirected edges are used to associate non-dimension attributes to their dimension levels. A directed edge (arrow) stands for a many-to-one relationship and a bi-directed edge captures a many-to-many relationship.

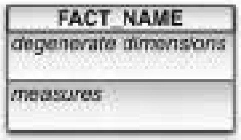
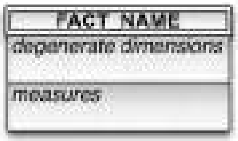
—All kinds of related or alternative "rolls-up-to" relationships (these relationships arise in case of multiple alternative hierarchies as well as in heterogeneous hierarchies of types non-covering and specialization) have to be visibly related in the model. We suggest to merge the respective edges into a single outgoing end. This way, it is possible to distinguish between exclusive and independent aggregation paths.

—Dashed lines are used to mark partial "rolls-up-to" relationships, dotted-lines link derived elements to their base elements.

—Abstract dimension categories consisting of a single value `ALL`, such as a superclass of multiple subclasses or the root node of a dimension hierarchy are shown as shaded nodes. The name of a root category is preceded by the "⊤" symbol.

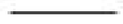
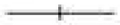
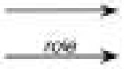
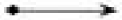
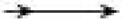
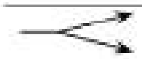
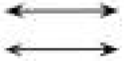
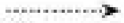
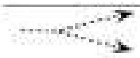
Table I gives an overview of the X-DFM extensions described above.

Table I: Graphical constructs of the proposed X-DFM notation

Element	Description
	**Fact** is a box-shaped node labeled by the fact's name and containing two sets of elements: 1) **degenerate dimension** attributes and 2) **measure** attributes. Both sets are allowed to be empty.
	**Degenerate fact** is a many-to-many fact-dimensional mapping extracted into a separate fact, shown by placing a double-lined frame around the fact's name.
● measure_name	**Measure attribute**, is a black circle-shaped node labeled by the attribute's name, placed inside the respective fact box.
○ attribute_name	**Dimension attribute** representing a non-abstract category is a circle-shaped node labeled by the attribute's name.
◎ attribute_name ◉ measure_name	**Derived dimension attribute / derived measure** is shown as a double-lined circle-shaped node. Optionally, dashed-lines can be used to connect the derived element to its base elements.
○ attribute_name	**Fact identifier** is a degenerate dimension attribute with one-to-one relationship to the fact, shown by underlining the attribute's name.
◍ category_name ◍ ⊤ category_name	**Abstract dimension category**, is a grey circle-shaped node labeled by the attribute's name. In case of a top-level category, the category name is shown as a subscript of the ⊤ symbol.

Continued on next page

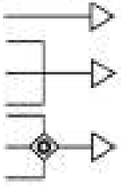
Table I – continued from previous page

Element	Description
attribute_name	**Non-dimension attribute** is a characteristic associated with its dimension attribute by an undirected edge.
attribute_name	**"Degree-of-belonging" attribute** is a property associated with a child category of a non-strict weighted "rolls-up-to" relationship.
	**Association relationship** is an undirected edge connecting a non-dimension attributes to their respective dimension attributes. The same construct is used for associating a fact with a dimension in one-to-one relationship with that fact.
	**Optional relationship** with a non-dimension attribute is specified by placing a dash across the edge.
	**"Rolls-up-to" relationship** is a many-to-one relationship between a fact and a dimension attribute or between a pair of dimension attributes, shown as a directed edge. In case of multiple roles of the same node, each edge is labeled by its role.
	**Complete "rolls-up-to" relationship** is a many-to-one relationship within a complete hierarchy.
	**Fuzzy "rolls-up-to" relationship** assigns child elements to parent elements dynamically based on some rules, and is shown as a double-pointed arrow.
	**Multiple alternative rolls-up-to relationships** are alternative aggregation paths, shown as directed edges merged into a single edge at the outgoing end.
	**Many-to-many relationship** between two dimension attributes is shown as a bi-directed edge[2]. In case of weak non-strictness, its edge is shown with a "light" arrow pointer.
	**Partial rollup** is an optional rolls-up-to relationship, shown as a dotted-lined directed edge.
	**Related partial rolls-up-to relationships**, or splitting paths, are mutually exclusive partial roll-up relationships in heterogeneous hierarchies, shown as dotted-lined directed edges merged into a single edge at the outgoing end.

Continued on next page

[2] Actually, bi-directional "rolls-up-to" relationships between categories are disallowed by the stratification condition, introduced later in this section. Therefore, this construct is admitted only at intermediate modeling stages.

Table I – continued from previous page

Element	Description
	**Generalization / specialization** is a one-to-one relationship between a category and its superclass/subclass, shown as a solid-line edge with a large hollow triangle at the superclass end. Multiple subclasses of a superclass category are shown in a shared-target style, i.e., as a tree rooted at the superclass. By default, specialization is disjoint. Overlapping subclasses are specified by placing a diamond with "o" symbol onto the edge at the point where it branches into subclass edges.

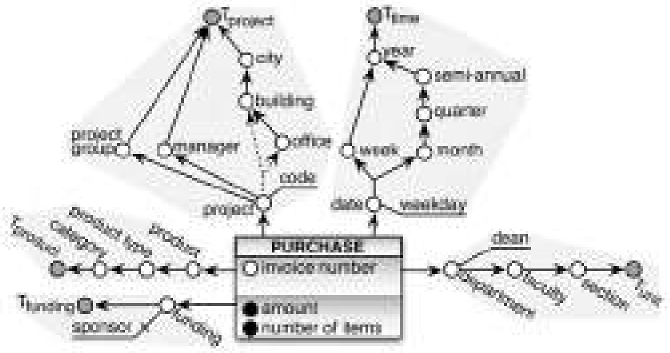


Figure 6. The revised sample fact schema in X-DFM

Figure 6 shows the results of modifying the original PURCHASE scheme from Figure 5, which now appears conform with each of the above X-DFM definitions.

5.2 Formalization

The output of the conceptual data warehouse design is a *multidimensional scheme*, i.e., a set of *fact schemes*, composed of facts, measures, dimensions, and hierarchies [Golfarelli et al. 1998].

Definition 5.1. A *fact*, denoted F, is a collection of uniformly structured data entries over a fact schema $\mathcal{F}$, with the latter defined as a pair $(\mathscr{M}^{\mathcal{F}}, \mathscr{D})$, where $\mathscr{M}^{\mathcal{F}} = \{\mathcal{M}_j, j = 1, \ldots, m\}$ is a set of measures and $\mathscr{D} = \{\mathcal{D}_i, i = 1, \ldots, n\}$ is a set of corresponding dimension schemes.

Definition 5.2. A *dimension*, denoted D is defined by its hierarchy scheme, or *intension*, $\mathcal{D}$ and the associated data set, or *extension*, E, so that $Type(E) = \mathcal{D}$.

Back to the example in Figure 6, the depicted fact scheme PURCHASE consists of two measures $\mathscr{M}^{\mathsf{PURCHASE}} = \{$amount, number of items$\}$, characterized by a set of six dimensions $\mathscr{D}^{\mathsf{PURCHASE}} = \{$funding, product, project, date, unit, invoice number$\}$.

A dimension scheme is a connected, directed graph, in which each vertex corresponds to an aggregation level and each edge represents a full or partial "rolls-up-to" relationship between the level, or, formally:

Definition 5.3. A *dimension scheme* is a quadruple $\mathcal{D} = (\mathscr{C}, \sqsubseteq_{\mathcal{D}}, \top_{\mathcal{D}}, \bot_{\mathcal{D}})$, where $\mathscr{C} = \{\mathcal{C}_k, k = 1, \ldots, p\}$ is set of category types, or dimension levels, in $\mathcal{D}$, $\sqsubseteq_{\mathcal{D}}$ is a partial order in $\mathscr{C}$, whereas $\top_{\mathcal{D}}$ and $\bot_{\mathcal{D}}$ are distinguished as the top and the bottom element of the ordering, respectively.

$\bot_{\mathcal{D}}$ corresponds to the finest grain of $\mathcal{D}$, i.e., the one at which $\mathcal{D}$ is connected to the fact scheme. $\top_{\mathcal{D}}$ corresponds to an abstract root node of the dimension's hierarchy that has a single value referred to as `ALL`: $\top_{\mathcal{D}} = \{\texttt{ALL}\}$.

Relation $\sqsubseteq_{\mathcal{D}}$ captures the containment relationships between category types. This containment may be *full*, denoted $\sqsubseteq_{\mathcal{D}}^{(\text{full})}$, or *partial*, denoted $\sqsubseteq_{\mathcal{D}}^{(\text{part})}$. Therefore, relation $\sqsubseteq_{\mathcal{D}}$ indicates the union of the two orders $\sqsubseteq_{\mathcal{D}}^{(\text{full})}$ and $\sqsubseteq_{\mathcal{D}}^{(\text{part})}$. Admission of partial containment, also known as *partial rolls-up-to* relationship, between category types is crucial for specifying heterogeneous dimension hierarchies.

