IDEA GROUP PUBLISHING **ITB13727**

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

This paper appears in the publication, **Data Warehouses and OLAP: Concepts, Architectures and Solutions**
edited by R. Wrembel; C. Koncilia © 2007, Idea Group Inc.

Chapter I

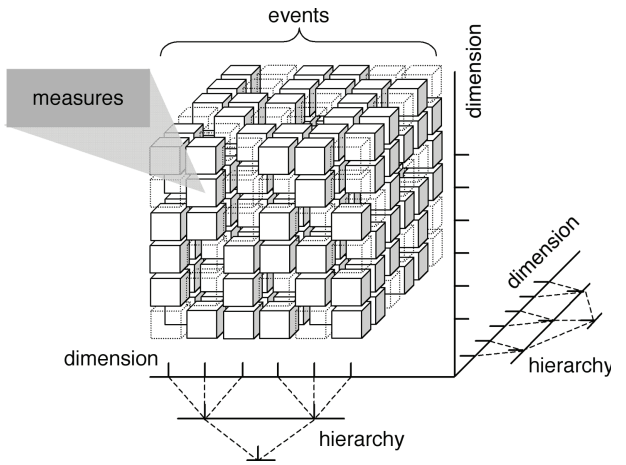
Conceptual Modeling Solutions for the Data Warehouse

Stefano Rizzi
DEIS-University of Bologna, Italy

Abstract

In the context of data warehouse design, a basic role is played by conceptual modeling, that provides a higher level of abstraction in describing the warehousing process and architecture in all its aspects, aimed at achieving independence of implementation issues. This chapter focuses on a conceptual model called the DFM that suits the variety of modeling situations that may be encountered in real projects of small to large complexity. The aim of the chapter is to propose a comprehensive set of solutions for conceptual modeling according to the DFM and to give the designer a practical guide for applying them in the context of a design methodology. Besides the basic concepts of multidimensional modeling, the other issues discussed are descriptive and cross-dimension attributes; convergences; shared, incomplete, recursive, and dynamic hierarchies; multiple and optional arcs; and additivity.

Figure 1. The cube metaphor for multidimensional modeling



Introduction

Operational databases are focused on recording transactions, thus they are prevalently characterized by an OLTP (online transaction processing) workload. Conversely, data warehouses (DWs) allow complex analysis of data aimed at decision support; the workload they support has completely different characteristics, and is widely known as OLAP (online analytical processing). Traditionally, OLAP applications are based on *multidimensional modeling* that intuitively represents data under the metaphor of a cube whose cells correspond to events that occurred in the business domain (Figure 1). Each event is quantified by a set of measures; each edge of the cube corresponds to a relevant dimension for analysis, typically associated to a hierarchy of attributes that further describe it. The multidimensional model has a twofold benefit. On the one hand, it is close to the way of thinking of data analyzers, who are used to the spreadsheet metaphor; therefore it helps users understand data. On the other hand, it supports performance improvement as its simple structure allows designers to predict the user intentions.

Multidimensional modeling and OLAP workloads require specialized design techniques. In the context of design, a basic role is played by *conceptual modeling* that provides a higher level of abstraction in describing the warehousing process and architecture in all its aspects, aimed at achieving independence of implementation issues. Conceptual modeling is widely recognized to be the necessary foundation for building a database that is well-documented and fully satisfies the user require-

ments; usually, it relies on a graphical notation that facilitates writing, understanding, and managing conceptual schemata by both designers and users.

Unfortunately, in the field of data warehousing there still is no consensus about a formalism for conceptual modeling (Sen & Sinha, 2005). The entity/relationship (E/R) model is widespread in the enterprises as a conceptual formalism to provide standard documentation for relational information systems, and a great deal of effort has been made to use E/R schemata as the input for designing nonrelational databases as well (Fahrner & Vossen, 1995); nevertheless, as E/R is oriented to support queries that navigate associations between data rather than synthesize them, it is not well suited for data warehousing (Kimball, 1996). Actually, the E/R model has enough expressivity to represent most concepts necessary for modeling a DW; on the other hand, in its basic form, it is not able to properly emphasize the key aspects of the multidimensional model, so that its usage for DWs is expensive from the point of view of the graphical notation and not intuitive (Golfarelli, Maio, & Rizzi, 1998).

Some designers claim to use star schemata for conceptual modeling. A *star schema* is the standard implementation of the multidimensional model on relational platforms; it is just a (denormalized) relational schema, so it merely defines a set of relations and integrity constraints. Using the star schema for conceptual modeling is like starting to build a complex software by writing the code, without the support of and static, functional, or dynamic model, which typically leads to very poor results from the points of view of adherence to user requirements, of maintenance, and of reuse.

For all these reasons, in the last few years the research literature has proposed several original approaches for modeling a DW, some based on extensions of E/R, some on extensions of UML. This chapter focuses on an ad hoc conceptual model, the *dimensional fact model* (DFM), that was first proposed in Golfarelli et al. (1998) and continuously enriched and refined during the following years in order to optimally suit the variety of modeling situations that may be encountered in real projects of small to large complexity. The aim of the chapter is to propose a comprehensive set of solutions for conceptual modeling according to the DFM and to give a practical guide for applying them in the context of a design methodology. Besides the basic concepts of multidimensional modeling, namely facts, dimensions, measures, and hierarchies, the other issues discussed are descriptive and cross-dimension attributes; convergences; shared, incomplete, recursive, and dynamic hierarchies; multiple and optional arcs; and additivity.

After reviewing the related literature in the next section, in the third and fourth sections, we introduce the constructs of DFM for basic and advanced modeling, respectively. Then, in the fifth section we briefly discuss the different methodological approaches to conceptual design. Finally, in the sixth section we outline the open issues in conceptual modeling, and in the last section we draw the conclusions.

Related Literature

In the context of data warehousing, the literature proposed several approaches to multidimensional modeling. Some of them have no graphical support and are aimed at establishing a formal foundation for representing cubes and hierarchies as well as an algebra for querying them (Agrawal, Gupta, & Sarawagi, 1995; Cabibbo & Torlone, 1998; Datta & Thomas, 1997; Franconi & Kamble, 2004a; Gyssens & Lakshmanan, 1997; Li & Wang, 1996; Pedersen & Jensen, 1999; Vassiliadis, 1998); since we believe that a distinguishing feature of conceptual models is that of providing a graphical support to be easily understood by both designers and users when discussing and validating requirements, we will not discuss them.

