A Data Warehouse Conceptual Data Model for Multidimensional Aggregation: a preliminary report

Enrico Franconi
Dept. of Computer Science, Univ. of Manchester
Manchester M13 9PL, UK
franconi@cs.man.ac.uk

Ulrike Sattler
RWTH, LuFG Theoretical Computer Science
D-52074 Aachen, Germany
uli@cantor.Informatik.rwth-aachen.de

ABSTRACT

This paper presents a proposal for a Data Warehouse Conceptual Data Model which allows for the description of both the relevant aggregated entities of the domain—together with their properties and their relationships with other relevant entities—and the relevant dimensions involved in building the aggregated entities. The proposed Data Warehouse Conceptual Data Model is able to capture the database schemata expressed in the most interesting traditional Semantic Data Models and Object-Oriented Data Models; it is able to introduce complex descriptions of the structure of aggregated entities and multiply hierarchically organised dimensions; it is based on Description Logics, a class of formalisms for which it is possible to study the expressivity in relation with decidability of reasoning problems and completeness of algorithms; it supports the most important reasoning services for the basic Data Warehouse operations.

1 Introduction

Data Warehouse—and especially OLAP—applications ask for the vital extension of the expressive power and functionality of traditional conceptual modeling formalisms in order to cope with *aggregation*. Still, there have been few attempts [Agrawal *et al.*, 1995, Gray *et al.*, 1996, Gyssens and Lakshmanan, 1997, Catarci *et al.*, 1995] to provide such an extended modeling formalism, despite the fact that (1) experiences in the field of databases have proved that conceptual modeling is crucial for the design, evolution, and optimisation of a database, (2) a great variety of data warehouse system are on the market, most of them providing some implementation of multidimensional aggregation, and (3) query optimisation is even more crucial for data warehouses than it is for databases—which makes semantic query optimisation using a conceptual model even more important. As a consequence of the absence of a such an extended modeling formalism, a comparison of different systems or language extensions for query optimisation is difficult: a common framework in which to translate and compare these extensions is missing, new query optimisation techniques developed for extended schema and/or query languages (see [Gupta *et al.*, 1995, Levy and Mumick, 1996, Srivastava *et al.*, 1996, Mumick and Shmueli, 1995] for query optimisation with aggregation, and [Levy *et al.*, 1996] for planning queries to heterogeneous sources) cannot be compared appropriately: in most cases, it can be easily seen that the optimisation algorithms transform queries to equivalent queries, but it remains open where one algorithm is better than another one, whether it is optimal or in how far it is incomplete.

In order to address these questions, a formal framework must be developed that encompasses the abstract principles of the data warehouse related extensions of traditional representation formalisms. We present in this paper some preliminary outcome from the reserach done within the "Foundations of Data Warehouse Quality" (DWQ) long term research project, funded by the European Commission (n. 22469) under the ESPRIT Programme. With respect to the global picture, the role of our research within DWQ is to study a formal framework at the *conceptual level* (see Figure 1). The conceptual data model we are investigating should be able to abstract and describe the entities and relations which are relevant both in the whole enterprise, and in the user analysis of such information. In the following, we will refer to this formalism as the Data Warehouse Conceptual Data Model.

In order to overcome the above mentioned lack of formalisation of semantic data warehouse problems, our goal is to develop a novel conceptual data model which:

- has to be equipped with well-defined semantics,
- should be expressive enough to capture the data models relevant in standard (relational) database technology and in more advanced Data Warehouse applications,

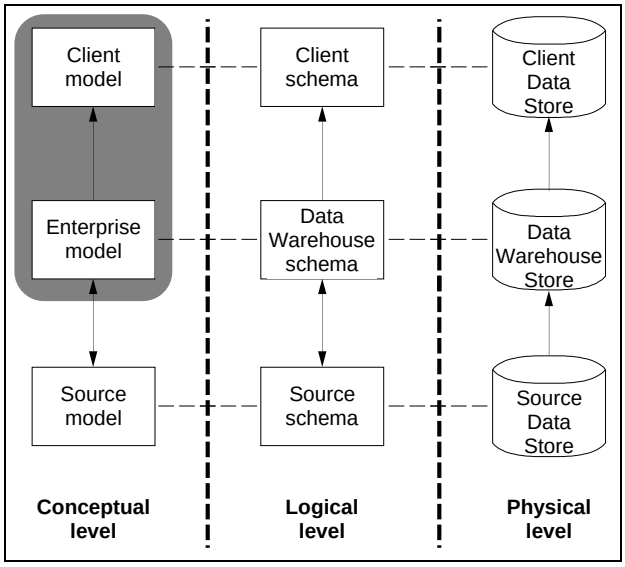

Figure 1: The role played by the Data Warehouse Conceptual Data Model with respect to the DWQ architecture.

- should give a formalisation of the operators on the data structures used in Data Warehouse applications,
- has to be able to capture inference problems relevant for reasoning in Data Warehouses like query optimisation, view reuse, update propagation, etc.

1.1 A Data Warehouse Conceptual Data Model

A DWCDM must provide means for the representation of a *multidimensional* conceptual view of data.
More precisely, a DWCDM provides the language for defining multidimensional information within a conceptual model in the data warehouse information base. As stated above, the model is of support for the conceptual design of a data warehouse, for query and view management, and for update propagation: it serves as a reference meta-model for deriving the inter-relations among entities, relations, aggregations, and for providing the integrity constraints necessary to reduce the design and maintenance costs of the data warehouse. Hence a DWCDM must be expressive enough to describe both the abstract business domain concerned with the specific application (*Enterprise model*)—just like a conceptual schema in the traditional database world—and the possible views of the enterprise information a user may want to analyse (*Client model*)—with particular emphasis on the aggregated views, which are peculiar to a data warehouse architecture (see Figure 1). A multidimensional modeling object in the logical perspective—e.g., a materialised view, a query, or a cube—should always be related with some (possibly aggregated) entity in the conceptual schema.

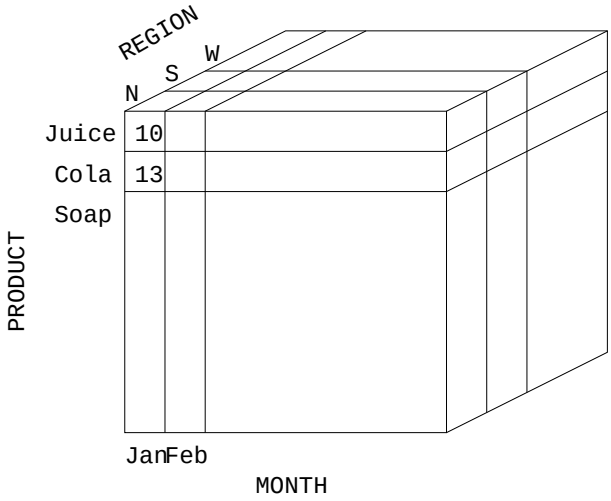

Figure 2: Sales volume as a function of product, time, location.