Predicates $\sqsubseteq$ and $\sqsubseteq^*$ specify *direct* and *transitive* containment relationship, respectively, between a pair of category types in $\mathscr{C}$. Partial and full direct containment predicates are denoted $\sqsubseteq^{(\text{part})}$ and $\sqsubseteq^{(\text{full})}$, respectively. Therefore, predicates $\sqsubseteq$ and $\sqsubseteq^*$ without fullness/partiality indication imply that the containment is either full or partial, or formally: $\mathcal{C} \sqsubseteq \mathcal{C}' \Leftrightarrow (\mathcal{C} \sqsubseteq^{(\text{full})} \mathcal{C}' \vee \mathcal{C} \sqsubseteq^{(\text{part})} \mathcal{C}')$. Partial containment between any two categories $\mathcal{C}$ and $\mathcal{C}'$ ($\mathcal{C} \sqsubseteq^{(\text{part})} \mathcal{C}'$) occurs when members of $\mathcal{C}$ are not required to have parent members in $\mathcal{C}'$. A pair of partial containment relationships of the same category $\mathcal{C}$ (i.e., $\mathcal{C} \sqsubseteq^{(\text{part})} \mathcal{C}' \wedge \mathcal{C} \sqsubseteq^{(\text{part})} \mathcal{C}''$) are *exclusive* if any member of $\mathcal{C}$ rolls-up either to $\mathcal{C}'$ or $\mathcal{C}''$, but never to both. A set of exclusive partial "rolls-up-to" relationships is denoted $\mathcal{C} \sqsubseteq^{\text{part}} (\mathcal{C}'|\mathcal{C}'')$.

The following properties hold for the partial order relation $\sqsubseteq_{\mathcal{D}}$ and its predicates:

(1) Antireflexivity: $\nexists \mathcal{C}_j \in \mathscr{C} : \mathcal{C}_j \sqsubseteq \mathcal{C}_j$.

(2) Antisymmetry: $\nexists (\mathcal{C}_i, \mathcal{C}_j) \in \mathscr{C} : (\mathcal{C}_i \sqsubseteq \mathcal{C}_j \wedge \mathcal{C}_j \sqsubseteq \mathcal{C}_i)$.

(3) Transitivity: $\forall (\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathscr{C} : (((\mathcal{C}_i \sqsubseteq \mathcal{C}_j \vee \mathcal{C}_i \sqsubseteq^* \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubseteq \mathcal{C}_k \vee \mathcal{C}_j \sqsubseteq^* \mathcal{C}_k)) \Rightarrow \mathcal{C}_i \sqsubseteq^* \mathcal{C}_k)$.

The first of the above properties implies that there is no support for reflexive "rolls-up-to" relationships, i.e., of a category with itself. A classical example of such reflexive relationship could be a supervisor hierarchy within a category employee. The second property disallows a bi-directional "rolls-up-to" relationship between any pair of categories as those would result in cyclic aggregation paths. Thereby, properties (1) and (2) guarantee acyclic termination of all aggregation paths. The property of transitivity defines recursive "rolls-up-to" relationships within a hierarchy. For instance, if date is contained in month and month is contained in year, then date is transitively contained in year.

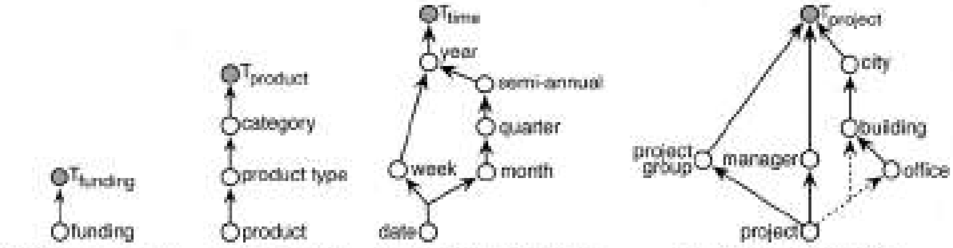
$\mathcal{C}_j$ is said to be a category type in $\mathscr{C}$, denoted $\mathcal{C}_j \in \mathscr{C}$. Dimension scheme defines a skeleton of the associated data tree, for which the following conditions hold:

(1) $\forall \mathcal{C}_j \in \mathscr{C} \backslash \{\top_{\mathcal{D}}\} : \mathcal{C}_j \sqsubseteq^{*(\text{full})} \top_{\mathcal{D}}$ (each non-top category type is fully contained in the top category type).

(2) $\forall \mathcal{C}_j \in \mathscr{C} \backslash \{\perp_{\mathcal{D}}\} : \perp_{\mathcal{D}} \sqsubseteq^* \mathcal{C}_j$ (the bottom category of dimension is fully or partially contained in all its other category types).

(3) $\nexists \mathcal{C}_j \in \mathscr{C} : \mathcal{C}_j \sqsubseteq \perp_{\mathcal{D}}$ (the bottom category type is childless).

In the simplest case, a dimension consists solely of the bottom and the top category types. A scheme of a single hierarchy is a lattice, whereas dimension schemes of multiple or parallel hierarchies may result in rather complex graph structures. Multiple hierarchies in $\mathcal{D}$ exist whenever there exists a category type at which at least two paths converge, or formally: $\exists \mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k, \in \mathcal{D} : \mathcal{C}_i \sqsubseteq \mathcal{C}_k \wedge \mathcal{C}_j \sqsubseteq \mathcal{C}_k$. Figure 7 shows examples of dimension schemes of various complexity.



(a) Non-hierarchy (b) Hierarchy (c) Multiple hierarchies (d) Parallel hierarchies

Figure 7. Dimension schemes as directed graphs of various complexity.

Definition 5.4. A *dimension category type* is a pair $\mathcal{C} = \langle \mathcal{A}^{\mathcal{C}}, \mathscr{A} \rangle$ where $\mathcal{A}^{\mathcal{C}}$ is the distinguished *dimension level* attribute and $\mathscr{A} = \{\mathcal{A}_r, r = 1, \ldots, x\}$ is a set of *property* attributes belonging to $\mathcal{C}$ and functionally dependent on $\mathcal{A}^{\mathcal{C}}$.

Definition 5.5. An *aggregation path* in $\mathcal{D}$ is any pair of category types $\mathcal{C}_i$, $\mathcal{C}_j$ such that $(\mathcal{C}_i, \mathcal{C}_j) \in \mathscr{C} \wedge \mathcal{C}_i \sqsubseteq^* \mathcal{C}_j$.

Having defined the scheme elements of the model, we proceed by defining dimension instances and their properties.

Definition 5.6. An *instance*, or *extension*, E associated with dimension scheme $\mathcal{D}$ is a pair $\langle C, \subseteq_E \rangle$, where $C = \{C_j, j = 1, \ldots, m\}$ is a set of categories such that $Type(C_j) = \mathcal{C}_j$ and $\subseteq_E$ is a partial order on $\cup_j C_j$, the union of all dimensional values in the individual categories.

Definition 5.7. *Dimension category* C of type $\mathcal{C}$ is a set of *member values* $C = \{e_i, i = 1, \ldots, n\}$ such that $\text{Type}(e_i) = \mathcal{C}$.

Distinction between the concepts *category* and *category type* is made in order to support modeling of fully and partially shared dimensions, in which the same category type, e.g., city, may be used as categories customer city, store city, etc.

The partial order $\subseteq_E$ relation on $\cup_j C_j$ is to be understood as follows: given $(e_1, e_2) \in \cup_j C_j$, $e_1 \subseteq e_2$ if e_1 is logically contained in e_2. Predicates $\subseteq$ and $\subseteq^*$ specify direct and transitive containment relationship, respectively, between a pair of dimension's member values. Notice that at the instance level, the containment relationship is always a *full* one. The total number of member values in category C_j is denoted $|C_j|$. The following conditions hold for dimension members:

(1) $\forall e_m \in C_i, \forall e_n \in C_j : e_m \subseteq e_n \Rightarrow C_i \sqsubseteq C_j$ (connectivity). This condition ensures, that the containment relationship between a pair of categories results from the containment relationship between the members of those categories and disallows "rolls-up-to" relationships between members of unrelated categories.

(2) $\nexists e_m \in C_i, \nexists e_n \in C_j : e_m = e_n \wedge \mathcal{C}_i \neq \mathcal{C}_j$ (disjointness of categories). Prohibiting any value to be a member of multiple category types enforces category sharing as well as disjointness of any pair of categories referring to different types.

(3) $\forall C_i : (\nexists e_m \in C_i, \nexists e_n \in C_i : e_m \subseteq^* e_j)$ (stratification, i.e. disallowance of containment relation between the members of the same category).

