The $\mathcal{GMD}$ Data Model and Algebra for Multidimensional Information*

Enrico Franconi[1] and Anand Kamble[1,2]

[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
franconi@inf.unibz.it, anand.kamble@unibz.it
[2] Department of Computer Science, University of Manchester, UK

Abstract. In this paper we introduce $\mathcal{GMD}$, an abstract but rich data model for representing multidimensional information, equipped with logic-based semantics and seamlessly integrated with a fully compositional algebra also equipped with logic-based semantics. The aim of this work is to propose an homogeneous approach to formally represent all the aspects of multidimensional data, as proposed by the various data models presented in the literature.

1 Introduction

In this paper we introduce a novel data model for multidimensional information, $\mathcal{GMD}$, generalising the $\mathcal{MD}$ data model first proposed in [Cabibbo and Torlone, 1998]. A preliminary introduction to the $\mathcal{GMD}$ data model discussing only the core representational abilities but not the algebra nor the extended features can be found in [Franconi and Kamble, 2003]. The aim of this work is not to propose yet another data model, but to find a very general *logic-based formalism* encompassing the main features of all the proposals for a logical data model in the data warehouse field, as for example summarised in [Vassiliadis and Sellis, 1999]. Our proposal is compatible with all these proposals, making therefore possible a formal comparison of the different expressivities of the models in the literature. The $\mathcal{GMD}$ data model provides a very precise and, we believe, very elegant and uniform way to model multidimensional information. It turns out that most of the proposals in the literature make many hidden assumptions which may harm the understanding of the advantages or disadvantages of the proposal itself. An embedding in our model would make all these assumptions explicit. For lack of space, in the last section of this paper we only briefly suggest the encodings. So far, we have considered, together with the classical basic star and snowflake ER-based models and multidimensional cubes, the logical data models introduced in [Cabibbo and Torlone, 1998, Golfarelli *et al.*, 1998, Agrawal *et al.*, 1997, Gray *et al.*, 1996, Vassiliadis, 1998, Vassiliadis and Skiadopoulos, 2000, Franconi and Sattler, 1999, Gyssens and Lakshmanan, 1997, Tsois *et al.*, 2001, Abello *et al.*, 2001].

* This work has been partially supported by the EU projects Sewasie, KnowledgeWeb, and Interop.

A. Persson and J. Stirna (Eds.): CAiSE 2004, LNCS 3084, pp. 446–462, 2004.

$\mathcal{GMD}$ is completely defined using a logic-based model theoretic approach. We start introducing the notion of data warehouse signature, which has a data warehouse state as its model theoretic counterpart in the semantics. The data model is able to speak in a well founded manner of facts, dimensions, levels, level hierarchies, level attributes, measures, domains. We then introduce the data warehouse schema, which is nothing else than a collection of compositional *fact definitions* (i.e., axioms on the structure of the cubes), which restricts (i.e., constrains) the set of legal data warehouse states associated to the schema. By systematically defining how the various operators used in a fact definition compositionally constrain the legal data warehouse states, we give a formal logic-based account of the $\mathcal{GMD}$ data model. We introduce the aggregation (roll-up) operator with compound aggregation functions, the derivation of attributes, the slice and the multislice operators, the join, union, intersection, and difference operators.

2 The $\mathcal{GMD}$ Logical Data Model

We introduce in this Section the $\mathcal{GMD}$ logical data model. A data warehouse signature gives the building blocks for a data warehouse, but is does not provide any cube definition yet. A data warehouse schema basically introduces the structures of the cubes that will populate the warehouse, together with the types allowed for the components of the structures. Moreover, a schema may contain the definition of complex cubes obtained by composing other cubes through algebraic operations,. The operations that we will introduce are the typical basic OLAP operation.

Definition 1 ($\mathcal{GMD}$ Signature). *A $\mathcal{GMD}$ signature is a tuple $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$, where*

- *$\mathcal{F}$ is a finite set of* ***fact names*** (like SALES, PURCHASES)
- *$\mathcal{D}$ is a finite set of* ***dimension names*** (like Date, Product)
- *$\mathcal{L}$ is a finite set of* ***level names*** (like year, month; brand, category), *each one associated to a finite set of* ***level elements*** (like 2003, 2004; heineken, drink)*; level elements are also called* ***dimension values***
- *$\mathcal{A}$ is a finite set of* ***level attribute names*** (like is-leap, country-of-origin)
- *$\mathcal{M}$ is a finite set of* ***measure names*** (like Price, UnitSales)
- *$\mathcal{V}$ is a finite set of* ***domain names*** (like String, Integer, Boolean), *each one associated to a finite set of* ***domain values****; domain values are also called* ***measure values***

Having just defined $\mathcal{GMD}$ signatures, we introduce now their semantics through a well founded model theory. We first define the notion of a data warehouse state, namely a collection of cells with their dimensions and measures, in agreement with the signature.

Definition 2 (Data Warehouse State). *A* data warehouse state *over the $\mathcal{GMD}$ signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ is a tuple $I = < \Delta, \Lambda, \Gamma, \cdot^I >$, where*

- Δ *is a non-empty finite set of* ***individual facts*** (elements in Δ are the object identifiers for the *cells* in a multidimensional cube)*;*
- Λ *is a finite set of* ***level elements****;*
- Γ *is a finite set of* ***domain values****;*
- $\cdot^I$ *is a function (the* interpretation function*) such that*

$$
\begin{array}{ll}
F^I \subseteq \Delta & \text{for each } F \in \mathcal{F}, \text{ where } F^I \text{ is disjoint from any other } E^I \text{ in } \mathcal{F} \\
L^I \subseteq \Lambda & \text{for each } L \in \mathcal{L}, \text{ where } L^I \text{ is disjoint from any other } H^I \text{ in } \mathcal{L} \\
V^I \subseteq \Gamma & \text{for each } V \in \mathcal{V}, \text{ where } V^I \text{ is disjoint from any other } W^I \text{ in } \mathcal{V} \\
D^I : \Delta \longrightarrow \Lambda & \text{for each } D \in \mathcal{D} \\
M^I : \Delta \longrightarrow \Gamma & \text{for each } M \in \mathcal{M} \\
(A_i^L)^I : L \longrightarrow \Gamma & \text{for each } L \in \mathcal{L} \quad \text{and } A_i^L \in \mathcal{A} \text{ for some } i
\end{array}
$$

The interpretation functions defines a specific data warehouse state given a $\mathcal{GMD}$ signature. It associates to a fact name a set of cells (individual facts), which are meant to form a cube. To each cell corresponds a level element for some dimension name: the sequence of these level elements is meant to be the "coordinate" of the cell. Moreover, to each cell corresponds a value for some measure name.