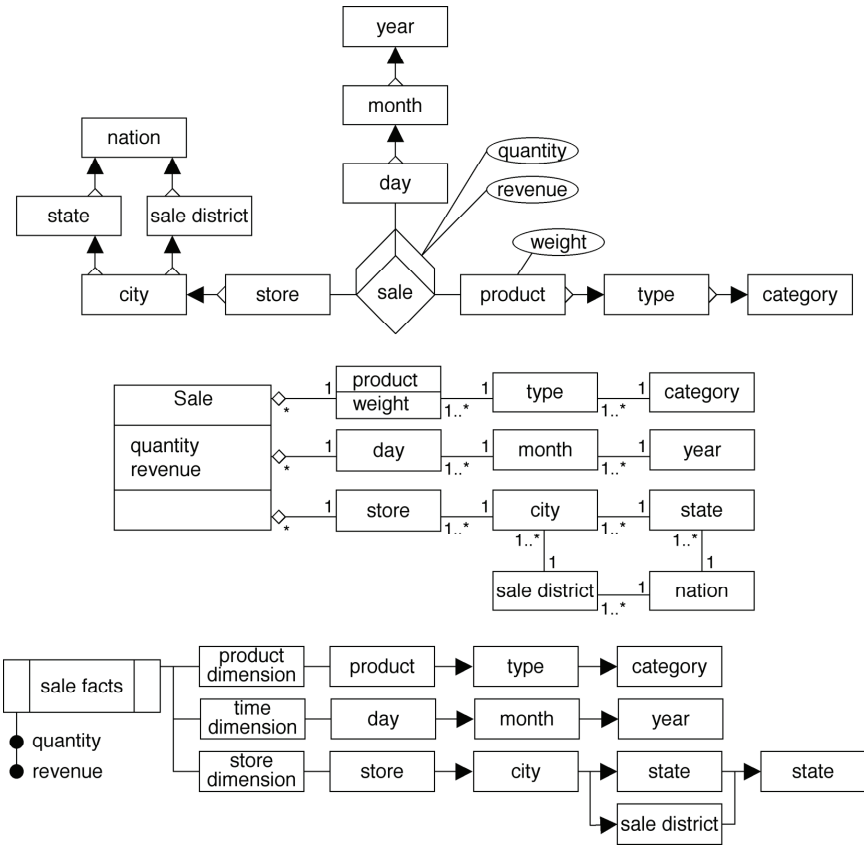
The approaches to "strict" conceptual modeling for DWs devised so far are summarized in Table 1. For each model, the table shows if it is associated to some method for conceptual design and if it is based on E/R, is object-oriented, or is an ad hoc model.

The discussion about whether E/R-based, object-oriented, or ad hoc models are preferable is controversial. Some claim that E/R extensions should be adopted since (1) E/R has been tested for years; (2) designers are familiar with E/R; (3) E/R has proven flexible and powerful enough to adapt to a variety of application domains; and (4) several important research results were obtained for the E/R (Sapia, Blaschka, Hofling, & Dinter, 1998; Tryfona, Busborg, & Borch Christiansen, 1999). On the other hand, advocates of object-oriented models argue that (1) they are more expressive and better represent static and dynamic properties of information systems; (2) they provide powerful mechanisms for expressing requirements and constraints; (3) object-orientation is currently the dominant trend in data modeling; and (4) UML, in particular, is a standard and is naturally extensible (Abelló, Samos, & Saltor, 2002; Luján-Mora, Trujillo, & Song, 2002). Finally, we believe that ad hoc models compensate for the lack of familiarity from designers with the fact that (1) they achieve better notational economy; (2) they give proper emphasis to the peculiarities of the multidimensional model, thus (3) they are more intuitive and

Table 1. Approaches to conceptual modeling

	E/R extension	object-oriented	ad hoc
no method	Franconi and Kamble (2004b); Sapia et al. (1998); Tryfona et al. (1999)	Abelló et al. (2002); Nguyen, Tjoa, and Wagner (2000)	Tsois et al. (2001)
method		Luján-Mora et al. (2002)	Golfarelli et al. (1998); Hüsemann et al. (2000)

Figure 2. The SALE fact modeled through a starER (Sapia et al., 1998), a UML class diagram (Luján-Mora et al., 2002), and a fact schema (Hüsemann, Lechtenbörger, & Vossen, 2000)



readable by nonexpert users. In particular, they can model some constraints related to functional dependencies (e.g., convergences and cross-dimensional attributes) in a simpler way than UML, that requires the use of formal expressions written, for instance, in OCL.

A comparison of the different models done by Tsois, Karayannidis, and Sellis (2001) pointed out that, abstracting from their graphical form, the core expressivity is similar. In confirmation of this, we show in Figure 2 how the same simple fact could be modeled through an E/R based, an object-oriented, and an ad hoc approach.

The Dimensional Fact Model: Basic Modeling

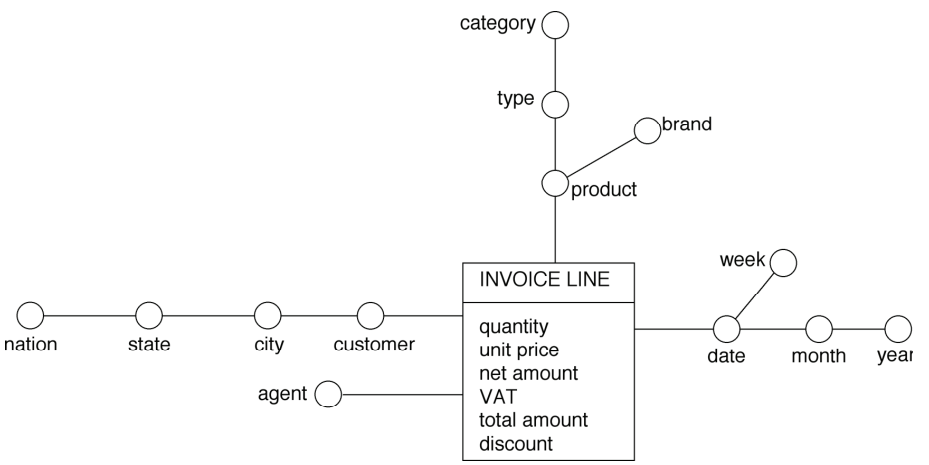
In this chapter we focus on an ad hoc model called the dimensional fact model. The DFM is a graphical conceptual model, specifically devised for multidimensional modeling, aimed at:

- Effectively supporting conceptual design
- Providing an environment on which user queries can be intuitively expressed
- Supporting the dialogue between the designer and the end users to refine the specification of requirements
- Creating a stable platform to ground logical design
- Providing an expressive and non-ambiguous design documentation

The representation of reality built using the DFM consists of a set of *fact schemata*. The basic concepts modeled are facts, measures, dimensions, and hierarchies. In the following we intuitively define these concepts, referring the reader to Figure 3 that depicts a simple fact schema for modeling invoices at line granularity; a formal definition of the same concepts can be found in Golfarelli et al. (1998).

