From Federated Databases to a Federated Data Warehouse System

Stefan Berger Michael Schrefl
University Linz, Data & Knowledge Engineering Group
4040 Linz, Altenberger Str. 69, Austria
(berger | schrefl)@dke.uni-linz.ac.at

Abstract

Although Data Warehousing is regarded as a mature technology now, the definition of a federated architecture for Data Warehouse (DW) integration remains an open research question. This paper identifies requirements on a Federated DW System and proposes an architecture supporting the tightly coupled integration of heterogeneous data marts into a global, logical schema. In order to enable the processing of queries in the federation, our approach provides a Dimension Algebra (DA) and Fact Algebra (FA) to define the mappings between the global and local schemas. Moreover, we demonstrate how to apply DA and FA expressions for dimension and fact integration and explain the benefits of such an approach.

1. Introduction

Data Warehouses (DWs) are sophisticated, highly specialized database systems optimized for decision support rather than transaction support. An organization's DW collects and consolidates the data on all subject areas that are helpful for decision making. Data Marts (DMs) are more specific repositories, designed on top of the DW to deliver a particular data subset to a particular set of users, e.g. the sales division. Business Intelligence tools provide analysis functionality, thus acting as the query interface between the users and the DMs [13].

As the result of increasing information needs or of mergers and acquisitions, different organizations have to integrate their independent DWs. Successful DW integration opens up a larger pool of information, broadening the knowledge base for the decision makers. Consequently, the participating organizations benefit from more accurate analysis and better decisions.

The simplest solution to perform the integration of autonomous DMs is on the physical level, copying the data from the sources to the global DW. However, the physical integration of huge information sources, especially organizationally independent DWs, is often not feasible due to privacy restrictions or technical limitations (e.g. storage).

Therefore, the integration of DMs is better performed on the logical level, by building a federation. The advantages of such an approach are well known from the field of databases. A Federated Database System (FDB) establishes a dedicated access layer on top of autonomous data sources to hide data heterogeneity from the applications and users [25, 18]. The local databases participating in the federation can be queried with a uniform language. Tightly coupled federated systems additionally provide a global schema expressed in the common, "canonical" data model [25].

The use of well-established FDB architectures for autonomous DMs, however, has an important drawback. The traditional data integration techniques employed in FDBs were developed for the relational data model. As such, they are optimized for dealing with heterogeneities among relational entities. However, DMs are typically based on the multi-dimensional data model—the relevant information is modelled as *measure variables* in so-called fact tables, that are contextualized by several *dimensions* representing business perspectives [13].

The complex properties specific to facts and dimensions (e.g. grain levels, roll-up hierachies [9]), cannot be represented adequately in FDB Systems. Basic functionality that should be provided by the federation layer (e.g. joining the fact data with the dimensions) is not available. Consequently, not all the heterogeneities among the autonomous DMs are hidden completely from the application layer, increasing the complexity of applications or user queries.

Several approaches towards Federated DW Systems were proposed in the literature (e.g. [7], [1], [2]). However, these systems are loosely-coupled—they do not support a global multi-dimensional schema. Consequently, a laborious ad-hoc integration of the data is necessary to utilize the distributed DMs together.

Analogously to FDB architectures, the "ideal" Federated DW is tightly coupled with its components, providing transparent access to autonomous, distributed DMs through a global schema. Under the global, "federated" schema, de-

1530-1605/08 $25.00 © 2008 IEEE

fined in a *canonical data model*, several conceptually independent layers repair structural and semantic heterogeneity among the DMs. Four different system layers are distinguished: the *application* or *presentation layer* uses the global schema residing on the *federation layer*; the intermediate *wrapper layer* defines the export and import schemas from the native schemas of the component systems, that in turn reside on the *foundation layer*. A query on the global schema is analyzed, decomposed into sub-queries and sent to the component systems. Upon receipt of the responses, the result data are consolidated and transformed to the global schema in the canonical model again. To the best of our knowledge, such an architecture has not yet been proposed for the Data Warehousing area.

Going beyond the ideas proposed in previous research on distributed DWs, we identify several requirements that constitute essential features of a tightly coupled Federated DW System. First, the canonical data model used for the global schema must provide the semantic richness to model the specific features of multi-dimensional data (e.g. hierarchies in dimensions). The global schema represents integrated data marts, establishing a level of abstraction from the component-specific schema and implementation details.

Second, the multi-dimensional canonical data model has to be independent of any implementation model. Thus, the federated system can support DW products of multiple vendors. Nonetheless, it should be possible to query the global schema using a uniform language.

Third, to integrate the autonomous data, the federation layer has to enable the definition of mappings between multi-dimensional schemas, e.g. using a language of conversion operators. When processing and evaluating queries on the global schema, the federated system uses the mappings to translate data between the heterogeneous schemas among the component systems. This process requires a global repository storing the metadata on the component schemas and the necessary transformations.

Fourth, the federated system must be easily adaptable to schema changes in the autonomous component systems. Thus, schema evolution of the local DMs—although resulting in an adapted mapping—does not enforce any changes on the application level, unless the new schema should be propagated also globally. Moreover, schema integration, especially dimension integration, is a laborious task. It should be possible to adapt the dimension mappings incrementally so that the task of adding new component DMs (and consequently, new mappings) to the federation is facilitated.

In this paper we present a Federated DW (FDW) system architecture meeting the above requirements. We introduce a platform-independent canonical data model used to specify the global multi-dimensional schema of the DM federation. In order to support a uniform query language with OLAP features, the canonical data model can be implemented on relational or multidimensional platforms.

An important contribution of our FDW architecture is the *dimension repository*, a novel component on the federated layer that replicates the consolidated dimensions. The fact data, in contrast, remain at the autonomous component DMs and are accessed by querying the global schema. In evaluating such a query, the federated layer joins the fact data transferred from the DMs with the replicated dimensions in the repository, thus reducing the overhead of distributed query processing.

To map the autonomous schemas to the global schema, we use a Fact Algebra (FA) and Dimension Algebra (DA). The schema mappings defined as FA and DA expressions—the wrapper layer of the federated architecture—are easily maintainable in an incremental way. Thus, the effects of schema evolution on the application layer are kept minimal.

The paper is organized as follows. Section 2 reviews the relevant literature. Section 3 gives an overview of the proposed FDW, comparing our approach to previous works. In Sections 4 and 5 we present the canonical data model and integration techniques used in our approach, respectively. The final remarks in Section 6 conclude the paper.

