The $\mathcal{GMD}$ Data Model for Multidimensional Information: a brief introduction

Enrico Franconi and Anand Kamble

Faculty of Computer Science, Free Univ. of Bozen-Bolzano, Italy
franconi@inf.unibz.it — anand.kamble@unibz.it

Abstract. In this paper we introduce a novel data model for multidimensional information, $\mathcal{GMD}$, generalising the $\mathcal{MD}$ data model first proposed in Cabibbo et al (EDBT-98). The aim of this work is not to propose yet another multidimensional data model, but to find the general precise formalism encompassing all the proposals for a logical data model in the data warehouse field. Our proposal is compatible with all these proposals, making therefore possible a formal comparison of the differences of the models in the literature, and to study formal properties or extensions of such data models. Starting with a logic-based definition of the semantics of the $\mathcal{GMD}$ data model and of the basic algebraic operations over it, we show how the most important approaches in DW modelling can be captured by it. The star and the snowflake schemas, Gray's cube, Agrawal's and Vassiliadis' models, $\mathcal{MD}$ and other multidimensional conceptual data 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, their limits, and their potentials.

1 Introduction

In this short 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]. The aim of this work is not to propose yet another data model, but to find the most general formalism encompassing 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. We believe that the $\mathcal{GMD}$ data model is already very useful since it 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.

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]. A complete account of both the $\mathcal{GMD}$ data model (including and extended algebra) and of the various encodings can be found in [Franconi and Kamble, 2003];

in this paper we just give a brief introduction to the basic principles of the data model.

$\mathcal{GMD}$ is completely defined using a logic-based approach. We start introducing a data warehouse schema, which is nothing else than a set of *fact definitions* 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 constrain the legal data warehouse states, we give a formal logic-based account of the $\mathcal{GMD}$ data model.

2 The syntax of the $\mathcal{GMD}$ data model

We introduce in this Section the notion of data warehouse schema. 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. The definition of a $\mathcal{GMD}$ schema that follows is explained step by step.

Definition 1 ($\mathcal{GMD}$ schema). *Consider the signature $< \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{M}, \mathcal{V}, \mathcal{A} >$, where $\mathcal{F}$ is a finite set of fact names, $\mathcal{D}$ is a finite set of dimension names, $\mathcal{L}$ is a finite set of level names – each one associated to a finite set of level element names, $\mathcal{M}$ is a finite set of measure names, $\mathcal{V}$ is a finite set of domain names – each one associated to a finite set of values, $\mathcal{A}$ is a finite set of level attributes.*

> ➠ We have just defined the alphabet of a data warehouse: we may have fact names (like SALES, PURCHASES), dimension names (like Date, Product), level name (like year, month, product-brand, product-category) and their level elements (like 2003, 2004, heineken, drink), measure names (like Price, UnitSales), domain names (like integers, strings), and level attributes (like is-leap, country-of-origin).

A $\mathcal{GMD}$ schema includes:

– *a finite set of* fact *definitions of the form*

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

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*, and we say that F is* based on *E. 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 $\{\text{Date}|_{\text{day}}, \text{Product}|_{\text{product}}, \text{Store}|_{\text{store}}\}$:
$\{\text{UnitSales}|_{\text{int}}, \text{SalePrice}|_{\text{int}}\}$

– *a partial order* $(\mathcal{L}, \leq)$ *on the levels in* $\mathcal{L}$.
We call $\ll$ *the immediate predecessor relation on* $\mathcal{L}$ *induced by* $\leq$.

➧ The partial order defines the taxonomy of levels. For example, day $\ll$ month $\ll$ quarter and day $\ll$ week; product $\ll$ type $\ll$ category

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

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

for each L_i, L_j *such that* $L_i \ll L_j$.
We call $\rho^*_{L_i,L_j}$ *the reflexive transitive closure of the roll-up functions inductively defined as follows:*

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

where

$$(\rho_{L_p,L_q} \cup \rho_{L_r,L_s})(x) = y \quad \text{iff} \begin{cases} \rho_{L_p,L_q}(x) = \rho_{L_r,L_s}(x) = y, \text{ or} \\ \rho_{L_p,L_q}(x) = y \text{ and } \rho_{L_r,L_s}(x) = \perp, \text{ or} \\ \rho_{L_p,L_q}(x) = \perp \text{ and } \rho_{L_r,L_s}(x) = y \end{cases}$$

➧ When in a schema various 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, $\rho_{\text{day,month}}(1/1/01)$ = Jan-01, $\rho_{\text{day,month}}(2/1/01)$ = Jan-01, ... $\rho_{\text{quarter,year}}(\text{Qtr1-01}) = 2001$, $\rho_{\text{quarter,year}}(\text{Qtr2-01}) = 2001$, ...