Definition 1: A *fact* is a focus of interest for the decision-making process; typically, it models a set of events occurring in the enterprise world. A

Figure 3. A basic fact schema for the INVOICE LINE fact



fact is graphically represented by a box with two sections, one for the fact name and one for the measures.

Examples of facts in the trade domain are sales, shipments, purchases, claims; in the financial domain: stock exchange transactions, contracts for insurance policies, granting of loans, bank statements, credit cards purchases. It is essential for a fact to have some *dynamic* aspects, that is, to evolve somehow across time.

Guideline 1: The concepts represented in the data source by frequently-updated archives are good candidates for facts; those represented by almost-static archives are not.

As a matter of fact, very few things are completely static; even the relationship between cities and regions might change, if some border were revised. Thus, the choice of facts should be based either on the average periodicity of changes, or on the specific interests of analysis. For instance, assigning a new sales manager to a sales department occurs less frequently than coupling a promotion to a product; thus, while the relationship between promotions and products is a good candidate to be modeled as a fact, that between sales managers and departments is not—except for the personnel manager, who is interested in analyzing the turnover!

Definition 2: A *measure* is a numerical property of a fact, and describes one of its quantitative aspects of interests for analysis. Measures are included in the bottom section of the fact.

For instance, each invoice line is measured by the number of units sold, the price per unit, the net amount, and so forth. The reason why measures should be numerical is that they are used for computations. A fact may also have no measures, if the only interesting thing to be recorded is the occurrence of events; in this case the fact scheme is said to be *empty* and is typically queried to count the events that occurred.

Definition 3: A *dimension* is a fact property with a finite domain and describes one of its analysis coordinates. The set of dimensions of a fact determines its finest representation granularity. Graphically, dimensions are represented as circles attached to the fact by straight lines.

Typical dimensions for the invoice fact are product, customer, agent, and date.

Guideline 2: At least one of the dimensions of the fact should represent time, at any granularity.

The relationship between measures and dimensions is expressed, at the instance level, by the concept of event.

Definition 4: A *primary event* is an occurrence of a fact, and is identified by a tuple of values, one for each dimension. Each primary event is described by one value for each measure.

Primary events are the elemental information which can be represented (in the cube metaphor, they correspond to the cube cells). In the invoice example they model the invoicing of one product to one customer made by one agent on one day; it is not possible to distinguish between invoices possibly made with different types (e.g., active, passive, returned, etc.) or in different hours of the day.

Guideline 3: If the granularity of primary events as determined by the set of dimensions is coarser than the granularity of tuples in the data source, measures should be defined as either aggregations of numerical attributes in the data source, or as counts of tuples.

Remarkably, some multidimensional models in the literature focus on treating dimensions and measures symmetrically (Agrawal et al., 1995; Gyssens & Lakshmanan, 1997). This is an important achievement from both the point of view of the uniformity of the logical model and that of the flexibility of OLAP operators. Nevertheless we claim that, at a conceptual level, distinguishing between measures and dimensions is important since it allows logical design to be more specifically aimed at the efficiency required by data warehousing applications.

Aggregation is the basic OLAP operation, since it allows significant information useful for decision support to be summarized from large amounts of data. From a conceptual point of view, aggregation is carried out on primary events thanks to the definition of dimension attributes and hierarchies.

Definition 5: A *dimension attribute* is a property, with a finite domain, of a dimension. Like dimensions, it is represented by a circle.

For instance, a product is described by its type, category, and brand; a customer, by its city and its nation. The relationships between dimension attributes are expressed by hierarchies.

Definition 6: A *hierarchy* is a directed tree, rooted in a dimension, whose nodes are all the dimension attributes that describe that dimension, and whose arcs model many-to-one associations between pairs of dimension attributes. Arcs are graphically represented by straight lines.

Guideline 4: Hierarchies should reproduce the pattern of interattribute functional dependencies expressed by the data source.

Hierarchies determine how primary events can be aggregated into secondary events and selected significantly for the decision-making process. The dimension in which a hierarchy is rooted defines its finest aggregation granularity, while the other dimension attributes define progressively coarser granularities. For instance, thanks to the existence of a many-to-one association between products and their categories, the invoicing events may be grouped according to the category of the products.

Definition 7: Given a set of dimension attributes, each tuple of their values identifies a *secondary event* that aggregates all the corresponding primary events. Each secondary event is described by a value for each measure that summarizes the values taken by the same measure in the corresponding primary events.

We close this section by surveying some alternative terminology used either in the literature or in the commercial tools. There is substantial agreement on using the term *dimensions* to designate the "entry points" to classify and identify events; while we refer in particular to the attribute determining the minimum fact granularity, sometimes the whole hierarchies are named as dimensions (for instance, the term "time dimension" often refers to the whole hierarchy built on dimension *date*). Measures are sometimes called *variables* or *metrics*. Finally, in some data warehousing tools, the term *hierarchy* denotes each single branch of the tree rooted in a dimension.

The Dimensional Fact Model: Advanced Modeling

The constructs we introduce in this section, with the support of Figure 4, are descriptive and cross-dimension attributes; convergences; shared, incomplete, recursive,

and dynamic hierarchies; multiple and optional arcs; and additivity. Though some of them are not necessary in the simplest and most common modeling situations, they are quite useful in order to better express the multitude of conceptual shades that characterize real-world scenarios. In particular we will see how, following the introduction of some of this constructs, hierarchies will no longer be defined as trees to become, in the general case, directed graphs.

Descriptive Attributes

In several cases it is useful to represent additional information about a dimension attribute, though it is not interesting to use such information for aggregation. For instance, the user may ask for knowing the address of each store, but the user will hardly be interested in aggregating sales according to the address of the store.

> **Definition 8:** A *descriptive attribute* specifies a property of a dimension attribute, to which is related by an *x*-to-one association. Descriptive attributes are not used for aggregation; they are always leaves of their hierarchy and are graphically represented by horizontal lines.