(4) $(Type(C_j) = \top_{\mathcal{D}}) \Rightarrow |C_j| = 1 \wedge C_j = \{\texttt{ALL}\}$ (top category consists of a single value ALL).

6. FACT TYPES AND FACTUAL RELATIONS

Classical designation of facts is to contain relevant measures of a business process. Normally, facts are modeled by specifying the measures of interest and the context (dimensions) for their analysis.

Definition 6.1. A fact scheme $\mathcal{F}$ is *measurable* if it has a non-empty set of measures, i.e., $\mathscr{M}^{\mathcal{F}} \neq \emptyset$.

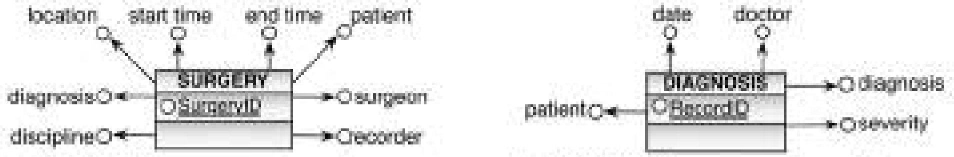
Kimball subdivides measurable facts into three classes: 1) *transactional*, 2) *periodic snapshots*, and 3) *cumulative snapshots* [Kimball 1996].

Technically, a fact type is given by a many-to-many relationship between a set of attributes. According to Kimball's laws, any many-to-many relationship is a fact by definition [Kimball 1996]. Some scenarios require storing many-to-many mappings in which no attribute qualifies to be a measure. Typical cases include recording of some events, where an event is given by a combination of simultaneously occurring dimensional characteristics. Such scenarios result in so called *factless fact tables* – a term introduced in [Kimball 1996]. However, *fact table* is a logical design construct corresponding to the concept of a *fact type*. Therefore, there is a need to define a conceptual equivalent of factless fact tables.

Definition 6.2. A fact scheme $\mathcal{F}$ is *non-measurable* if its set of measures is empty, i.e., $\mathscr{M}^{\mathcal{F}} = \emptyset$.

Major usage scenarios of non-measurable facts are *event tracking* and *coverage* tables [Kimball 1996]. The former model events as a robust set of many-to-many relationships between multiple dimensions, while the latter is used to track events that were eligible but did not happen (e.g, product items not bought by any customers). Another difference is that event tracking facts are *primary*, i.e., not derivable or dependent on other facts, while coverage facts are *secondary* as the latter are always related to some primary fact table.

To exemplify the concepts defined in this section, we borrow a case study concerned with surgical workflow analysis from [Mansmann et al. 2007a]. A data warehouse is used for storing the records of executed surgical interventions. A surgical process is decomposed into *activities*, or work steps, describing the actions of the surgeons and other participants, and into *events* describing discrete occurrences. Surgical process executions themselves can also be viewed as events,



(a) SURGERY as a non-measurable event tracking fact scheme

(b) DIAGNOSIS as a non-measurable coverage fact scheme related to SURGERY

Figure 8. Examples of non-measurable fact schemes.

or event tracking facts, of type SURGERY. Figure 8(a) gives an example of the resulting non-measurable fact scheme (for simplicity, only the bottom dimension categories are shown at this stage).

An example of a useful coverage fact in this scenario could be a record of all patient diagnoses. A patient may have multiple diagnoses that evolve in time. SURGERY facts capture only the primary diagnosis of each executed surgical instance. A complete patient diagnosis history is managed in a coverage fact DIAGNOSIS, depicted in Figure 8(b).

Whenever facts correspond to actual events, there may exist a dimensional attribute with primary key properties, i.e. whose values are unique for each fact entry. For example, each SURGERY instance has a unique SurgeryID. [Kimball 1996] introduces the concept of a *degenerate dimension* to handle such *id*-like attributes, while the DF model of [Golfarelli et al. 1998] handles them as *non-dimension attributes of a fact*.

Definition 6.3. Dimension $\mathcal{D}$ is *degenerate* if it has a single category $\mathcal{C}$ consisting solely of a dimension level attribute, i.e., $\mathcal{C} = \{\mathcal{A}^{\mathcal{C}}, \varnothing\}$.

Definition 6.4. A degenerate dimension $\mathcal{D}$ is called a *fact identifier* if its attribute has a one-to-one relationship with its fact: $\forall(e_1, e_2) \in \mathcal{C} : e_1 \neq e_2$.

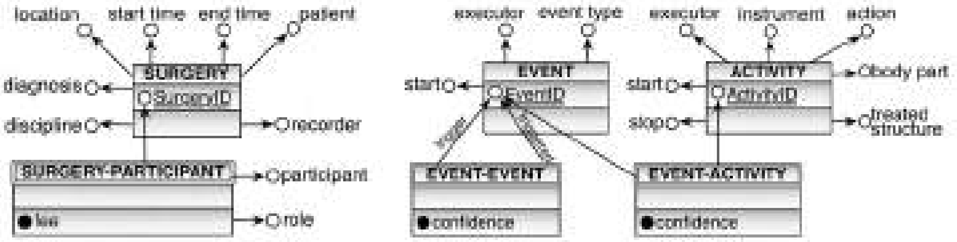
Since a degenerate dimension is only valid in the context of its fact, X-DFM places it inside the fact's node as shown in the examples in Figure 8. A fact identifier is shown with a double-underlined name. Fact identifier dimensions are crucial for modeling inter-factual relationships, as discussed later in this section.

There may exist a many-to-many mapping of a fact with some of its dimensional characteristic or even with another fact. [Giovinazzo 2000] proposes a concept of a *degenerate fact*, defined as a measure recorded in the intersection table of a many-to-many relationship between a pair of facts or a fact and a dimension. We suggest to distinguish between the following types of fact degeneration:

—*Satellite fact* scheme $\mathcal{F}'$ extracts a many-to-many relationship between fact scheme $\mathcal{F}$ and one or several dimension schemes $\{\mathcal{D}'\}$ along with the corresponding measure characteristics of this relationship into a separate fact. Thereby, $\mathcal{F}$ acts as a dimension of $\mathcal{F}'$. The term *satellite* reflects the accompanying nature of this fact with respect to its base fact.

—*Association fact* scheme $\mathcal{F}''$ extracts a many-to-many relationship between a pair of fact schemes $\mathcal{F}$ and $\mathcal{F}'$ along with the corresponding measure characteristics of this relationship into a separate fact. A special case of a self-association fact ($\mathcal{F} = \mathcal{F}'$) occurs when $\mathcal{F}$ acts as two different dimensions in $\mathcal{F}''$.

Back to the example in Figure 8(a), an attempt to replace a many-to-one mapping with surgeon by a many-to-many participant mapping would yield a satellite fact SURGERY-PARTICIPANT, shown in Figure 9(a), for storing all participants of each surgery with fee as a measure of this relationship.



(a) SURGERY-PARTICIPANT as a satellite fact of SURGERY

(b) EVENT-ACTIVITY as a fact association and EVENT-EVENT as a fact self-association

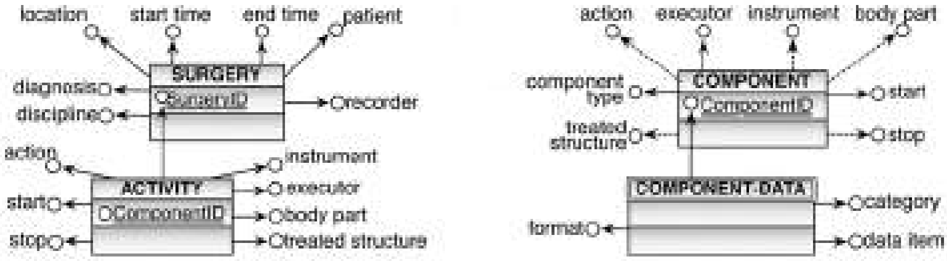
Figure 9. Examples of satellite fact schemes.

An example of an association fact is a trigger relationship between the facts of type EVENT and ACTIVITY (e.g., event X triggered activity Y). Figure 9(b) shows the resulting EVENT-ACTIVITY fact and its base facts acting as dimensions. Similarly, a self-association of EVENT is defined to store a trigger relationship between pairs of events and is also shown in Figure 9(b) as EVENT-EVENT fact.

Similar to dimension levels, a pair of facts $\mathcal{F}$ and $\mathcal{F}'$ may form a fact hierarchy, or *fact rollup*, denoted $\mathcal{F} \sqsubseteq \mathcal{F}'$, if the relationship between them is many-to-one. Such relationships typically arise between event tracking facts modeling events at different granularity levels. In our example, ACTIVITY rolls-up to SURGERY, as depicted in Figure 10(a).

A *fact generalization/specialization* arises when heterogeneous fact types are extracted into a superclass fact type in part of their common characteristics.

