AN AUTOMATIC METHOD FOR THE DESIGN OF MULTIDIMENSIONAL SCHEMAS FROM OBJECT ORIENTED DATABASES

YASSER HACHAICHI
Multimedia, Information Systems and Advanced Computing Laboratory, University of Sfax
Higher Institute of Business Administration
BP 1013, Sfax, 3018, Tunisia
Yasser.hachaichi@fsegs.rnu.tn

JAMEL FEKI
Multimedia, Information Systems and Advanced Computing Laboratory, University of Sfax
Faculty of Economics and Management
BP 1018, Sfax, 3018, Tunisia
Jamel.feki@fsegs.rnu.tn

A data warehouse (DW) is a large data repository system designed for decision-making purposes. Its design relies on a specific model called multidimensional. This multidimensional model supports analyses of huge volumes of data that trace the enterprise’s activities over time. Several design methods were proposed to build multidimensional schemas from either the relational data model or the entity-relationship data model. Almost all proposals that treated the object-oriented data model assume the presence of the data source UML class-diagram. However, in practice, either such a diagram does not exist or is obsolete due to multiple changes/evolutions of the information system. Furthermore, these few proposals require an intense manual intervention of the designer, which requires a high expertise both in the DW domain and in the object database domain. To overcome these disadvantages, this work proposes an automatic DW schema design method starting from an object database (schema and its instances). This method applies a set of extraction rules to identify multidimensional concepts and to generate star schemas. It is defined for the standard ODMG model and, thus, can be adapted with slight changes for other object database models. In addition, its extraction rules have the merit of being independent of the domain semantics. Furthermore, they automatically generate schemas classified according to their analytical potential; this classification helps the DW designer in selecting the most relevant schemas among the generated ones. Finally, being automatic, our method is supported by a tool-set that also prepares for the automatic generation of the Extract Transform and Load procedures used to load the DW.

Keywords: Multidimensional model; data warehouse; automatic design; object-oriented database.
1. Introduction

Organizations generate huge volumes of data pertinent to various aspects of their business processes, such as customers’ management, suppliers’ management and procurement. In addition, they need to access and analyze this data to support either their daily operations or business decisions. Thus, today’s organizations have two types of information systems: operational information systems and decision support systems (DSS). The operational information system supports the execution of the daily business operations. On the other hand, a DSS manages historical information used to analyze the business performance over time in order to take appropriate business decisions.

Data warehousing is a technology that intends to provide decision makers access to various levels of information. A data warehouse (DW) provides an architectural model for the flow of data from the operational information system to decision support environments. It is periodically populated with data from operational information systems, e.g., for equipment management, accounting, inventory and customer management. Essentially, a DW collects all relevant data into one central system, organizes data efficiently so it is consistent and convenient for many purposes such as retrieving/using or keeping old data for historical analyses. In addition, a DW is a central data repository used to build and load data marts (DM); each DM contains an extract of the DW and is oriented to a specific subject of decisional analyses.

On the other hand, given their functional differences, designing a DW/DM requires a methodology different from those commonly adopted for operational information systems. In fact, current commercial software tools only assist the administrators in the DW/DM structure specification and production of analytical results; hence, they suppose that the DW/DM schemas are designed beforehand. This shortage motivated the proposition of several design methods for DW/DM schema. These methods differ in three aspects: their approaches (top-down, bottom-up, mixed) the data model they assume (relational, object-oriented versus XML), and their degree of automation (automatic, semi-automatic versus manual). The current state-of-the-art shows that several DW/DM design methods have been proposed for relational databases with various degree of automation. However, XML and object-oriented databases still interest researchers. In this paper, we focus on object-oriented databases (ODB) as they have better treated complex objects increasingly in use in today’s computer applications. In addition, as set forth by Barry, “generally, an ODB is a good choice when you have all three factors: business need, high performance, and complex data. Recently, people have also been considering an ODB even when their data is not particularly complex. An ODB can allow for a smaller team and faster development because there is only one data model.” Examples of ODBMS (Object Data Base Management System) in use in industrial applications are given in ODBMS FAQ; for instance, British Airways uses the
Versant Object Database for its *Origin and Destination Revenue Management System.*

To our knowledge, there are very few DW/DM schema design methodologies that start from an ODB. Almost all methods rely on an object-oriented data source start from a UML class diagram. However, in practice, the organization either does not always have such a diagram or even when it has one it is often obsolete: not up to date to reflect the evolution/maintenance of the operational information system. In addition, the few proposed methods for the design of DW/DM based on objects require human intervention with high expertise in both domains OLAP (On-Line Analytical Processing) and ODB. Furthermore, in the design process, these works do not address some specificities of the object data sources, such as methods and structured attributes.