There are two main reasons why a descriptive attribute should not be used for aggregation:

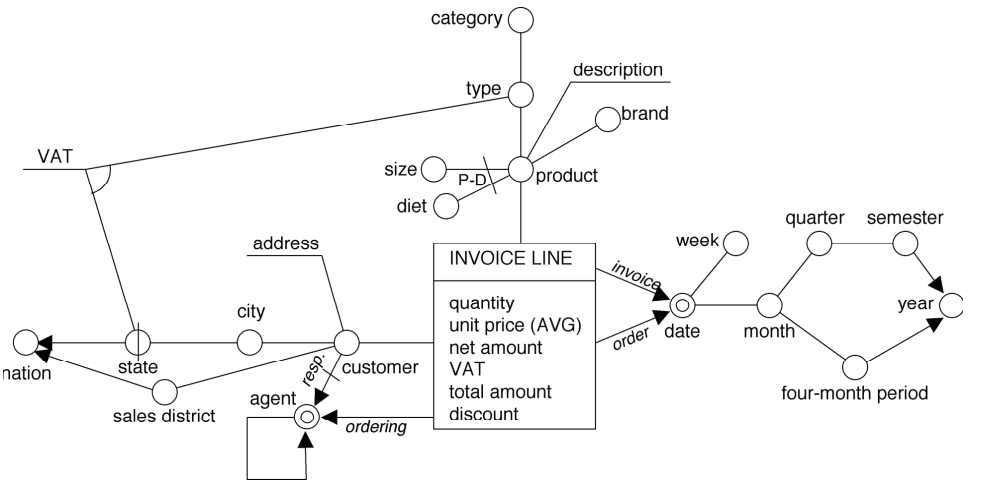
> **Guideline 5:** A descriptive attribute either has a continuously-valued domain (for instance, the weight of a product), or is related to a dimension attribute by a one-to-one association (for instance, the address of a customer).

Cross-Dimension Attributes

> **Definition 9:** A *cross-dimension attribute* is a (either dimension or descriptive) attribute whose value is determined by the combination of two or more dimension attributes, possibly belonging to different hierarchies. It is denoted by connecting through a curve line the arcs that determine it.

For instance, if the VAT on a product depends on both the product category and the state where the product is sold, it can be represented by a cross-dimension attribute as shown in Figure 4.

Figure 4. The complete fact schema for the INVOICE LINE fact



Convergence

Consider the geographic hierarchy on dimension *customer* (Figure 4): customers live in cities, which are grouped into states belonging to nations. Suppose that customers are grouped into sales districts as well, and that no inclusion relationships exist between districts and cities/states; on the other hand, sales districts never cross the nation boundaries. In this case, each customer belongs to exactly one nation whichever of the two paths is followed (customer → city → state → nation or customer → sales district → nation).

> **Definition 10:** A *convergence* takes place when two dimension attributes within a hierarchy are connected by two or more alternative paths of many-to-one associations. Convergences are represented by letting two or more arcs converge on the same dimension attribute.

The existence of apparently equal attributes does not always determine a convergence. If in the invoice fact we had a brand city attribute on the product hierarchy, representing the city where a brand is manufactured, there would be no convergence with attribute (customer) city, since a product manufactured in a city can obviously be sold to customers of other cities as well.

Optional Arcs

Definition 11: An *optional arc* models the fact that an association represented within the fact scheme is undefined for a subset of the events. An optional arc is graphically denoted by marking it with a dash.

For instance, attribute diet takes a value only for food products; for the other products, it is undefined.

In the presence of a set of optional arcs exiting from the same dimension attribute, their *coverage* can be denoted in order to pose a constraint on the optionalities involved. Like for IS-A hierarchies in the E/R model, the coverage of a set of optional arcs is characterized by two independent coordinates. Let *a* be a dimension attribute, and b_1,..., b_m be its children attributes connected by optional arcs:

- The coverage is *total* if each value of *a* always corresponds to a value for at least one of its children; conversely, if some values of *a* exist for which all of its children are undefined, the coverage is said to be *partial*.
- The coverage is *disjoint* if each value of *a* corresponds to a value for, at most, one of its children; conversely, if some values of *a* exist that correspond to values for two or more children, the coverage is said to be *overlapped*.

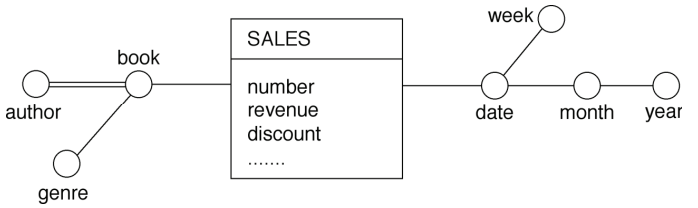
Thus, overall, there are four possible coverages, denoted by T-D, T-O, P-D, and P-O. Figure 4 shows an example of optionality annotated with its coverage. We assume that products can have three types: food, clothing, and household, since expiration date and size are defined only for, respectively, food and clothing, the coverage is partial and disjoint.

Multiple Arcs

In most cases, as already said, hierarchies include attributes related by many-to-one associations. On the other hand, in some situations it is necessary to include also attributes that, for a single value taken by their father attribute, take several values.

Definition 12: A *multiple arc* is an arc, within a hierarchy, modeling a many-to-many association between the two dimension attributes it connects. Graphically, it is denoted by doubling the line that represents the arc.

Figure 5. The fact schema for the SALES fact



Consider the fact schema modeling the sales of books in a library, represented in Figure 5, whose dimensions are *date* and book. Users will probably be interested in analyzing sales for each book author; on the other hand, since some books have two or more authors, the relationship between book and author must be modeled as a multiple arc.

> **Guideline 6:** In presence of many-to-many associations, summarizability is no longer guaranteed, unless the multiple arc is properly *weighted*. Multiple arcs should be used sparingly since, in ROLAP logical design, they require complex solutions.

Summarizability is the property of correcting summarizing measures along hierarchies (Lenz & Shoshani, 1997). Weights restore summarizability, but their introduction is artificial in several cases; for instance, in the book sales fact, each author of a multiauthored book should be assigned a normalized weight expressing her "contribution" to the book.