In our example, EVENT and ACTIVITY are two heterogeneous component types of a surgery. To be treated as the same class in part of their common attributes,



(a) Fact rollup of ACTIVITY to SURGERY

(b) Heterogeneous fact type COMPONENT

Figure 10. Examples of (a) containment and (b) identity relationships between fact schemes.

the two fact types are made subclasses of a class COMPONENT, as shown in Figure 11. Fact generalization is especially favorable if heterogeneous classes have common degenerate facts, as is the case with COMPONENT-DATA, modeled as a satellite of the generalized fact type COMPONENT.

Finally, fact types can be divided into homogeneous and heterogeneous. Fact scheme is homogeneous if it disallows partial rollup relations between the fact and any of its dimensions, and is heterogeneous otherwise. Heterogeneous fact type is a result of storing non-uniformly structured facts as the same type, i.e. avoiding specialization. Figure 10(b) shows a heterogeneous fact type COMPONENT storing all characteristics of both subclasses EVENT and ACTIVITY.

All fact types considered so far are called *primary* as they store fine-grained non-derived fact entries. All types of facts that are derivable from the primary facts are called *secondary*. The latter can also be categorized according to the way they were obtained:

—*Aggregation* fact type contains the measures of its base fact type, however, at a coarser granularity.

—*Drill-across* fact type contains measures obtained by combining the measures of multiple related fact types.

—*Partition* fact type contains a subset of fact entries from its base fact type.

—*Transformation* fact type is drawn by defining a measure from a dimension category (*push*) or turning a measure into a dimension (*pull*).

6.1 Duality of Fact and Dimension Roles

In this section we encountered multiple examples, in which one fact acts as a dimension another fact. That might seem paradox, but it has its legitimacy. Structurally, both facts and dimensions are given by a graph of part-whole relationships between their categories. The difference is that the aggregation graph of a dimension depends on its proper semantics, while the aggregation graph of a fact depends on the aggregation hierarchies of its analysis dimensions [Abelló et al. 2001]. Fact and dimension roles are fixed only in the context of isolated fact schemes. What happens to those roles in the context multidimensional multi-fact schemes?

We suggest that these roles are determined by the focus of the analytical task at hand. According to the definition, facts represent the focus of the analysis. This focus varies from one query to another. For example, querying an association fact turns the base fact type(s) of the latter into dimensions. Altogether, multidimensionality implies that what is considered a fact by one analyst could be considered

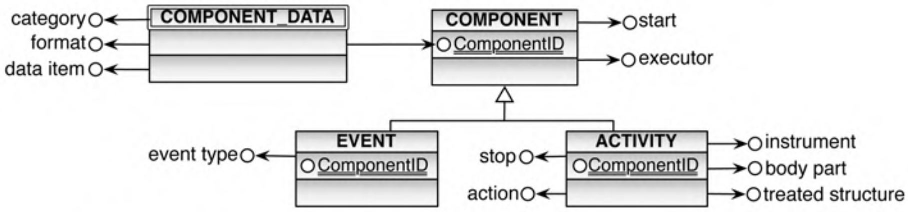


Figure 11. Fact generalization of classes EVENT and ACTIVITY as a superclass COMPONENT.

a dimension by another one, and vice versa.

The first interchangeability case is concerned with a fact scheme acting as dimension of another fact scheme. Fact scheme $\mathcal{F}'$ can be treated as a dimension in fact scheme $\mathcal{F}''$ while querying its measures when $\mathcal{F}''$ contains the fact identifier dimension of $\mathcal{F}'$. This relationship may be encountered in satellite facts and fact rollups of event tracking facts.

Lemma 6.5. *Fact schema $\mathcal{F}'$ plays a role of a dimension with respect to fact schema $\mathcal{F}''$ if both schemes are connected via the fact identifier dimension of $\mathcal{F}'$. Dimensions within $\mathcal{F}'$ are to be treated as parallel hierarchies, with at the fact identifier of $\mathcal{F}'$ as the bottom-level category.*

One implication of this interchangeability is that it results in different conceptual schemes for the same data fragment, depending on the focus of the analysis. Figure 12 illustrates the example of two conceptual views at the satellite fact relationship between SURGERY-PARTICIPANT and SURGERY. The left-side scheme is focus-independent whereas the right-hand one focuses on the valid analysis paths for measure fee of SURGERY-PARTICIPANT. Thereby, fact scheme SURGERY is transformed into a dimension surgery, in which all dimensions of SURGERY turn into parallel hierarchies, diverging from the category SurgeryID. The validity of regarding the fact identifier of SURGERY as a bottom-level category in surgery is given by the fact that the latter has the same grain as SURGERY fact entries, and thus, has a many-to-one, i.e., "rolls-up-to" relationship, to all other dimensions.

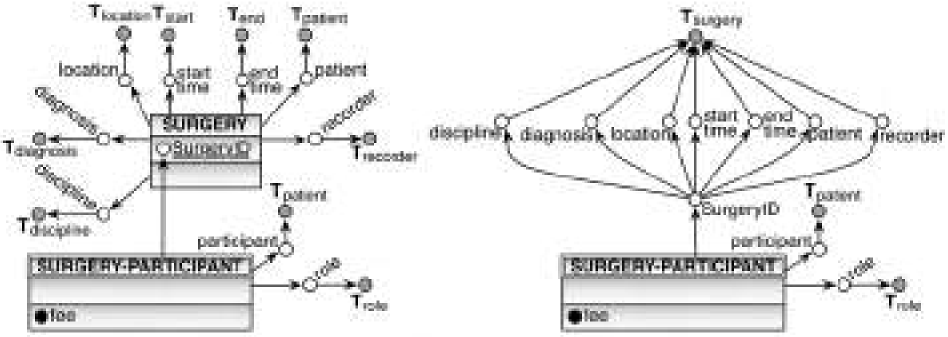


Figure 12. Fact scheme SURGERY in the role of a dimension in fact scheme SURGERY-PARTICIPANT.

Another kind of interchangeability related to treating dimensions as measures, and vice versa. Support of advanced OLAP operators, such as `PUSH` for converting a dimension category into a measure and `PULL` for converting a measure into a dimension, as well as `DRILL-ACROSS` for combining measures from multiple related fact schemes, is a challenge not handled by statical conceptual models. The result of applying these operators is a new conceptual multidimensional scheme. Our solution for supporting scheme-transforming operators at the conceptual level is straightforward—to explicitly model the outcome schemes of such operations. Figure 13 exemplifies this idea by showing the conceptual consequences of "pushing" a dimension category hospital into a newly defined measure #Hospitals (specified as

`COUNT(HospitalID)`). Notice, that the "pushed" category itself as well as all categories that roll-up to it have to be removed from the dimension's scheme as their granularities. Besides, the degenerate dimension SurgeryID loses its fact identifier property due to the changed granularity of the fact entries.

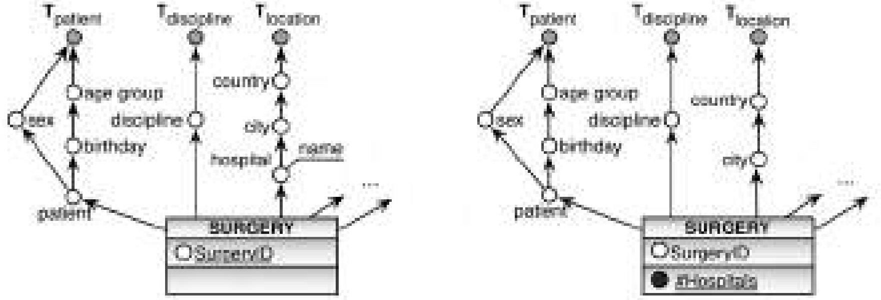


Figure 13. Transformation of the original fact scheme (left-side) caused by a `PUSH` operation (right).

7. CLASSIFICATION OF DIMENSIONS

In this section we refine the definitions related to dimensions and their hierarchies, using the example of the fact scheme PURCHASE introduced in Section 5. A dimension hierarchy is based on a hierarchical characteristic, also referred to as the *analysis criterion*, upon which the hierarchy is defined. For instance, in the dimension project (see Figure 6), a hierarchy given by office → building → city orders projects according to the location criteria, whereas manager creates a supervision hierarchy and project group category groups projects thematically. To be able to classify dimensions according to their hierarchy types, in this section the concept *dimension* will be redefined in terms of its constituting hierarchies.

Definition 7.1. A *hierarchical domain* is a non-empty set V_H with the only defined predicates = (identity), $\sqsubseteq$ (child/parent relationship), and $\sqsubseteq^*$ (transitive closure, or descendant/ancestor relationship) such that the graph $G_{\sqsubseteq}$ over the nodes $\{e_i\}$ of V_H is a tree. A *data hierarchy* H defined over V_H is *strict* if it disallows multi-parent relationships, i.e., $\forall(e_1, e_2, e_3) \in V_H : (e_1 \sqsubseteq e_2 \wedge e_1 \sqsubseteq e_3) \Rightarrow e_2 \neq e_3$.