In the following, we will briefly introduce the ideas behind a multidimensional data model and compare it with a traditional relational data model. A more comprehensive introduction has been done in the forthcoming book "Fundamentals of Data Warehousing" [Baader *et al.*, 1999], Chapter 4 on *Multidimensional Aggregation*.
Relational database tables contain records (or rows). Each record consists of fields (or columns). In a normal relational database, a number of fields in each record (keys) may uniquely identify each record. In contrast, the multidimensional data model is an n-dimensional array (sometimes called a *hypercube* or *cube*). Each dimension has an associated hierarchy of levels of consolidated data. For instance, a spatial dimension might have a hierarchy with levels such as country, region, city, office.
Measures (which are also known as variables or metrics)—like Sales in the example, or budget, revenue, inventory, etc.—in a multidimensional array correspond to columns in a relational database table whose values functionally depend on the values of other columns. Values within a table column correspond to values for that measure in a multidimensional array: measures associate values with points in the multi-dimensional world. For example, the measure of the sales of the product Cola, in the northern region, in January, is 13,000. Thus, a dimension acts as an index for identifying values within a multi-dimensional array. If one member of the dimension is selected, then the remaining dimensions in which a range of members (or all members) are selected defines a sub-cube. If all but two dimensions have a single member selected, the remaining two dimensions define a spreadsheet (or a slice or a page). If all dimensions have a single member selected, then a single cell is defined. Dimensions offer a very concise, intuitive way of organising and selecting data for retrieval, exploration and analysis. Usual pre-defined dimension levels

(or Roll-Ups) for aggregating data in DW are: temporal (e.g., year vs. month), geographical/spatial (e.g., Rome vs. Italy), organisational (meaning the hierarchical breakdowns of your organisation, e.g., Institute vs. Department), and physical (e.g., Car vs. Engine).
A value in a single cell may represent an *aggregated* measure computed from more specific data at some lower level of the same dimension. Aggregation involves computing *aggregation functions* – according to the attribute hierarchy within dimensions or to cross-dimensional formulas – for one or more dimensions. For example, the value 13,000 for the sales in January, may have been consolidated as the sum of the disaggregated values of the weekly (or day-by-day) sales. Another example introducing an aggregation grounded on a different dimension is the cost of a product – e.g., a car – as being the sum of the costs of all of its components.
In order to provide an adequate conceptualization of multidimensional information, a Data Warehouse Conceptual Data Model should provide the possibility of explicitly modeling the relevant *aggregations* and *dimensions*.
According to a conservative point of view, a desirable Data Warehouse Conceptual Data Model should extend some standard modeling formalism (such as Entity-Relationship or OMT) to allow for the description of both aggregated entities of the domain – together with their properties and their relationships with the other relevant entities – and the dimensions involved. This document is about a proposal for a Data Warehouse Conceptual Data Model having the following properties:

- it is able to capture the database schemata expressed in the most interesting Semantic Data Models and Object-Oriented Data Models;
- descriptions of dimensions and aggregated entities are part of the model;
- it is based on *Description Logics*, a class of formalisms for which it is possible to study expressivity in relation to the decidability of the reasoning problems and the completeness of algorithms;
- it supports the most important reasoning services for the basic Data Warehouse operations.

1.2 Reasoning with a Data Warehouse Conceptual Model

Assuming that the syntax, semantics, and operators of the Data Warehouse Conceptual Data Model are defined according to the requirements stated above, it remains to specify relevant inference problems, to investigate these problems with respect to their computational complexity, and to develop reasoning algorithms for them. This will provide a theoretical and algorithmic basis which can be used for the design and evolution of a data warehouse and for semantic query optimisation.
As in traditional representation formalisms, many inference problems can be reduced to satisfiability and containment. Satisfiability of classes (or of queries, by means of the classes they represent) is the problem whether there exists a world such that each class of a given set of classes has at least one instance. Containment asks whether one class is more general than another one, that is whether each instance of the latter class is always also an instance of the more general one. It is well-known that solutions to these problems can be used for optimising queries: for example, if a query (resp. the class it represents) is not satisfiable, we do not need to process it since its result is surely empty. If a query is contained in a materialised view, then this view can be used to process the query instead of searching in a larger table.
Now, data warehouse applications confront us with a third inference problem. Aggregation is the central means to summarise and condense the information contained in the various sources. It occurs (1) when integrating data from sources, (2) when building views for the data marts, and (3) in ad-hoc queries. As queries to the sources or to larger views are far more expensive than those to smaller views, we are confronted with a new problem, namely, given a query involving aggregation and a (materialised) view, can this query be computed using (the aggregations contained in) this view. This depends on whether the aggregations contained in the view are still fine-grained enough to compute the aggregations required by the query. For example, suppose the users asks the system to compute a query Q, namely the total profit of all product groups for each year and each region. If a (materialised) view V exists which contains the profit for the product groups food and non-food for all quarters for all regions, then the total profit can be computed by simply summing up those partial profits for each year (given that we sell only food and non-food and that nothing is both food and non-food). Please note that the query Q is not contained in the view V in the classical reading of containment. Nevertheless, V can be used to compute Q.
In the following, we will call this relationship between a query (or a view) and a view (or a query) *refinement*. The main difference between the containment relation and the refinement relation is the following: for a view V to be contained in another view V', each element of V is also an element of V', and, roughly spoken, they can be obtained simply by erasing some lines or columns of V'. For a view V to be more coarse-grained than another view V', erasing is no longer sufficient. It might be necessary to aggregate some elements of V' to build an element of V.
The last reasoning task to be cited here is the retrieval of all those instances in a given data base which satisfy certain properties. Traditionally, these properties are specified using an expressive query language like SQL, conjunctive queries, QBE, etc. This high expressiveness is possible since, in general, to answer a query (that

is, to retrieve all instances satisfying the query) is less complex than deciding, for example, if a given query is satisfiable. Summing up, we are confronted with four reasoning services or problems:

- to decide whether queries (or views) are satisfiable,
- whether one is contained in another,
- whether one is refined by another, and
- to answer a query.

The first three reasoning problems belong to the intensional reasoning problems, whereas the last one belongs to the extensional reasoning problems. In general, intensional reasoning is more complicated than extensional reasoning. As a consequence, given that all these problems should be decidable, one may use a more expressive language to formulate extensional problems than the language used to formulate intensional problems. A logical approach for reasoning is surely useful not only for integrating heterogeneous sources and optimising (aggregate) queries, but also for update propagation, DW design and DW evolution. In update propagation, the information provided by integrity constraints (expressed in some logic) can be used to reduce the maintenance cost of the DW. This information along with reasoning mechanisms for checking query containment or query refinement (more generally, query rewriting over views) can be used for optimal DW design, incremental DW design, and DW evolution.

The paper is organised as follows. After having introduced the notion of DWCDM, Section 2 will propose a basic modeling language—based on Description Logics—which is expressive enough to capture the traditional semantic data models and Object-Oriented data models. The core part of this document (Section 3) is the proposal of a conceptual modeling language which emphasises the possibility to describe the explicit *structure* of aggregations instead of the way the values of the measures of those aggregations are computed through aggregation functions.

2 The basic Modeling Language

The formal language for the basic conceptual level representation is based on Description Logics. Description Logics[1] are formalisms designed for a logical reconstruction of representation tools such as *frames*, *Object-Oriented* and *semantic* data models, *semantic networks*, KL-ONE-like languages [Woods and Schmolze, 1992], *type systems*, and *feature logics*. Nowadays, description logics are also considered the most important unifying formalism for the many object-centred representation languages used in areas other than Knowledge

[1]Description Logics have been also called *Frame-Based Description Languages*, *Term Subsumption Languages*, *Terminological Logics*, *Taxonomic Logics*, *Concept Languages* or KL-ONE-*like languages*.

C, D	$\rightarrow$	$A \mid$	A	(primitive concept)
		$\top \mid$	`top`	(top)
		$\bot \mid$	`bottom`	(bottom)
		$\neg C \mid$	(`not` C)	(complement)
		$C \sqcap D \mid$	(`and` C D ...)	(conjunction)
		$C \sqcup D \mid$	(`or` C D ...)	(disjunction)
		$\forall R.C \mid$	(`all` R C)	(univ. quantifier)
		$\exists R.C \mid$	(`some` R C)	(exist. quantifier)
		$f\uparrow \mid$	(`undefined` f)	(undefinedness)
		$f : C$	(`in` f C)	(selection)
R, S	$\rightarrow$	$P \mid$	P	(primitive role)
		$f \mid$	f	(feature)
		$R^{-1} \mid$	(`inverse` R)	(inverse role)
		$R\|_C \mid$	(`restrict` R C)	(range restriction)
		$R \circ S$	(`compose` R S ...)	(role chain)
f, g	$\rightarrow$	$p \mid$	p	(primitive feature)
		$f \circ g$	(`compose` f g ...)	(feature chain)

Figure 3: Syntax rules for the $\mathcal{ALCFI}^+$ Description Logic.

