Multi-Granular Schemas for Data Integration

M. Andrea Rodríguez and Loreto Bravo

Universidad de Concepción, Chile
Edmundo Larenas 215, 4070409 Concepción, Chile
{arodriguez,lbravo}@inf.udec.cl

Abstract. Data can contain information at different levels of granularities but this metadata is generally left implicit in the data model. If we want to take advantage of different levels of granularity when integrating data, we need to first extend database schemas to include granularity information. In this article we (i) provide a multi-granular domain schema that is used in the formalization of database schemas so that each attribute is assigned a certain granularity; and (ii) explore the issues when integrating data at different granularities and suggest possible global schemas and instances.

1 Introduction

The notion of granularity is relevant to many aspects of data representation, and its formalization is fundamental for data integration. Granularity relates to data quality since it introduces bounds to the level of detail in which data is represented. Despite its relevance, granularity is usually implicit in a data model. This rises consequence for data integration, since no useful information can be extracted from data that may be semantically related but that are represented at different granularities.

As a motivating example, consider the integration of two different databases that store information about earthquakes occurred in different populated areas of the world. One of the databases stores the location of earthquakes by the epicenter's geographic coordinates, stores the time of occurrence by day and hour, and stores the magnitude in the Richter scale. The second database stores the location of earthquakes by the name of the administrative district that contains the epicenter, stores the time of occurrence by day, hour and minute, and stores the magnitude in the same scale. To solve this problem, we first need a database schema language that can specify both databases using different granularities. Then, the integration of both databases is not trivial. Although both databases store the magnitude in the same scale, they store temporal and spatial information at different granularities. We then need to formalization the schema of integration to be able to materialize the integrated database.

Although there exist different studies that model and query data at different granularities [1–3], none of them have addressed the integration of database schemas that differ in the granularity to represent different attributes. Unlike other studies, we do not focus on a particular domain, instead, we provide a general definition of schema integration, where granularity can be applied to any of the attribute domains. Even more, unlike classical approaches in data warehousing, data is not necessarily stored at the finest level of granularity upon which aggregation functions derive data at other coarser levels. In summary, the main contributions of this work are twofold. We start by formalizing a multi-granular domain schema and multi-granular database schema, in which attributes take values from domains that can be represented at different granularities. A second contribution is the formalization of a global schema given source multi-granular schemas and data. The goal is to obtain a global schema that provides at least the same information as the source data. Future work is the materialization of the database instance of the global schema.

The organization of the paper is as follows. In Section 2 we introduce our formalization of domain schemas and databases with multiple granularities. In Section 3 we study the problem of integrating two instances with data at different levels of granularity under some restrictions and we propose two possible global schemas and instances to integrate this data. Related work is presented in Section 4 to highlight the novelty of the work. Finally, Section 5 provides a discussion about more general settings.

2 Domain schemas and Databases

A *domain schema* is a tuple $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$, where (i) $\mathcal{U}$ is the domain associated with Ψ, (ii) ℓ is the set of granularity identifiers (or labels), (iii) $\mathcal{I}$ is the set of granule identifiers (or labels), (iv) ρ is a function $\rho : \mathcal{I} \rightarrow 2^{\mathcal{U}}$ that maps granules identifiers to subsets of the domain, and (v) τ is a function $\ell \rightarrow 2^{\mathcal{I}}$ such that for all $G \in \ell$ if $i, j \in \tau(G)$ then $\rho(i) \cap \rho(j) = \emptyset$. To simplify the presentation we will asume that for $i, j \in \mathcal{I}$, $i \neq j$ iff $\rho(i) \neq \rho(j)$, and that for $G_1, G_2 \in \ell$, $G_1 \neq G_2$ iff $\tau(G_1) \neq \tau(G_2)$.

Given a domain schema $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$ and granularities $G_1, G_2 \in \ell$: (i) G_1 is *finer or equal than* G_2, denoted $G_1 \preceq_{\Psi} G_2$, iff for all $i \in \tau(G_1)$ there exists $j \in \tau(G_2)$ such that $\rho(i) \subseteq \rho(j)$; (ii) G_1 *aggregates to* G_2, denoted $G_1 \triangleleft_{\Psi} G_2$, iff for all $j \in \tau(G_2)$ there exists $i_1, \ldots, i_n \in \tau(G_1)$ such that $\rho(i_1) \cup ... \cup \rho(i_n) = \rho(j)$; and (iii) G_1 is a *partition* of G_2, denoted by $G_1 \trianglelefteq_{\Psi} G_2$, if $G_1 \triangleleft_{\Psi} G_2$ and $G_1 \prec_{\Psi} G_2$. When the domain schema is clear from the context we will use $\prec$, $\triangleleft$ and $\trianglelefteq$.

A particular instance of a domain schema is the sets of granularities defined over the time domain. A *time domain* is a pair $(T; \leq)$, where T is a non-empty set of time instants, and $\leq$ is a total order in T [4–6]. By saying that T is a set of time instants we do not impose the idea that the time is only discrete. Time domain is continuous (dense) if for all $t < t'$ there exists t'' such that $t < t'' < t'$. The next example illustrates the case of multiple time granularities for representing academic activities of universities. In this setting the set of granule identifiers is an ordered set of index values.