In the context of OLAP we are concerned only with *structured* hierarchies, i.e., those whose instances adhere to a certain schema.

Definition 7.2. A *hierarchy scheme* $\mathcal{H}$ within a dimension scheme $\mathcal{D}$ is a 5-tuple $\mathcal{H} = (\mathscr{C}^{\mathcal{H}}, \sqsubseteq_{\mathscr{C}}, \sqsubseteq_{\mathcal{D}}, \top_{\mathcal{D}}, \bot_{\mathcal{D}})$ for which holds:
$\nexists(\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathscr{C}^{\mathcal{H}} : \mathcal{C}_i \sqsubseteq \mathcal{C}_j \wedge \mathcal{C}_i \sqsubseteq \mathcal{C}_k$, i.e., no category is allowed to have multiple full "rolls-up-to" relationships.

Thereby, any hierarchy in a dimension has the same bottom and top category types as the dimension itself. According to the above definition, a scheme is guarantied to yield a single hierarchy by excluding multiple full roll-up relationships of the same category. Notice that it does not prohibit heterogeneous schemes, produced by related partial roll-ups.

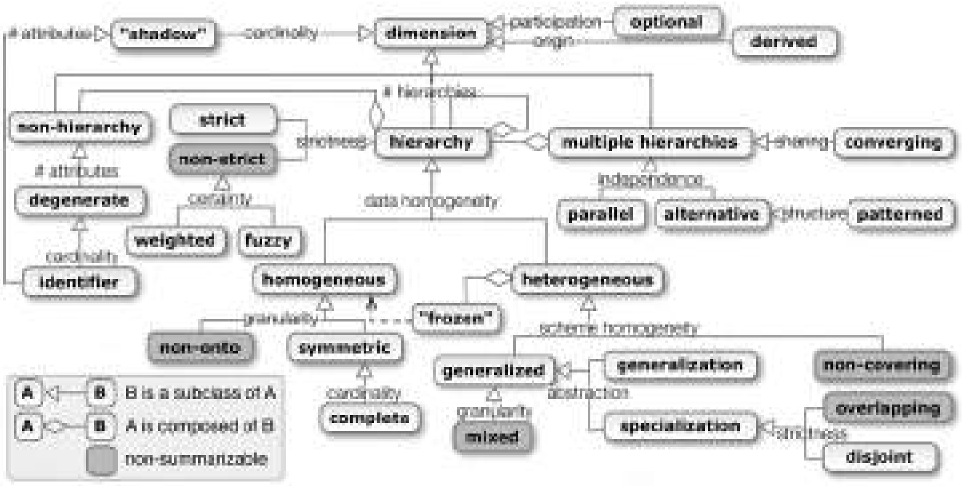


Figure 14. Categorization of dimension types.

The *instance*, or the *extension*, of a hierarchy results from specifying the elements for each category type and the partial order between them.

Definition 7.3. A *hierarchy (instance)* H associated with hierarchy scheme $\mathcal{H}$ is a pair $H = (C, \subseteq_H)$, where $C = \{C_j, j = 1, \ldots, m\}$ is a set of categories such that $Type(C_j) = \mathcal{C}_j$, $\mathcal{C}_j \in \mathcal{H}$, and $\subseteq_H$ is a partial order on $\cup_j C_j$, the union of all dimensional values in the individual categories.

A *dimension extension* D associated with schema $\mathcal{D}$ is a merge graph of all its hierarchy instances H. A dimension scheme can similarly be obtained by merging its hierarchy schemes, however, in each case of multiple outgoing "rolls-up-to" relationships, it is necessary to specify whether those are multiple alternative or parallel hierarchies.

A dimension is decomposable into a set of hierarchy schemes $\{\mathcal{H}_i, i = 1, \ldots, n\}$ and the associated set of hierarchy instances $\{H_i, i = 1, \ldots, n\}$. With the above definition we achieve a coverage of a broad variety of hierarchical behaviors within a dimension. For instance, it is not prohibited to have multiple hierarchies, shared category types, or partial roll-up relationships.

We derive a classification of dimension types by investigating various properties of their schemes and extensions. Adherence or violation of a particular property is used for drawing a classification of dimension hierarchies. The properties used as classification criteria are: 1) the number of hierarchies $|\mathcal{H}|$ in $\mathcal{D}$, 2) homogeneity of $\mathcal{D}$, 3) types of attributes, and 4) cardinality of "rolls-up-to" relationships, 5) sharing of categories. Dimension types are identified starting from more general categories and proceeding to their subclasses and special cases, with the entire resulting categorization scheme depicted in Figure 14. The edges of the classification are labeled by the respective discrimination criteria. A more detailed variant of fact scheme PURCHASE from Section 5, shown in Figure 15, will serve as an illustrating example for different types of dimensions presented in this section.

Typically, a relationship between a fact and any of its dimensions is many-to-one, i.e., each fact maps to exactly one member of each dimension. A degenerate case of a dimension having a one-to-one relationship with the fact is called *"shadow"* dimension [Giovinazzo 2000]. A special case of a "shadow" dimension is a fact identifier that consists of a single attribute with a primary key property with respect to its fact.

A "rolls-up-to" relationship between the fact and the dimension may be *full* or *partial*. In case of a partial containment, the dimension is called *optional*.

A dimension, derived from one or several other dimensions, is called derived. In the PURCHASE scheme depicted in Figure 15, an example of such dimension is delay derived by subtracting order date from receipt date.

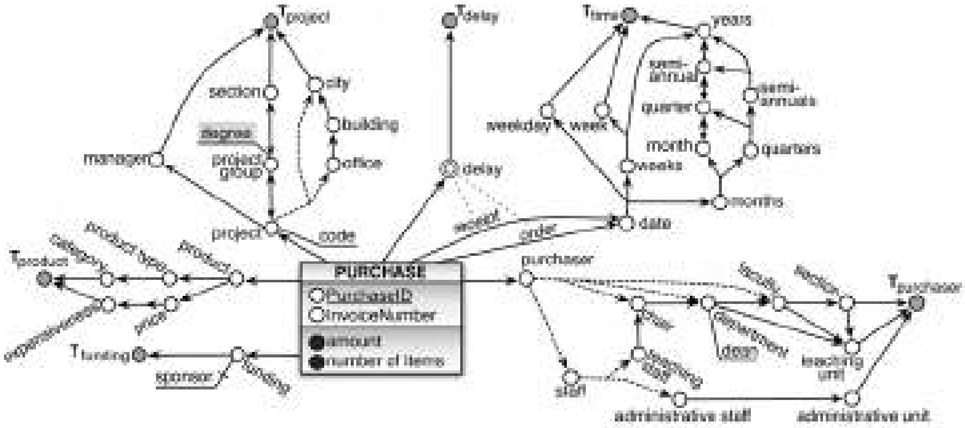


Figure 15. Fact scheme PURCHASE with added complexity.

Considering the number of hierarchies, a dimension is a *hierarchy*, *non-hierarchy*, or *multiple hierarchies*, with funding, product, and project as the examples of the respective types shown in Figure 15.

A dimension is a *non-hierarchy* if its scheme is composed solely of the bottom and the top category types:

$$\textbf{Non-hierarchy}(D)\colon\ \mathscr{C} = \{\bot_{\mathcal{D}}, \top_{\mathcal{D}}\}.$$

A dimension is a *hierarchy* if its scheme is a hierarchy with at least one category type, which is neither a bottom nor a top one:

$$\textbf{Hierarchy}(D)\colon\ |\mathcal{H}| = 1 \wedge \exists \mathcal{C}_i \in \mathcal{D} : \mathcal{C}_i \neq \bot_{\mathcal{D}} \wedge \mathcal{C}_i \neq \top_{\mathcal{D}}.$$

A dimension with more than one hierarchy represents *multiple hierarchies*:

$$\textbf{Multiple}(D)\colon\ |\mathcal{H}| > 1.$$

A non-hierarchical dimension that has been "stripped off" to a single category with a single attribute is called *degenerate* [Kimball 1996]:

$$\textbf{Degenerate}(D)\colon\ \textbf{Non-hierarchy}(D) \wedge \bot_{\mathcal{D}} = \{\mathcal{A}^{\bot}, \varnothing\}.$$

A degenerate dimension is a *fact identifier*if it uniquely identifies the fact's entries:

$$\mathbf{Identifier}(D)\text{: } \mathbf{Degenerate}(D) \wedge (\forall (e_i, e_j) \in \bot_{\mathcal{D}} : e_1 \neq e_2).$$

InvoiceNumber and PurchaseID are examples of a degenerate and a fact identifier dimension of scheme PURCHASE in Figure 15, respectively.