– *a finite set of level attribute definitions:*

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

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

➧ Level attributes are properties associated to levels. For example, product $\doteq$ $\{\text{prodname}|_{\text{string}}, \text{prodnum}|_{\text{int}}, \text{prodsize}|_{\text{int}}, \text{prodweight}|_{\text{int}}\}$

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

$$N \doteq \mathsf{f}(M)$$

where $N, M \in \mathcal{M}$*, and* f *is an aggregation function* $\mathsf{f} : \mathcal{B}(V) \mapsto W$*, for some* $V, W \in \mathcal{V}$*.* $\mathcal{B}(V)$ *is the finite set of all bags obtainable from values in* V *whose cardinality is bound by some finite integer* Ω*.*

➧ 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) and Avg-SalePrice $\doteq$ average(SalePrice)

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*.*

➧ 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.

– *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*

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

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:*

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

- *each measure in the aggregated fact* G *is computed via an aggregation function from some measure of the defined fact* F *it is based on:*

$$N_1 \doteq \mathsf{f}_1(M_{j(1)}) \quad \ldots \quad N_q \doteq \mathsf{f}_q(M_{j(q)})$$

Moreover the range and the domain of the aggregation function should be in agreement with the domains specified respectively in the aggregated fact G *and in the fact* F *it is based on.*

➠ Here we give a more precise characterisation of an aggregated fact: 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 $\{\text{Date}|_{\text{day}}, \text{Product}|_{\text{product}}, \text{Store}|_{\text{store}}\}$:
$\{\text{UnitSales}|_{\text{int}}, \text{SalePrice}|_{\text{int}}\}$

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 $\{\text{Date}|_{\text{month}}, \text{Product}|_{\text{type}}\}$:
$\{\text{Total-UnitSales}|_{\text{int}}, \text{Avg-SalePrice}|_{\text{real}}\}$

with the following aggregated measures:

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

2.1 Example

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

- 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, storenumb}
- 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:
  $\rho_{\text{day,month}}(1/1/01) = \text{Jan-01}$, $\rho_{\text{day,month}}(2/1/01) = \text{Jan-01}$, ...
  $\rho_{\text{month,quarter}}(\text{Jan-01}) = \text{Qtr1-01}$, $\rho_{\text{month,quarter}}(\text{Feb-01}) = \text{Qtr1-01}$, ...
  $\rho_{\text{quarter,year}}(\text{Qtr1-01}) = 2001$, $\rho_{\text{quarter,year}}(\text{Qtr2-01}) = 2001$, ...
  $\rho^*_{\text{day,year}}(1/1/01) = 2001$, $\rho^*_{\text{day,year}}(2/1/01) = 2001$, ...
  ...

– Level Attributes:
  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)

3 $\mathcal{GMD}$ Semantics

Having just defined the syntax of $\mathcal{GMD}$ schemas, we introduce now their semantics through a well founded model theory. We define the notion of a data warehouse state, namely a specific data warehouse, and we formalise when a data warehouse state is actually in agreement with the constraints imposed by a $\mathcal{GMD}$ schema.

Definition 2 (Data Warehouse State). *A* data warehouse state *over a schema with the 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 (or* cells*) of cardinality smaller than* Ω*;*

  ➧ Elements in Δ are the object identifiers for the cells in a multi-dimensional cube; we call them individual facts.

– Λ *is a finite set of level elements;*
– Γ *is a finite set of domain elements;*
– $\cdot^I$ *is a function (the interpretation function) such that*

$F^I \subseteq \Delta$	*for each* $F \in \mathcal{F}$*, where* F^I *is disjoint from any other* E^I *such that* $E \in \mathcal{F}$
$L^I \subseteq \Lambda$	*for each* $L \in \mathcal{L}$*, where* L^I *is disjoint from any other* H^I *such that* $H \in \mathcal{L}$
$V^I \subseteq \Gamma$	*for each* $V \in \mathcal{V}$*, where* V^I *is disjoint from any other* W^I *such that* $W \in \mathcal{V}$
$D^I = \Delta \mapsto \Lambda$	*for each* $D \in \mathcal{D}$
$M^I = \Delta \mapsto \Gamma$	*for each* $M \in \mathcal{M}$
$(A_i^L)^I = L \mapsto \Gamma$	*for each* $L \in \mathcal{L}$ *and* $A_i^L \in \mathcal{A}$ *for some* i

(Note: in the paper we will omit the $\cdot^I$ *interpretation function applied to some symbol whenever this is non ambiguous)*

➧ The interpretation functions defines a specific data warehouse state given a $\mathcal{GMD}$ signature, regardless from any fact definition. 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. Since fact definitions in the schema are not considered yet at this stage, the dimensions and the measures associated to cells are still arbitrary. In the following, we will introduce the notion of *legal* data warehouse state, which is the data warehouse state which conforms to the constraints imposed by the fact definitions. 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.