Example 1. Consider the following domain schema $\Psi_{time} = (\mathcal{U}_{time}, \ell_{time}, \mathcal{I}_{time}, \rho_{time}, \tau_{time})$, with $\ell_{time} = \{month, term, year\}$ and the elements in $\mathcal{I}_{time}$, as well as functions ρ_{time} and τ_{time} are represented graphically in Figure 1. The only relations that hold among these granularities are $month \trianglelefteq year$ and $term \prec year$. $\square$

Another instance of a domain schema is the domain representing foodstuffs. Different granularities to represent a food product can be defined using classification criteria such as group, calorie, and brand. This is illustrated in the following example.

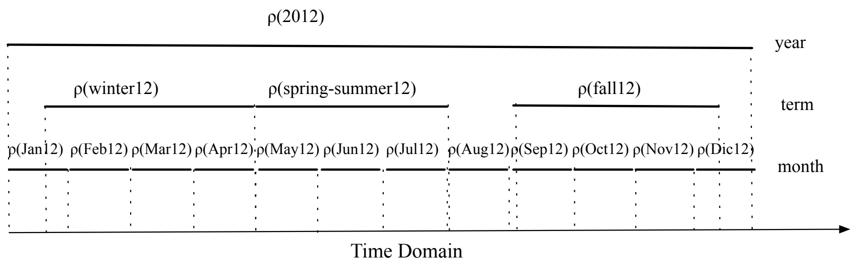
ρ(2012)
year
ρ(winter12)
ρ(spring-summer12)
ρ(fall12)
term
ρ(Jan12) ρ(Feb12) ρ(Mar12) ρ(Apr12) ρ(May12) ρ(Jun12) ρ(Jul12) ρ(Aug12) ρ(Sep12) ρ(Oct12) ρ(Nov12) ρ(Dic12)
month
Time Domain

Fig. 1. Instance a of a domain schema over the time domain to represent academic activities of universities

Example 2. Consider a product domain schema $\mathit{\Psi}_{prod} = (\mathcal{U}_{prod}, \ell_{prod}, \mathcal{I}_{prod}, \rho_{prod}, \tau_{prod})$, where:
$\mathcal{U}_{prod} = \{P_1, P_2, P_3, P_4, P_5, \ldots, P_{10}\}$
$\ell_{prod} = \{Product, Group, Calorie, Brand\}$
$\mathcal{I}_{prod} = \{P_1, \ldots, P_{10}, \mathit{Diaries}, \mathit{Bread}, \mathit{Meat}, \mathit{Low}, \mathit{Medium}, \mathit{High}, \mathit{Soprole}, \ldots, \mathit{Baker}\}$
$\tau_{prod}(\mathit{Product}) = \{P_1, P_2, P_3, P_4, P_5, \ldots, P_{10}\}$
$\tau_{prod}(\mathit{Group}) = \{\mathit{Diaries}, \mathit{Bread}, \mathit{Meat}\}$
$\tau_{prod}(\mathit{Calorie}) = \{\mathit{Low}, \mathit{Medium}, \mathit{High}\}$
$\tau_{prod}(\mathit{Brand}) = \{\mathit{Soprole}, \mathit{Nestle}, \mathit{Alpura}, \mathit{Bimbo}, \mathit{Baker}\}$
$\rho_{prod}(P_i) = \{P_i\}$ for $i \in [1, 10]$
$\rho_{prod}(\mathit{Soprole}) = \{P_1, P_2\}$
$\rho_{prod}(\mathit{Nestle}) = \{P_3, P_4\}$
$\rho_{prod}(\mathit{Alpura}) = \{P_5, P_6\}$
$\rho_{prod}(\mathit{Bimbo}) = \{P_7, P_8\}$
$\rho_{prod}(\mathit{Baker}) = \{P_9, P_{10}\}$
$\rho_{prod}(\mathit{Low}) = \{P_1, P_2, P_5\}$
$\rho_{prod}(\mathit{Medium}) = \{P_4, P_7, P_9, P_{10}\}$
$\rho_{prod}(\mathit{High}) = \{P_8, P_6\}$
$\rho_{prod}(\mathit{Dairies}) = \{P_1, P_2, P_3, P_5, P_8\}$
$\rho_{prod}(\mathit{Bread}) = \{P_7, P_9\}$
$\rho_{prod}(\mathit{Meat}) = \{P_4\}$
The relationships between granularities in $\mathit{\Psi}_{Prod}$ are: $\mathit{Product} \trianglelefteq \mathit{Brand}$, $\mathit{Product} \trianglelefteq \mathit{Calorie}$ and $\mathit{Product} \triangleleft \mathit{Group}$. $\square$

The previous example resembles data warehousing (DW). In an homogeneous and strict DWs, if a category A rolls-up to a category B then $A \trianglelefteq B$. Unlike datawarehousing, a domain schema enables to store data at different levels of granularities and not only at the finest granularity.