2. Related Work

Database integration has been researched for several decades. Halevy et al. [10] survey recent progress achieved by the data integration community and list future challenges. Federated Database Systems [25] are a well-known example of systems applying data integration techniques.

Two different strategies for describing an integrated schema of heterogeneous data sources are known: global-as-view (GAV) and local-as-view (LAV), specifying the global schema resp. the local schemas as view expressions over the other components [10]. GAV mappings facilitate query processing, whereas LAV mappings are easier to maintain and evolve. The data integration tutorial of Lenzerini [17] contains a detailed discussion of the challenges and possible solutions in these contexts.

Dimension integration has been addressed by several recent works in the fields of distributed and federated Data Warehousing. For example, [27] developed a visual tool for dimension integration based on the approach in [7]. Their tool allows the user to specify mappings between dimensions and check them for correctness. A graph model of dimensions is used in [11, 12] in order to detect similar levels and hierarchies in heterogeneous dimensions. An architecture using XML tools for dimension integration and query processing in a federation of DMs is developed in [21] and [5]. All these approaches achieve a loosely coupled federation of DMs without a global schema.

The approach closest to our work is described in [6] and [7], where the authors introduce the notion of dimension

compatibility as the prerequisite for successful integration of autonomous DMs. The query processing in [6] focuses primarily on drill-across operations over the autonomous DMs. In their approach, the federation between the DMs is again loosely coupled.

In [7] the authors additionally discuss a tightly coupled variant of their approach in the sense that they compute a global materialized view from the autonomous DMs. Obviously, their idea is a feasible solution for DMs with relatively small amounts of fact data. Given the potentially huge data volume in DWs, however, an approach using a federated layer with schema mappings seems more appropriate for most real-world settings.

The conceptual modelling of DW integration is supported by the UML profile proposed in [19, 20]. Their approach allows to visualize dependencies between dimension schemas using so-called schema mapping diagrams. However, the schema mapping diagrams cannot be used for distributed query processing.

OLAP query processing in distributed DW systems without a federated layer is addressed in [1] and [4]. The Skalla system [1] provides a framework for the complete evaluation of distributed queries, whereas DWS-AQA [4] investigates approximate query answering to optimize response time at the cost of some accuracy of the result. An interesting idea of both approaches is the replication of dimension data at all local sites to improve the query performance.

Due to the increasing variety of DW tools on the market, the field of DW metadata integration has been receiving attention recently. Two industry standards were developed aiming to support interoperability between DW tools [24]: the Open Information Model (OIM) and the Common Warehouse Metamodel (CWM). These two competing specifications are compared in [29]. As pointed out by [24], however, both the OIM and CWM standards lack expressivity to represent all the complex semantics of multi-dimensional models that is encountered in DW integration.

3. Overview of the Federated DW Architecture

The main goal of the Federated DW (FDW) introduced in this paper is to provide a tightly coupled architecture with a global multi-dimensional schema, abstracting from the structural heterogeneities among the autonomous DMs. The architecture enables the definition of the global schema in terms of mappings from the DMs to the global DW. Based on the Sheth and Larson general five-level architecture [25], we extend previous approaches (e.g. [7], [21], [1]) with a dimension repository component and a framework for dimension and fact integration. Note that Fig. 1 shows the five schema levels known from [25] on four system layers.

The novel aspects of the FDW architecture are twofold: (1) the so-called *dimension repository* stores the consolidated dimension schemas and replicates the dimensional data (see Fig. 1). This idea is motivated by the Skalla and DWS-AQA approaches [1, 4] indicating that local dimension replicates improve the query performance. Moreover, dimensions—similar to domain ontologies—typically evolve slowly and are relatively small in size [15, 1]. When answering queries to the global schema, it is sufficient to fetch the fact data from the DMs, subsequently joining it with the dimensions out of the repository. Thus, the dimension repository allows to reduce the overhead of distributed query processing. (2) Our approach provides a comprehensive framework tackling the interrelated problems of dimension integration and fact integration that arise in the context of distributed and federated Data Warehousing.

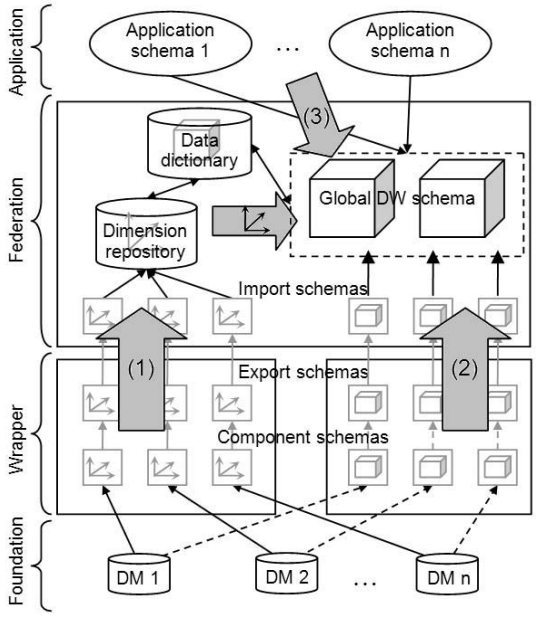

Figure 1. Federated Data Warehouse architecture.

The proposed FDW architecture is depicted in Fig. 1, of which the following concepts build its cornerstones:

Dimension integration (label (1) in Fig. 1): in the first place, the federation layer of the system needs to understand the common dimensional context of the fact data in the component DMs. Therefore, the heterogeneous dimensions found in the DMs are represented in the canonical model and integrated, using mappings expressed by a Dimension Algebra. The result of the dimension integration process is the *dimension repository*, consisting of both the integrated schemas and copies of the dimension members.

Fact integration (label (2) in Fig. 1): in order to consolidate a global schema of the multi-dimensional data in the component DMs, the autonomous fact tables are integrated using mappings expressed by a Fact Algebra. The fact integration process results in the definition of the import

schemas that have to conform to the *global DW* schema. The mappings and global metadata are stored in the *data dictionary*. When processing a user query on the global schema, the federated system uses the mappings in the data dictionary to decompose the query and retrieve the results from the local DMs. The wrapper layer transforms all data to the global schema according to the mappings.

User requirements modelling (label (3) in Fig. 1): the cube definitions of the global schema should correspond to the information needs of the applications and users that work on top of the federated layer. Object oriented modelling techniques, e.g. the UML, are commonly used to model the user requirements. In our architecture, UML use case diagrams can be used to express the "desired" cubes of the global schema. In general, requirements modelling is well researched [22] and out of the scope of our paper.