Shared Hierarchies

Sometimes, large portions of hierarchies are replicated twice or more in the same fact schema. A typical example is the temporal hierarchy: a fact frequently has more than one dimension of type date, with different semantics, and it may be useful to define on each of them a temporal hierarchy month-week-year. Another example are geographic hierarchies, that may be defined starting from any location attribute in the fact schema. To avoid redundancy, the DFM provides a graphical shorthand for denoting hierarchy sharing. Figure 4 shows two examples of shared hierarchies. Fact INVOICE LINE has two date dimensions, with semantics invoice date and order date,

respectively. This is denoted by doubling the circle that represents attribute date and specifying two *roles* invoice and order on the entering arcs. The second shared hierarchy is the one on agent, that may have two roles: the ordering agent, that is a dimension, and the agent who is responsible for a customer (optional).

Guideline 8: Explicitly representing shared hierarchies on the fact schema is important since, during ROLAP logical design, it enables ad hoc solutions aimed at avoiding replication of data in dimension tables.

Ragged Hierarchies

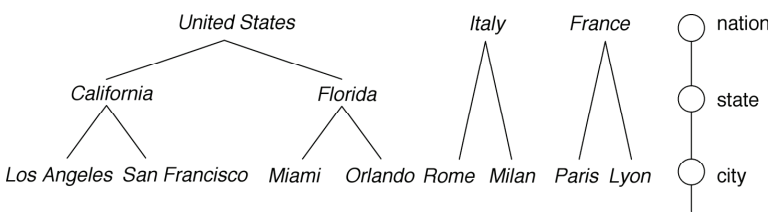
Let $a_1,..., a_n$ be a sequence of dimension attributes that define a path within a hierarchy (such as city, state, nation). Up to now we assumed that, for each value of a_1, exactly one value for every other attribute on the path exists. In the previous case, this is actually true for each city in the U.S., while it is false for most European countries where no decomposition in states is defined (see Figure 6).

Definition 13: A *ragged* (or *incomplete*) *hierarchy* is a hierarchy where, for some instances, the values of one or more attributes are missing (since undefined or unknown). A ragged hierarchy is graphically denoted by marking with a dash the attributes whose values may be missing.

As stated by Niemi (2001), within a ragged hierarchy each aggregation level has precise and consistent semantics, but the different hierarchy instances may have different length since one or more levels are missing, making the interlevel relationships not uniform (the father of "San Francisco" belongs to level state, the father of "Rome" to level nation).

There is a noticeable difference between a ragged hierarchy and an optional arc. In the first case we model the fact that, for some hierarchy instances, there is no value for one or more attributes *in any position of the hierarchy*. Conversely, through an

Figure 6. Ragged geographic hierarchies



optional arc we model the fact that there is no value for an attribute *and for all of its descendents*.

> **Guideline 9:** Ragged hierarchies may lead to summarizability problems. A way for avoiding them is to fragment a fact into two or more facts, each including a subset of the hierarchies characterized by uniform interlevel relationships.

Thus, in the invoice example, fragmenting INVOICE LINE into U.S. INVOICE LINE and E.U. INVOICE LINE (the first with the state attribute, the second without state) restores the completeness of the geographic hierarchy.

Unbalanced Hierarchies

> **Definition 14:** An *unbalanced* (or *recursive*) *hierarchy* is a hierarchy where, though interattribute relationships are consistent, the instances may have different length. Graphically, it is represented by introducing a cycle within the hierarchy.

A typical example of unbalanced hierarchy is the one that models the dependence interrelationships between working persons. Figure 4 includes an unbalanced hierarchy on sale agents: there are no fixed roles for the different agents, and the different "leaf" agents have a variable number of supervisor agents above them.

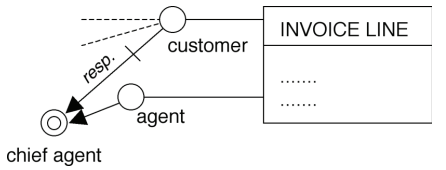
> **Guideline 10:** Recursive hierarchies lead to complex solutions during ROLAP logical design and to poor querying performance. A way for avoiding them is to "unroll" them for a given number of times.

For instance, in the agent example, if the user states that two is the maximum number of interesting levels for the dependence relationship, the customer hierarchy could be transformed as in Figure 7.

Dynamic Hierarchies

Time is a key factor in data warehousing systems, since the decision process is often based on the evaluation of historical series and on the comparison between snapshots of the enterprise taken at different moments. The multidimensional models implicitly assume that the only dynamic components described in a cube are the events that

Figure 7. Unrolling the agent hierarchy



instantiate it; hierarchies are traditionally considered to be static. Of course this is not correct: sales manager alternate, though slowly, on different departments; new products are added every week to those already being sold; the product categories change, and their relationship with products change; sales districts can be modified, and a customer may be moved from one district to another.[1]

The conceptual representation of hierarchy dynamicity is strictly related to its impact on user queries. In fact, in presence of a dynamic hierarchy we may picture three different temporal scenarios for analyzing events (SAP, 1998):

- **Today for yesterday:** All events are referred to the current configuration of hierarchies. Thus, assuming on January 1, 2005 the responsible agent for customer Smith has changed from Mr. Black to Mr. White, and that a new customer O'Hara has been acquired and assigned to Mr. Black, when computing the agent commissions all invoices for Smith are attributed to Mr. White, while only invoices for O'Hara are attributed to Mr. Black.
- **Yesterday for today:** All events are referred to some past configuration of hierarchies. In the previous example, all invoices for Smith are attributed to Mr. Black, while invoices for O'Hara are not considered.
- **Today or yesterday (or historical truth):** Each event is referred to the configuration hierarchies had at the time the event occurred. Thus, the invoices for Smith up to 2004 and those for O'Hara are attributed to Mr. Black, while invoices for Smith from 2005 are attributed to Mr. White.

While in the agent example, dynamicity concerns an arc of a hierarchy, the one expressing the many-to-one association between customer and agent, in some cases it may as well concern a dimension attribute: for instance, the name of a product category may change. Even in this case, the different scenarios are defined in much the same way as before.

On the conceptual schema, it is useful to denote which scenarios the user is interested for each arc and attribute, since this heavily impacts on the specific solutions to be adopted during logical design. By default, we will assume that the only interesting scenario is today for yesterday—it is the most common one, and the one whose

Table 2. Temporal scenarios for the INVOICE fact

arc/attribute	today for yesterday	yesterday for today	today or yesterday
customer-resp. agent	YES	YES	YES
customer-city	YES		YES
sale district			YES

Table 3. Valid aggregation operators for the three types of measures (Lenz, 1997)

	temporal hierarchies	nontemporal hierarchies
flow measures	SUM, AVG, MIN, MAX	SUM, AVG, MIN, MAX
stock measures	AVG, MIN, MAX	SUM, AVG, MIN, MAX
unit measures	AVG, MIN, MAX	AVG, MIN, MAX

implementation on the star schema is simplest. If some attributes or arcs require different scenarios, the designer should specify them on a table like Table 2.