Databases. A database schema is a tuple $\mathit{\Sigma} = (\mathcal{M}, \mathcal{R}, \mathit{Dom})$, where: (a) $\mathcal{M}$ is a set of domain schema, (b) $\mathcal{R}$ is a set of relational schemas and (c) Dom is a function that, given a relation $R \in \mathcal{R}$ and an attribute $A \in R$, returns a tuple $(\mathit{\Psi}_{RA}, G_{RA})$ where

$\Psi_{RA} = (\mathcal{U}_{RA}, \ell_{RA}, \mathcal{I}_{RA}, \rho_{RA}, \tau_{RA}) \in \mathcal{M}$ and $G_{RA} \in \ell_{RA}$. Intuitively, Dom returns the domain schema and granularity associated with attribute $A \in R$.

To simplify the presentation of results, we will assume that there are no two $\Psi_1, \Psi_2 \in \mathcal{M}$ that refer to the same domain $\mathcal{U}$. Function $\mathsf{schemaOfDomain}(\mathcal{M}, \mathcal{U})$ returns the domain schema in $\mathcal{M}$ defined over domain $\mathcal{U}$.

A database instance D of a schema Σ is a finite collection of ground atoms of the form $R(c_1, \ldots, c_i, \ldots, c_l)$, where (a) $R(B_1, \ldots, B_i, \ldots, B_l) \in \mathcal{R}$, and (b) every c_i with $i \in [1, l]$ is such that $c_i \in \tau_{RB_i}(G_{RB_i})$ where $Dom(R, B_i) = (\Psi_{RB_i}, G_{RB_i})$ and $\Psi_{RB_i} = (\mathcal{U}_{RB_i}, \ell_{RB_i}, \mathcal{I}_{RB_i}, \rho_{RB_i}, \tau_{RB_i})$.

3 Database Integration

We want to explore the challenges that arise when merging data with different levels of granularity. For example, given several data sources with data at different granularity levels, which is the best granularity to be used in the global schema? At what level of granularity should the data be stored in the merged instances? How do integrity constraints affect this process? What type of query language can we use to deal with this different levels of granularity?

In a first stage we decided to restrict to a simple base case with two source databases to be integrated and a global schema that share the same set of relations in $\mathcal{R}$ and have no integrity constraints. In this way, each relation in a source is mapped to the same relation in the global schema. In this setting, we will want to define the granularity associated with every attribute in the global schema and the level of granularity at which the data should be stored.

Schemas $\Sigma_1 = (\mathcal{M}_1, \mathcal{R}_1, Dom_1)$ and $\Sigma_2 = (\mathcal{M}_2, \mathcal{R}_2, Dom_2)$ are *domain compatible* if $\mathcal{M}_1 = \mathcal{M}_2$, $R_1 = R_2$ and for every $R \in \mathcal{R}_1$ and every $A \in R_1$, there exists Ψ, G_1 and G_2 such that $Dom_1(R, A) = (\Psi, G_1)$ and $Dom_2(R, A) = (\Psi, G_2)$. This is, the two schemas share the same set of domain schemas and relational schemas and every attribute is associated to the same domain schema (even if it is at different levels of granularity) in the different schemas.

In this article we concentrate on the integration of databases defined over domain compatible schemas. In this setting, we will consider two cases: *Domain Invariant Integration* and *Finest Domain Integration*. In the former, we require global schema to share the same set of domains $\mathcal{M}$ as the sources. In the latter setting, we allow $\mathcal{M}_{\mathcal{G}}$ to differ from the sources but require it to contain the finest granularity schema that generalizes the domains in the sources.

3.1 Domain Invariant Integration

Given two domain compatible source databases with schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$ and instances D_1 and D_2, respectively, we want to define a global schema $\Sigma_{\mathcal{G}} = (\mathcal{M}, \mathcal{R}, Dom_{\mathcal{G}})$ such that at least the following conditions hold:

(A.1) For every $R \in \mathcal{R}$ and every $A \in R$, if $Dom_1(R, A) = Dom_2(R, A)$ then $Dom_{\mathcal{G}}(R, A) = Dom_1(R, A) = Dom_2(R, A)$.

(A.2) For every $R \in \mathcal{R}$ and every $A \in R$, if $Dom_1(R, A) = (\Psi, G_1)$, $Dom_2(R, A) = (\Psi, G_2)$ and $Dom_{\mathcal{G}}(R, A) = (\Psi, G_{\mathcal{G}})$, then $G_1 \preceq G_{\mathcal{G}}$ and $G_2 \preceq G_{\mathcal{G}}$.

The first condition will ensure that if two attributes have the same granularity, the attribute in the global schema will have the same granularity. On the other hand, if they are different, the second condition ensures that the granularity in the global schema is coarser or equal than both of them.

In order to find the global schema that provides the finest granularity for each attribute such that these properties are satisfied, we define the join operator. Given a domain schema $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$ and $G_1, G_2 \in \ell$, the *Join Operator* is

$$Join(\Psi, G_1, G_2) = \begin{cases} G \text{ if } G \in \ell, G_1 \preceq G, G_2 \preceq G, \nexists G' \in \ell (G_1 \preceq G', G_2 \preceq G', G' \prec G) \\ \perp \text{ otherwise} \end{cases}$$

Thus, the join operator of G_1 and G_2 returns the finest granularity in ℓ such that it subsumes both G_1 and G_2; if that granularity does not exist, it return $\perp$.