In this paper, we propose a DM schema design method from ODB (schema and its instances). This method is supported by a software engineering tool and is domain independent because it relies on the structural properties of the data source independently of its semantics. It automatically applies a set of rules to extract, from the ODB, multidimensional concepts (i.e., candidate facts, dimensions) and generates star schemas. Moreover, it keeps track of the origin (e.g., object name, attribute name, data type, length) of each multidimensional concept in the generated DM schemas; this traceability helps the automatic generation of ETL (Extract Transform and Load) procedures to load the DM. Finally, to ensure its adaptability to various ODBMS, we base our method on the standard object-model ODMG (Object Database Management Group); thus, any ODBMS compliant with this standard can benefit from our method for the design of DM schemas.

The remainder of this paper is organized as follows. Section 2 presents the ODMG object model as a standard, and the multidimensional model. Section 3 puts our method in its scope by overviewing DW/DM bottom-up design approaches based on object-oriented data models. Section 4 describes our design method of DM from an ODB. Section 5 overviews the CAME-BDO toolset which supports our DM schema design method. The evaluation of CAME-BDO is discussed in Sec. 6. Finally, the conclusion summarizes the presented work and outlines its perspectives.

2. Background and Terminology

To ensure a *generic* DM schema design method, we relied on the object data model standardized by the ODMG, a consortium of the leading ODB vendors. In this section, we present a conceptual overview for each of the ODMG object model and the multidimensional model.

2.1. The ODMG object data model

We overview the object model supported by the ODBMS compliant to the last release of the standard ODMG 3.0.
The object model specifies the kind of semantics that can be defined explicitly for an ODBMS. Among other things, the semantics of the object model determines the characteristics of objects, how objects can be related to one another, and how objects can be named and identified. The object model specifies the constructs that are supported by an ODBMS.\textsuperscript{58}

The basic modeling concepts are the object and the literal. Each object has a unique identifier whereas a literal has no identifier. In addition,

- Objects and literals can be categorized by their types. All elements of a given type have a common range of states (i.e., the same set of properties) and a common behavior (i.e., the same set of operations). An object is sometimes referred to as an instance of its type.
- The state of an object is defined by the values it carries for the set of properties. These properties can be attributes of the object itself or relationships with other objects.
- The behavior of an object is defined by the set of operations that can be executed on or by the object. Operations may have a list of input and output typed parameters and may return a typed result; and
- An ODBMS has an appropriate meta-model according to which instances of ODBs are stored and managed.

The ODMG Object meta-model (Fig. 1) specifies what is meant by objects, literals, types, operations, properties, attributes, relationships, and so forth. It includes significantly richer semantics than the relational model does, by declaring relationships and operations explicitly. In addition, the ODMG standard is based on a common object model and uses several aspects of OMG’s object model. It supports types (interface) and classes (implementation), encapsulation, inheritance and polymorphism.

![Fig. 1. ODMG standard Meta-model: main concepts.\textsuperscript{14}]
We next shortly explain the basic concepts of the ODMG Model; for more details
the reader is referred to the Object Data Standard: ODMG 3.0.14

- A **Relationship** is a property of an object. The ODMG Model supports only binary
  relationships, i.e., relationships between two types (i.e., object or literal). A
  relationship is defined explicitly by declaring their *traversal paths* which enable
  applications to use logical connections between objects. Traversal paths are
  declared in pairs, one for each traversal direction of the relationship.
- A **Key** uniquely identifies an instance of a type. Simple and compound keys are
  supported.
- **Object IDentiifiers (OID)** are unique within a storage domain. The value of an
  object OID never changes over its lifetime and is never re-assigned.

For the inheritance of state and behavior, the ODMG object model defines the
**extends** relationship. The *extends* relationship applies only to *object* types; thus, only
classes (but not literals) may inherit state. It is a single inheritance relationship
between two classes whereby the subordinate class inherits all of the properties and
all of the behavior of the class that it extends.

In terms of typing, as illustrated in Fig. 2, the ODMG object model defines:

- **Collection Types** are available to build complex objects. They include \( Set\{t\} \),
  \( Bag\{t\} \), \( List\{t\} \) and \( Array\{t\} \) collections.
- The **Atomic Literals** supported are: Long, Short, Unsigned Long, Unsigned Short,
  Float, Double, Boolean, Octet, Char, String and Enum.
- The **Structured Literals** supported are: Date, Time, Timestamp and Interval.