In order to provide a mechanism for the definition of schema mappings, we introduce the *Dimension Algebra* (see subsection 5.1) and *Fact Algebra* (see subsection 5.2) and will discuss how they help to solve the problems of multi-dimensional data integration. Note that our algebra-based approach is similar to the global-as-view data integration strategy [10, 17], since it maps the local DM schemas to the global DW schema.

The main benefits of our approach are the following: (1) it allows the integration of autonomous DMs to a global schema, whilst the DMs retain schema and data autonomy; (2) the tightly coupled federation of DMs allows the users and applications to operate on the global schema, providing a level of abstractions from the heterogeneities among the DMs; and (3) the proposed architecture supports DMs that are implemented in relational or multi-dimensional physical platforms. The next Section of the paper will introduce the canonical data model used by the FDW.

4. Multi-dimensional Canonical Data Model

This Section discusses the conceptual multi-dimensional data model that is used as the canonical model [25] of the FDW depicted in Fig. 1. The data model supports the essential concepts *fact* and *dimension*, refining a previous proposal [7]. It allows the definition of the global multi-dimensional schema independent of any implementation aspects.

Intuitively, a DM defines one or more measure variables within fact tables, categorized by some dimensions that are organized in hierarchies of levels. A cube is a data structure linking a fact table to one or more dimensions, comprising a multi-dimensional space that stores the factual data. Facts and dimensions consist of both their schema *and* the corresponding instances. For the dimension instances, we use the commonly accepted term *dimension members* (or *members* for short) [28] throughout the paper.

The model described here extends the so-called *MD* model [7], introducing additional properties of the *dimension* construct in order to cope with *all* multi-dimensional heterogeneities analyzed in [3]. In particular, we introduce (1) the support for non-dimensional attributes in a dimension's level schema (definition 4.3), and (2) the functions $level$ and $members$ specifying the relationship between the hierarchy and roll-up functions of levels (defs. 4.5 and 4.6). These functions are necessary to define operators changing the hierarchy in both a level's schema and instance (see defs. 5.5 and 5.6). Moreover, we refine the model concepts *data mart*, *cube* and *dimension* as follows (defs. 4.1 and 4.2): we regard a local DM as a *universe of discourse* for the declaration of dimensions, i.e. the constructs *cube* and *dimension* are first-class citizens of the model.

Definition 4.1: A (local) *Data Mart* $DM = \{C_1, ..., C_n, D_1, ..., D_m\}$ consists of a non-empty set of Cubes C_i and a non-empty set of dimensions D_j.

Definition 4.2: A *Cube* $C = [F_C, D_C]$ consists of a set of facts or cells F_C that are linked to a set of dimensions $D_C \subseteq \{D_1, ..., D_m\}$. We say that D_C represents the multi-dimensional *context* of the cells F_C. The number k of dimensions in D_C is referred to as the *dimensionality of* C.

Intuitively, a data mart provides a "view" on the fact and dimension tables in a local DW, probably including slice and dice operations. Every fact table of the DM, in turn, corresponds to a cube (def. 4.2). Note that def. 4.2 does not impose any restriction on the logical model of a cube.

Now let $\{\tau_1, ..., \tau_m\}$ be a finite set of data types (e.g. integers) with their domain defined by function $dom(\tau)$.

Definition 4.3: A *Dimension* $D \in \{D_1, ..., D_m\}$ of DM has the following properties:

- the *dimension schema* $S_D = (L_D, S(L_D), H_D)$ containing (I) the finite, non-empty set of Levels $L_D = \{l_1, ..., l_j, ..., l_m, l_{all}\}$ with level schema $S(L_D) = \{S_{l_1}, ..., S_{l_m}\}$ and (II) a hierarchy $H_D \subseteq L_D \times L_D$, where H_D forms a lattice. If $l_1, l_2 \in H_D$, we write $l_1 \mapsto l_2$ and we say l_1 "rolls-up to" l_2.
- the *level schema* $S_{l_j} \in S(L_D)$ of a level l_j is an attribute schema $(k_j, a_{j1}, ..., a_{jk})$ with name l_j, key k_j (the dimensional or "roll-up attribute") and optional non-dimensional attributes $a_{j1}, ..., a_{jk}$, denoted as $l_j.k, l_j.a_1, ..., l_j.a_k$, respectively. Every attribute $A_k \in S_{l_j}$ has a domain $dom(A_k) = dom(\tau_k)$.
- the *dimension instance* $d(S_D)$ over schema S_D with name d containing (I) a set of members V_d with each $v \in V_d$ being a tuple over a level schema S_{l_j}, and (II) a family of "roll-up" relationships ρ_d between the member subsets $T_j \subseteq V_d$ (defined later).

Definition 4.4: The *base level* l_0 of hierarchy H_D, representing the finest grain of the dimension's members, is the bottom element of the lattice H_D. The *all-level* or l_{all} is the top element of the lattice H_D.

Definition 4.5: Let D be a dimension with schema S_D, level schema $S(L_D)$ and instance $d(S_D)$. We define the following functions over D:

- $level : V_d \rightarrow L_D$ returns the level l_i corresponding to a given $v_i \in V_d$.
- $members : L_D \rightarrow 2^{V_d}$ returns the set $T_i = \{v \in V_d \mid level(v) = l_i\}$ containing all members $v \in V_d$ belonging to level l_i.

Definition 4.6: Let $l, k \in L_D$ be two levels, $l \neq k$, of a dimension D, and $T_l = members(l)$, $T_k = members(k)$. The roll-up function $\rho^{l \mapsto k}$ is defined for each pair $l, k \in H_D$. The roll-up function $\rho^{l \mapsto k}$ is consistent iff $\forall v \in T_l : \rho^{l \mapsto k}(v) = w \wedge w \in T_k$. The family of roll-up functions ρ_d (see def. 4.3) contains all $\rho^{l \mapsto k}$ defined in this way.