Definition 1. Given schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$ let the *join schema* $\Sigma_1 \bowtie \Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_{\Sigma_1 \bowtie \Sigma_2})$ where for every $R \in \mathcal{R}$ and $A \in R$:

$$Dom_{\Sigma_1 \bowtie \Sigma_2}(R, A) = \begin{cases} G & \text{if } G = Join(\Psi, G_1, G_2) \neq \perp, Dom_1(R, A) = (\Psi, G_1) \\ & \text{ and } Dom_2(R, A) = (\Psi, G_2) \\ \perp & \text{otherwise} \end{cases}$$

If any of this granularities is $\perp$, then there is no join schema that allows the integration of the databases under the domains in $\mathcal{M}$. □

The join schema of Σ_1 and Σ_2 is a good candidate for a global schema since it satisfies conditions (A.1) and (A.2). Its main drawback is the fact that it does not always exist.

Example 3. Consider the database schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$ where $\Psi_{Prod} = (\mathcal{U}_{prod}, \ell_{prod}, \mathcal{I}_{prod}, \rho_{prod}, \tau_{prod}) \in \mathcal{M}$, $\mathcal{R} = \{Diet(product, amount)\}$, $Dom_1(Diet, product) = (\Psi_{prod}, Group)$ and $Dom_2(Diet, product) = (\Psi_{prod}, Calorie)$. There is no $G \in \ell_{prod}$ such that $Calorie \preceq G$ and $Group \preceq G$. Therefore, $Join(\Psi_{prod}, Group, Calorie) = \perp$ and the join schema $\Sigma_1 \bowtie \Sigma_2$ does not exist. Now, let $\Sigma_3 = (\mathcal{M}, \mathcal{R}, Dom_3)$ with $Dom_3(Diet, product) = (\Psi_{prod}, Product)$. In this case, $Join(\Psi_{prod}, Product, Group) = Group$ and, therefore, $\Sigma_1 \bowtie \Sigma_3 = (\mathcal{M}, \mathcal{R}, Dom_{\Sigma_1 \bowtie \Sigma_3})$, where $Dom_{\Sigma_1 \bowtie \Sigma_3}(Diet, product) = (\Psi_{prod}, Group)$. □

Now, in order to define the global instance given a global schema, we need to ensure that the data in the global schema are the same as the one contained in the sources but probably at a higher level of granularity. For example, we can have a month in the source and the associated year in the global schema.

Definition 2. Given instances D_1 and D_2 defined respectively over schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$, and a global schema $\Sigma_{\mathcal{G}}$, a *global instance* $D_{\mathcal{G}}$ over the schema $\Sigma_{\mathcal{G}}$ is such that $R(c_1, \ldots, c_i, \ldots, c_l) \in D_{\mathcal{G}}$ if and only if:

– There exists $R(B_1, \ldots, B_i, \ldots, B_l) \in \mathcal{R}$,
– For every $i \in [1, l]$, $c_i \in \tau(G_{RB_i})$ where $Dom_{\Sigma_{\mathcal{G}}}(R, B_i) = (\Psi_{RB_i}, G_{RB_i})$.
– There exists $R(c_1^s, \ldots, c_i^s, \ldots, c_l^s) \in D_1$ or $R(c_1^s, \ldots, c_i^s, \ldots, c_l^s) \in D_1$ such that for every $i \in [1, l]$, it holds that $\rho_{RB_i}(c_i^s) \subseteq \rho_{RB_i}(c_i)$ where $Dom_{\Sigma_{\mathcal{G}}}(R, B_i) = (\Psi_{RB_i}, G_{RB_i})$. □

It is easy to see that if the join schema exists, then there will exist a *unique* global instance for the join schema.

Since sometimes there is no join schema to use as a global schema, in the next section we study the consequences of lifting the restriction of forcing the global schema to use the same set of domains $\mathcal{M}$ as the sources.

3.2 Finest Domain Integration

When integrating two granularities G_1 and G_2, it might be the case that its associated domain schema has no granularity which is coarser than both of them. To solve this limitation, we consider in this section the possibility of adding to the domain schema the finest granularity that is coarser than both G_1 and G_2. More formally, given two domain compatible schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$, we want to find a global schema $\Sigma_\mathcal{G} = (\mathcal{M}_\mathcal{G}, \mathcal{R}, Dom_\mathcal{G})$ for which the following conditions hold:

(B.1) For every $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau) \in \mathcal{M}$ there is a $\Psi_\mathcal{G} = (\mathcal{U}_\mathcal{G}, \ell_\mathcal{G}, \mathcal{I}_\mathcal{G}, \rho_\mathcal{G}, \tau_\mathcal{G}) \in \mathcal{M}_\mathcal{G}$ such that $\mathcal{U}_\mathcal{G} = \mathcal{U}$; $\ell \subseteq \ell_\mathcal{G}$; $\mathcal{I} \subseteq \mathcal{I}_\mathcal{G}$; for every $i \in \mathcal{I}$, $\rho(i) = \rho_\mathcal{G}(i)$; and for every $G \in \ell$, $\tau(G) = \tau_\mathcal{G}(G)$. Furthermore, only domains with this property belong to $\mathcal{M}_\mathcal{G}$.

(B.2) For every $R \in \mathcal{R}$ and every $A \in R$, if $Dom_1(R, A) = Dom_2(R, A)$ then $Dom_\mathcal{G}(R, A) = Dom_1(R, A) = Dom_2(R, A)$.