Representation. In particular, [Buchheit *et al.*, 1994, Calvanese *et al.*, 1994, Calvanese *et al.*, 1995] propose a formal mapping between description logics, Semantic Data Models, and Object-Oriented formalisms. Important characteristics of Description Logics are high expressivity together with decidability, which guarantee that reasoning algorithms always terminate with the correct answers. Unlike Object-Oriented systems, description logics do not stress the representation of the behavioural aspect of information, for which they are still considered inadequate.

In this section we give a brief introduction to a basic description logic, which will serve as the basic representation language for our DWCDM proposal. With respect to the formal apparatus, we will strictly follow the concept language formalism introduced by [Schmidt-Schauß and Smolka, 1991] and further elaborated, for example, by [Donini *et al.*, 1991a, Donini *et al.*, 1991b, Donini *et al.*, 1992, Buchheit *et al.*, 1993, Donini *et al.*, 1994, De Giacomo and Lenzerini, 1995, De Giacomo and Lenzerini, 1996]: in this perspective, Description Logics are considered as a *structured* fragment of predicate logic. $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] is the minimal description language including full negation and disjunction—i.e., propositional calculus, and it is a notational variant of the propositional modal logic $\boldsymbol{K}_{(m)}$ [Schild, 1991].

The basic types of a concept language are *concepts*, *roles*, and *features*. A concept is a description gathering the common properties among a collection of individuals; from a logical point of view it is a unary predicate ranging over the domain of individuals. Interrelationships between these individuals are represented either by means of roles (which are interpreted as binary relations over the domain of individuals) or by means of

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.\, C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid \forall j.\, (i,j) \in R^{\mathcal{I}} \Rightarrow j \in C^{\mathcal{I}}\} \\
(\exists R.\, C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid \exists j.\, (i,j) \in R^{\mathcal{I}} \wedge j \in C^{\mathcal{I}}\} \\
(f\!\uparrow)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus dom\, f^{\mathcal{I}} \\
(f : C)^{\mathcal{I}} &= \{i \in dom\, f^{\mathcal{I}} \mid f^{\mathcal{I}}(i) \in C^{\mathcal{I}}\} \\
\\
(R^{-1})^{\mathcal{I}} &= \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (j,i) \in R^{\mathcal{I}}\} \\
(R|_C)^{\mathcal{I}} &= R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \times C^{\mathcal{I}}) \\
(R \circ S)^{\mathcal{I}} &= R^{\mathcal{I}} \circ S^{\mathcal{I}}
\end{aligned}
$$

Figure 4: The semantics of $\mathcal{ALCFI}^+$.

features (which are interpreted as partial functions over the domain of individuals). Both roles and features can be used to individuals to certain properties. In the following, we will consider the Description Logic $\mathcal{ALCFI}^+$, extending $\mathcal{ALC}$ with features (i.e., functional roles), inverse roles, role composition, and role restrictions.

According to the syntax rules of Figure 3, $\mathcal{ALCFI}^+$ *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A), *roles* (denoted by the letter R, S), and *features* (denoted by the letters f, g); roles are built out of *primitive roles* (denoted by the letter P) and features are built out of *primitive features* (denoted by the letter p); it is worth noting that features are considered as special cases of roles.

Let us now consider the formal semantics of the $\mathcal{ALCFI}^+$. We define the *meaning* of concepts as sets of individuals—as for unary predicates—and the meaning of roles as sets of pairs of individuals—as for binary predicates. Formally, an *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a set $\Delta^{\mathcal{I}}$ of individuals (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) mapping every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every feature to a partial function from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$, such that the equations in Figure 4 are satisfied.

For example, we can consider the concept of HAPPY FATHERS, defined using the primitive concepts `Man`, `Doctor`, `Rich`, `Famous` and the roles `CHILD`, `FRIEND`. The concept HAPPY FATHERS can be expressed in $\mathcal{ALCFI}^+$ as

$$\texttt{Man} \sqcap (\exists \texttt{CHILD}.\, \top) \sqcap \\ \quad \forall \texttt{CHILD}.\, (\texttt{Doctor} \sqcap \exists \texttt{FRIEND}.\, (\texttt{Rich} \sqcup \texttt{Famous})),$$

i.e., those men having some child and all of whose children are doctors having some friend who is rich or famous.

A *knowledge base*, in this context, is a finite set Σ of *terminological axioms*; it can also be called a *terminology* or TBox. For a concept name A, and (possibly complex) concepts C, D, terminological axioms are of the form $A \doteq C$ (concept definition), $A \sqsubseteq C$ (primitive concept definition), $C \sqsubseteq D$ (general inclusion statement). An interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ if and only if the interpretation of C is included in the interpretation of D, i.e., $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It is clear that the last kind of axiom is a generalization of the first two: concept definitions of the type $A \doteq C$ – where A is an atomic concept – can be reduced to the pair of axioms $(A \sqsubseteq C)$ and $(C \sqsubseteq A)$. Another class of terminological axioms – pertaining to roles R, S – are of the form $R \sqsubseteq S$. Again, an interpretation $\mathcal{I}$ satisfies $R \sqsubseteq S$ if and only if the interpretation of R – which is now a set of *pairs* of individuals – is included in the interpretation of S, i.e., $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a knowledge base Σ iff every terminological axiom of Σ is satisfied by $\mathcal{I}$. If Σ has a model, then it is *satisfiable*; thus, checking for KB satisfiability is deciding whether there is at least one model for the knowledge base. Σ *logically implies* an axiom α (written $\Sigma \models \alpha$) if α is satisfied by every model of Σ. We say that a concept C is *subsumed* by a concept D in a knowledge base Σ (written $\Sigma \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of Σ. For example, the concept

$$\texttt{Person} \sqcap (\exists \texttt{CHILD}.\, \texttt{Person})$$

denoting the class of PARENTS—i.e., the persons having at least a child which is a person—subsumes the concept

$$\texttt{Man} \sqcap (\exists \texttt{CHILD}.\, \top) \sqcap \\ \quad \forall \texttt{CHILD}.\, (\texttt{Doctor} \sqcap \exists \texttt{FRIEND}.\, (\texttt{Rich} \sqcup \texttt{Famous}))$$

denoting the class of HAPPY FATHERS – with respect to the following knowledge base Σ:

$$\texttt{Doctor} \doteq \texttt{Person} \sqcap \exists \texttt{DEGREE}.\, \texttt{Phd},$$
$$\texttt{Man} \doteq \texttt{Person} \sqcap \texttt{sex} : \texttt{Male},$$

i.e., every happy father is also a person having at least one child, given the background knowledge that men are male persons, and that doctors are persons.

A concept C is satisfiable, given a knowledge base Σ, if there is at least one model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$, i.e. $\Sigma \not\models C \equiv \bot$. For example, the concept

$$(\exists \texttt{CHILD}.\, \texttt{Man}) \sqcap (\forall \texttt{CHILD}.\, (\texttt{sex} : \neg \texttt{Male}))$$

is unsatisfiable with respect to the above knowledge base Σ. In fact, an individual whose children are not male cannot have a child being a man.

Concept subsumption can be reduced to concept satisfiability since C is subsumed by D in Σ if and only if $(C \sqcap \neg D)$ is unsatisfiable in Σ.