Up to this stage, no cube definition is considered yet. That is, there is still no schema associated to the signature. Therefore, the dimensions and the measures associated to cells are still completely arbitrary. We now introduce the notion of $\mathcal{GMD}$ schema, which may contain various types of constraints and definitions. The simplest one is the definition of a basic fact, which is a cube whose dimensions and measures are well defined.

Definition 3 ($\mathcal{GMD}$ Schema: Fact Definitions). *A $\mathcal{GMD}$ schema includes a finite set of* fact *definitions of the form*

$$F \doteq E\,\{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\},$$

where $E,F \in \mathcal{F}, D_i \in \mathcal{D}, L_i \in \mathcal{L}, M_j \in \mathcal{M}, V_j \in \mathcal{V}$.

We call the fact name F a defined fact*. We say that F is* based on *E; that the fact F has the listed dimensions each one restricted to the corresponding level; and that the fact F has the listed measures each one restricted to the corresponding domain.. A fact name not appearing at the left hand side of a definition is called an undefined fact. We will generally call* fact *either a defined fact or an undefined fact. A fact based on an undefined fact is called* basic fact*. A fact based on a defined fact is called* aggregated fact*. A fact is* dimensionless *if $n = 0$; it is* measureless *if $m = 0$. The orderings in a defined fact among dimensions and among measures are irrelevant.*

We have here introduced the building block of a $\mathcal{GMD}$ schema: the fact definition. A basic fact corresponds to the base data of any data warehouse: it is the cube structure that contains all the data on which any other cube will be built upon. In the following example, BASIC-SALES is a basic fact, including base data about sale transactions, organised by date, product, and store (which

are the dimensions of the fact) which are respectively restricted to the levels day, product, and store, and with unit sales and sale price as measures:

BASIC-SALES $\doteq$
SALES $\{\text{Date}|_{\text{day}}, \text{Product}|_{\text{product}}, \text{Store}|_{\text{store}}\}$:
$\{\text{UnitSales}|_{\text{int}}, \text{SalePrice}|_{\text{int}}, \text{UnitCost}|_{\text{int}}\}$

Level attribute names are properties associated to levels; for example:

product $\doteq$ $\{\text{prodname}|_{\text{string}}, \text{prodnum}|_{\text{int}}, \text{prodsize}|_{\text{int}}, \text{prodweight}|_{\text{int}}\}$

In the following, in order to give a semantics to fact definitions, we will introduce the notion of *legal* data warehouse state, which is the data warehouse state which conforms to the constraints imposed by the cube definitions. In general, a data warehouse state will be called legal for a given $\mathcal{GMD}$ schema if it is a data warehouse state in the signature of the $\mathcal{GMD}$ schema and it satisfies the additional conditions found in the $\mathcal{GMD}$ schema. In this case, we want that the data warehouse state satisfies the cube conditions as defined in the schema.

Please note that in the following we will omit the $\cdot^I$ interpretation function applied to some symbol whenever this is non ambiguous.

Definition 4 (Legal Data Warehouse State: The Cube Conditions). *A data warehouse state* $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ *over the signature* $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ *is* legal *with respect to a* $\mathcal{GMD}$ *schema if for each fact* $F \doteq E\,\{D_1\,|_{L_1}, \ldots, D_n\,|_{L_n}\} : \{M_1\,|_{V_1}, \ldots, M_m\,|_{V_m}\}$ *in the schema:*

1. *the function associated to a dimension which does not appear in a fact is undefined for its cells:*

   $\forall f.\ F(f) \rightarrow f \notin \mathit{dom}(D)$

   for each $D \in \mathcal{D}$ *such that* $D \neq D_i$ *for each* $i \leq n$, *where* $\mathit{dom}(D)$ *is the domain of the function* D
2. *each cell of a fact has a unique set of dimension values at the appropriate level:*

   $\forall f.\ F(f) \rightarrow \exists l_1, \ldots, l_n.\ D_1(f) = l_1 \wedge L_1(l_1) \wedge \ldots \wedge D_n(f) = l_n \wedge L_n(l_n)$

3. *a set of dimension values identifies a unique cell within a fact:*

   $\forall f, f', l_1, \ldots, l_n.\ F(f) \wedge F(f') \wedge$
   $D_1(f) = l_1 \wedge D_1(f') = l_1 \wedge \ldots \wedge D_n(f) = l_n \wedge D_n(f') = l_n \rightarrow$
   $f = f'$

4. *the function associated to a measure which does not appear in a fact is undefined for its cells:*

   $\forall f.\ F(f) \rightarrow f \notin \mathit{dom}(M)$

   for each $M \in \mathcal{M}$ *such that* $M \neq M_i$ *for each* $i \leq n$

5. *each cell of a fact has a unique set of measures:*

$$\forall f.\ F(f) \rightarrow \exists m_1, \ldots, m_m.\\ M_1(f) = m_1 \wedge V_1(m_1) \wedge \ldots \wedge M_m(f) = m_m \wedge V_m(m_m)$$

Condition 1 states that the level elements associated to a cell of a fact should correspond only to the dimensions declared in the fact definition of the schema. That is, a cell has only the declared dimensions in any legal data warehouse state. Condition 2 states that the level elements associated to a cell of a fact are unique for each dimension declared for the fact in the schema. So, a cell has a unique dimension value for each declared dimension in any legal data warehouse state. Condition 3 states that a sequence of level elements associated to a cell of a fact are associated only to that cell. Therefore, the sequence of dimension values can really be seen as an identifying *coordinate* for the cell. In other words, these conditions enforce the legal data warehouse state to really model a cube according the specification given in the schema. Condition 4 states that the measure values associated to a cell of a fact in a legal data warehouse state should correspond only to the measures explicitly declared in the fact definition of the schema. Condition 5 states that the measure values associated to a cell of a fact are unique for each measure explicitly declared for the fact in the schema. So, a cell has a unique measure value for each declared measure in any legal data warehouse state.

3 Aggregated Cubes

We now introduce the first algebraic component in a $\mathcal{GMD}$ schema: the definition of aggregated cubes. An aggregated cube is based on another defined cube if in the schema it is defined how the measures of the aggregated cube can be computed from the measures of the cube it is based on. Moreover, it is possible to aggregate a cube by changing the levels of the involved dimensions.

Definition 5 ($\mathcal{GMD}$ Schema: Aggregated Facts). *A $\mathcal{GMD}$ schema may also include:*

- *a finite set of* measure definitions *of the form*

$$N \doteq \mathsf{f}(\mathsf{g}(M_1, \ldots, M_k))$$

  where $N, M_1, \ldots, M_k \in \mathcal{M}$, f is an aggregation function $\mathsf{f} : \mathcal{B}(V) \longrightarrow W$ for some $V, W \in \mathcal{V}$, and g is a function (called attribute function*) $\mathsf{g} : V_1 \times \ldots \times V_k \longrightarrow V$ for some $V_1, \ldots, V_k, V \in \mathcal{V}$. $\mathcal{B}(V)$ is the finite set of all bags obtainable from domain values in V whose cardinality is bound by some finite integer Ω.*
- *a partial order $(\mathcal{L}, \leq)$ on the levels in $\mathcal{L}$.*
  We call $\ll$ the immediate predecessor relation on $\mathcal{L}$ induced by $\leq$.