(B.3) For every $R \in \mathcal{R}$ and every $A \in R$, if $Dom_1(R, A) = (\Psi, G_1)$, $Dom_2(R, A) = (\Psi, G_2)$ and $Dom_\mathcal{G}(R, A) = (\Psi_\mathcal{G}, G_{RA_i})$, with $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$ and $\Psi_\mathcal{G} = (\mathcal{U}_\mathcal{G}, \ell_\mathcal{G}, \mathcal{I}_\mathcal{G}, \rho_\mathcal{G}, \tau_\mathcal{G})$, then $\mathcal{U} = \mathcal{U}_\mathcal{G}$, $G_1 \preceq_{\Psi_\mathcal{G}} G_{RA_i}$ and $G_2 \preceq_{\Psi_\mathcal{G}} G_{RA_i}$.

The first conditions ensures that $\mathcal{M}'$ is an extension of $\mathcal{M}$, this is, it contains a domain for the same set of $\mathcal{U}$ as $\mathcal{M}$, and to each of them, it can only add new indices and new granularities without modifying the ones that already exist in $\mathcal{M}$. Condition (B.2) ensures that if two attributes have the same granularity in the sources, the attribute in the global schema will not be modified. On the other hand, if they are different, condition (B.3) requires that for each pair of granularities to merge, there exists a domain schema $\Psi_\mathcal{G}$ that extends the domain schema Ψ by adding a new granularity that is coarser than both of them. As before, we would like the global schema to use the finest granularity for each attribute such that these conditions hold. In order to achieve this, we first need to pair up the granularities and indices that will need to be merged.

Definition 3. Given two domain compatible schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$, $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$ and $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau) \in \mathcal{M}$, the *set of granularities to merge* of Ψ for Σ_1 and Σ_2 is $\mathit{GtoMerge}(\Psi, \Sigma_1, \Sigma_2) = \{\{G_1, G_2\} \mid R \in \mathcal{R},\ A \in R,\ Dom_1(R, A) = (\Psi, G_1)$ and $Dom_2(R, A) = (\Psi, G_2)\}$. Also, we will denote the set of attributes that share a specific pair of granularities by $\mathit{Att}(\Sigma_1, \Sigma_2, \{G_1, G_2\}) = \{(R, A) \mid \Psi \in \mathcal{M},$ $Dom_i(R, A) = (\Psi, G_1)$, $Dom_j(R, A) = (\Psi, G_2)$ and $(i, j) \in \{(1, 2), (2, 1)\}\}$.

Let $\wp(\Psi, G_1, G_2)$ be the maximal partition of $\tau(G_1) \cup \tau(G_2)$ such that for every distinct sets $S_1, S_2 \in \wp(\Psi, G_1, G_2)$ and every $i \in S_1$ and $j \in S_2$, $\rho(i) \cap \rho(j) = \emptyset$. $\square$

Taking into consideration the pairs of granularities that need to be merged, we define the new domain schema that can include new granularities that merge the granularities that need to be integrated.

Definition 4. Given two domain compatible schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$ and $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau) \in \mathcal{M}$, the *merged* domain schema of Ψ for Σ_1 and Σ_2 is $Merged(\Psi, \Sigma_1, \Sigma_2) = (\mathcal{U},' \ell', \mathcal{I}', \rho', \tau')$ where:

1. $\mathcal{U}' = \mathcal{U}$,
2. $\ell' \supseteq \ell$
3. $\mathcal{I}' \supseteq \mathcal{I}$
4. For every $i \in \mathcal{I}$, $\rho'(i) = \rho(i)$;
5. For every $G \in \ell$, $\tau'(G) = \tau(G)$;
6. For every $\{G_1, G_2\} \in GtoMerge(\Psi, \Sigma_1, \Sigma_2)$, there exists $G' \in \ell'$ such that $G_1 \preceq G'$ and $G_2 \preceq G'$ and that it is not possible to define a granularity G" over $\mathcal{U}$ such that $G" \prec G'$, $G_1 \preceq G"$ and $G_1 \preceq G"$.
7. There is no other schema $\Psi" = (\mathcal{U}", \ell", \mathcal{I}", \rho", \tau")$ that satisfies conditions (1) to (6) and such that $\ell" \subsetneq \ell'$ and/or $\mathcal{I}" \subsetneq \mathcal{I}'$. □

By condition (1) the merged domain schema will be defined over the same sources domain schema. Next, by conditions (2)-(5), the merged schema will contain at least the same granularity labels and indices as Ψ with the same meaning. Condition (6) ensures that for each pair of granularities G_1 and G_2 that need to be merged, there is another granularity (not necessarily new) for which both of them are finer, and which is also the finest one that satisfies this requirement.