Additivity

Aggregation requires defining a proper operator to compose the measure values characterizing primary events into measure values characterizing each secondary event. From this point of view, we may distinguish three types of measures (Lenz & Shoshani, 1997):

- **Flow measures:** They refer to a time period, and are cumulatively evaluated at the end of that period. Examples are the number of products sold in a day, the monthly revenue, the number of those born in a year.
- **Stock measures:** They are evaluated at particular moments in time. Examples are the number of products in a warehouse, the number of inhabitants of a city, the temperature measured by a gauge.
- **Unit measures:** They are evaluated at particular moments in time, but they are expressed in relative terms. Examples are the unit price of a product, the discount percentage, the exchange rate of a currency.

The aggregation operators that can be used on the three types of measures are summarized in Table 3.

Definition 15: A measure is said to be *additive* along a dimension if its values can be aggregated along the corresponding hierarchy by the sum operator, otherwise it is called *nonadditive*. A nonadditive measure is *nonaggregable* if no other aggregation operator can be used on it.

Table 3 shows that, in general, flow measures are additive along all dimensions, stock measures are nonadditive along temporal hierarchies, and unit measures are nonadditive along all dimensions.

On the invoice scheme, most measures are additive. For instance, quantity has flow type: the total quantity invoiced in a month is the sum of the quantities invoiced in the single days of that month. Measure unit price has unit type and is nonadditive along all dimensions. Though it cannot be summed up, it can still be aggregated by using operators such as average, maximum, and minimum.

Since additivity is the most frequent case, in order to simplify the graphic notation in the DFM, only the exceptions are represented explicitly. In particular, a measure is connected to the dimensions along which it is nonadditive by a dashed line labeled with the other aggregation operators (if any) which can be used instead. If a measure is aggregated through the same operator along all dimensions, that operator can be simply reported on its side (see for instance unit price in Figure 4).

Approaches to Conceptual Design

In this section we discuss how conceptual design can be framed within a methodology for DW design. The approaches to DW design are usually classified in two categories (Winter & Strauch, 2003):

- Data-driven (or supply-driven) approaches that design the DW starting from a detailed analysis of the data sources; user requirements impact on design by allowing the designer to select which chunks of data are relevant for decision making and by determining their structure according to the multidimensional model (Golfarelli et al., 1998; Hüsemann et al., 2000).
- Requirement-driven (or demand-driven) approaches start from determining the information requirements of end users, and how to map these requirements onto the available data sources is investigated only *a posteriori* (Prakash & Gosain, 2003; Schiefer, List & Bruckner, 2002).

While data-driven approaches somehow simplify the design of ETL (extraction, transformation, and loading), since each data in the DW is rooted in one or more attributes

of the sources, they give user requirements a secondary role in determining the information contents for analysis, and give the designer little support in identifying facts, dimensions, and measures. Conversely, requirement-driven approaches bring user requirements to the foreground, but require a larger effort when designing ETL.

Data-Driven Approaches

Data-driven approaches are feasible when all of the following are true: (1) detailed knowledge of data sources is available *a priori* or easily achievable; (2) the source schemata exhibit a good degree of normalization; (3) the complexity of source schemata is not high. In practice, when the chosen architecture for the DW relies on a *reconciled level* (or *operational data store*) these requirements are largely satisfied: in fact, normalization and detailed knowledge are guaranteed by the source integration process. The same holds, thanks to a careful source recognition activity, in the frequent case when the source is a single relational database, well-designed and not very large.

In a data-driven approach, requirement analysis is typically carried out informally, based on simple requirement glossaries (Lechtenbörger, 2001) rather than on formal diagrams. Conceptual design is then heavily rooted on source schemata and can be largely automated. In particular, the designer is actively supported in identifying dimensions and measures, in building hierarchies, in detecting convergences and shared hierarchies. For instance, the approach proposed by Golfarelli et al. (1998) consists of five steps that, starting from the source schema expressed either by an E/R schema or a relational schema, create the conceptual schema for the DW:

1. Choose facts of interest on the source schema
2. For each fact, build an *attribute tree* that captures the functional dependencies expressed by the source schema
3. Edit the attribute trees by adding/deleting attributes and functional dependencies
4. Choose dimensions and measures
5. Create the fact schemata

While step 2 is completely automated, some advanced constructs of the DFM are manually applied by the designer during step 5.

On-the-field experience shows that, when applicable, the data-driven approach is preferable since it reduces the overall time necessary for design. In fact, not only conceptual design can be partially automated, but even ETL design is made easier since the mapping between the data sources and the DW is derived at no additional cost during conceptual design.

Requirement-Driven Approaches

Conversely, within a requirement-driven framework, in the absence of knowledge of the source schema, the building of hierarchies cannot be automated; the main assurance of a satisfactory result is the skill and experience of the designer, and the designer's ability to interact with the domain experts. In this case it may be worth adopting formal techniques for specifying requirements in order to more accurately capture users' needs; for instance, the goal-oriented approach proposed by Giorgini, Rizzi, and Garzetti (2005) is based on an extension of the Tropos formalism and includes the following steps:

1. Create, in the Tropos formalism, an *organizational model* that represents the stakeholders, their relationships, their goals as well as the relevant facts for the organization and the attributes that describe them.
2. Create, in the Tropos formalism, a *decisional model* that expresses the analysis goals of decision makers and their information needs.
3. Create preliminary fact schemata from the decisional model.
4. Edit the fact schemata, for instance, by detecting functional dependencies between dimensions, recognizing optional dimensions, and unifying measures that only differ for the aggregation operator.

This approach is, in our view, more difficult to pursue than the previous one. Nevertheless, it is the only alternative when a detailed analysis of data sources cannot be made (for instance, when the DW is fed from an ERP system), or when the sources come from legacy systems whose complexity discourages recognition and normalization.

Mixed Approaches

Finally, also a few *mixed* approaches to design have been devised, aimed at joining the facilities of data-driven approaches with the guarantees of requirement-driven ones (Bonifati, Cattaneo, Ceri, Fuggetta, & Paraboschi, 2001; Giorgini et al., 2005). Here the user requirements, captured by means of a goal-oriented formalism, are matched with the schema of the source database to drive the algorithm that generates the conceptual schema for the DW. For instance, the approach proposed by Giorgini et al. (2005) encompasses three phases:

1. Create, in the Tropos formalism, an *organizational model* that represents the stakeholders, their relationships, their goals, as well as the relevant facts for the organization and the attributes that describe them.