Definition 4.7: The *facts* or *cells* F_C of Cube C are composed of:

- the *fact schema* $S_C = \{A_C, M_C\}$ with (I) a set of dimension attributes $A_C = \{A_1, ..., A_n\}$, (II) a set of measure attributes $M_C = \{M_1, ..., M_m\}$. Each $A_i \in A_C$ is linked with a level $l_i \in L_{D_i}$ of the dimensions $D_i \in D_C$ (see def. 4.2), each $M_j \in M_C$ with a τ_j. The domain of the attributes in S_C is defined as $dom(A_i) = members(l_i)$ and $dom(M_j) = dom(\tau_j)$.
- the *fact instance* $c(S_C)$, a set of tuples over $\{[dom(A_1) \times ... \times dom(A_n)], [dom(M_1) \times ... \times dom(M_m)]\}$. A tuple $f \in c(S_C)$ is called a *"cell"* or *"fact"*. Moreover, we call the values $[f(A_1), ..., f(A_n)]$ the *"coordinate"* of a cell, modelling the multi-dimensional context for the measures $[f(M_1), ..., f(M_m)]$.

Definition 4.8: Let $f_1 \in c_1(S_{C_1})$ and $f_2 \in c_2(S_{C_2})$ be two fact sets with exactly the same dimension attributes, i.e. $A_{C_1} = A_{C_2} = A$. The cells f_1 and f_2 are called *overlapping* iff $f_1(A) = f_2(A)$ since the measure attributes M_{C_1} and M_{C_2} may still be different.

Note that the multi-dimensional model presented above is a conceptual model—as such, it can be implemented on several platforms to support a preferred DW tool and query language upon the global schema. For example, the global schema can be implemented on a relational system in order to use SQL [14] to query the global DW and SQL-MDi [3] to specify the schema mappings. However, since we concentrate on the conceptual architecture of the FDW, this aspect lies outside the scope of this paper.

Example 4.1: Assume '*med*' be some cube in a DM named 'DM1' that records medication costs of patients. Fig. 2 depicts a sample subset of the cells or facts in cube *med*. For the sake of simplicity, we choose a tabular presentation. The cube defines a single dimension *date* with levels L_{date} = $\{day, month, year, all\}$, level schema $S(L_{date})$ = $\{(day), (month, name), (year)\}$ and hiearchy H_{date} = $\{day \mapsto month, month \mapsto year, year \mapsto all\}$, i.e. *day* is the base level. The member set V_{date} consists of the two tuples depicted in Fig. 2 (bottom). Note that, intuitively, the roll-up functions ρ_d correspond to the functional dependencies within the V_{date}-tupels; e.g. $\rho^{month \mapsto year}$ = {*(02-06 $\mapsto$ 2006)*}. The *med* cells are given with schema S_{med} = $\{A_{med}$ = {*date*}, M_{med} = {*cost_p, cost_m*}$\}$, and the instance over the domain *dom(date)* = *day.day*, *dom(cost_p)* = *float*, *dom(cost_m)* = *float* and instance $c(S_{med})$ with the tuples depicted in Fig. 2 (top).

DM1.med

date	cost_p	cost_m
23-02-06	356.0	425.0
23-02-06	125.2	1742.0
25-02-06	473.0	903.8

DM1.date

[day] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Feb. 06'	2006	ALL
25-02-06	02-06 'Feb. 06'	2006	ALL

Figure 2. Example cube "med".

5. DW Integration with Algebra Expressions

This section discusses the essential concepts of our FDW architecture that were briefly presented in Section 3—how to define import mappings from dimensions and facts of local DMs to the global DW using Dimension Algebra (DA) and Fact Algebra (FA) expressions.

5.1. Integrating Dimensions

Dimension integration is recognized by the DW research community as a fundamental challenge in distributed and federated Data Warehousing (e.g. [27], [7], [11], [12], see Section 2). Indeed, a DM federation is useless if no common semantics of the dimensional data is defined.

In contrast to the previous approaches, the FDW maintains the global *dimension repository* that both stores the federated dimension schema and replicates the members. Thus, the dimension repository holds the comprehensive model of the common multi-dimensional context of fact data within the federation. This approach is motivated by the slowly evolving nature of the dimensional data [15, 1].

Using the dimension repository, the federation layer is able to "understand" the dimensional context of a fact set retrieved from a component DM, thus eliminating the need to transfer the dimension table(s) separately. Consequently, the overhead for processing a federated query is reduced.

The dimension integration process in our FDW consists of the following steps, depicted in Fig. 3: *(1) Translation*. First, the public parts of the local dimension schemata have

to be translated from the native model to the canonical data model. *(2) Definition*. The global system does not necessarily need every dimension defined in the component systems. The export dimension schema contains only the subset of dimensions needed in the global schema. *(3) Transformation*. Dimension schemas are mapped from export schemas to import schemas using DA expressions. Thus, the DA expressions transform the dimensions to a homogeneous, common representation. *(4) Merging*. The dimension definitions in the import schemas, obtained from the application of DA expressions, are merged to derive the global dimensions, that are replicated and stored in the dimension repository (see algorithm 5.1).

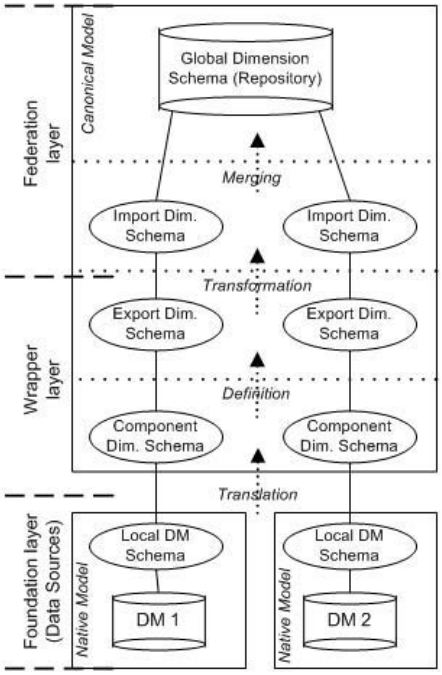

Figure 3. Dimension integration process

Only steps (1) and (2) of the process are well researched in the literature. Schema translation techniques [23] and wrappers solve the respective problems of defining the component and export schemas. In order to obtain the global dimension repository, schema transformation and merging techniques have to be applied to solve the remaining steps (3) and (4). Our goal of specifying the import schemas in a transformation language is not supported adequately by previous approaches, although the work of [7] contributes in this direction. However, the scope of their transformation operators is rather limited, given the variety of possible heterogeneities [3].

To support schema transformation with a language of transformation operators, our FDW employs the so-called *Dimension Algebra* (DA). The DA is an essential tool to create import schemas [25] during the dimension integration process in the FDW. Its operators are used to manipulate dimension schema or instance objects (see def. 4.3), thus defining a global dimension from an input dimension in terms of a mapping expressed in a sequence of DA operators (cf. def. 5.1). Our algebra takes over the three operators σ, π and ψ (select, project, aggregate) proposed by [7] (see examples 5.1 – 5.3) and defines five additional operators. Due to space limitations we refer to previously published work [3] that identifies the conflicts to be overcome by these operators and explains their necessity.