2.1 Expressivity of the basic Modeling Language

The basic Description Logic introduced in the previous section was designed such that it is able to

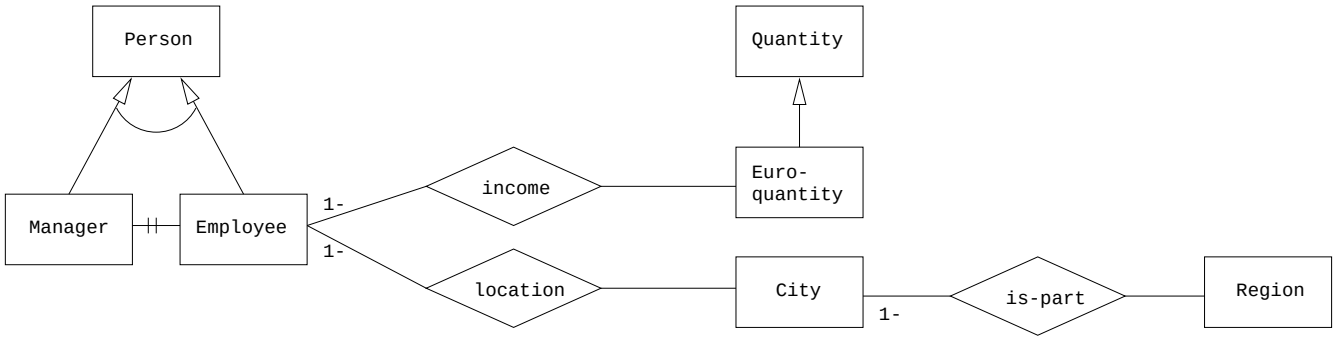

Figure 5: A schema expressed in the Extended Entity-Relationship conceptual data model.

capture database schemata expressed in the most interesting Semantic Data Models and Object-Oriented Data Models, and by the fact that it is compatible with the $\mathcal{DLR}$ Description Logic first introduced in [De Giacomo and Lenzerini, 1999]. We have chosen a limited expressivity to meet the following two goals:

- the language should be compatible with the extensions proposed in this document for handling multidimensional aggregation. In particular, the intention is to have decidable satisfiability and logical implication problems. We will briefly mention how the most general extension of a Description Logic including built-in predicates with aggregation functions makes the logic undecidable, even for very weak base Description Logics. Thus, our efforts were concentrated in the direction of a language for multidimensional aggregation *without* explicit aggregation functions.
- the final language should be implementable with the current technology. In particular, we refer to the current academic implementations of expressive Description Logics, namely the system FaCT [Horrocks, 1997]. It has been recently argued [Horrocks and Sattler, 1999] that the logic we are considering here allows for the implementation of sound and complete reasoning algorithms that should behave quite well in realistic applications.

Next, we will sketch the relationships between the $\mathcal{ALCFI}^+$ Description Logic and Entity-Relationship, Object-Oriented, and $\mathcal{DLR}$ data models. These relationships will demonstrate the adequacy of the proposed language as a general language for the conceptual level of an information system. In Section 3, we will extend the basic language with the capability to handle multidimensional aggregation.

Semantic and Object-Oriented Data Models

The most common semantic data model for database design is the Entity-Relationship (ER) model [Chen, 1976]. Figure 5 shows a simple schema represented in an extended version of the basic ER model, called EER, including ISA relationships. [Calvanese *et al.*, 1994] show how a schema expressed in an EER conceptual data model can be expressed in a suitable description logic theory – whose models correspond with legal database states of the EER schemas – allowing for reasoning services such as satisfiability of a schema or logical implication. Moreover, a description logic allows for a greater expressivity than the original EER framework, in terms of full disjunction and negation, and entity definitions by means of both necessary and sufficient conditions.

The proposals of [Calvanese *et al.*, 1994] and the more advanced one presented in [De Giacomo and Lenzerini, 1999] use the $\mathcal{ALUNI}$ and the $\mathcal{DLR}$ description logics respectively. Those logics allow for the correct translation of the cardinality restrictions of an EER schema, whereas $\mathcal{ALCFI}^+$ only allows the translation of simple cardinality restrictions, i.e., those involving only either 'zero' or 'one' as minimum cardinality. We believe that in most real domains these are the only relevant cardinality restrictions.

We will not describe in detail the transformation from EER schemata to $\mathcal{ALCFI}^+$ knowledge bases. It is only important to say that the relations are *reified* in the description logic theory, i.e., they become concepts with n special feature names denoting the n arguments of the n-ary relation. For example, the relation `INCOME` becomes a concept with the two features: `incomer` – relating to the first argument of the relation, i.e., an employee – and `incoming` – relating to the second argument of the relation, i.e., a Euro quantity.

The translation of the EER schema of Figure 5 is presented in Figure 6. Please note that every role name which appears in the formalization of an EER schema is to be considered a functional role name in the translated description logic theory; in our example `incomer`, `incoming`, `locator`, `place`, `whole`, and `part` are primitive features.

[Calvanese *et al.*, 1994] also presents the relationship between a Description Logic and a generic Object-Oriented formalism. The translation of the structural part of an O-O schema into a description logic knowledge base is similar to the one sketched for ER schemas; we will not go into details here. We simply point out that the $\mathcal{ALCFI}^+$ Description Logic is expressive

$\texttt{INCOME} \sqsubseteq \texttt{incomer} : \texttt{Employee} \sqcap \texttt{incoming} : \texttt{Euro-quantity}$
$\texttt{LOCATION} \sqsubseteq \texttt{locator} : \texttt{Employee} \sqcap \texttt{place} : \texttt{City}$
$\texttt{IS-PART} \sqsubseteq \texttt{part} : \texttt{City} \sqcap \texttt{whole} : \texttt{Region}$
$\texttt{Employee} \sqsubseteq \texttt{Person} \sqcap \exists \texttt{incomer}^{-1}.\texttt{INCOME} \sqcap \exists \texttt{locator}^{-1}.\texttt{LOCATION}$
$\texttt{Manager} \sqsubseteq \texttt{Person}$
$\texttt{Euro-quantity} \sqsubseteq \texttt{Quantity}$
$\texttt{City} \sqsubseteq \exists \texttt{part}^{-1}.\texttt{IS-PART}$

Figure 6: DL translation of the ER schema.

enough to capture that translation.

The Description Logic $\mathcal{DLR}$

$\mathcal{DLR}$ is the DWCDM proposed by [De Giacomo and Lenzerini, 1999]. There is no competition between $\mathcal{DLR}$ and $\mathcal{ALCFI}^+$. In fact, it turns out that it is always possible to translate a $\mathcal{DLR}$ knowledge base—including n-ary relations but *without* cardinality constraints ≥ 2, negation or conjunction of relations—into an $\mathcal{ALCFI}^+$ knowledge base. This emerges from a careful reading of the encoding of $\mathcal{DLR}$ knowledge bases into $\mathcal{CIQ}$ knowledge bases: the only operators used in the translation which are present in $\mathcal{CIQ}$ but not in $\mathcal{ALCFI}^+$ are the counting existential quantifiers $\exists^{\geq n}$ with $n \geq 2$, and the negation or conjunction of relations.
Please note the notational ambiguity of the "standard" description logics *feature selection* operator ":" which is used in $\mathcal{ALCFI}^+$, with the use of the same symbol in $\mathcal{DLR}$. In the $\mathcal{DLR}$ notation, the terminological axiom resulting from the translation of the above ER schema

$$\texttt{INCOME} \sqsubseteq (\texttt{incomer} : \texttt{Employee}) \sqcap (\texttt{incoming} : \texttt{Euro-quantity})$$

states that the *binary relation* `INCOME` is defined as having an `Employee` as its first argument (the `incomer`) and a `Euro-quantity` as its second argument (the `incoming`). However, we have seen that this is encoded in $\mathcal{ALCFI}^+$ using the very same terminological axiom, where `INCOME` is now a concept and not a binary relation.