2. Create, in the Tropos formalism, a *decisional model* that expresses the analysis goals of decision makers and their information needs.
3. Map facts, dimensions, and measures identified during requirement analysis onto entities in the source schema.
4. Generate a preliminary conceptual schema by navigating the functional dependencies expressed by the source schema.
5. Edit the fact schemata to fully meet the user expectations.

Note that, though step 4 may be based on the same algorithm employed in step 2 of the data-driven approach, here navigation is not "blind" but rather it is actively biased by the user requirements. Thus, the preliminary fact schemata generated here may be considerably simpler and smaller than those obtained in the data-driven approach. Besides, while in that approach the analyst is asked for identifying facts, dimensions, and measures directly on the source schema, here such identification is driven by the diagrams developed during requirement analysis.

Overall, the mixed framework is recommendable when source schemata are well-known but their size and complexity are substantial. In fact, the cost for a more careful and formal analysis of requirement is balanced by the quickening of conceptual design.

Open Issues

A lot of work has been done in the field of conceptual modeling for DWs; nevertheless some very important issues still remain open. We report some of them in this section, as they emerged during joint discussion at the *Perspective Seminar on "Data Warehousing at the Crossroads"* that took place at Dagstuhl, Germany on August 2004.

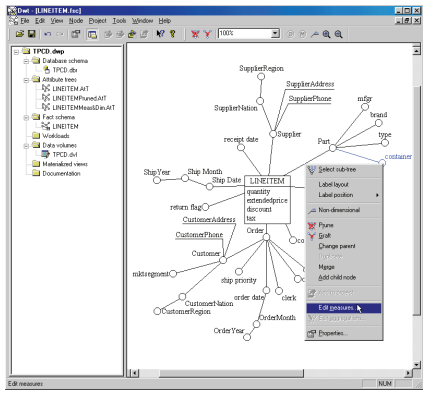
- **Lack of a standard:** Though several conceptual models have been proposed, none of them has been accepted as a standard so far, and all vendors propose their own proprietary design methods. We see two main reasons for this: (1) though the conceptual models devised are semantically rich, some of the modeled properties cannot be expressed in the target logical models, so the translation from conceptual to logical is incomplete; and (2) commercial CASE tools currently enable designers to directly draw logical schemata, thus no industrial push is given to any of the models. On the other hand, a unified conceptual model for DWs, implemented by sophisticated CASE tools, would be a valuable support for both the research and industrial communities.

- **Design patterns:** In software engineering, design patterns are a precious support for designers since they propose standard solutions to address common modeling problems. Recently, some preliminary attempts have been made to identify relevant patterns for multidimensional design, aimed at assisting DW designers during their modeling tasks by providing an approach for recognizing dimensions in a systematic and usable way (Jones & Song, 2005). Though we agree that DW design would undoubtedly benefit from adopting a pattern-based approach, and we also recognize the utility of patterns in increasing the effectiveness of teaching how to design, we believe that further research is necessary in order to achieve a more comprehensive characterization of multidimensional patterns for both conceptual and logical design.
- **Modeling security:** Information security is a serious requirement that must be carefully considered in software engineering, not in isolation but as an issue underlying all stages of the development life cycle, from requirement analysis to implementation and maintenance. The problem of information security is even bigger in DWs, as these systems are used to discover crucial business information in strategic decision making. Some approaches to security in DWs, focused, for instance, on access control and multilevel security, can be found in the literature (see, for instance, Priebe & Pernul, 2000), but neither of them treats security as comprising all stages of the DW development cycle. Besides, the classical security model used in transactional databases, centered on tables, rows, and attributes, is unsuitable for DW and should be replaced by an ad hoc model centered on the main concepts of multidimensional modeling—such as facts, dimensions, and measures.
- **Modeling ETL:** ETL is a cornerstone of the data warehousing process, and its design and implementation may easily take 50% of the total time for setting up a DW. In the literature some approaches were devised for conceptual modeling of the ETL process from either the functional (Vassiliadis, Simitsis, & Skiadopoulos, 2002), the dynamic (Bouzeghoub, Fabret, & Matulovic, 1999), or the static (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1998) points of view. Recently, also some interesting work on translating conceptual into logical ETL schemata has been done (Simitsis, 2005). Nevertheless, issues such as the optimization of ETL logical schemata are not very well understood. Besides, there is a need for techniques that automatically propagate changes occurred in the source schemas to the ETL process.

Conclusion

In this chapter we have proposed a set of solutions for conceptual modeling of a DW according to the DFM. Since 1998, the DFM has been successfully adopted, in real

Figure 8. Editing a fact schema in WAND



DW projects mainly in the fields of retail, large distribution, telecommunications, health, justice, and instruction, where it has proved expressive enough to capture a wide variety of modeling situations. Remarkably, in most projects the DFM was also used to directly support dialogue with end users aimed at validating requirements, and to express the expected workload for the DW to be used for logical and physical design. This was made possible by the adoption of a CASE tool named WAND (warehouse integrated designer), entirely developed at the University of Bologna, that assists the designer in structuring a DW. WAND carries out data-driven conceptual design in a semiautomatic fashion starting from the logical scheme of the source database (see Figure 8), allows for a core workload to be defined on the conceptual scheme, and carries out workload-based logical design to produce an optimized relational scheme for the DW (Golfarelli & Rizzi, 2001).

Overall, our on-the-field experience confirmed that adopting conceptual modeling within a DW project brings great advantages since:

- Conceptual schemata are the best support for discussing, verifying, and refining user specifications since they achieve the optimal trade-off between expressivity and clarity. Star schemata could hardly be used to this purpose.
- For the same reason, conceptual schemata are an irreplaceable component of the documentation for the DW project.
- They provide a solid and platform-independent foundation for logical and physical design.
- They are an effective support for maintaining and extending the DW.
- They make turn-over of designers and administrators on a DW project quicker and simpler.

References

Abelló, A., Samos, J., & Saltor, F. (2002, July 17-19). YAM2 (Yet another multidimensional model): An extension of UML. In *Proceedings of the International Database Engineering & Applications Symposium* (pp. 172-181). Edmonton, Canada.