Figure 3 depicts an example of ODB modeling the ‘Estimate management for an
enterprise’. In this example, object classes are shown as rectangles; operations are
distinguished from attributes by the symbol \( \Box \); attributes and operations are linked

![Fig. 2. The ODMG full set of Built-in types.](image-url)
to their object by horizontal lines; attribute and operation types are specified between parentheses “(N)umeric, (S)tring, (B)oolian, (D)ate, (I)nterval”; key attributes are tagged with K; relationships are drawn by arrowed lines between objects as follows: \( \rightarrow \) : one-to-one, \( \rightarrow \rightarrow \) : one-to-many, \( \rightarrow \rightarrow \rightarrow \) : many-to-many and \( \rightarrow \) : extends relationship.

In this example, an Estimate is elaborated for a Client and contains Estimate details. Each Estimate detail concerns one Model, needs a Study to define associated Materials, Articles and Tasks. A Task can be Automatic, and therefore is executed by Machine, Manual i.e., carried out by Personnel, or Semi-automatic requiring both Automatic and Manual interventions. We will refer to this example to illustrate our DM design method.

2.2. The multidimensional model

A conceptual schema is used to define logical schemas supported by a class of software systems (e.g., relational DBMS). Then, a logical schema is translated into a physical one, supported by a specific software system (e.g., Oracle). In the data warehousing context, it was early realized that traditional conceptual models for database modeling, such as the E/R model, do not provide a suitable means to describe the fundamental aspects of such applications. The crucial point is that, in designing a DW, there is the need to represent explicitly certain important characteristics of the information contained therein. These characteristics are not related to the abstract representation of real-world concepts, but rather to the final goal of the data warehouse: supporting data analyses oriented to decision making.

In the last few years, multidimensional modeling has attracted the attention of several researchers who defined various solutions each focusing on the set of information they considered strictly relevant. Some of these solutions have not or
have limited graphical support, and aim at establishing a formal foundation for representing cubes, hierarchies and an algebra for querying them. On the other hand, we believe that a distinguishing feature of DW conceptual models is that of providing a graphical support to be easily understood by both designers and decision makers when discussing requirement and validating conceptual solutions. Hence, we opted to use the Dimensional Fact Model (DFM) which is a graphical conceptual model, specifically devised for multidimensional design. DFM was first proposed in 1998 by Golfarelli et al. and has been continuously enriched and refined in order to suit optimally the variety of modeling situations that may be encountered in real projects from small to large complexity.

A conceptual design according to the DFM consists of a set of fact schemas (thereafter multidimensional schemas) where the basic concepts are facts, measures, dimensions and hierarchies. Figure 4 is a meta-model grouping these concepts.

A formal definition of these concepts can be found in the DFM. Here, we informally present these concepts through the example shown in Fig. 5 which depicts an introductory multidimensional schema according to the DFM model. This schema allows decision makers to analyze the Sales fact.
A fact is a focus of interest for the decision-making process; typically, it models a set of events occurring in the organization and/or its environment. A fact is graphically represented by a box divided into two compartments, the upper compartment is for the fact name and the bottom one is for the fact measures. Examples of facts in the commercial domain are sales, shipments, purchases; others in the financial domain are stock exchange transactions, contracts for insurance policies, etc.

A measure is a numerical property of a fact; it describes a quantitative aspect of interest for decisional analyses. For instance, the fact Sales has two measures: quantity and unitPrice. Measures should be numerical because they are used for computing aggregated values using aggregate functions (e.g., Sum, Average, Count). Rarely, a fact may have no measures, this happens when the only interesting thing to be recorded is the occurrence of events; in this case the fact is said to be empty and is typically queried to count the events that occurred.

A dimension is directly linked to a fact considered as an association linking dimensions. Dimensions of a fact set the context of recording its measures and, therefore represent the fact analysis coordinates. Graphically, dimensions are represented as rectangles attached to the fact by straight lines. Typical dimensions for the Sales fact (Fig. 5) are Product, Customer, Store and Date. Usually one of the fact dimensions represents the time which is necessary to extract time series from the DW data. In addition, note that, in the multidimensional schema of Fig. 5, each measure depends on all dimensions; i.e., concerns one product, sold to one customer, delivered from one store at a given date.