A data warehouse state 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:*
  - *the function associated to a dimension which does not appear in a fact is undefined for its cells:*
    $\forall f.\ F(f) \rightarrow f \notin \mathsf{dom}(D)$
    for each $D \in \mathcal{D}$ such that $D \neq D_i$ for each $i \leq n$
    ➧ This condition 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.
  - *each cell of a fact has a unique set of dimension values at the appropriate level:*
    $\forall f.\ F(f) \rightarrow \exists l_1, \ldots, l_n.\ D_1(f) = l_1 \wedge L_1(l_1) \wedge \ldots \wedge D_n(f) = l_n \wedge L_n(l_n)$
    ➧ This condition 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 value for each declared dimension in any legal data warehouse state.
  - *a set of dimension values identifies a unique cell within a fact:*
    $\forall f, f', l_1, \ldots, l_n.$
    $F(f) \wedge F(f') \wedge$
    $D_1(f) = l_1 \wedge D_1(f') = l_1 \wedge \ldots \wedge D_n(f) = l_n \wedge D_n(f') = l_n \rightarrow$
    $f = f'$
    ➧ This condition 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.

- *the function associated to a measure which does not appear in a fact is undefined for its cells:*
  $\forall f.\ F(f) \rightarrow f \notin \mathsf{dom}(M)$
  for each $M \in \mathcal{M}$ *such that* $M \neq M_i$ *for each* $i \leq n$
  ➠ This condition 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.
- *each cell of a fact has a unique set of measures:*
  $\forall f.\ F(f) \rightarrow \exists m_1, \ldots, m_m.\ M_1(f) = m_1 \wedge V_1(m_1) \wedge \ldots \wedge M_m(f) = m_m \wedge V_m(m_m)$
  ➠ This condition 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.

– *for each aggregated fact and for the defined fact it is based on in the schema:*

$$F \doteq E\,\{D_1 \mid_{L_1}, \ldots, D_n \mid_{L_n}\} : \{M_1 \mid_{V_1}, \ldots, M_m \mid_{V_m}\}$$
$$G \doteq F\,\{D_1 \mid_{R_1}, \ldots, D_p \mid_{R_p}\} : \{N_1 \mid_{W_1}, \ldots, N_q \mid_{W_q}\}$$
$$N_1 \doteq \mathsf{f}_1(M_{j(1)}) \quad \ldots \quad N_q \doteq \mathsf{f}_q(M_{j(q)})$$

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

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

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

➠ This condition 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.

According to the definition, 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.

3.1 Example

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

Date(g_1) = Jan-01	Product(g_1) = Dairy
Date(g_2) = Feb-01	Product(g_2) = Dairy
Date(g_3) = Jan-01	Product(g_3) = Drink
Date(g_4) = Feb-01	Product(g_4) = Drink
Date(g_5) = Mar-01	Product(g_5) = Dairy
Date(g_6) = Mar-01	Product(g_6) = Drink

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

daynum(day) = 1 prodweight(product) = 100gm storenum(store) = S101

4 $\mathcal{GMD}$ Extensions

For lack of space, in this brief report it is impossible to introduce the full $\mathcal{GMD}$ framework [Franconi and Kamble, 2003], which includes a full algebra in addition to the basic aggregation operation introduced in this paper. We will just mention the main extensions with respect to what has been presented here, and the main results.

The full $\mathcal{GMD}$ schema language includes also the possibility to define aggregated measures with respect to the application of a function to a set of original measures, pretty much like in SQL. For example, it is possible to have an aggregated cube with a measure total-profit being the sum of the differences between the cost and the price in the original cube; the difference is applied cell by cell in

the original cube (generating a profit virtual measure), and then the aggregation computes the sum of all the profits.

Two selection operators are also in the full $\mathcal{GMD}$ language. The slice operation simply selects the cells of a cube corresponding to a specific value for a dimension, resulting in a cube which contains a subset of the cells of the original one and one less dimension. 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.

A fact-join operation is 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 understandable. For similar reasons we do not allow a general union operator (like the one proposed in [Gray *et al.*, 1996]).

As we were mentioning in the introduction, one main result 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, we are able to clarify ambiguous parts, and we argue about the utility of some of the operators presented in the literature.

The other main result is in the proposal of a novel conceptual data model for multidimensional information, that extends and clarifies the one presented in [Franconi and Sattler, 1999].

References

[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. Technical report, Free University of Bozen-Bolzano, Italy, 2003. Forthcoming.

[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.

[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.