3 Structuring Aggregation

We introduce an extension of the basic description logic which does not explicitly include aggregation functions, but does make possible the description of the explicit *structure* of aggregations. Thus, the conceptual language will be able to abstract properties of aggregations, their interrelationships, and, most notably, their components: a Data Warehouse Conceptual Schema may contain detailed descriptions of the structure of aggregates. This result is obtained by making aggregations first class citizens of the representation language: it is possible to describe properties of aggregations in the same way that it is possible to describe properties of classes of individuals; it is possible to describe the components of an aggregations, and the relationships that the properties of the components may have with the properties of the aggregation itself; it is possible to build aggregations out of other aggregations, i.e., it is possible for an aggregation to be explicitly composed by other aggregations. This approach closely resembles the one pursued by [Catarci *et al.*, 1995, De Giacomo and Naggar, 1996], in the sense of proposing a conceptual data model in which aggregations are first-class entities intensionally described by means of their components. As we have pointed out, the description of an aggregation is not going to include a specification of *how* it is specifically built out of some particular components; that is, an aggregation function is not explicitly determined.
The basic idea for this extension is to introduce explicitly into the language a special binary relation, i.e., a role having particular properties, written "$\succeq$" (to be read "`aggregates`"), which relates an aggregation with its components. This is a radical departure from [Catarci *et al.*, 1995, De Giacomo and Naggar, 1996], where the relation among aggregations and components is not explicitly present in the language as a role.
The paper [Artale *et al.*, 1996b] surveys several alternative approaches for representing the structure of aggregates according to this idea, while [Artale *et al.*, 1996a] surveys the technical problems which are introduced in a description logic with such a representation for aggregates. In the following, section 3.1 will introduce the basic framework for the conceptual modeling of multidimensional aggregations, together with an explanatory example. In section 3.2 a simple extension of the description logic $\mathcal{ALCFI}^+$ – including an explicit `aggregates` relation – is introduced, and the previous example will be formalized using that language.

3.1 Multidimensional Aggregations

As stated in [Agrawal *et al.*, 1995], a "good" data warehouse system should support user-definable *multiple* hierarchies along *arbitrary* dimensions. In section 1.1 we have briefly defined a dimension as an index for identifying measures within a multidimensional data model. A dimension is basically a domain, which may possibly be structured in hierarchies of levels. For example, in the context of a statistical study of the happiness of employees (see Figure 7), possible dimensions are income and location; chosen dimension levels may be Euro quantity and city. A partitioning of a dimension defines a particular level for that dimension. For instance, a spatial dimension (like *location* above) might have a hierarchy with levels such as country, region, city, office. A set of levels of different dimensions (one level per dimension) defines a hypercube for a measure depending on those dimensions. For example, "average age" can be a measure depending on the levels *income-category* (distinguishing between rich and poor) in the income dimension, and *region* in the spatial dimension.

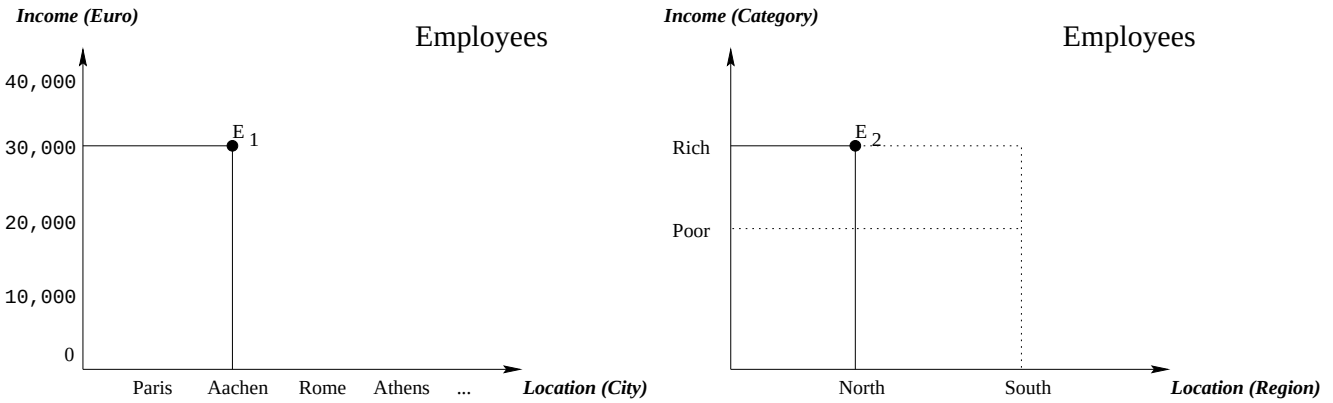

Figure 7: The cube of employees by income in Euros and city, and the *consolidated* cube of employees by income category and region.

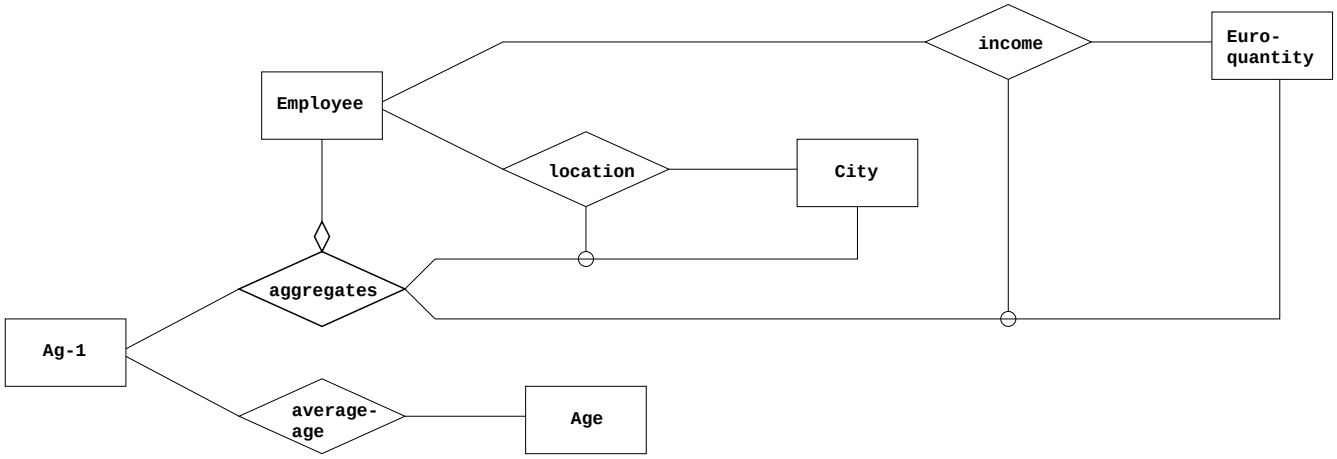

Figure 8: A Conceptual Data Warehouse Schema for the information base of the left table of Figure 7.

In the conceptual data model, "dimension" is a synonym for a domain of an attribute (or of attributes) that is structured by a hierarchy and/or an order. In order to support multiple hierarchies, the data model must provide means for defining and structuring these hierarchies, and for arbitrary aggregation along the hierarchies.

A conceptual data model where both multidimensional aggregations and multiple hierarchically organised dimensions can be abstracted and described could provide support for query languages in multidimensional data models. In fact, in the few attempts where a *cube algebra* introduces the notion of multiple dimensions and of levels within dimensions (e.g., [Cabibbo and Torlone, 1997]) the Data Warehouse Conceptual Schema could serve as a *reference meta-model* for deriving the inter-relations among levels and dimensions. [Hacid and Sattler, 1997] presents a proposal for an extension of the cube algebra introduced in [Agrawal *et al.*, 1995], which makes explicit use of structured dimensions.

Let us now consider a concrete example of a conceptual schema for the information displayed in the two tables of Figure 7. Each cell in the bi-dimensional cube on the left denotes the aggregation composed by all the employees having some income (in Euros) and working in some location (a city). In particular, cell E_1 is the aggregation composed of all those employees having an income of approximately 30,000 Euros and working in Aachen. It is clear that E_1 may include more than one employee, and it may itself have some properties which depend on all of its components. For example, E_1 may have the property `count` which says how many employees—i.e., how many components of it—actually have an income of 30,000 Euros and work in Aachen, and the property `average-age` with the average age from all those employees. Of course, these properties may be computed by some aggregation function from the properties of the components.