– *a finite set of* roll-up *partial functions between level elements*

$$\rho_{L_i,L_j} : L_i \longrightarrow L_j$$

for each L_i, L_j *such that* $L_i \ll L_j$, *and* $(\rho_{L_i,L_{i_1}} \circ \rho_{L_{i_1},L_{i_2}} \circ \cdots \circ \rho_{L_{i_{n-1}},L_{i_n}} \circ \rho_{L_{i_n},L_j}) = (\rho_{L_i,L_{k_1}} \circ \rho_{L_{k_1},L_{k_2}} \circ \cdots \circ \rho_{L_{k_{n-1}},L_{k_n}} \circ \rho_{L_{k_n},L_j})$ *for any two paths in the partial order between any two level elements* L_i *and* L_j.
We call $\rho^*_{L_i,L_j}$ *the reflexive transitive closure of the roll-up functions inductively defined as follows:*

$$\rho^*_{L_i,L_i} = \mathsf{id}$$
$$\rho^*_{L_i,L_j} = \rho_{L_i,L_k} \circ \rho^*_{L_k,L_j} \qquad \text{for some } k \text{ such that } L_i \ll L_k$$

– *a finite set of level attribute names definitions:*

$$L \doteq \{A_1 \mid_{V_1}, \ldots, A_n \mid_{V_n}\}$$

where $L \in \mathcal{L}$, $A_i \in \mathcal{A}$ *and* $V_i \in \mathcal{V}$ *for each* i, $1 \leq i \leq n$.

Levels and facts are subject to additional syntactical well-foundedness conditions:

– *The connected components of* $(\mathcal{L}, \leq)$ *must have a unique least element each, which is called* basic level.
– *For each undefined fact there can be at most one basic fact based on it (this allows us to disregard undefined facts, which are in one-to-one correspondence with basic facts).*
– *Each aggregated fact must be congruent with the defined fact it is based on, i.e., for each aggregated fact* G *and for the defined fact* F *it is based on such that*

$$F \doteq E\,\{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$$
$$G \doteq F\,\{D_1 \mid_{R_1}, \ldots, D_p \mid_{R_p}\} : \{N_1 \mid_{W_1}, \ldots, N_q \mid_{W_q}\}$$

the following must hold (for some reordering on the dimensions):

- *the dimensions in the aggregated fact* G *are among the dimensions of the fact* F *it is based on:*

$$p \leq n$$

- *the level of a dimension in the aggregated fact* G *is above the level of the corresponding dimension in the fact* F *it is based on:*

$$L_i \leq R_i \quad \text{for each } i \leq p$$

- *each measure* N_i *in the aggregated fact* G *is computed via an aggregation function from some measure of the defined fact* F *it is based on:*
$N_i \doteq \mathsf{f}_i(\mathsf{g}_i(M_{j_i(1)}, \ldots, M_{j_i(k_i)}))$
for each $i \leq q$ *and for some* $k_i \leq m$, *where* j_i *is a permutation function Moreover the range and the domain of the aggregation functions* f_i *and the attribute functions* g_i *should be in agreement each other and with the domains specified in the aggregated fact* G *and in the fact* F *it is based on. If* g_i *is the identity function then it can be omitted. For measureless facts, the aggregated measure may only have the form* $N_i \doteq \mathsf{count}(\star)$.

Measure definitions are used to compute values of measures in an aggregated fact from values of the fact it is based on. For example:

Total-UnitSales $\doteq$ sum(UnitSales)
Total-Profit $\doteq$ sum(SalePrice - UnitCost)

The partial order defines the taxonomy of levels. For example: day $\ll$ month $\ll$ quarter and day $\ll$ week; product $\ll$ type $\ll$ category. When in a schema various hierarchically organised levels are introduced for a dimension, it is also necessary to introduce a roll-up function for them. A roll-up function defines how elements of one level map to elements of a superior level. Since we just require for the roll-up function to be a partial order, it is possible to have elements of a level which roll-up to an upper level, while other elements may skip that upper level to be mapped to a superior one. For example, $\rho_{\text{day,month}}(1/1/01) = \text{Jan-01}$, $\rho_{\text{day,month}}(2/1/01) = \text{Jan-01}$, $\ldots\, \rho_{\text{quarter,year}}(\text{Qtr1-01}) = 2001$, $\rho_{\text{quarter,year}}(\text{Qtr2-01}) = 2001$, ...
The basic level contains the finest grained level elements, on top of which all the facts are identified. For example, store $\ll$ city $\ll$ country; `store` is a basic level.

In the definition above, a precise characterisation of an aggregated fact is given: its dimensions should be among the dimensions of the fact it is based on, its levels should be generalised from the corresponding ones in the fact it is based on, and its measures should be all computed from the fact it is based on. For example, given the basic fact BASIC-SALES:

BASIC-SALES $\doteq$
SALES $\{\text{Date}|_{\text{day}}, \text{Product}|_{\text{product}}, \text{Store}|_{\text{store}}\}$:
$\{\text{UnitSales}|_{\text{int}}, \text{SalePrice}|_{\text{int}}, \text{UnitCost}|_{\text{int}}\}$

the following SALES-BY-MONTH-AND-TYPE is an aggregated fact computed from the BASIC-SALES fact:

SALES-BY-MONTH-AND-TYPE $\doteq$
BASIC-SALES $\{\text{Date}|_{\text{month}}, \text{Product}|_{\text{type}}\}$:
$\{\text{Total-UnitSales}|_{\text{int}}, \text{Avg-SalePrice}|_{\text{real}}, \text{Total-Profit}|_{\text{int}}\}$

with the following aggregated measures:

Total-UnitSales $\doteq$ sum(UnitSales)
Avg-SalePrice $\doteq$ average(SalePrice)
Total-Profit $\doteq$ sum(SalePrice - UnitCost)

Consider now as an example the measureless fact:

STUD-ENROL $\doteq$ ENROL $\{\text{Date}|_{\text{year}}, \text{Student}|_{\text{student}}, \text{Course}|_{\text{course}}\}$

The number of student per year is obtained as follows:

ENROL-BY-YEAR $\doteq$ STUD-ENROL $\{\text{Date}|_{\text{year}}\}$: $\{\text{No\_of\_student}|_{\text{int}}\}$

No_of_students $\doteq$ count($\star$)

Again, a data warehouse state is legal for a given $\mathcal{GMD}$ schema if it is a data warehouse state in the signature of the $\mathcal{GMD}$ schema and it satisfies the additional conditions found in the $\mathcal{GMD}$ schema. In this case, we want that the data warehouse state satisfies the additional *aggregated* cube definitions as defined in the schema.