The operators of the proposed DA are applied to an input dimension D, resulting in an output dimension D' in which either the original schema S_D or its instance $d(S_D)$ is modified compared to D. As such, several operators can be combined easily to form more complex expressions:

Definition 5.1 (DA expression): A Dimension Algebra *expression* is a sequence of the DA operators σ, π, ψ [7] and ζ, δ, γ, Ω (see defs. 5.2 – 5.5) that is applied to some input dimension D, deriving an output dimension D'.

For the following examples, let dt denote the instance $d(S_{date})$ of dimension 'DM1.date' shown in Fig. 2.

Example 5.1 (σ – select): Applying $\sigma_{\{23-02-06\}}(dt)$, we obtain dimension $date'$ with the following $d'(S_{date'})$:

DM1.date'

[day] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Feb. 06'	2006	ALL

Example 5.2 (π – project): $\pi_{\{day\}}(dt)$ obtains dimension $date'$ with $L_{date'} = \{day\}$, $S(L_{date'}) = \{(day)\}$ and $H_{date'} = \{day \mapsto all\}$ and the following $d'(S_{date'})$:

DM1.date'

[day] $\mapsto$	[ALL]
23-02-06	ALL
25-02-06	ALL

Note from the above example 5.2 that the all-level can never be eliminated with the π-operator.

Example 5.3 (ψ – aggregate): $\psi_{month}(dt)$ obtains $date'$ with $L_{date'} = \{month, year, all\}$, $S(L_{date'}) = \{(month, name), (year)\}$, $H_{date'} = \{month \mapsto year, year \mapsto all\}$ and the following $d'(S_{date'})$:

DM1.date'

[month] name $\mapsto$	[year] $\mapsto$	[ALL]
02-06 'Feb.'	2006	ALL

In what follows, we define the DA operators *rename* (ζ), *change* (δ), *convert* (γ) and *override rollup* (Ω) formally. Let D be a dimension with schema S_D = $\{L_D, S(L_D), H_D\}$ and instance $d(S_D)$ = $\{V_d, \rho_d\}$, on which we apply a DA expression obtaining dimension D' with schema S'_D and instance $d'(S'_D)$.

Definition 5.2 (ζ – rename): Operator ζ changes the name of some attribute in d'. In particular, ζ can be applied to the following objects:

- Rename a dimension level $l \in L_D$: $\zeta_{l' \leftarrow l}(d)$.
- Rename an attribute $l.a$ of level schema $S_l \in S(L_D)$: $\zeta_{a' \leftarrow l.a}(d)$.

Example 5.4: Applying $\zeta_{time \leftarrow day}(dt)$, we obtain dimension $date'$ with the following $d'(S_{date'})$:

DM1.date'

[time] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Feb. 06'	2006	ALL
25-02-06	02-06 'Feb. 06'	2006	ALL

Definition 5.3 (δ – change): Let $T_l = members(l)$ be a subset of V_d and $v, w \in dom(l.a_i)$ be values for the non-dimensional attribute $l.a_i$, with $v \neq w$. $\delta_{w \leftarrow v(l.a_i)}(d)$ computes a new member-subset T'_l in d' such that $T'_l = \{t'(S_l) \mid \exists t \in T_l : t'(S_l \setminus a_i) = t(S_l \setminus a_i), t(a_i) = v, t'(a_i) = w\}$.

Example 5.5: $\delta_{\text{'}Febr.\text{'} \leftarrow \text{'}Feb.\text{'}(month.name)}(dt)$ obtains dimension $date'$ with the following $d'(S_{date'})$:

DM1.date'

[day] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Feb.'	2006	ALL
25-02-06	02-06 'Feb.'	2006	ALL

Definition 5.4 (γ – convert): Let $l.a \in S_l$ be some non-dimensional attribute, $T_l = members(l)$ and θ be an operator over $dom(a)$. The result of $\gamma_{\theta a}(d)$ is d' with member set $T'_l = \{t(S_l) \mid \exists t' \in T_l : t(S_l \setminus a) = t'(S_l \setminus a), t(a) = \theta a\}$.

Example 5.6: Assuming that *trunc(2)* is an operator over $dom(month.name)$, $\gamma_{month.name \;\; trunc(2)}(dt)$ results in the dimension $date'$ with the following $d'(S_{date'})$:

DM1.date'

[day] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Fe'	2006	ALL
25-02-06	02-06 'Fe'	2006	ALL

Definition 5.5 (Ω – override rollup): Let $m, v \in V_d$ be members of some dimension with $l_m = level(m)$, $l_v = level(v)$, such that $l_m \mapsto l_v \in H_D$ and $v \neq \rho^{l_m \mapsto l_v}(m)$. The result of $\Omega_{m \mapsto v}(d)$ is d' in which the result of $\rho^{l_m \mapsto l_v}(m)$ is changed to v.

Example 5.7: Assume that V_{date} in Fig. 2 contains an additional $year$-value of *2005*. Then, $\Omega_{02-06 \rightarrow 2005}(dt)$ obtains dimension $date'$ with the following $d'(S_{date'})$:

DM1.date'

[day] $\mapsto$	[month] name $\mapsto$	[year] $\mapsto$	[ALL]
23-02-06	02-06 'Feb.'	2005	ALL
25-02-06	02-06 'Feb.'	2005	ALL

To compute the dimension repository from the import dimension schemas, defined by DA expressions, the algorithm *mergeDim* is applied (see algorithm 5.1). Assume that $DM_i.D$ retrieves dimension D in DM_i of the FDW. From an input set of n dimensions D having identical names, with each D stored in a different DM, the algorithm returns a single output dimension $DM.D$ and computes its set of levels $DM.L_D$, its hierarchy $DM.H_D$ and member set $DM.V_D$. Finally, the family of roll-up functions $DM.\rho_d$ is computed (cf. definition 4.3). The result of the algorithm, i.e. the output dimension $DW.D$, is *consistent* iff definition 5.6 holds. All dimensions in the repository must be consistent.