An adequate basic conceptual schema for this simple multidimensional information base should include entities such as `Employee`, `City`, and `Euro-quantity` and relations such as `income`, and `location`. Moreover, the schema should also include an additional *aggregated entity*, say `Ag-1`, namely the class denoting the aggregations of employees by city as location and Euro quan-

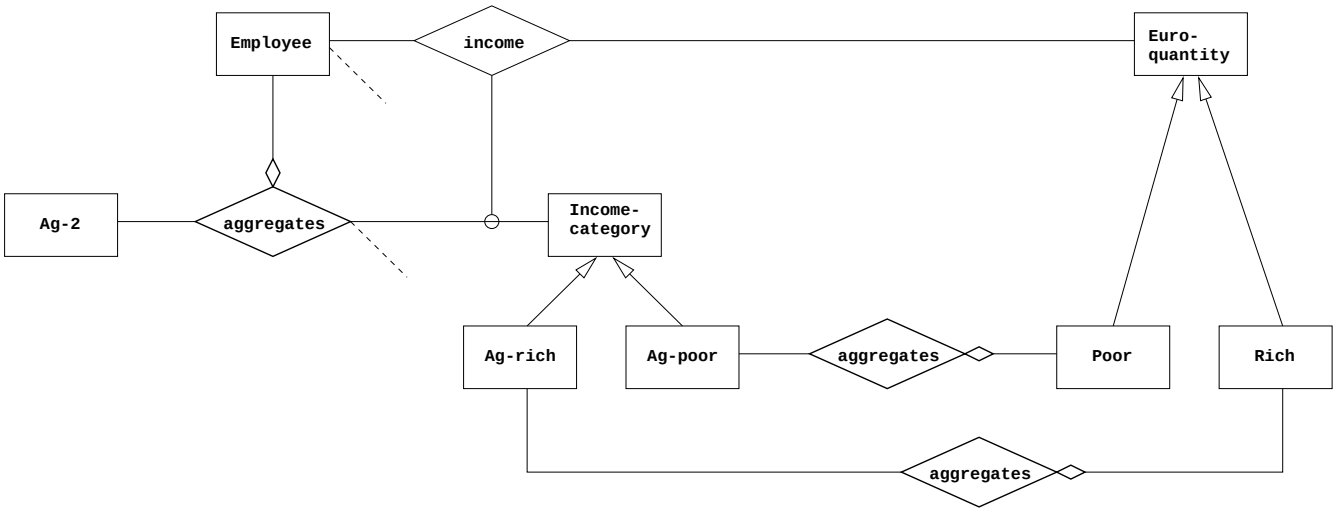

Figure 9: The extensions for the Conceptual Data Warehouse Schema of the information base of Figure 7.

tities as income; such an aggregated entity should also have properties such as `count` and `average-age`. We can also say that `Ag-1` aggregates employees according to the (basic) level `Euro-quantity` and the (basic) level `City` of the dimensions `income` and `location` respectively. It is clear that E_1 is one of the aggregations denoted by `Ag-1`.

Figure 8 represents the schema in a variant of the Entity-Relationship data model. Please note that the particular way of representing aggregated entities in the figure is inspired by [Catarci *et al.*, 1995, De Giacomo and Naggar, 1996]. However, in the following sections we will show how our formalisation of this representation differs fromand extends the original one, being grounded on the special binary relation `aggregates`. Moreover, it is clear from the schema that in the conceptual description there is no specification on how aggregation functions possibly connect properties of components with properties of aggregates: it is unknown whether the average age of the aggregation of employees `Ag-1` may be computed by means of an *average* function from the age of each employee which is part of the aggregation.

If we also consider as part of the multidimensional information base the *aggregated view* represented by the table on the right of Figure 7–denoting the aggregation composed by the employees having some income (aggregated in simple categories such as rich and poor) and working in some location (aggregated in northern and southern regions)—more conceptual entities come into play.

Cell E_2 is the aggregation composed by employees having a high income and working in a northern region. Analogously with E_1, E_2 may have the property `count` which says how many employees actually have a high income and work in a northern region, and the property `average-age` which *computes* the average age from all those employees.

Thus, we need to add both a new aggregated entity and the definitions of the newly introduced levels for the dimensions `income` and `location`. The new aggregated entity, let's call it `Ag-2`, aggregates employees according to the level `Income-category` and the level `Region` of the dimensions `income` and `location` respectively. It is clear that E_2 is one of the aggregations denoted by `Ag-1`. The level `Income-category` is obtained by aggregating Euro quantities into the two aggregations `Ag-rich` and `Ag-poor` according to the partitioning of the `Euro-quantity` entity into the two sub-entities, `Rich` and `Poor` respectively. The level `Region` is obtained by aggregating cities into the two aggregations `Ag-north` and `Ag-south` according to the partitioning of the `City` entity into the two sub-entities `North` and `South` respectively. To sum up, we have identified a new aggregated entity `Ag-2` and the new dimensional entities `Income-category`, `Rich`, `Poor`, `Ag-rich`, `Ag-poor`, `Region`, `North`, `South`, `Ag-north`, and `Ag-south`.

Figure 9 represents the extensions required to the original schema (for reasons of space, some dimensions have been left out).

3.2 The Language for Structured Aggregations

In this section a simple extension of the Description Logic $\mathcal{ALCFI}^+$ is introduced. The language is augmented with an explicit "`aggregates`" binary relation—i.e., a role—with its inverse "`is-aggregated`", and with a special class denoting atomic aggregates—i.e. aggregations having no components. The syntax of the Description Logic is extended with the following rules:

$$C, D \rightarrow \mho \qquad \text{(atomic aggregate)}$$

$$R, S \rightarrow \succeq \mid \qquad \text{(aggregation role)}$$
$$\preceq \qquad \text{(decomposition role)}$$

The semantics of the new operators is defined as follows:

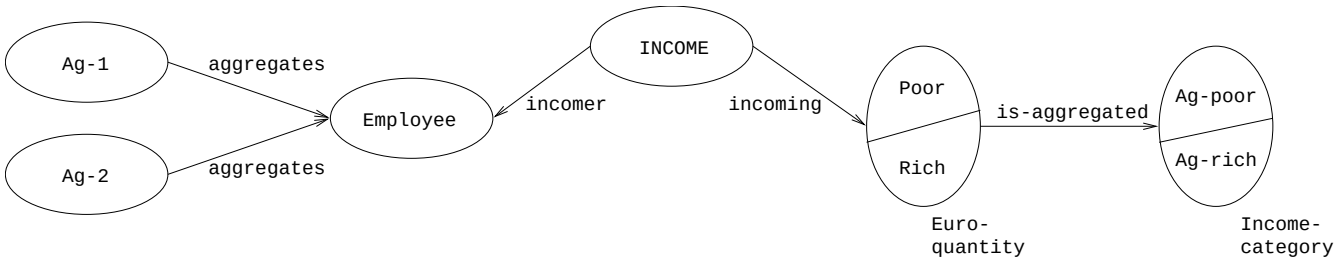

Figure 10: The role paths in the Description Logic theory.

$$\begin{aligned} \succeq^{\mathcal{I}} &\quad \text{is a transitive relation in } \Delta^{\mathcal{I}} \\ \mathcal{U}^{\mathcal{I}} = &\quad \{i \in \Delta^{\mathcal{I}} \mid \forall j.\, (i,j) \notin \succeq^{\mathcal{I}}\} \\ \preceq^{\mathcal{I}} = &\quad \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (j,i) \in \succeq^{\mathcal{I}}\} \end{aligned}$$

where it is worth noting that the aggregates role $\succeq$ must be interpreted as a transitive relation. In fact, if we consider as an example the aggregated entity `Car`, described by having wheels as components which in turn have tires as components:

$$\texttt{Car} \doteq \exists \succeq.\, (\texttt{Wheel} \sqcap \exists \succeq.\, \texttt{Tire}),$$

then the fact that tires are also parts of cars is logically implied by the above terminological axiom, i.e.,

$$\texttt{Car} \sqsubseteq \exists \succeq.\, \texttt{Tire}.$$

is a consequence of the definition of car. This deduction is valid only if the `aggregates` relation is transitive.
In order to understand the full potential of the formalism, we will reconsider the example given for an EER data model in the previous section. We start to formalise the basic part of the schema of Figure 8, i.e., the pure EER part, in a similar way to the formalisation presented in Section 2.1. Please recall that every role name which appears in the translation of an EER schema in a Description Logic knowledge base—with the exception of the aggregation roles—is to be considered a functional role name.