Definition 6 (Legal Data Warehouse State: Aggregated Cubes). *A data warehouse state* $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ *over the* $\mathcal{GMD}$ *signature* $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ *is* legal *with respect to a* $\mathcal{GMD}$ *schema if, in addition to the conditions stated in Definition 4, (a) the cardinality of* Δ *is smaller than* Ω*, (b) for each level attribute name definition* $L \doteq \{A_1 \mid_{V_1}, \ldots, A_n \mid_{V_n}\}$*, the interpretation of level attribute names* $(A_i^L)^I = L \longrightarrow \Gamma$ *is in agreement with the domains specified in the definition itself, and (c) for each aggregated fact and for the defined fact it is based on in the schema:*

$F \doteq E\ \{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$
$G \doteq F\ \{D_1 \mid_{R_1}, \ldots, D_p \mid_{R_p}\} : \{N_1 \mid_{W_1}, \ldots, N_q \mid_{W_q}\}$
$N_i \doteq \mathsf{f}_i(\mathsf{g}_i(M_{j_i(1)}, \ldots, M_{j_i(k)}))$ *for each* $i \leq q$ *and for some* $k \leq m$*, where* j *is a permutation function*

each aggregated measure function actually computes the aggregation of the values in the corresponding measure of the fact the aggregation is based on:

$$\begin{aligned}\forall g, v.\ N_i(g) = v \leftrightarrow \exists r_1, \ldots, r_p.\ & G(g) \wedge D_1(g) = r_1 \wedge \ldots \wedge D_p(g) = r_p \wedge \\ & v = \mathsf{f}_i(\{\!\!\{\mathsf{g}_i(M_{j_i(1)}(f), \ldots, M_{j_i(k)}(f)) \mid \\ & \qquad \exists l_1, \ldots, l_p.\ F(f) \wedge \\ & \qquad D_1(f) = l_1 \wedge \ldots \wedge D_p(f) = l_p \wedge \\ & \qquad \rho^*_{L_1,R_1}(l_1) = r_1 \wedge \ldots \wedge \rho^*_{L_p,R_p}(l_p) = r_p\}\!\!\})\end{aligned}$$

for each $i \leq q$*, where* $\{\!\!\{ \cdot \}\!\!\}$ *denotes a bag.*

The legal data warehouse condition expressed above guarantees that if a fact is the aggregation of another fact, then in a legal data warehouse state the measures associated to the cells of the aggregated cube should be actually computed by applying the aggregation function to the measures of the corresponding cells in the original cube. The correspondence between a cell in the aggregated cube and a set of cells in the original cube is found by looking how their coordinates – which are level elements – are mapped through the roll-up function dimension by dimension.

To sum up, a legal data warehouse state for a $\mathcal{GMD}$ schema is a bunch of multidimensional cubes, whose cells carry measure values. Each cube conforms to the fact definition given in the $\mathcal{GMD}$ schema, i.e., the coordinates are in agreement with the dimensions and the levels specified, and the measures are of the correct type. If a cube is the aggregation of another cube, in a legal data warehouse state it is enforced that the measures of the aggregated cubes are correctly computed from the measures of the original cube.

4 Example

The following $\mathcal{GMD}$ schema summarises the examples shown in the previous Sections:

- Signature:
  - $\mathcal{F}$ = {SALES, BASIC-SALES, SALES-BY-MONTH-AND-TYPE, PURCHASES}
  - $\mathcal{M}$ = {UnitSales, Price, Total-UnitSales, Avg-Price}
  - $\mathcal{D}$ = {Date, Product, Store}
  - $\mathcal{L}$ = {day, week, month, quarter, year, product, type, category, brand, store, city, country}
    day = {1/1/01, 2/1/01, ..., 1/1/02, 2/1/02, ...}
    month = {Jan-01, Feb-01, ..., Jan-02, Feb-02, ...}
    quarter = {Qtr1-01, Qtr2-01, ..., Qtr1-02, Qtr2-02, ...}
    year = {2001, 2002}
    ...
  - $\mathcal{V}$ = {int, real, string}
  - $\mathcal{A}$ = {dayname, prodname, prodsize, prodweight, storenum}
- Partial order over levels:
  - day $\ll$ month $\ll$ quarter $\ll$ year, day $\ll$ week; `day` is a basic level
  - product $\ll$ type $\ll$ category, product $\ll$ brand; `product` is a basic level
  - store $\ll$ city $\ll$ country; `store` is a basic level
- Roll-up functions:
  $\rho_{\text{day,month}}(1/1/01)$ = Jan-01, $\rho_{\text{day,month}}(2/1/01)$ = Jan-01, ...
  $\rho_{\text{month,quarter}}$(Jan-01) = Qtr1-01, $\rho_{\text{month,quarter}}$(Feb-01) = Qtr1-01, ...
  $\rho_{\text{quarter,year}}$(Qtr1-01) = 2001, $\rho_{\text{quarter,year}}$(Qtr2-01) = 2001, ...
  $\rho^*_{\text{day,year}}(1/1/01)$ = 2001, $\rho^*_{\text{day,year}}(2/1/01)$ = 2001, ...
  ...
- Level Attribute names:
  day $\doteq$ {dayname$|_{\text{string}}$, daynum$|_{\text{int}}$}
  product $\doteq$ {prodname$|_{\text{string}}$, prodnum$|_{\text{int}}$, prodsize$|_{\text{int}}$, prodweight$|_{\text{int}}$}
  store $\doteq$ {storename$|_{\text{string}}$, storenum$|_{\text{int}}$, address$|_{\text{string}}$}
- Facts:
  BASIC-SALES $\doteq$
  SALES {Date$|_{\text{day}}$, Product$|_{\text{product}}$, Store$|_{\text{store}}$} : {UnitSales$|_{\text{int}}$, SalePrice$|_{\text{int}}$}
  SALES-BY-MONTH-AND-TYPE $\doteq$
  BASIC-SALES {Date$|_{\text{month}}$, Product$|_{\text{type}}$} : {Total-UnitSales$|_{\text{int}}$, Avg-SalePrice$|_{\text{real}}$}
- Measures:
  Total-UnitSales $\doteq$ sum(UnitSales)
  Avg-SalePrice $\doteq$ average(SalePrice)

A possible legal data warehouse state for (part of) the previous example $\mathcal{GMD}$ schema is shown in the following.