Agrawal, R., Gupta, A., & Sarawagi, S. (1995). *Modeling multidimensional databases* (IBM Research Report). IBM Almaden Research Center, San Jose, CA.

Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A., & Paraboschi, S. (2001). Designing data marts for data warehouses. *ACM Transactions on Software Engineering and Methodology*, *10*(4), 452-483.

Bouzeghoub, M., Fabret, F., & Matulovic, M. (1999). Modeling data warehouse refreshment process as a workflow application. In *Proceedings of the International Workshop on Design and Management of Data Warehouses*, Heidelberg, Germany.

Cabibbo, L., & Torlone, R. (1998, March 23-27). A logical approach to multidimensional databases. In *Proceedings of the International Conference on Extending Database Technology* (pp. 183-197). Valencia, Spain.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (1998, August 20-22). Information integration: Conceptual modeling and reasoning support. In *Proceedings of the International Conference on Cooperative Information Systems* (pp. 280-291). New York.

Datta, A., & Thomas, H. (1997). A conceptual model and algebra for on-line analytical processing in data warehouses. In *Proceedings of the Workshop for Information Technology and Systems* (pp. 91-100).

Fahrner, C., & Vossen, G. (1995). A survey of database transformations based on the entity-relationship model. *Data & Knowledge Engineering*, *15*(3), 213-250.

Franconi, E., & Kamble, A. (2004a, June 7-11). The GMD data model and algebra for multidimensional information. In *Proceedings of the Conference on Advanced Information Systems Engineering* (pp. 446-462). Riga, Latvia.

Franconi, E., & Kamble, A. (2004b). A data warehouse conceptual data model. In *Proceedings of the International Conference on Statistical and Scientific Database Management* (pp. 435-436).

Giorgini, P., Rizzi, S., & Garzetti, M. (2005, November 4-5). Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP* (pp. 47-56). Bremen, Germany.

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

Golfarelli, M., & Rizzi, S. (2001, April 2-6). WAND: A CASE tool for data warehouse design. In *Demo Proceedings of the International Conference on Data Engineering* (pp. 7-9). Heidelberg, Germany.

Gyssens, M., & Lakshmanan, L. V. S. (1997). A foundation for multi-dimensional databases. In *Proceedings of the International Conference on Very Large Data Bases* (pp. 106-115), Athens, Greece.

Hüsemann, B., Lechtenbörger, J., & Vossen, G. (2000). Conceptual data warehouse design. In *Proceedings of the International Workshop on Design and Management of Data Warehouses*, Stockholm, Sweden.

Jones, M. E., & Song, I. Y. (2005). Dimensional modeling: Identifying, classifying & applying patterns. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP* (pp. 29-38). Bremen, Germany.

Kimball, R. (1996). *The data warehouse toolkit*. New York: John Wiley & Sons.

Lechtenbörger , J. (2001). *Data warehouse schema design* (Tech. Rep. No. 79). DISDBIS Akademische Verlagsgesellschaft Aka GmbH, Germany.

Lenz, H. J., & Shoshani, A. (1997). Summarizability in OLAP and statistical databases. In *Proceedings of the 9th International Conference on Statistical and Scientific Database Management* (pp. 132-143). Washington, DC.

Li, C., & Wang, X. S. (1996). A data model for supporting on-line analytical processing. In *Proceedings of the International Conference on Information and Knowledge Management* (pp. 81-88). Rockville, Maryland.

Luján-Mora, S., Trujillo, J., & Song, I. Y. (2002). Extending the UML for multidimensional modeling. In *Proceedings of the International Conference on the Unified Modeling Language* (pp. 290-304). Dresden, Germany.

Niemi, T., Nummenmaa, J., & Thanisch, P. (2001, June 4). Logical multidimensional database design for ragged and unbalanced aggregation. *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses,* Interlaken, Switzerland (p. 7).

Nguyen, T. B., Tjoa, A. M., & Wagner, R. (2000). An object-oriented multidimensional data model for OLAP. In *Proceedings of the International Conference on Web-Age Information Management* (pp. 69-82). Shanghai, China.

Pedersen, T. B., & Jensen, C. (1999). Multidimensional data modeling for complex data. In *Proceedings of the International Conference on Data Engineering* (pp. 336-345). Sydney, Austrialia.

Prakash, N., & Gosain, A. (2003). Requirements driven data warehouse development. In *Proceedings of the Conference on Advanced Information Systems Engineering—Short Papers*, Klagenfurt/Velden, Austria.

Priebe, T., & Pernul, G. (2000). Towards OLAP security design: Survey and research issues. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP* (pp. 33-40). Washington, DC.

SAP. (1998). *Data modeling with BW*. SAP America Inc. and SAP AG, Rockville, MD.

Sapia, C., Blaschka, M., Hofling, G., & Dinter, B. (1998). Extending the E/R model for the multidimensional paradigm. In *Proceedings of the International Conference on Conceptual Modeling*, Singapore.

Schiefer, J., List, B., & Bruckner, R. (2002). A holistic approach for managing requirements of data warehouse systems. In *Proceedings of the Americas Conference on Information Systems*.

Sen, A., & Sinha, A. P. (2005). A comparison of data warehousing methodologies. *Communications of the ACM*, *48*(3), 79-84.

Simitsis, A. (2005). Mapping conceptual to logical models for ETL processes. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP* (pp. 67-76). Bremen, Germany.

Tryfona, N., Busborg, F., & Borch Christiansen, J. G. (1999). starER: A conceptual model for data warehouse design. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP,* Kansas City, Kansas (pp. 3-8).

Tsois, A., Karayannidis, N., & Sellis, T. (2001). MAC: Conceptual data modeling for OLAP. In *Proceedings of the International Workshop on Design and Management of Data Warehouses* (pp. 5.1-5.11). Interlaken, Switzerland.

Vassiliadis, P. (1998). Modeling multidimensional databases, cubes and cube operations. In *Proceedings of the 10th International Conference on Statistical and Scientific Database Management*, Capri, Italy.

Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002, November 8). Conceptual modeling for ETL processes. In *Proceedings of the ACM International Workshop on Data Warehousing and OLAP* (pp. 14-21). McLean, VA.

Winter, R., & Strauch, B. (2003). A method for demand-driven information requirements analysis in data warehousing projects. In *Proceedings of the Hawaii International Conference on System Sciences,* Kona (pp. 1359-1365).

Endnote

[1] In this chapter we will only consider dynamicity at the *instance* level. Dynamicity at the *schema* level is related to the problem of evolution of DWs and is outside the scope of this chapter.