$$\begin{aligned} &\texttt{INCOME} \sqsubseteq \texttt{incomer} : \texttt{Employee} \sqcap \\ &\qquad \texttt{incoming} : \texttt{Euro-quantity} \\ &\texttt{LOCATION} \sqsubseteq \texttt{locator} : \texttt{Employee} \sqcap \texttt{place} : \texttt{City} \end{aligned}$$

The aggregated entity `Ag-1` is defined as being an aggregation composed by those employees having an income in Euros and a location in a city:

$$\begin{aligned} &\texttt{AVERAGE-AGE} \sqsubseteq \texttt{theme} : \texttt{Ag-1} \sqcap \texttt{age} : \texttt{Age} \\ &\texttt{Ag-1} \sqsubseteq \neg\mathcal{U} \sqcap \forall \succeq.\texttt{Employee} \sqcap \\ &\qquad \forall(\succeq \circ \texttt{incomer}^{-1}|_{\texttt{INCOME}} \circ \texttt{incoming}).\texttt{Euro-quantity} \sqcap \\ &\qquad \forall(\succeq \circ \texttt{locator}^{-1}|_{\texttt{LOCATION}} \circ \texttt{place}).\texttt{City} \end{aligned}$$

Figure 10 may help the reader to better understand the complex role paths appearing in the definitions of aggregations. `Ag-1` is the class denoting all the possible aggregations composed by employees which are `incomers` of some `INCOME` in `Euro-quantity`. In a similar way it is possible to reconstruct the path through the `LOCATION` dimension. According to [De Giacomo and Naggar, 1996], `Ag-1` is a *simple aggregate declaration*.
We now introduce the aggregations defining the new level in the income dimension (compare with Figure 9).

$$\begin{aligned} &\texttt{Rich} \doteq \texttt{Euro-quantity} \sqcap \neg\texttt{Poor} \\ &\texttt{Euro-quantity} \doteq \texttt{Rich} \sqcup \texttt{Poor} \\ &\texttt{Ag-rich} \doteq \neg\mathcal{U} \sqcap \forall \succeq.\texttt{Rich} \\ &\texttt{Ag-poor} \doteq \neg\mathcal{U} \sqcap \forall \succeq.\texttt{Poor} \\ &\texttt{Income-category} \doteq \texttt{Ag-rich} \sqcup \texttt{Ag-poor} \end{aligned}$$

In an analogous way it is possible to define the new level for the location dimension. The aggregated entity `Ag-2` is defined as being an aggregation composed of those employees having an income category and a location in a region:

$$\begin{aligned} \texttt{Ag-2} \sqsubseteq\ & \neg\mathcal{U} \sqcap \forall \succeq.\texttt{Employee} \sqcap \\ & \big(\forall(\succeq \circ \texttt{incomer}^{-1}|_{\texttt{INCOME}} \circ \texttt{incoming}|_{\texttt{Euro-quantity}} \circ \preceq).\texttt{Ag-rich} \sqcup \\ & \quad \forall(\succeq \circ \texttt{incomer}^{-1}|_{\texttt{INCOME}} \circ \texttt{incoming}|_{\texttt{Euro-quantity}} \circ \preceq).\texttt{Ag-poor}\big) \sqcap \\ & \big(\forall(\succeq \circ \texttt{locator}^{-1}|_{\texttt{LOCATION}} \circ \texttt{place}|_{\texttt{city}} \circ \preceq).\texttt{Ag-north} \sqcup \\ & \quad \forall(\succeq \circ \texttt{locator}^{-1}|_{\texttt{LOCATION}} \circ \texttt{place}|_{\texttt{city}} \circ \preceq).\texttt{Ag-south}\big) \end{aligned}$$

Recall that `Ag-2` is the class of all aggregations such that each one of them aggregates employees having the same income (at the level of income category) and the same location (at the level of region). This is opposed to `Ag-1` aggregating employees having the same income (at the level of Euros) and the same location (at the level of city). According to [De Giacomo and Naggar, 1996], `Ag-2` is a *complex aggregate declaration*.
Each aggregation of employees belonging to the class denoted by `Ag-2` includes either only rich employees (incomers of some income which is a Euro quantity in the rich aggregation of Euro quantities) or only poor employees (incomers of some income which is a Euro quantity in the poor aggregation of Euro quantities). In a similar way, each aggregation of `Ag-2` includes either only northern employees, or only southern employees. Thus, aggregations denoted by `Ag-2` may be of four possible types: northern rich employees, northern poor employees, southern rich employees, or southern poor employees.
In order to enforce the abovementioned cardinalities on

the extension of the introduced aggregations, we should add the following constraints (which are, however, not supported by the current Description Logic):

$$\begin{array}{l} |\ \texttt{Ag-1}\ | = |\ \texttt{City}\ | \cdot |\ \texttt{Euro-quantity}\ | \\ |\ \texttt{Ag-north}\ | = |\ \texttt{Ag-south}\ | = |\ \texttt{Ag-rich}\ | = |\ \texttt{Ag-poor}\ | = 1 \\ |\ \texttt{Ag-2}\ | = (|\ \texttt{Ag-rich}\ | + |\ \texttt{Ag-poor}\ |) \cdot \\ \qquad\qquad (|\ \texttt{Ag-north}\ | + |\ \texttt{Ag-south}\ |) = 4 \end{array}$$

Note that *simple aggregate declarations* always introduce a unique aggregation: thus, simple aggregates do have a cardinality equal to one. *Complex aggregate declarations* introduce aggregations in a number equal to the product of the cardinalities of the chosen level for each dimension.

It is worth noting that the complex roles

$$(\succeq \circ \texttt{incomer}^{-1}\ |_{\text{INCOME}} \circ \texttt{incoming}\ |_{\text{Euro-quantity}} \circ \preceq)$$
$$(\succeq \circ \texttt{locator}^{-1}\ |_{\text{LOCATION}} \circ \texttt{place}\ |_{\text{City}} \circ \preceq)$$

represent the precise and complete internal structure of the interrelationship between the *target* of an aggregation—in this case `Employee` which is related to the aggregation via the `aggregates` relation—and the dimensions defining the way the aggregation is built—in this case the income and location dimensions.
In [Catarci *et al.*, 1995, De Giacomo and Naggar, 1996], the definition of those interrelationships were left unspecified, and there was an asymmetry in the semantic treatment of *simple* aggregate declarations (like `Income-category` in our example) – using the *membership* relation to denote the components – and *complex* aggregate declarations (like `Ag-1` and `Ag-2`) – using the *target* relation to denote the components.