BASIC-SALES^I = $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$
SALES-BY-MONTH-AND-TYPE^I = $\{g_1, g_2, g_3, g_4, g_5, g_6\}$

Date(s_1) = 1/1/01	Product(s_1) = Organic-milk-1l	Store(s_1) = Fair-trade-central
Date(s_2) = 7/1/01	Product(s_2) = Organic-yogh-125g	Store(s_2) = Fair-trade-central
Date(s_3) = 7/1/01	Product(s_3) = Organic-milk-1l	Store(s_3) = Ali-grocery
Date(s_4) = 10/2/01	Product(s_4) = Organic-milk-1l	Store(s_4) = Barbacan-store
Date(s_5) = 28/2/01	Product(s_5) = Organic-beer-6pack	Store(s_5) = Fair-trade-central
Date(s_6) = 2/3/01	Product(s_6) = Organic-milk-1l	Store(s_6) = Fair-trade-central
Date(s_7) = 12/3/01	Product(s_7) = Organic-beer-6pack	Store(s_7) = Ali-grocery

UnitSales(s_1) = 100	EuroSalePrice(s_1) = 71,00
UnitSales(s_2) = 500	EuroSalePrice(s_2) = 250,00
UnitSales(s_3) = 230	EuroSalePrice(s_3) = 138,00
UnitSales(s_4) = 300	EuroSalePrice(s_4) = 210,00
UnitSales(s_5) = 210	EuroSalePrice(s_5) = 420,00
UnitSales(s_6) = 150	EuroSalePrice(s_6) = 105,00
UnitSales(s_7) = 100	EuroSalePrice(s_7) = 200,00

$\text{Date}(g_1) = \text{Jan-01}$ $\text{Product}(g_1) = \text{Dairy}$
$\text{Date}(g_2) = \text{Feb-01}$ $\text{Product}(g_2) = \text{Dairy}$
$\text{Date}(g_3) = \text{Jan-01}$ $\text{Product}(g_3) = \text{Drink}$
$\text{Date}(g_4) = \text{Feb-01}$ $\text{Product}(g_4) = \text{Drink}$
$\text{Date}(g_5) = \text{Mar-01}$ $\text{Product}(g_5) = \text{Dairy}$
$\text{Date}(g_6) = \text{Mar-01}$ $\text{Product}(g_6) = \text{Drink}$

$\text{Total-UnitSales}(g_1) = 830$ $\text{Avg-EuroSalePrice}(g_1) = 153{,}00$
$\text{Total-UnitSales}(g_2) = 300$ $\text{Avg-EuroSalePrice}(g_2) = 210{,}00$
$\text{Total-UnitSales}(g_3) = 0$ $\text{Avg-EuroSalePrice}(g_3) = 0{,}00$
$\text{Total-UnitSales}(g_4) = 210$ $\text{Avg-EuroSalePrice}(g_4) = 420{,}00$
$\text{Total-UnitSales}(g_5) = 150$ $\text{Avg-EuroSalePrice}(g_5) = 105{,}00$
$\text{Total-UnitSales}(g_6) = 100$ $\text{Avg-EuroSalePrice}(g_6) = 200{,}00$

$\text{daynum(day)} = 1$ $\text{prodweight(product)} = 100\text{gm}$ $\text{storenum(store)} = \text{S101}$

5 $\mathcal{GMD}$ Full Algebra

On top the data model described so far, we now introduced a full cube algebra, which has the very desirable property of being compositional (any new cube is always introduced by means of an additional definition in the schema, possibly making use of other cube definitions), and of being equipped with a formal semantics (in the sense that a logic based definition of *legal* data warehouse state is given for all valid algebraic constructs). We introduce the operator to add derived measures to cubes, the operator to create slices and multislices to cubes, the join operator between two cubes, and finally the union, the intersection, and the difference operators between pairs of cubes.

We start by defining the *derived measure* operator: a new cube G can be computed from a cube F by just adding a measure whose value can be computed from the other measures of F.

Definition 7 (Derived Measures). *A $\mathcal{GMD}$ schema may also include definitions of the kind:*

$$F \doteq E\,\{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$$
$$G \doteq F + \{N \mid_V\}$$
$$N \doteq \mathsf{g}(M_{j(1)}, \ldots, M_{j(k)})$$

where $k \leq m$, and g is a function $\mathsf{g} : V_1 \times \ldots \times V_k \longrightarrow V$ for some $V_1, \ldots, V_k, V \in \mathcal{V}$ in agreement with the various domain constraints.

A data warehouse state $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ over the $\mathcal{GMD}$ signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ is legal *with respect to a $\mathcal{GMD}$ schema if, in addition to the conditions to be satisfied by the other parts of the schema, the following holds:*

$$\begin{aligned}\forall f, g, l_1, \ldots, l_n, v_1, \ldots, v_m.\ & (F(f)\ \wedge D_1(f) = l_1 \wedge \ldots \wedge D_n(f) = l_n\ \wedge \\ & M_1(f) = v_1 \wedge \ldots \wedge M_m(f) = v_m) \\ & \leftrightarrow \\ & (G(g) \wedge D_1(g) = l_1 \wedge \ldots \wedge D_n(g) = l_n\ \wedge \\ & M_1(g) = v_1 \wedge \ldots \wedge M_m(g) = v_m \wedge \\ & N(g) = \mathsf{g}(M_{j(1)}(f), \ldots, M_{j(k)}(f)))\end{aligned}$$

Moreover, the fact G should satisfy the cube conditions as specified in Definition 4 with respect to the same dimension, levels and measures as for fact F with the additional measure N.

As an example we could have as part of the schema:

DERIVED-SALES $\doteq$ SALES + $\{\text{Profit}|_{\text{int}}\}$
Profit $\doteq$ SalePrice - UnitCost

Two selection operators are also available in the full $\mathcal{GMD}$ algebra. The slice operation simply selects the cells of a cube corresponding to specific values for some dimension, resulting in a cube which contains a subset of the cells of the original one and fewer dimensions. The multislice allows for the selection of ranges of values for a dimension, so that the resulting cube will contain a subset of the cells of the original one but retains the selected dimension.

Definition 8 (Slice/Multislice). *A $\mathcal{GMD}$ schema may also include definitions of the kind:*

(slice)
$F \doteq E\ \{D_1\ |_{L_1}, \ldots, D_n\ |_{L_n}\} : \{M_1\ |_{V_1}, \ldots, M_m\ |_{V_m}\}$
$G \doteq F[D_{i+1}\ |_{l_{i+1}}, \ldots, D_n\ |_{l_n}]$

where $1 \leq i \leq n$ and l_j is level element of a level L_j for each j, $i \leq j \leq n$.