Definition 5. Given two domain compatible schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$, let the *merged schema* of Σ_1 and Σ_2 be $\Sigma_1 \otimes \Sigma_2 = (\mathcal{M}_\otimes, \mathcal{R}, Dom_\otimes)$ where

- $\mathcal{M}_\otimes = \{\Psi' \mid \Psi' = Merged(\Psi, \Sigma_1, \Sigma_2)$ and $\Psi \in \mathcal{M}\}$; and
- For every $R \in \mathcal{R}$ and $A \in R$, $Dom_\otimes(R, A) = (\Psi_\otimes, Join(\Psi', G_1, G_2))$ where $Dom_1(R, A) = (\Psi, G_1)$, $Dom_2(R, A) = (\Psi, G_2)$ and $\Psi' = Merged(\Psi, \Sigma_1, \Sigma_2)$. □

The merged schema is a good candidate to be a global schema since it satisfies properties (B.1) to (B.3).

Example 4. (example 3 continued) When merging Σ_1 and Σ_2 we need to compute $\Psi'_{prod} = Merged(\Psi_{prod}, Group, Calories) = (\mathcal{U}_{prod}, \ell_{prod} \cup \{GroupCalories\}, \mathcal{I}_{prod} \cup \{\delta_1\}, \rho_\otimes, \tau_\otimes)$, where $\rho_\otimes(G) = \rho_{prod}(G)$ for every $G \in \ell_{prod}$, $\rho_\otimes(\delta_1) = \{P_1, P_2, P_3, P_5, P_6, P_8\}$, $\tau_\otimes(i) = \tau_{prod}(i)$ for every $i \in \mathcal{I}_{prod}$, $\tau_\otimes(GroupCalories) = \{\delta_1, Medium\}$. Thus, the merged schema $\Sigma_1 \otimes \Sigma_2 = (\mathcal{M}_\otimes, \mathcal{R}, Dom_\otimes)$ where $\mathcal{M}_\otimes = \{\Psi'_{prod}\}$ and $Dom_\otimes(Diet, product) = (\Psi'_{prod}, GroupCalories)$.

On the other hand, when integrating schemas Σ_1 and Σ_3 the join and the merge schema coincide. □

A good property of the merged schema is that it always exists and is unique up to renaming of granularities in $\ell_\otimes$ and indices en $\mathcal{I}_\otimes$. Also, when merging schemas Σ_1 and Σ_2, for every domain schema Ψ in them, and for every $G_1, G_2 \in \ell_\Psi$, if the join schema exists and is $Join(\Psi, G_1, G_2) = G_\mathcal{J}$, then for $\Psi' = Merged(\Psi, G_1, G_2)$ it holds that the $Join(\Psi', G_1, G_2) = G' \prec_{\Psi'} G_\mathcal{J}$. Thus, schema $\Sigma_1 \otimes \Sigma_2$ can contain finer grained attributes than $\Sigma_1 \bowtie \Sigma_2$. A drawback of the merged schema is that the new granularity names (in $\ell_\otimes$) and indices (in $\mathcal{I}_\otimes$) will be less intuitive. Indeed, in Example 4

Algorithm 1: MergedSchema

```
   Input  : Domain compatible schemas $\Sigma_1 = (\mathcal{M}, \mathcal{R}, Dom_1)$ and $\Sigma_2 = (\mathcal{M}, \mathcal{R}, Dom_2)$
   Output : Merged domain schemas $\Sigma_1 \otimes \Sigma_2$
1  $\mathcal{M}_\otimes \leftarrow \emptyset$
2  for $(\mathcal{U}, \ell, \mathcal{I}, \rho, \tau) \in \mathcal{M}$ do
3      $\Psi' \leftarrow (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$
4      for $\{G_1, G_2\} \in GtoMerge(\Psi, \Sigma_1, \Sigma_2)$ do
           /* Add to $\Psi$ a granularity that merges $G_1$ and $G_2$ */
5          $(G', \Psi') \leftarrow$ MergedGranularity$(\Psi', G_1, G_2)$
           /* Removes equivalent indices and granularities that
              could have been added in lines homomorphic indices
              and granularities */
6          $G' \leftarrow$ CleanUp$(\Psi', G')$
7          $\mathcal{M}_\otimes \leftarrow \mathcal{M}_\otimes \cup \{\Psi'\}$
8          for $(R, A) \in Att(\Sigma_1, \Sigma_2, \{G_1, G_2\})$ do
9              $Dom_\otimes(R, A) = G'$
10 return $(\mathcal{M}_\otimes, \mathcal{R}, Dom_\otimes)$;
```

the index δ_1 and the granularity name $GroupCalories$ are less intuitive than the indices and granularity names already in the domain schema before the merge.

Computing the Merged Schema. The merged schema of two domain compatible schemas Σ_1 and Σ_2 can be computed by Algorithm 1 called MergedSchema which relies on the following property of the merged schema: for each pair of granularities that need to be merged, the merged schema contains a new granularity G' that contains one index for each partition in $\wp(\Psi, G_1, G_2)$ and the ρ of each of this indices will contain exactly the union of the ρ of the indices in the corresponding partition. Relying on this observation, the subroutine MergedGranularity (see Algorithm 2) returns granularity G' and the new domain schema which now contains this new granularity G'. Algorithm MergedSchema calls MergedGranularity for all the combinations of G_1 and G_2 that need to be merged. It might be the case that some of the new granularities and indices added in lines 4-5 of the Algorithm will result in non-minimal schemas, in the sense that there might be two indices i and j in a domain schema for which $\rho(i) = \rho(j)$, also there might be granularities G_1 and G_2 which contain the same indices. Method CleanUp (Algorithm 3) removes all these indices and granularities that are not needed. To achieve this, it first replaces any new index in G' by one in the original schema which maps to the same elements in $\mathcal{U}$ if it exists. Next, it makes sure that there is no other granularity with the same indices. After the CleanUp, Algorithm MergeSchema in lines 8-9 associates to each merged attribute its new granularity through function $DOm_\otimes$. Finally, in line 10 the merged schema for Σ_1 and Σ_2 is returned.