3.3 Aggregation Functions

In order to correctly model the notion of an aggregate, we can not restrict ourselves to describing its *meronymic* structure – i.e., the existence of an aggregation relation between the aggregated entity and its components – rather we must be able to express *how* the aggregate is related to its components, and how the components are "glued together" to form an aggregate.
An important result of the research within the DWQ project identifies the borders for the possible extensions of a Data Warehouse Conceptual Data Model towards the explicit inclusion of aggregation functions [Baader and Sattler, 1998]. It has turned out that the *explicit* presence of arbitrary aggregation functions, when viewed as a means to define new attribute values for aggregated entities, and built-in predicates in a concrete domain increases the expressive power of the basic conceptual model in such a way that all interesting inference problems may easily become undecidable. Moreover, this result is very tightly bounded: extending $\mathcal{FL}_0$, a very weak Description Logic allowing only conjunction and universal value restrictions (which is included in $\mathcal{ALCFI}^+$) with a weak form of aggregation, already

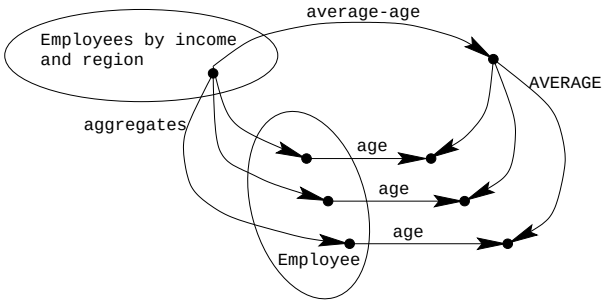

Figure 11: The average age of employees aggregated by income and region.

leads to the undecidability of satisfiability and subsumption. On the other hand, recent research has shown that appropriate restrictions of the allowed aggregation functions yield decidability of these problems. These results concern (1) the use of aggregation functions in nested concepts and (2) concrete domains like the integers, the non-negative integers, the rationals, and the reals.
Given the restrictions posed by these negative results, we follow in this paper a mere structural approach. In order to represent a weak kind of dependency between components and aggregates we make use of the so-called *structural descriptions* in Description Logic. The syntax of the Description Logic is extended with the following rule:

$$C, D \quad \rightarrow \quad \rightleftharpoons R_1\ R_2.\ S \qquad \text{(role constraint)}$$

The extensional semantics of the new operator is the following:

$$(\rightleftharpoons R_1\ R_2.\ S)^{\mathcal{I}} = \{i \in \Delta^{\mathcal{I}} \mid \forall j,k.\ (i,j) \in R_1^{\mathcal{I}} \wedge (i,k) \in R_2^{\mathcal{I}} \Rightarrow (j,k) \in S^{\mathcal{I}}\}$$

Structural descriptions may serve to emulate a simplified version of aggregate functions. Consider the example of the aggregated entity `Ag-2` (introduced in Section 3.2) denoting aggregations of employees by income category (poor and rich) and region (north and south). The functional attribute `average-age` of such an aggregated entity may be represented as the `AVERAGE` of the values of the functional attribute `age` of the employees composing the aggregate itself:

$$\texttt{Ag-2} \sqsubseteq \rightleftharpoons \texttt{average-age}\ (\succeq |_{\text{Employee}} \circ \texttt{age}).\ \texttt{AVERAGE}$$

This definition states that for each aggregate in `Ag-2` there is a relation of type `AVERAGE` between the value of its attribute `average-age` and all the values of the attribute `age` for each employee composing the aggregate—see Figure 11.

4 Conclusions

We have introduced a *Data Warehouse Conceptual Data Model*, extending the most interesting traditional Semantic Data Models and Object-Oriented Data Models, which allows the representation of a multidimensional conceptual view of data. We have seen how the proposed conceptual data model is able to introduce complex descriptions of the structure of aggregated entities and multiply hierarchically organised dimensions. In order to support multiple hierarchies, the data model provides means for defining and structuring these hierarchies, and for arbitrary aggregation along the hierarchies. The proposed framework does not yet include an explicit treatment of the temporal and spatial dimensions. Our future work will be devoted to a further development of the data model in order to explicitly consider temporal and spatial dimensions, and a study of the expressivity in relation with decidability and complexity of the *refinement* reasoning task.

REFERENCES

[Agrawal *et al.*, 1995] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. Technical report, IBM Almaden Research Center, San Jose, California, 1995. Appeared in Proc. ICDE '97.

[Artale *et al.*, 1996a] A. Artale, E. Franconi, and N. Guarino. Open problems for part-whole relations. In *Proceedings of the 1996 International Workshop on Description Logics (DL'96)*, pages 70–73, Boston MA, November 1996. AAAI Press.

[Artale *et al.*, 1996b] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, 20:347–383, 1996.

[Baader and Sattler, 1998] Franz Baader and Ulrike Sattler. Description logics with concrete domains and aggregation. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 336–340, 1998.

[Baader *et al.*, 1999] Franz Baader, Enrico Franconi, and Ulrike Sattler. *Multidimensional Data Models and Aggregation*, chapter 4. Springer-Verlag, 1999. Edited by M. Jarke, M. Lenzerini, Y. Vassilious and P. Vassiliadis.

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. In *Proc. of the 13 th IJCAI*, pages 704–709, Chambery, France, August 1993.

[Buchheit *et al.*, 1994] Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, and Martin Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.

[Cabibbo and Torlone, 1997] Luca Cabibbo and Riccardo Torlone. Querying multidimensional databases. In *proc. Sixth Int. Workshop on Database Programming Languages (DBPL-97)*, pages 253–269, 1997.

[Calvanese *et al.*, 1994] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. A unified framework for class-based representation formalisms. In *Proc. of KR-94*, Bonn D, May 1994.

[Calvanese *et al.*, 1995] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and reasoning. In *In Proc. of the 4th International Conference on Deductive and Object-Oriented Databases, DOOD'95*, volume LNCS-1013, pages 229–246, Singapore, 1995. Springer-Verlag.

[Catarci *et al.*, 1995] Tiziana Catarci, Giovanna D'Angolini, and Maurizio Lenzerini. Conceptual language for statistical data modeling. *Data & Knowledge Engineering (DKE)*, 17:93–125, 1995.

[Chen, 1976] P.P. Chen. The entity-relationship model: toward a unified view of data. *ACM Transactions On Database Systems*, 1(1):9–36, 1976.

[De Giacomo and Lenzerini, 1995] Giuseppe De Giacomo and Maurizio Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14 th IJCAI*, Montreal, Canada, 1995.

[De Giacomo and Lenzerini, 1996] G. De Giacomo and M. Lenzerini. Tbox and abox reasoning in expressive description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann, 1996.

[De Giacomo and Lenzerini, 1999] G. De Giacomo and M. Lenzerini. Description logic framework for information integration. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13. Morgan Kaufmann, 1999.

[De Giacomo and Naggar, 1996] G. De Giacomo and P. Naggar. Conceptual data model with structured objects for statistical databases. In *Proceedings of the Eighth International Conference on Statistical Database Management Systems (SSDBM'96)*, pages 168–175. IEEE Computer Society Press, 1996.

[Donini *et al.*, 1991a] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2 nd International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162, Cambridge, MA, 1991.

[Donini *et al.*, 1991b] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of the 12 th IJCAI*, pages 458–465, Sidney, Australia, August 1991.

[Donini *et al.*, 1992] F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[Donini *et al.*, 1994] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: from subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.

[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. Int'l Conf. on Data Engineering*, 1996.

[Gupta *et al.*, 1995] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. In *Proceedings of the 21. International Conference on Very Large Data Bases (VLDB-95)*, 1995.

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

[Hacid and Sattler, 1997] M. S. Hacid and U. Sattler. A multidimensional data model with structured dimensions. In *Proc. IEEE Knowledge and Data Engineering Workshop 97*, Newport, CA, USA, 1997.

[Horrocks and Sattler, 1999] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 1999.

[Horrocks, 1997] Ian Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, November 1997.

[Levy and Mumick, 1996] A. Y. Levy and I. S. Mumick. Reasoning with aggregation constraints. In *Proceedings of the International Conference on Extending Database Technology (EDBT-96)*, Avignon, France, 1996.

[Levy *et al.*, 1996] AlonY. Levy, Anand Rajaraman, and JoannJ. Ordille. Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pages 40–47, 1996.

[Mumick and Shmueli, 1995] I. S. Mumick and O. Shmueli. How expressive is stratified aggregation. *Annals of Mathematics and Artificial Intelligence*, 15(3-4), 1995.

[Schild, 1991] Klaus D. Schild. A correspondence theory for terminological logics: preliminary report. In *Proc. of the 12th IJCAI*, pages 466–471, October 1991.

[Schmidt-Schauß and Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Srivastava *et al.*, 1996] D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. In *Proceedings of the 22. International Conference on Very Large Data Bases (VLDB-96)*, Bombay, India, 1996.

[Woods and Schmolze, 1992] William A. Woods and James G. Schmolze. The KL-ONE family. *Computer and Mathematics with Applications, special issue: Semantic Networks in Artificial Intelligence*, 23(2-5):133–177, March-May 1992.