Algorithm 5.1 (mergeDim):
Input: $DM_1.D, ..., DM_n.D$
Output: $DM.D$
begin
1 $DM.L_D = \bigcup_{i=1...n}\{DM_i.L_D\}$;
2 $DM.H_D = \bigcup_{i=1...n}\{DM_i.H_D\}$;
3 $DM.V_d = \bigcup_{i=1...n}\{members(DM_i.V_d)\}$;
4 $DM.\rho_d = \bigcup_{l,k} \rho^{l \mapsto k} : \; l, k \in DM.H_D$;
end;

Definition 5.6: The dimension $DM.D$ computed by algorithm 5.1 is *consistent*, iff (1) $DM.H_D$ forms a lattice, and (2) $\rho^{l \mapsto k}$ is consistent in the sense of definition 4.6 for all pairs $l, k \in DM.H_D$.

5.2. Integrating Facts

We call fact integration the process of defining the federated fact schema from heterogeneous cubes in the local autonomous DMs (see Fig. 4). In order to process queries on the global DW successfully, the federation layer needs a model of the distribution of fact data among the component DMs. For that purpose, the global *data dictionary* stores the mappings from the component fact schemas to the global fact schema and the schema metadata.

The fact integration process in the FDW consists of the following steps, depicted in Fig. 4: *(1) Translation*. The public parts of the local fact schemas are translated from the native to the canonical data model. *(2) Definition*. The export fact schemas are specified, containing only the subsets of fact schemas that are needed in the global schema. *(3) Transformation (local integration)*. The export fact schemas are mapped to the global DW schema, thus defining the import fact schemas. During this step, FA operators repair the heterogeneities among fact schemas. The context of the transformation step is local—that means, the result of the FA operators is computed from the local fact schema alone. *(4) Merging (global integration)*. If any overlapping fact sets (see def. 4.8) occur among the import fact schemas, the adequate extension matching operation has to be chosen according to the semantic relationship of the fact subsets. In this context, the allocation mappings are global since the federation layer needs information out of several local fact schemas for the computation of the mapping result.

Fact integration in relation with dimension integration has not been addressed as a problem in the literature. It can be argued that well-known data integration techniques [17] "do the job"—but only for very simple fact sets (e.g. with degenerate dimensions). These techniques, however, cannot

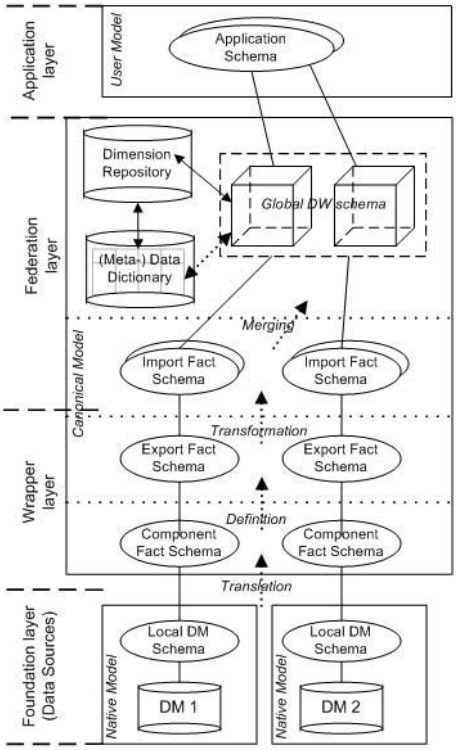

Figure 4. Fact integration process

handle adequately the semantics of dimensions and hierarchies specific to the multi-dimensional data model. Several heterogeneities among DMs must be resolved in an interrelated way, as discussed in [3].

To tackle the fact integration problem, we propose a fact transformation language called *Fact Algebra* (FA). The FA is an essential tool of our FDW to create import schemas during the fact integration process. It provides operators that can be used to manipulate the schema or instance of facts. As stated in [3], the cells of autonomous data cubes are a source of heterogeneity in a DM federation like the dimensions. Overall, the FA defines a set of eight operators that are necessary to resolve the heterogeneities among facts.

Analogously to the DA, the proposed FA operators are applied to some input fact F, resulting in another output fact F', of which either the schema $s(F')$ or the instance $i(F')$ is modified compared to the original F. Consequently, several FA operators are easy to combine, as stated below:

Definition 5.7 (FA expression): A Fact Algebra expression is a sequence of the FA operators σ, π, λ, ζ, γ, ε, χ and ξ (see defs. 5.8 – 5.11) that is applied to some input fact set F_C, deriving an output fact set F'_C. Alternatively, operator μ (def. 5.12) is applicable on two overlapping input fact sets F_{C_1}, F_{C_2}, producing an output fact set F'_C.

We will now formally define the FA operators *select* (σ), *project* (π), *delete measure* (λ), *rename* (ζ), *convert* (γ), *enrich dimensions* (ε), *merge measures* (χ) and *split measure* (ξ) that are applied in the local context of fact integration. In the remainder of this subsection, let F_C be a fact set with schema $S_C = \{A_F, M_F\}$, instance $c(S_C)$ and F' be the result of the DA expression with schema S'_C, instance $c'(S'_C)$.

The four basic operators σ, π, λ and ζ are defined like in the relational algebra [8] and are to be used as follows:

- Select: $\sigma_{[P]}(c)$ computes the fact set F' containing all tuples $t \in F$ satisfying the predicate(s) P.
- Project: $\pi_{(L \subset A_C)}(c)$ reduces the *dimensionality* of F' by projecting on one or more dimensional attributes.
- Delete measure: $\lambda_{(N \subset M_F)}(c)$ computes a fact set F' reducing the measure attributes to the projection N.
- Rename: $\zeta_{A' \leftarrow A}(c)$ and $\zeta_{M' \leftarrow M}(c)$ rename dimensional attribute $A \in A_C$ to A' and measure attribute $M \in M_C$ to M' in F', respectively.

Definition 5.8 (γ – convert): Let $M' \in M_C$ be a measure attribute of F_C and θ be an operator defined over $dom(M')$, and $v \in dom(M')$ be some legal M'-value. The result of $\gamma_{M'\theta v}(c)$ is F' with schema $S'_C = S_C$ and $c'(S'_C) = \{t' \mid \exists t \in c : t(S_C \setminus M') = t'(S_C \setminus M'), t'(M') = t(M')\theta v\}$.

For the examples in the remainder of this subsection, let F_{med} denote the fact set depicted in Fig. 2 with schema S_{med} and instance $md(S_{med})$.