Algorithm 2: MergedGranularity

```
Input  : Schema Domain $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$ and granularities $G_1, G_2 \in \ell$
Output : $(G', \Psi')$ where $G'$ is a granularity that minimally merges $G_1$ and $G_2$ and $\Psi'$
         extends $\Psi$ to consider it
1  $\Psi' \leftarrow (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$
2  $G' \leftarrow$ fresh granularity label not already in $\ell$
3  $\tau(G') \leftarrow \emptyset$
4  for $S \in \wp(\Psi, G_1, G_2)$ do
5      $i \leftarrow$ fresh index not already in $\mathcal{I}$
6      $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
7      $\rho(i) \leftarrow \bigcup_{j \in S} \rho(j)$
8      $\tau(G') \leftarrow \tau(G') \cup \{i\}$
9      $\ell \leftarrow \ell \cup \{G'\}$
10 return $(G', \Psi')$;
```

Algorithm 3: CleanUp

```
Input  : Schema Domain $\Psi = (\mathcal{U}, \ell, \mathcal{I}, \rho, \tau)$ and a granularities $G \in \ell$
Output : A granularity $G'$ equivalent to $G$
1  for $i \in \tau(G')$ do
2      for $j \in (\mathcal{I} \setminus \tau(G'))$ do
           /* Condition can be satisfied only once for each $i$   */
3          if $\rho(i) = \rho(j)$ then
4              $G' \leftarrow (G' \setminus \{i\}) \cup \{j\}$
5              $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$
6  $GtoReturn \leftarrow G'$
7  for $G_i \in (\ell \setminus \{G'\})$ do
       /* Condition can be satisfied only once.               */
8      if $\tau(G_i) = \tau(G')$ then
9          $GtoReturn \leftarrow G_i$
10         $\ell \leftarrow (\ell \setminus \{G'\})$
11 return $GtoReturn$;
```

4 Related work

The concept of granularity is not new. Important advances have been done in the formalization of granularity in the temporal and spatial domains, since their applications need abstraction mechanisms for handling domains that are continuous in nature. The formalization of temporal granularity by Bettini *et. al.* [4, 5] has been basis for several different studies that explore temporal and spatial granularity. The work by Worboys [7, 8] provides theoretical foundation, using notions similar to rough set theory, to define imprecision of spatial data at different granularities. For conceptual modeling, Khatri *et al.* [9] propose an annotation-based spatio-temporal conceptual model that accounts for the semantic related to spatial and temporal granularity. The work by Camossi *et* *al.* [6] extends ODMG type system [10] to handle spatio-temporal data with specific types to define spatio-temporal properties at multiple granularities. They provide geometric converse operators to implement changes on granularity. Similarly, Bertino *et al.* [1] extends object-relational model based on OpenGis specifications described in SQL3 to represent temporal dimension in this model and the multi-representation of spatio-temporal granularities.

From another perspective, a schema model for datawarehousing defines dimensions over fact tables. From fact tables, data is aggregated along a dimension forming a hierarchy of finer-than relationships of granularity. As Iftikhar and Pedersen already highlighted [2, 3], current models cannot store data at different levels of granularity, since they require all dimensions attributes to be given concrete values. They propose extensions to current models that include a time dimension granularity defined as a single hierarchy. Thus, fact data is associated with a time dimension in a particular granularity. The work in [3] can be seen as a materialization of our multi-granular database schema. Unlike the work in [3], however, our model is applicable to not only temporal granularity but also spatial or semantic granularity of a conceptual classification.

Our work relates to the problem described in [11], which characterizes the derivation problem for summary data. The idea is to compare summary data in statistical databases based on common but not identical classification criterion. This classification criterion can be seen as a granularity of atomic data, where each class corresponds to a granularity index that maps to a portion of the underlying domain (granule). The problem is then to integrate different summarized data and extract useful information from them. Unlike the datawarehouse context, summary datasets may not have atomic data from which summarized data is obtained. Data is stored in terms of different classification criterion, which is similar to our problem of having datasets at different level of granularities without the atomic facts of the underlying domain.

Despite previous work, to the best of our knowledge, there is no work that formalizes granularity in a more general relational context. We want here to formalize granularity not only for time-depending attributes, but also for attributes defined by different classification criteria. Our approach could then be extended to deal with particularities of each attribute domain. Like the work in [11], we also want to be able to extract useful information from different datasets whose data are represented at different granularities.

5 Discussion

In this section we discuss the issues that arise if we consider constraints or allow instances where each attribute can contain different levels of granularity.

Integrity Constraints. In the previous sections, we have considered that the global instance contains all the tuples of the data sources, where some of their attributes may have been modified to a coarser level of granularity. If we now consider that the schema contains integrity constraints, new issues arise, which are illustrated by the following example.