Hierarchical dimensions can be recursively decomposed into their constituent sub-dimensions by recursively stripping off the bottom-level category. For example, building $\sqsubseteq$ city $\sqsubseteq \top_{\mathsf{project}}$ and city $\sqsubseteq \top_{\mathsf{project}}$ are sub-dimensions of project.

Definition 7.4. Dimension D' is a *sub-dimension* of D if its scheme $\mathcal{D}'$ is a subgraph of $\mathcal{D}$ and each $\sqsubseteq_{H'}$, $H' \in D'$, is a restriction of $\sqsubseteq_H$, $H \in D$, to the corresponding categories.

7.1 Dimension Hierarchy Types

Dimension hierarchies are categorized along two orthogonal properties of *strictness* and *homogeneity.*

Strictness means that each member of a lower level of a hierarchy belongs to only one member of a higher level. A hierarchy is *strict* if it disallows many-to-many cardinalities in its “rolls-up-to” relationships:

$$\mathbf{Strict}(H)\text{:}$$
$$\forall (C_i \sqsubseteq C_j) \in H, \forall e_1 \in C_i, \forall (e_2, e_3) \in C_j : ((e_1 \subseteq e_2 \wedge e_1 \subseteq e_3) \Rightarrow e_2 = e_3).$$

A *non-strict* hierarchy has at least one many-to-many “rolls-up-to” relationship:

$$\mathbf{Non\text{-}strict}(H)\text{:}$$
$$\exists (C_i \sqsubseteq C_j) \in H, \exists e_1 \in C_i, \exists (e_2, e_3) \in C_j : (e_1 \subseteq e_2 \wedge e_1 \subseteq e_3 \wedge e_2 \neq e_3).$$

In our example, such relationship exists between project and project group where a project may be associated with multiple project groups. Non-strict mappings are not summarizable. However, there exist two augmented non-strict hierarchy types that guarantee correct summarization[3]:

—*Weighted non-strict* hierarchy restores summarizability by specifying each element's degree or probability of belonging to each of its parent elements. The relation between project group and section is an example of such mapping, supplied with an obligatory “degree-of-belonging” attribute degree in Figure 15. Further details on this type may be found in [Mansmann and Scholl 2007].

—*Fuzzy* hierarchies is a very special type of non-strict mapping in which child elements are assigned to parent elements dynamically using some rules, so that the belonging relationship may very from one query to another [Laurent 2001]. However, at any single point in time, the mapping is strict. Consider an example of a fuzzy category expensiveness in Figure 15. Members of price are assigned to the members of expensiveness based on complex rules, e.g., analysis the overall price scale of the products already purchased.

Homogeneity of a hierarchy is assessed by testing it for existence of partial containment relationships. Dimension project contains a homogeneous hierarchy project $\sqsubseteq$ manager $\sqsubseteq \top_{\mathsf{project}}$. However, the hierarchy project $\sqsubseteq^{\text{part}}$ (office | city), office $\sqsubseteq$ building $\sqsubseteq$ city $\sqsubseteq \top_{\mathsf{project}}$ is heterogeneous as project values are allowed to roll-up

[3] Neither of these hierarchy types are implemented by standard OLAP systems.

either to office or to city. This behavior is the result of including both internal projects, i.e., with an office location, and external ones situated in other cities.

A hierarchy is *homogeneous* if all "rolls-up-to" relationships within it are full:

$$\textbf{Homogeneous}(H)\text{: } \nexists(C_i, C_j) \in H : C_i \sqsubseteq^{(\text{part})} C_j.$$

A *heterogeneous* hierarchy admits multiple exclusive paths, i.e., related partial "rolls-up-to" relationships, between its categories:

$$\textbf{Heterogeneous}(H)\text{: } \exists(C_i, C_j) \in H : C_i \sqsubseteq^{(\text{part})} C_j.$$

Homogeneous hierarchies should be tested for symmetry and completeness. A hierarchy is *symmetric*, or *onto*, if all its levels are mandatory, i.e., if each non-bottom member has at lease one descendant element at the bottom level:

$$\textbf{Onto}(H)\text{: } (\textbf{Homogeneous}(H) \wedge \\ \forall C_i \in H, Type(C_i) \neq \bot_{\mathcal{D}}, \forall e_1 \in C_i : (\exists e_2, Type(e_2) = \bot_{\mathcal{D}} : e_2 \subseteq^* e_1)).$$

A hierarchy is *asymmetric*, or *non-onto*, if it tolerates childless members in non-bottom categories:

$$\textbf{Non-onto}(H)\text{: } \exists e_1 \in C_i, Type(C_i) \neq \bot_{\mathcal{D}} : (\nexists e_2, Type(e_2) = \bot_{\mathcal{D}} : e_2 \subseteq^* e_1).$$

The hierarchy office $\sqsubseteq$ building $\sqsubseteq$ city is symmetric. A non-onto mapping occurs in administrative staff $\sqsubseteq$ administrative division, as a division may appear to have no staff in purchaser role.

Completeness means that all child-level members fully roll-up to the same parent category and that the extension of the parent category consists of those child members only [Luján-Mora et al. 2006]. The mapping between month and quarter is complete.

Heterogeneous hierarchies occur whenever members of the same category roll-up along different paths. A term *frozen* dimension is introduced in [Hurtado and Mendelzon 2002] to denote minimal homogeneous dimension instances representing different structures, which are implicitly combined in a heterogeneous dimension. Typically, heterogeneous hierarchies result from generalization / specialization relationship between some categories.

Generalized hierarchy contains categories that can be represented by a generalization relationship. The categories at which the alternative paths split and join are called *splitting* and *joining* levels, respectively. Dimension purchaser is an example of a generalized hierarchy with the bottom category purchaser as the splitting level and teaching unit as the joining level. We distinguish between specialization and generalization hierarchies:

—*Generalization* hierarchy uses superclasses for uniting multiple categories to treat their members as one category in part of their common characteristics. For example, teaching unit is introduced to treat the members of chair, faculty, and department as one class. The actual members belong to the subclass categories and the superclass is introduced upon it.

—*Specialization* hierarchy emerges when a category, originally treated as a single class, is divided into subclasses to refine its characteristics for the analysis, as can be observed at the example of staff category, which is subdivided into teaching staff and administrative staff. The actual members belong to the superclass

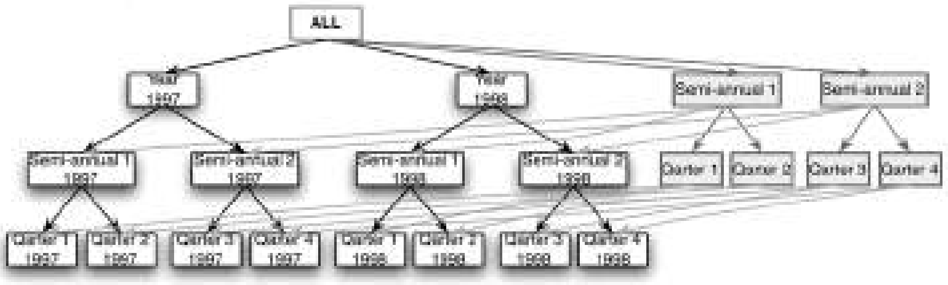


Figure 16. Adding categories semi-annual and quarter (right) to the time hierarchy (left).

and the subclasses are introduced upon it. Specialization may be *disjoint* (each entity belongs to exactly one subclass) or *overlapping*. For example, overlapping subcategories of staff allow the same element to belong to both teaching staff and administrative staff.

A generalized hierarchy is *mixed* if its granularity is non-uniform due to a "rolls-up-to" relationship between subclasses of the same superclass. In our example, the granularity of purchaser is mixed as its subclasses chair, department, and faculty build a hierarchy of their own.

A *non-covering* hierarchy is obtained by allowing roll-up relationships of a category to be partial, so that its members may skip levels. For instance, members of project roll-up to office for internal projects and to city for external ones.

7.2 Types of Multiple Hierarchies

Multiple hierarchies are subdivided into *alternative* and *parallel* according to whether their constituent hierarchies refer to the same or to different analysis criteria, respectively.

Multiple alternative hierarchies are non-exclusive aggregation paths based on the same analysis criterion with at least one shared level in the dimension scheme. Time dimension is a classical example of multiple alternative hierarchies at various granularity levels.

Patterned hierarchies are a special case of multiple alternatives in which a certain regularity, or pattern, re-occurs in the "rolls-up-to" relationships of its members. The classical example is time dimension. Treating time as an ordinary dimension would yield a schema shown in Figure 6. However, recognition of the reoccurring patterns, such as that each year consist of the 1st and the 2nd semi-annual, of the same four quarters, the same 12 months, etc., makes it possible to generate further valid hierarchies. This is done by explicitly representing the type of the patterned category as its parent category. For instance, members of quarters roll up to the category quarter ("Quarter 1 1997" is a child of "Quarter 1"), as illustrated in Figure 16. The new dimension scheme of time supports additional aggregation paths, as shown in Figure 15. A comprehensive framework for identifying frequently occurring dimensional design patterns, such as temporal, action, location, object, stakeholder, qualifier, and combination, is presented in [Jones and Song 2005].