Example 5.8: Applying $\gamma_{cost\_m+25}(md)$, we obtain fact set F'_{med} with the following cells $md'(S'_{med})$:

DM1.md'

date	cost_p	cost_m
23-02-06	356.0	450.0
23-02-06	125.2	1767.0
25-02-06	473.0	928.8

Definition 5.9 (ε – enrich dimensions): Let $A' \notin A_F$ be the name of a dimension attribute and $l_i \in S_D$ be a level definition of some dimension D of the global dimension repository. $\varepsilon_{A'=v}(c)$ results in F' with schema $S'_C = \{[A_F \cup A'], M_F\}$ and instance $c'(S'_C) = \{t' \mid \exists t \in i(F) : t'(F) = t(F), t'(A') = v\}$. Thus, $\varepsilon_{A'=v}(c)$ allows to increase the dimensionality of F'.

Example 5.9: Operator $\varepsilon_{source='dm1'}(md)$ obtains fact set F'_{med} with the following cells $md'(S'_{med})$:

DM1.md'

date	*source*	cost_p	cost_m
23-02-06	'dm1'	356.0	450.0
23-02-06	'dm1'	125.2	1767.0
25-02-06	'dm1'	473.0	928.8

Finally, the *pivot* operators χ and ξ can be used to repair the more complex schema-instance conflicts among cubes [3]. Two different pivot-options are available: *merge measure attributes* (χ) or *split measure attribute* (ξ)—see

defs. 5.10 and 5.11, respectively. Intuitively, χ merges several measures into a single one, preserving their context by extracting an additional dimension attribute. On the other hand, ξ transforms part of the fact context (i.e. one of the dimension attributes) into one or several new measure attributes. For further details, refer to the discussion in [3].

For definitions 5.10 and 5.11, let $M' \in M_C$, $A' \in A_C$ be some measure and dimension attribute of F_C, respectively.

Definition 5.10 (χ – merge measures): Let $M \subseteq M_C = \{M_1, ..., M_k\}$ denote the set of measure attributes to be merged, $M_x \notin M_C$ and $A_x \notin A_C$ be new attribute names. $\chi_{M \Rightarrow M_x, A_x}(c)$ creates F' with schema $S'_C = \{[A_C \cup A_x], [(M_C \setminus M) \cup \{M_x\}]\}$ and instance $c'(S'_C) = \bigcup_{i=1...n}\{t \mid \exists r \in c : t(A_x) = M_i,\ t(M_x) = r(M_i),\ t(A_C) = r(A_C), t(M_C \setminus M) = r(M_C \setminus M)\}$.

Example 5.10: $\xi_{\{cost\_p, cost\_m\} \Rightarrow costs, ccat}(md)$ results in fact set F'_{med} with the following cells $md'(S'_{med})$:

DM1.md'

date	*cat*	costs
23-02-06	cost_p	356.0
23-02-06	cost_m	425.0
23-02-06	cost_p	125.2
23-02-06	cost_m	1742.0
25-02-06	cost_p	473.0
25-02-06	cost_m	903.8

Definition 5.11 (ξ – split measure): Let $M = \{t(A') \mid t \in c(F_C)\}$ denote the set of all A'-values in c such that $M = \{M_1, ..., M_m\}$, and $B = A_C \setminus A'$ be the "remaining" dimensions. $\xi_{M' \Rightarrow A'}(c)$ creates F' with schema $S'_C = \{[A_C \setminus A'], [(M_C \setminus M') \cup M]\}$ and instance $c'(S'_C) = \{t(S'_C) \mid \exists t_1, ..., t_m \in c : \forall i, j = 1...m : t_i(B) = t_j(B) \wedge t(B) = t_1(B) \wedge t(M_1) = t_1(A') \wedge t(M_2) = t_2(A') \wedge ... \wedge t(M_m) = t_m(A')\}$.

Example 5.11: Applying $\chi_{cat \Rightarrow costs}(md')$ results in fact set F''_{med} with the following cells $md''(S''_{med})$, that is equivalent to the original fact set med of Fig. 2:

DM1.md"

date	cost_p	cost_m
23-02-06	356.0	425.0
23-02-06	125.2	1742.0
25-02-06	473.0	903.8

Note that a higher number of measures results in higher expressivity of the fact schema, thus fewer cells, and vice versa. In other words, *split measure* reduces the number of cells, whereas *merge measures* leads to an increased number of cells.

During the last step of the fact integration process the import fact schemas, defined by FA expressions, are mapped to the global DW. If one or more subsets of the facts have overlapping coordinates, the semantic relationship between the fact extensions has to be specified by the merge facts (μ) operator (see below). The μ operator represents the global context of the fact integration process.

Definition 5.12 (μ – merge facts): Let F_{C_1}, F_{C_2} be two fact sets with schemas $G = S_{C_1}, H = S_{C_2}$ with $G \cap H = \{A_1, ..., A_n, M_1, ..., M_m\}$ and *Op* = { *prefer, sum, avg, min, max* } a pre-defined set of operators. Further, let $N \subseteq \{M_1, ..., M_m\}$ be the subset of measures in both fact sets for which an operator $\theta \in$ *Op* should be applied. $\mu(G[N, \theta]H)$ computes the result fact set F' with schema $S' = G \cup H$ and instance $c'(S') = \{t(S') \mid \exists t_g \in G, \exists t_h \in H : t(G \setminus N) = t_g(G \setminus N), t(H \setminus N) = t_h(H \setminus N), t(N) = t_g(N)\ \theta\ t_h(N)\}$.

Example 5.12: Assume a second fact set *'meds'* of some cube *'DM2'* be given as depicted below, with dimension *date = DM1.date* (see example 4.1). Applying F' = $\mu(med\ [\{cost\_p, cost\_m\}, avg]\ meds)$, we obtain as result:

DM2.meds

date	cost_p	cost_m
23-02-06	298.0	607.0
24-02-06	1872.4	941.0

F' ($\mu(med\ [\{cost\_p, cost\_m\}, avg]\ meds)$)

date	cost_p	cost_m
23-02-06	327.0	516.0
24-02-06	1872.4	941.0
25-02-06	473.0	903.8

The FA operator *merge facts* (μ) computes semantically meaningful measures for fact subsets with overlapping coordinates [3]. The *prefer*-operator is appropriate for facts on identical identities (with "stock" semantics [16]), whereas the other set operators can be used for "context-related" entities [3] (with "flow" semantics [16]). Note that in most cases the sum-operator is the best choice. Obviously, only a human is able to choose an appropriate set operator θ according to the semantic relationship between the fact sets.