Example 5. Consider different databases that store information about relevant touristic landmarks. Each source database contains the predicate $\mathit{LandMark}(\mathit{name}, \mathit{location})$

and functional dependency $name \to location$. Attribute *location* may be defined over a domain at different levels of granularities. For example, while a data source may store a location in terms of provinces, another data source stores it in terms of countries.

Since we have a functional dependency constraints, an integrated database will not store two tuples in $LandMark$ with the same value in attribute $name$. A global schema will need to be such that the functional dependency constraints must be satisfied. Tuples with the same $name$ will need to be analyzed and, when possible, the conversion of attributes will need to be applied.

We say that a conflict of data integration exists when a data stored in one database cannot be converted to the data in another database. For example, a database stores tuple (*Aconcagua mountain*, *Argentina*) and another stores (*Aconcagua mountain*, *Chile*). In this example, both tuples are at the same granularity of countries and, therefore, they are inconsistent because they should have the same value in attribute *location*. An inconsistency can also happen when storing the location at different levels of granularity. For example, a database stores tuple (*Aconcagua mountain*, *San Juan Province*) using a granularity of provinces and another stores (*Aconcagua mountain*, *Chile*) using a granularity of countries. For this example, there is no way to convert the value of *location* in one tuple into the value of the other tuple.

Although values of attribute $location$ can be different in two databases for the same value of attribute $name$, there are cases of no conflict. We call this situation *synergy* as in [11]. For example, a database stores tuple (*Aconcagua mountain*, *San Juan Province*) and another stores (*Aconcagua mountain*, *Argentina*). In such case, we would expect to keep the tuples with the fine granularity which is not in conflict with the coarse granularity. $\square$

Instances with several granularities in the same attribute. One of the limitations of how global instances have been defined is that we loose some of the information when the global schema uses a coarser granularity in any of the attributes. For example, consider the domain schema Ψ_{Prod} and an attribute A with granularity $Product$ in the source and $Calories$ in the global schema. If when merging we replace every product value in attribute A by its associated Low, $Medium$ and $High$, we will be loosing some information that may turn to be useful.

This is why we could consider a modification of the definition of database instance to allow for each attribute to contain indices in or below the granularity of the attribute. More formally, a database instance D of a schema Σ could alternatively be defined as a finite collection of ground atoms of the form $R(c_1, \ldots, c_i, \ldots, c_l)$, where (a) $R(B_1, \ldots, B_i, \ldots, B_l) \in \mathcal{R}$, and (b) every c_i is such that $c_i \in \mathcal{I}$, there exists $g \in \tau_{RB_i}(G_{RB_i})$ for which $\rho_{RB_i}(c_i) \subseteq \rho_{RB_i}(g)$ where $Dom(R, B_i) = (\Psi_{RB_i}, G_{RB_i})$.

With this alternative definition, when merging instances D_1 over Σ_1 and D_2 over Σ_2 we can use the global instance $D_{\mathcal{G}} = D_1 \cup D_2$ over either the join or merge schema. The problem now becomes the way in which we can query this type of instances and how answers should be displayed. A query language, like SQL, could be augmented to request the level of granularity at which we want the answers to be displayed.

Acknowledgements We would like to thank reviewers for their valuable comments that help to improve this paper. This work is funded by Bicentenario Program PSD 57. Andrea Rodríguez is also partially funded by Fondef D09I1185.

References

1. Bertino, E., Camossi, E., Bertolotto, M.: Multi-granular spatio-temporal object models: concepts and research directions. In: ICOODB'09, Springer-Verlag (2010) 132–148
2. Iftikhar, N., Pedersen, T.B.: Schema design alternatives for multi-granular data warehousing. In: DEXA'10, Springer-Verlag (2010) 111–125
3. Iftikhar, N., Pedersen, T.B.: Gradual data aggregation in multi-granular fact tables on resource-constrained systems. In: KES'10, Springer-Verlag (2010) 349–358
4. Bettini, C., Wang, X.S., Jajodia, S.: A general framework for time granularity and its application to temporal reasoning. Annals of Mathematics and Artificial Intelligence **22**(1-2) (1998) 29–58
5. Bettini, C., Dyreson, C.E., Evans, W.S., Snodgrass, R.T., Wang, X.S.: A glossary of time granularity concepts. In: Temporal Databases, Dagstuhl. (1997) 406–413
6. Camossi, E., Bertolotto, M., Bertino, E.: A multigranular object-oriented framework supporting spatio-temporal granularity conversions. International Journal of Geographical Information Science **20**(5) (2006) 511–534
7. Worboys, M.F.: Imprecision in finite resolution spatial data. GeoInformatica **2**(3) (1998) 257–279
8. Worboys, M.F.: Computation with imprecise geospatial data. Computers, Environment, and Urban Systems **22**(2) (1998) 85–106
9. Khatri, V., Ram, S., Snodgrass, R.T., O'Brien, G.M.: Supporting user-defined granularities in a spatiotemporal conceptual model. Annals of Mathematics and Artificial Intelligence **36**(1-2) (2002) 195–232
10. Cattell, R.G., Barry, D.K.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann (2000)
11. Malvestuto, F.M.: The derivation problem for summary data. In: SIGMOD Conference, ACM Press (1988) 82–89