Parallel hierarchies in a dimension account for different analysis criteria, as may

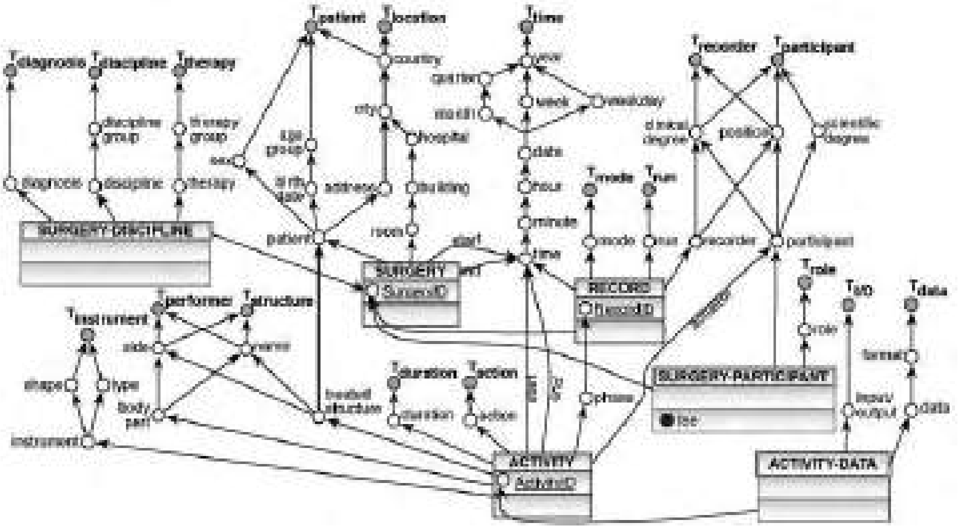


Figure 17. Multidimensional scheme of a surgical process with shared dimension categories.

be observed in project dimension: 1) location hierarchy (office ⊑ building ⊑ city), 2) manager hierarchy, and 3) subject hierarchy (project group ⊑ section).

Multiple hierarchies are *converging*, or *dependent*, if their paths meet at some upper level, as is the case with the time dimension whose paths converge in year.

7.3 Types of Dimension Sharing

Our approach allows to construct a universal data model that comprises multiple facts linked to each other via shared dimensions. The resulting multi-fact scheme, commonly known as a *fact constellation* or a *galaxy*, reveals the true structure of the multidimensional space showing all valid aggregation paths and drill-across options. Moreover, in the logical design phase, shared dimensions can be implemented in a non-redundant fashion, thus facilitating the maintenance of the data warehouse.

The smallest shareable unit is a dimension category type. Our formalization distinguishes between a dimension category type $\mathcal{C}$ and an actual category of that type C, denoted $Type(C) = \mathcal{C}$. This distinction allows us to achieve the highest level of sharing in a whole multidimensional scheme, i.e., inside and across fact schemes, within and between dimensions. To recognize different associations of a category type, the role of each usage may be placed as a label of the respective incoming edge. Figure 17 shows a variant of a surgical process scheme modeled with the highest degree of dimension sharing. We identify three levels of dimension sharing, namely *i*) *full*, *ii*) *partial*, and *iii*) *inclusion*.

Full sharing occurs when a pair of dimensions D and D' have identical dimension schemes ($\mathcal{D} = \mathcal{D}'$). In that case, a single dimension scheme is modeled, whose bottom-level category is referenced by two incoming fact-dimensional relationships. As an example, consider the use of time dimension as start and stop dimensions of SURGERY fact as well as of ACTIVITY fact. Notice how different usages of the same dimension level node are distinguished by means of labeled incoming edges.

A special case of full sharing is a fact scheme acting as a dimension, e.g., fact ACTIVITY as dimension of its satellite fact ACTIVITY-DATA.

Inclusion occurs when the entire dimension scheme of D is contained in the scheme of D' ($\mathcal{D} \subset \mathcal{D}'$), i.e., when D' rolls-up to D. For example, patient dimension of SURGERY is included into treated structure hierarchy of ACTIVITY.

Partial sharing occurs when a pair of dimension schemes $\mathcal{D}$ and $\mathcal{D}'$ converge at a category, non-bottom for either of them ($\exists \mathcal{C}_i \in \mathcal{D} \backslash \{\bot_{\mathcal{D}}\}, \exists \mathcal{C}_j \in \mathcal{D}' \backslash \{\bot_{\mathcal{D}'}\} : \mathcal{C}_i = \mathcal{C}_j$). Partial sharing is possible within the same dimension incase of multiple alternative or parallel hierarchies (year category in time), between two dimensions of the same fact scheme (city category in patient and location of SURGERY), or between two dimensions of different fact schemes (position category in recorder of RECORD and participant of SURGERY-PARTICIPANT).

Notice how presence of distinct top-level categories helps distinguish between seemingly and truly converging paths. The former result in case of category sharing between dimensions. For instance, even though country is the highest aggregation level in both location and participant, each of these dimensions ends at its own top-level node. True path convergence occurs within the same dimension, as in the case of time, where multiple paths converge in year.

Explicit modeling of shared categories at the conceptual level does not impose any particular logical design scheme. On contrary, it is beneficial for generating rich metadata for a proper navigation hierarchy irrespective of the implementation.

8. EVALUATION

We evaluated the concepts proposed in this paper against an extensive set of properties and requirements found in the state-of-the-art literature on multidimensional modeling. The results are presented in Table II. The enumerated multidimensional properties are grouped into five categories (separated by a double line in the table) to enable a more systematic overview, namely, 1) the level of sharing, 2) facts and measures, 3) dimensions and hierarchies, 4) dynamic features, and 5) implementation. The properties chosen for the evaluation essentially correspond to the requirements formulated in Section 4. However, they were adjusted in accordance with insights gained in subsequent formalization sections.

Not surprisingly, our classification and the proposed graphical notation achieve a full coverage of the defined requirements. After all, the objective of this work was to study and systematize a wealth of scattered concepts related to the conceptual data warehouse design, proposed by various researchers in the last years. To the best of our knowledge, our proposed formalization, classification, and graphical notation are the most comprehensive and coherent among the existing ones.

The issue of the implementation remains the subject of future work and has to be preceded by a more exhaustive evaluation and verification.

9. CONCLUSIONS

In this work we presented the results of an exhaustive effort on a systematic summarization and categorization of various extensions of the original multidimensional data model proposed by researchers and practitioners in the recent years. The aim of those extensions is to overcome the rigidness of the model with respect to complex data and non-conventional application domains. First, we formulated the modeling requirements and presented the two element of the conceptual framework – graphical notation and formalization. The terminology was clarified and a formal description of the model's constructs was provided in form of a classification framework. Our proposed classification evolved in two successive phases: 1) classification of fact types, their roles and interrelationships, and 2) classification of dimension and hierarchy types. The classification is concluded by examining the types of dimension sharing in multi-fact schemes.

All presented concepts were supplied with illustrative examples from two real-world applications. Last not least, we proposed a semantically rich graphical notation X-DFM, which extends the existing and widely used Dimensional Fact Model (DFM). The construct set of X-DFM is complete and unambiguous in representing all multidimensional model properties introduced in this paper.

Our future work will be focused on developing a framework for the logical data warehouse design, capable of handling the full set of conceptual extensions described in this work.

REFERENCES

Abelló, A., Samos, J., and Saltor, F. 2001. A framework for the classification and description of multidimensional data models. In *DEXA '01: Proceedings of the 12th International Conference on Database and Expert Systems Applications*. 668–677.

Abelló, A., Samos, J., and Saltor, F. 2001. Understanding facts in a multidimensional object-oriented model. In *DOLAP '01: Proceedings of the 4th ACM international workshop on Data Warehousing and OLAP*. 32–39.

Bauer, A., Ed. 2004. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung*. dpunkt-Verlag, Heidelberg.

Codd, E. F., Codd, S. B., and Salley, C. T. 1993. Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. *Technical report, E.F.Codd & Associates*.

Giovinazzo, W. A. 2000. *Object-oriented data warehouse design: building a star schema*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Golfarelli, M., Maio, D., and Rizzi, S. 1998. The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems 7,* 2-3, 215–247.

Hümmer, W., Lehner, W., Bauer, A., and Schlesinger, L. 2002. A decathlon in multidimensional modeling: Open issues and some solutions. In *DaWaK 2000: Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*. 275–285.