6. Conclusions

In this paper we introduced a conceptual Federated DW architecture supporting the tightly coupled integration of independent DMs into a global DW schema. While the global schema hides data heterogeneity from the users, the local DMs retain data and schema autonomy. The canonical data model of our FDW architecture supports DMs implemented on relational and multi-dimensional physical platforms alike. As such, the architecture is ideal for independent organizations that want to share their DWs.

Our current and future work involves the implementation of a prototype demonstrating the viability of our approach. The prototype is implemented using off-the-shelf software as far as possible. Particularly, the Oracle 10g and SQL Server 2005 database systems host the dimension and metadata repositories. Moreover, Java technology is used

to implement three additional system components: *query parser* and *query processor* for SQL-MDi [3] as well as the DM schema *integration tool*.

The major challenges to overcome are twofold, namely the optimization of the distributed query plans and the robustness of the system. Firstly, the evaluation of queries over the global schema is the crucial factor for achieving a satisfactory performance in DM federations, as pointed out previously [1, 4]. The prototype will include optimization algorithms for query decomposition and query planning similar to those developed in [1]. Secondly, the transmission of the query results is the second major factor on the usefulness of the FDW approach. In case that one or some of the local DMs do not respond to a query request, the federated system should still evaluate the query result. For this purpose, the prototype will apply an "approximate query answering" technique like in [4].

References

[1] M. O. Akinde, M. H. Böhlen, T. Johnson, L. V. S. Lakshmanan, and D. Srivastava. Efficient OLAP Query Processing in Distributed Data Warehouses. *Inf. Syst.*, 28(1-2): pp. 111–135, 2003.

[2] M. Banek, A. M. Tjoa, and N. Stolba. Integrating Different Grain Levels in a Medical Data Warehouse Federation. In Tjoa and Tho [26], pp. 185–194.

[3] S. Berger and M. Schrefl. Analysing Multi-dimensional Data Across Autonomous Data Warehouses. In Tjoa and Tho [26], pp. 120–133.

[4] J. Bernardino, P. Furtado, and H. Madeira. DWS-AQA: A Cost Effective Approach for Very Large Data Warehouses. In M. A. Nascimento, M. T. Özsu, and O. R. Zaïane, editors, *IDEAS*, pp. 233–242. IEEE Computer Society, 2002.

[5] N. T. Binh, A. M. Tjoa, and O. Mangisengi. Metacube XTM: A Multidimensional Metadata Approach for Semantic Web Warehousing Systems. In Y. Kambayashi, M. K. Mohania, and W. Wöß, editors, *DaWaK*, volume 2737 of *LNCS*, pp. 76–88. Springer, 2003.

[6] L. Cabibbo and R. Torlone. On the Integration of Autonomous Data Marts. In *SSDBM*, pp. 223–234. IEEE Computer Society, 2004.

[7] L. Cabibbo and R. Torlone. Integrating Heterogeneous Multidimensional Databases. In J. Frew, editor, *SSDBM*, pp. 205–214, 2005.

[8] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6): pp. 377–387, 1970.

[9] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *Int. J. Cooperative Inf. Syst.*, 7(2-3): pp. 215–247, 1998.

[10] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data Integration: the Teenage Years. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*, pp. 9–16. ACM, 2006.

[11] C. A. Hurtado, C. Gutiérrez, and A. O. Mendelzon. Capturing Summarizability with Integrity Constraints in OLAP. *ACM Trans. Database Syst.*, 30(3): pp. 854–886, 2005.

[12] C. A. Hurtado and A. O. Mendelzon. Reasoning About Summarizability in Heterogeneous Multidimensional Schemas. In J. V. den Bussche and V. Vianu, editors, *ICDT*, volume 1973 of *LNCS*, pp. 375–389. Springer, 2001.

[13] W. Inmon. *Building the Data Warehouse*. John Wiley & Sons, New York, 4^{th} edition, 2005.

[14] International Organization for Standardization. *ISO/IEC 9075:1992: Title: Information technology — Database languages — SQL*.

[15] R. Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Datawarehouses*. John Wiley & Sons, 2^{nd} edition, 2002.

[16] H.-J. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In Y. E. Ioannidis and D. M. Hansen, editors, *SSDBM*, pp. 132–143. IEEE Computer Society, 1997.

[17] M. Lenzerini. Data Integration: a Theoretical Perspective. In L. Popa, editor, *PODS*, pp. 233–246. ACM, 2002.

[18] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Comput. Surv.*, 22(3): pp. 267–293, 1990.

[19] S. Luján-Mora, J. Trujillo, and I.-Y. Song. A UML Profile for Multidimensional Modeling in Data Warehouses. *Data Knowl. Eng.*, 59(3): pp. 725–769, 2006.

[20] S. Luján-Mora, P. Vassiliadis, and J. Trujillo. Data Mapping Diagrams for Data Warehouse design with UML. In P. Atzeni, W. W. Chu, H. Lu, S. Zhou, and T. W. Ling, editors, *ER*, volume 3288 of *LNCS*, pp. 191–204. Springer, 2004.

[21] O. Mangisengi, W. Eßmayr, J. Huber, and E. Weippl. XML-based OLAP Query Processing in a Federated Data Warehouses. In *ICEIS (1)*, pp. 71–78, 2003.

[22] B. Nuseibeh and S. Easterbrook. Requirements Engineering: a Roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pp. 35–46, New York, NY, USA, 2000. ACM Press.

[23] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB J.*, 10(4): pp. 334–350, 2001.

[24] S. Rizzi, A. Abelló, J. Lechtenbörger, and J. Trujillo. Research in Data Warehouse Modeling and Design: Dead or Alive? In I.-Y. Song and P. Vassiliadis, editors, *DOLAP*, pp. 3–10. ACM, 2006.

[25] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Comput. Surv.*, 22(3): pp. 183–236, 1990.

[26] A. M. Tjoa and N. Tho, editors. *Proceedings of the eighth International Conference on Data Warehousing Knowledge Discovery (DaWaK), September 4–8, Krakow, Poland*, 2006.

[27] R. Torlone and I. Panella. Design and Development of a Tool for Integrating Heterogeneous Data Warehouses. In A. M. Tjoa and J. Trujillo, editors, *DaWaK*, volume 3589 of *LNCS*, pp. 105–114. Springer, 2005.

[28] P. Vassiliadis and T. K. Sellis. A Survey of Logical Models for OLAP Databases. *SIGMOD Record*, 28(4): pp. 64–69, 1999.

[29] T. Vetterli, A. Vaduva, and M. Staudt. Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metadata. *SIGMOD Rec.*, 29(3): pp. 68–75, 2000.