(multislice)
$F \doteq E\ \{D_1\ |_{L_1}, \ldots, D_n\ |_{L_n}\} : \{M_1\ |_{V_1}, \ldots, M_m\ |_{V_m}\}$
$G \doteq F[D_{i+1}\ |_{X_{i+1}}, \ldots, D_n\ |_{X_n}]$

where $1 \leq i \leq n$ and $X_j \subseteq L_j$ for $i \leq j \leq n$.

A data warehouse state $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ over the $\mathcal{GMD}$ signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ is legal *with respect to a $\mathcal{GMD}$ schema if, in addition to the conditions to be satisfied by the other parts of the schema, the following holds:*

(slice)

$$\begin{aligned}\forall f, g, l_1, \ldots, l_i, v_1, \ldots, v_m.\ & (G(g) \wedge D_1(g) = l_1 \wedge \ldots \wedge D_i(g) = l_i \wedge \\ & M_1(g) = v_1 \wedge \ldots \wedge M_m(g) = v_m) \leftrightarrow \\ & (F(f) \wedge D_1(f) = l_1 \wedge \ldots \wedge D_i(f) = l_i \wedge \\ & D_{i+1}(f) = l_{i+1} \wedge \ldots \wedge D_n(f) = l_n \wedge \\ & M_1(f) = v_1 \wedge \ldots \wedge M_m(f) = v_m)\end{aligned}$$

Moreover, the fact G should satisfy the cube conditions as specified in Definition 4 with respect to the same dimension, levels and measures as for fact F less the dimensions $D_{i+1}, \ldots, D_n$.

(multislice)

$$\begin{aligned}\forall f, g, l_1, \ldots, l_i, l_{i+1} \in X_{i+1}, \ldots, l_n \in X_n, v_1, \ldots, v_m.\ & \\ (G(g) \wedge D_1(g) = l_1 \wedge \ldots \wedge D_i(g) = l_i \wedge & \\ D_{i+1}(g) = l_{i+1} \wedge \ldots \wedge D_n(g) = l_n \wedge & \\ M_1(g) = v_1 \wedge \ldots \wedge M_m(g) = v_m) \leftrightarrow & \\ (F(f) \wedge D_1(f) = l_1 \wedge \ldots \wedge D_i(f) = l_i \wedge & \\ D_{i+1}(f) = l_{i+1} \wedge \ldots \wedge D_n(f) = l_n \wedge & \\ M_1(f) = v_1 \wedge \ldots \wedge M_m(f) = v_m) &\end{aligned}$$

Moreover, the fact G should satisfy the cube conditions as specified in Definition 4 with respect to the same dimension, levels and measures as for fact F but for the dimensions $D_{i+1}, \ldots, D_n$ the levels being $X_{i+1}, \ldots, X_n$.

As an example we could have as part of the schema:

SALES-BY-TYPE-IN-JAN'02 $\doteq$
SALES-BY-MONTH-AND-TYPE [Date$|_{\text{jan'02}}$]
SALES-BY-MONTH-AND-TYPE-IN-1ST-QTR'02 $\doteq$
SALES-BY-MONTH-AND-TYPE [Date$|_{\{\text{jan'02,feb'02,mar'02}\}}$]

The $\mathcal{GMD}$ algebra includes a join operation defined only between cubes sharing the same dimensions and the same levels. We argue that a more general join operation is meaningless in a cube algebra, since it may leads to cubes whose measures are no more consistent.

Definition 9 (Join). *A $\mathcal{GMD}$ schema may also include definitions of the kind:*

$F \doteq$ *E-1* $\{D_1 |_{L_1}, \ldots, D_n |_{L_n}\} : \{M_1 |_{V_1}, \ldots, M_m |_{V_m}\}$
$G \doteq$ *E-2* $\{D_1 |_{L_1}, \ldots, D_n |_{L_n}\} : \{N_1 |_{W_1}, \ldots, N_q |_{W_q}\}$
$H \doteq F \bowtie G$

A data warehouse state $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ over the $\mathcal{GMD}$ signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ is legal *with respect to a $\mathcal{GMD}$ schema if, in addition to the conditions to be satisfied by the other parts of the schema, the following holds:*

$$\begin{aligned}
&\forall f, g, h, l_1, \ldots, l_n, v_1, \ldots, v_m, w_1, \ldots, w_q. \\
&\quad (H(h) \wedge D_1(h) = l_1 \wedge \ldots \wedge D_n(h) = l_n \wedge \\
&\quad M_1(h) = v_1 \wedge \ldots \wedge M_m(h) = v_m \wedge \\
&\quad N_1(h) = w_1 \wedge \ldots \wedge N_q(h) = w_q) \leftrightarrow \\
&\qquad (F(f) \wedge D_1(f) = l_1 \wedge \ldots \wedge D_n(f) = l_n \wedge \\
&\qquad M_1(f) = v_1 \wedge \ldots \wedge M_m(f) = v_m \wedge \\
&\qquad G(g) \wedge D_1(g) = l_1 \wedge \ldots \wedge D_n(g) = l_n \wedge \\
&\qquad N_1(g) = w_1 \wedge \ldots \wedge N_q(g) = w_q)
\end{aligned}$$

Moreover, the fact H should satisfy the cube conditions as specified in Definition 4 with respect to the same dimension, levels as for the facts F, G and the union of the measures of F, G.

As an example we could have as part of the schema:

SALES-BY-MONTH-AND-TYPE $\doteq$
BASIC-SALES $\{\text{Date}|_{\text{month}}, \text{Product} |_{\text{product}}, \text{Store} |_{\text{store}}\}$:
$\{\text{Total-Sale-Price} |_{real}\}$
PURCHASES-BY-MONTH-AND-TYPE $\doteq$
BASIC-PURCHASES $\{\text{Date}|_{\text{month}}, \text{Product} |_{\text{product}}, \text{Store} |_{\text{store}}\}$:
$\{\text{Total-Cost} |_{real}\}$
SALES&PURCHASES-BY-MONTH-AND-TYPE $\doteq$
SALES-BY-MONTH-AND-TYPE $\bowtie$ PURCHASES-BY-MONTH-AND-TYPE

Finally, we introduce briefly the union, intersection, and difference operators. In order to be compatible for these operators, two facts should have the same dimensions, levels, and measures; we do not allow a general union operator like the one proposed in [Gray *et al.*, 1996], but it can be shown how the algebra proposed by Gray can be reconstructed using the $\mathcal{GMD}$ algebra.

Definition 10 (Union, Intersection, Difference). *A $\mathcal{GMD}$ schema may also include definitions of the kind:*

$$F \doteq \textit{E-1}\ \{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$$
$$G \doteq \textit{E-2}\ \{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$$
$$H \doteq F \bigotimes G$$

where $\bigotimes$ is one of $\{\bigcup, \bigcap, \setminus\}$.