Hurtado, C. A. and Mendelzon, A. O. 2001. Reasoning about summarizability in heterogeneous multidimensional schemas. In *ICDT 2001: Proceedings of the 8th International Conference on Database Theory*. 375–389.

Hurtado, C. A. and Mendelzon, A. O. 2002. OLAP dimension constraints. In *PODS '02: Proceedings of the 21st ACM Symposium on Principles of Database Systems*. 169–179.

Jagadish, H. V., Lakshmanan, L. V. S., and Srivastava, D. 1999. What can hierarchies do for data warehouses? In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*. 530–541.

Jensen, C. S., Kligys, A., Pedersen, T. B., and Timko, I. 2002. Multidimensional data modeling for location-based services. In *GIS '02: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, 55–61.

Jones, M. E. and Song, I.-Y. 2005. Dimensional modeling: identifying, classifying & applying patterns. In *DOLAP '05: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP*. ACM, New York, NY, USA, 29–38.

Kimball, R. 1996. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses.* John Wiley & Sons, Inc., New York, NY, USA.

Laurent, A. 2001. Generating fuzzy summaries from fuzzy multidimensional databases. In *IDA '01: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis.* 24–33.

Lechtenbörger, J. and Vossen, G. 2003. Multidimensional normal forms for data warehouse design. *Information Systems 28,* 5, 415–434.

Lehner, W., Albrecht, J., and Wedekind, H. 1998. Normal forms for multidimensional databases. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management.* 63–72.

Lenz, H.-J. and Shoshani, A. 1997. Summarizability in OLAP and statistical data bases. In *Proceedings of the 9th International Conference on Scientific and Statistical Database Management.* 132–143.

Luján-Mora, S., Trujillo, J., and Song, I.-Y. 2002. Multidimensional modeling with UML package diagrams. In *ER '02: Proceedings of the 21st International Conference on Conceptual Modeling.* LNCS, vol. 2503. Springer, London, UK, 199–213.

Luján-Mora, S., Trujillo, J., and Song, I.-Y. 2006. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering 59*, 725–769.

Malinowski, E. and Zimányi, E. 2004. OLAP hierarchies: A conceptual perspective. In *CAiSE'04: Proceedings of the 16th International Conference on Advanced Information Systems Engineering.* 477–491.

Malinowski, E. and Zimányi, E. 2006. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering 59,* 2, 348–377.

Mansmann, S., Neumuth, T., and Scholl, M. H. 2007a. Multidimensional Data Modeling for Business Process Analysis. In *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling.* LNCS, vol. 4801. Springer, Auckland, New Zealand, 23–38.

Mansmann, S., Neumuth, T., and Scholl, M. H. 2007b. OLAP Technology for Business Process Intelligence: Challenges and Solutions. In *DaWaK'07: Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery.* LNCS. Springer Verlag, 111–122.

Mansmann, S. and Scholl, M. H. 2006. Extending Visual OLAP for Handling Irregular Dimensional Hierarchies. In *DaWaK'06: Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery.* LNCS. Springer Verlag, 95–105.

Mansmann, S. and Scholl, M. H. 2007. Empowering the OLAP Technology to Support Complex Dimension Hierarchies. *International Journal of Data Warehousing and Mining 3,* 4, 31–50.

Niemi, T., Nummenmaa, J., and Thanisch, P. 2001. Logical multidimensional database design for ragged and unbalanced aggregation. In *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses.* 7.1–7.8.

Park, B.-K., Han, H., and Song, I.-Y. 2005. Xml-olap: A multidimensional analysis framework for xml warehouses. In *DaWaK'05: Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery.* LNCS, vol. 3589. Springer, 32–42.

Pedersen, T. B. and Jensen, C. S. 2001. Multidimensional database technology. *IEEE Computer 34,* 12, 40–46.

Pedersen, T. B., Jensen, C. S., and Dyreson, C. E. 2001. A foundation for capturing and querying complex multidimensional data. *Information Systems 26,* 5, 383–423.

Pourabbas, E. and Rafanelli, M. 2000. Hierarchies and relative operators in the olap environment. *SIGMOD Record 29,* 1, 32–37.

Rafanelli, M. and Shoshani, A. 1990. STORM: A statistical object representation model. In *Proceedings of 5th International Conference on Statistical and Scientific Database Management.* 14–29.

Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. 2007. A conceptual model for multidimensional analysis of documents. In *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling.* LNCS, vol. 4801. Springer, Auckland, New Zealand, 550–565.

SONG, I.-Y., ROWEN, W., MEDSKER, C., AND EWEN, E. F. 2001. An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling. In *DMDW'01: Proceedings of the 3^{rd} International Workshop on Design and Management of Data Warehouses.* CEUR Workshop Proceedings, vol. 39. CEUR-WS.org, 6.1–6.13.

TRUJILLO, J., PALOMAR, M., GOMEZ, J., AND SONG, I.-Y. 2001. Designing data warehouses with OO conceptual models. *Computer 34*, 12, 66–75.

VINNIK, S. AND MANSMANN, F. 2006. From analysis to interactive exploration: Building visual hierarchies from OLAP cubes. In *EDBT 2006: Proceedings of the 10^{th} International Conference on Extending Database Technology.* 496–514.

ZUBEK, T. AND SINNWELL, M. 1999. Datawarehousing has more colours than just black & white. In *VLDB '99: Proceedings of 25^{th} International Conference on Very Large Data Bases.* 726–729.

Svetlana Mansmann received a Diploma in International Economic Relations from the Belarusian State University in 1999, and the M.Sc. degree in information engineering from the University of Konstanz (Germany) in 2003. She is currently completing a Ph.D. degree in computer science, working as a Research Assistant in the Databases and Information Systems Group at the University of Konstanz. Previous research fields are DSS and e-Government. Recent publications focus on data warehousing, business process intelligence, and visual analysis. She received best paper awards from ICECE 2005 (International Conference on Engineering and Computer Education), DaWaK 2006, and DaWaK 2007 (International Conference on Data Warehousing and Knowledge Discovery). Mrs. Mansmann is an associate member of the Phd Program "Explorative Analysis and Exploration of Large Information Spaces" at the University of Konstanz.

Marc H. Scholl received his M.Sc. (Dipl.-Inform.) and Ph.D. (Dr.-Ing.) degrees in computer science from the Technical University of Darmstadt, Germany, in 1982 and 1988, respectively. He is currently a full Professor of Computer Science at the University of Konstanz, Germany. From 1998 to 2004 he served as Vice-President of the University, responsible for the information infrastructure. Previous positions held include an Associate Professorship at the University of Ulm, Germany (1992-94) and an Assistant Professorship at ETH Zurich, Switzerland (1989-92). Current research topics include XML and databases, query processing, DBMS architecture, digital libraries, data warehousing, and e-Government. Prof. Scholl is a member of IEEE Computer Society, ACM and ACM SIGMOD, EDBT (currently EDBT President), the ICDT Council, and the German Computer Society (GI).

Table II. Evaluating *X*-DFM against multidimensional modeling requirements.

Property	**Support**	**Explanation**
Sharing of elements	+	Explicit sharing at the dimensional scheme node level within and across dimensions and facts
Fact constellations / galaxies	+	Realized via element sharing at the level of dimension category types
Many-to-many fact-dimension mapping	+	Extracted into satellite facts
Degenerate facts	+	A special case of a satellite fact
Non-measurable facts	+	Empty set off measures, default measure `COUNT(*)` is added
Fact in a dimension role	+	Fact scheme viewed as parallel hierarchies
Fact rollup	+	Special case of fact in a dimension role
Fact generalization	+	Special case of a fact constellation
Fact association	+	Multiple facts in dimension roles
Atomic measures	+	Provided by DFM
Derived measures	+	Stored as a formula referencing its input atomic measures
Additivity	+	Provided by DFM
Degenerate dimensions	+	Category node residing within the fact
Fact identifier dimensions	+	Special case of degenerate dimension
Derived dimensions	+	Modeled explicitly, with derived categories linked to their input categories
Partial rollups	+	Dashed edge arrows
Related partial rollups	+	Dashed-line edge arrows with a merged end
Dimension roles	+	Shared dimension level node with labeled edges
Sub-dimension sharing	+	Special case of element sharing
Multiple alternative hierarchies	+	Outgoing roll-up edges with a merged end
Parallel hierarchies	+	Provided by DFM
Non-strict hierarchies	+	Edge augmented by cardinality information
Generalized hierarchies	+	Common superclass dimension category
Dynamic measure specification from arbitrary attribute(s)	+	Generation of the accordingly adjusted version of the fact scheme
Measure used as a dimension	+	Generation of the accordingly adjusted version of the fact scheme
Dimension attributes in measure role	+	Special case of dynamic measure specification
Drill-across	+	Partially provided by DFM as fact scheme overlap, improved via sharing of elements
Graphical notation	+	Original DFM with modifications and extensions
Implementation	–	Future work