A data warehouse state $I = < \Delta, \Lambda, \Gamma, \cdot^I >$ over the $\mathcal{GMD}$ signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$ is legal *with respect to a $\mathcal{GMD}$ schema if, in addition to the conditions to be satisfied by the other parts of the schema, the following holds:*

$$\begin{array}{l}
\forall f, g, h, l_1, \ldots, l_n, v_1, \ldots, v_m. \\
\quad (H(h) \wedge D_1(h) = l_1 \wedge \ldots \wedge D_n(h) = l_n \wedge \\
\quad M_1(h) = v_1 \wedge \ldots \wedge M_m(h) = v_m) \leftrightarrow \\
\qquad (\ (F(f) \wedge D_1(f) = l_1 \wedge \ldots \wedge D_n(f) = l_n \wedge \\
\qquad M_1(f) = v_1 \wedge \ldots \wedge M_m(f) = v_m) \\
\qquad \bigoplus \\
\qquad (G(g) \wedge D_1(g) = l_1 \wedge \ldots \wedge D_n(g) = l_n \wedge \\
\qquad M_1(g) = v_1 \wedge \ldots \wedge M_m(g) = v_m)\ )
\end{array}$$

where $\bigoplus$ is "$\vee$" in the case of union*, is "$\wedge$" in the case of* intersection*, is "$\wedge\neg$" in the case of* difference*. Moreover, the fact H should satisfy the cube conditions as specified in Definition 4 with respect to the same dimension, levels, and measures as for the facts F, G.*

6 Related Work

As we were mentioning in the introduction, one outcome of the formal definition of the $\mathcal{GMD}$ data model is in the full encoding of many data warehouse logical data models as $\mathcal{GMD}$ schemas. We are able in this way to give an homogeneous semantics (in terms of legal data warehouse states) to the logical model and the algebras proposed in all these different approaches. The star and the snowflake schemas, Gray's cube, Agrawal's and Vassiliadis' models, $\mathcal{MD}$ and other multidimensional conceptual data models can be captured uniformly by $\mathcal{GMD}$. In this way it is possible to formally understand the real differences in expressivity of the various models.

The classical relational based *star model* comprises a single fact table at the centre and multiple dimension tables around connected to it. The fact table consists of a set of dimension attributes forming the primary key and several

measure non-key numeric attributes. Each dimension attribute is also foreign key to a dimension table. Each dimension table consists of attributes including a primary key and several non-key attributes, and it represents the properties of elements of a single level for the dimension. The star model does not explicitly provide the support for dimension hierarchies. In the *snowflake model* the dimension hierarchy is explicitly represented by normalising the dimension tables, while in the *fact constellation model* multiple fact tables may share dimension tables. It can be easily seen how the star, the snowflake, and the fact constellation models can be encoded into corresponding $\mathcal{GMD}$ schemas, in a way that the legal data warehouse states identified by the encoded $\mathcal{GMD}$ schema are the same possible instantiations of the original schema.

The other classical data model for multidimensional data is the *cube model*, which contains n-dimensional arrays where each dimension is associated to a hierarchy of levels of consolidated data. The data is represented by means of matrices whose indexes range over natural numbers. This structure has an obvious mapping into the $\mathcal{GMD}$ data model.

The $\mathcal{MD}$ data model was introduced by [Cabibbo and Torlone, 1998] as a first proposal of a homogeneous logical data model for multidimensional data. The central element of an $\mathcal{MD}$ schema is a *f-table* representing factual multidimensional data. A *f-table* is the abstract logical representation of a multidimensional cube, and it is a function associating symbolic coordinates (one per involved dimension) to measures. Dimensions are organised into hierarchies of levels according to the various granularity of basic data. A $\mathcal{MD}$ dimension consists of a finite set of levels with partial ordering on these levels. Within a dimension, levels are related through roll-up functions. Levels can have level descriptions that provide information about the levels. It is possible to encode a $\mathcal{MD}$ schema into an equivalent $\mathcal{GMD}$ schema. However, $\mathcal{GMD}$ is richer than $\mathcal{MD}$. First, in $\mathcal{GMD}$ each set of connected levels is rooted through a basic level – the unique least level in the dimension hierarchy, and there is partial order among the levels (a notion stressed by, e.g., [Vassiliadis, 1998]); whereas in $\mathcal{MD}$ a uniqueness of a least element (level) has not been considered in the partial ordering of levels. Then, in $\mathcal{MD}$, the join of fact tables is like a join of relational tables; the join contains non-common dimensions apart from the common ones. An instance of the resulting fact table (join) may be erroneous for an f-table entry; whereas in $\mathcal{GMD}$, a join is possible only between the cubes sharing same dimensions and levels, and the join takes place only on common dimension values so that the each cell is consistent with respect to the measure values. Moreover, $\mathcal{GMD}$, supports correct aggregations (summarisations) irrespective of the path chosen for the rolling up thanks to the application of transitive reflexive roll-up functions, and an aggregated fact can be computed from another aggregated fact. Finally, the $\mathcal{GMD}$ model includes in its core definition all the algebraic operators, which are consistently and, most importantly, compositionally defined over the basic notion of cube.

In [Agrawal *et al.*, 1997], a logical data model has been proposed based on the notion of multidimensional cube, together with an algebraic query language

over it. The model is characterised by a symmetric treatment for dimensions and measures. The most notable difference with $\mathcal{GMD}$ is the lack of compositionality in the basic definitions of the model and in the algebraic operators. The dimension hierarchies are implemented using a special query language operator whereas level hierarchies are part of the core $\mathcal{GMD}$ model. The algebra is grounded on operators dealing with *destruction* and *restriction* of dimensions, general join of cubes, and merge among levels within a cube. For example, the aggregation is defined as the join of two cubes with a subsequent merge. It can be shown that the algebra of cubes of [Agrawal *et al.*, 1997] can be encoded in $\mathcal{GMD}$.

The cube operator introduced in [Gray *et al.*, 1996] expands a relational table by aggregating over all possible combinations of the attributes of the relation. A cube operator is an n-dimensional generalisation of the SQL GROUP BY operator. For n attributes in the select-list, it results in 2^n group-by computations. The data cube operator builds a table containing all these values. The $\mathcal{GMD}$ data model can not represent directly a cube generated by means of the cube operator, since it is impossible to directly represent in the same fact table data at different levels for the same dimension; this is captured in Gray's cube with the special aggregated level element all. We propose a possible encoding of the cube operator in $\mathcal{GMD}$ which goes around this problem. The idea is based on extending the set of level element names with the special constant all for each level, and then building the complete cube by means of a sequence of aggregations and unions. It should be noted, however, that the introduction of the special constant all may affect the consistency of the aggregated cubes in general, and so it should be used only in the special case of generating the completed cube, which itself shouldn't be used anymore for further aggregations (like in Gray's approach).

[Vassiliadis and Skiadopoulos, 2000], propose a logical model equipped with a lattice of dimension levels, so that the values of each dimension level can be grouped into a single all special value. It is emphasised that a cube is always a view over an underlying data set – which corresponds to the $\mathcal{GMD}$ basic level. Cubes (views) are computed over a base cube which holds the most detailed data. Aggregation is carried out with a special operator called *navigate* always defined from the base data, while in $\mathcal{GMD}$ it is possible to aggregate already aggregated data. It is possible to show that the composition of aggregations in $\mathcal{GMD}$ down to the basic level corresponds to the navigation as proposed by [Vassiliadis and Skiadopoulos, 2000].

In [Golfarelli *et al.*, 1998], a multidimensional data model called *Dimensional Fact Model* (DFM) has been introduced. DFM is a graphical model, independent on the logical (multidimensional/relational) model, but substantially based on the star model. DFM is a quasi-tree (weakly connected) of attributes, whose root is a fact. A fact is represented by a box at the centre of the diagram associated with a fact name and its multiple measures; the dimension attributes are linked to the fact and are represented by circles. Non-dimensional attributes are leaf nodes linked to the fact. Subtrees rooted in dimensions represent the

hierarchy of levels for the dimension. The hierarchies are constrained by x-to-one (i.e., many-to-one, etc) relationships between nodes (representing the levels). Each n-tuple of values taken from the domains of n dimensions of a fact defines an elemental cell called a primary fact instance where one unit of information for the data warehouse can be represented. Aggregation at different levels of abstraction (e.g. roll-up) is called a secondary fact instance which aggregates the set of primary fact instances. A query language is also provided for computing fact instances (primary and secondary) with n-dimensions and selections (using boolean predicates). DFM can easily be encoded in all its aspects as a $\mathcal{GMD}$ schema.

The most important aspect of DFM is that it is associated with a very powerful data warehouse design methodology and many supporting tools are available for it. Since it is possible to encode in $\mathcal{GMD}$ various steps of the data warehouse design methodology, it becomes possible to support those stages of the methodology by means of some automated tool which can be proved correct with respect to the $\mathcal{GMD}$ semantics. Our main line of research for the future is to extend and adapt the powerful data warehouse design methodologies proposed by [Golfarelli *et al.*, 1998] to the full $\mathcal{GMD}$ data model. We will do this in the spirit of the work done in [Franconi and Sattler, 1999].

[Lehner *et al.*, 1998] introduce a generalised multidimensional normal form (GMNF) which ensures the summarisability but restricts the roll-up function between the hierarchical levels to be total (similar to [Hutardo *et al.*, 1999]), whereas in $\mathcal{GMD}$ the relationship between hierarchical levels is a more general partial roll-up function. [Jagadish *et al.*, 1999] model hierarchies in a relational way by using SQL, overcoming the limitations on modelling hierarchies in the snowflake schema. In this way there is a reduction in the number of joins that are required to join the level hierarchies (dimension tables) while querying the snowflake schema. Also in this case the roll-up functions are considered total.

7 Conclusions

In this paper we have introduced the $\mathcal{GMD}$ data model and algebra, an abstract but rich data model for representing multidimensional information, equipped with logic-based semantics and seamlessly integrated with a fully compositional algebra also equipped with logic-based semantics. The aim of this work is to propose an homogeneous approach to formally represent all the aspects of multidimensional data, as proposed by the various data models presented in the literature. We hinted how actually $\mathcal{GMD}$ captures these various proposals. Starting with the $\mathcal{GMD}$ data model, our current research work is to adapt the conceptual data warehouse methodologies appeared in the literature – in the spirit of [Golfarelli *et al.*, 1998, Franconi and Sattler, 1999].

References

[Abello *et al.*, 2001] A. Abello, J. Samos, and F. Saltor. Understanding analysis dimensions in a multidimensional object-oriented model. In *Proc. of the International Workshop on Design and Management of Data Warehouses (DMDW'2001),Interlaken, Switzerland*, pages 4–1–4–9, 2001.

[Agrawal *et al.*, 1997] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proc. of ICDE-97*, 1997.

[Cabibbo and Torlone, 1998] Luca Cabibbo and Riccardo Torlone. A logical approach to multidimensional databases. In *Proc. of EDBT-98*, 1998.

[Franconi and Kamble, 2003] Enrico Franconi and Anand S. Kamble. The $\mathcal{GMD}$ data model for multidimensional information. In *Proc. 5th International Conference on Data Warehousing and Knowledge Discovery*, pages 55–65, 2003.

[Franconi and Sattler, 1999] E. Franconi and U. Sattler. A data warehouse conceptual data model for multidimensional aggregation. In *Proc. of the Workshop on Design and Management of Data Warehouses (DMDW-99)*, 1999.

[Golfarelli *et al.*, 1998] M. Golfarelli, D. Maio, and S. Rizzi. The dimensional fact model: a conceptual model for data warehouses. *IJCIS*, 7(2-3):215–247, 1998.

[Gray *et al.*, 1996] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: a relational aggregation operator generalizing group-by, cross-tabs and subtotals. In *Proc. of ICDE-96*, 1996.

[Gyssens and Lakshmanan, 1997] M. Gyssens and L.V.S. Lakshmanan. A foundation for multi-dimensional databases. In *Proc. of VLDB-97*, pages 106–115, 1997.

[Hutardo *et al.*, 1999] Carlos Hutardo, Alberto Mendelzon, and A. Vaisman. Maintaining data cube under dimension updates. In *Proc. 15th IEEE-ICDE International Conference*, 1999.

[Jagadish *et al.*, 1999] H.V. Jagadish, Laks V.S. Lakshmanan, and Divesh Srivastava. What can hierarchies do for data warehouses? In *Proc. 25th International Conference on Very Large Databases (VLDB)*, pages 530–541, 1999.

[Lehner *et al.*, 1998] W. Lehner, H. Albrecht, and H. Wedekind. Normal forms for multidimensional databases. In *Proc. 10th International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 63–73, 1998.

[Tsois *et al.*, 2001] A. Tsois, N. Karayiannidis, and T. Sellis. MAC: Conceptual data modelling for OLAP. In *Proc. of the International Workshop on Design and Management of Warehouses (DMDW-2001)*, pages 5–1–5–13, 2001.

[Vassiliadis and Sellis, 1999] P. Vassiliadis and T. Sellis. A survey of logical models for OLAP databases. In *SIGMOD Record*, volume 28, pages 64–69, December 1999.

[Vassiliadis and Skiadopoulos, 2000] P. Vassiliadis and S. Skiadopoulos. Modelling and optimisation issues for multidimensional databases. In *Proc. of CAiSE-2000*, pages 482–497, 2000.

[Vassiliadis, 1998] P. Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In *Proc. of the 10th SSDBM Conference*, Capri, Italy, July 1998.