Aggregation is the basic OLAP operation since it produces summarized information from large amounts of detailed data. An aggregation is carried out on measures thanks to dimensions. For instance, we can compute the total amount of sales (sum of quantity * unit price) by Product (or even by any combination of dimensions). In addition, this total can be obtained at different levels of details (by Product type or category) thanks to the definition of dimensional attributes organized into hierarchies.

A dimensional attribute (also called parameter) is a property of a dimension. It is graphically represented by a circle. Relationships between dimensional attributes are expressed by hierarchies. A hierarchy of a dimension $d$ is a directed graph, rooted from the identifier of $d$, whose all nodes are dimensional attributes of $d$ and, whose arcs model many-to-one associations between pairs of dimensional attributes. Hierarchies determine how primary events can be aggregated into secondary events and selected significantly for the decision-making process. In Fig. 5, the hierarchies of the Product dimension enable us to aggregate measures by type and markGroup or by type and category.

Note that each dimensional attribute may functionally determine some non-dimensional attributes; these latter are called descriptive (or weak) attributes. They are linked to their corresponding dimensional attributes by a line. For example, in Fig. 5, the dimensional attribute StoreID has one descriptive attribute called sname (for store name).
3. Related Works

In the literature, DW/DM development approaches are classified into three categories: (1) **Data-driven approaches**\(^1\) which rely on the analysis of the corporate data model of the OLTP (On-Line Transaction Processing) system and its relevant transactions; (2) **User-driven approaches**\(^11\) which start from a set of analytical requirements defined by the decision makers of the future (DSS); and (3) **Mixed approaches**\(^7\) which combine data-driven and user-driven approaches in order to profit of their offered advantages.

User-driven (aka. top-down) approaches presume that users have enough expertise in expressing their analytical requirements in order to design schemas loadable from the organization OLTP system. Mixed approaches are advisable when the data source model (i.e., logical schema) is well known and has substantial size and complexity.\(^28\) Finally, data-driven (aka. bottom-up) approaches benefit from two major advantages: first, they help decision makers since they offer potential multidimensional schemas built on the source data model and, second they guarantee that the organization’s OLTP system can feed the user-selected schemas with the needed data. In data-driven approaches, user-requirements elicitation is voluntary neglected. In fact, Inmon\(^35\) argued that user-requirements are the last thing to be considered in a DSS development since they are well understood after the DW is populated with data and query-results are analyzed by decision makers. Considering these advantages, we elected to propose a DM design method within the data-driven approaches category.

Current DW/DM design methods consider either E/R,\(^8\) relational,\(^18\) XML\(^32\) or object models.\(^22\) Since this paper treats ODB, in this section we restrict our review to works that consider object models.

From our point of view, the research combining the DW and object paradigm fields follows three main lines.

The first research line applies the object paradigm for the multidimensional modeling of the DW schema. Works of this line propose multidimensional conceptual object-oriented models based on UML (Unified Modeling Language)\(^1\) ; some of them focus their model on a specific area of the DW such as temporal aspects\(^47\) security or access control,\(^21\) requirements\(^11\) and association rules mining models.\(^63\)

The second research line addresses the problem of how to build a DW from an object model. More precisely, it aims at the design of multidimensional conceptual models from object data sources.\(^22\)

In the third research line, we gather works that implement a DW by means of an object or object-relational language/databases.\(^15\) They mainly address the efficient acquisition, storage, query, change control, and schema integration of data into an object/object-relational DW.

Since, our work aims to design multidimensional conceptual models from object data sources therefore it fits into the second research line. Due to space limitation,
in the remainder of this section, we limit ourselves to the context of the second research line where we review the main referred works.

In this context, Gandhi and Jain\textsuperscript{22} propose an approach called object-oriented methodology for DW design. It is a two-step approach:

(1) Transformation of an object model into a relational schema; the authors use three trivial transformation rules: (i) an object class maps to a table, (ii) an association maps to a table, (iii) a generalization maps to a super class plus a series of sub class tables. They also supply manually details that are missing from an object model, such as primary key. (2) Transformation of the obtained relational schema into a DW schema; to do so, the authors define five transformations to be applied manually; nevertheless, they do not precise in which case each transformation should be applied.

On their side, Prat et al.\textsuperscript{50} propose a conceptual design phase that starts with the definition of a UML class diagram representing the decision makers’ initial requirements. This definition uses no multidimensional concepts, thereby enabling maximal reuse of traditional methods commonly used for OLTP systems engineering. The designed UML model is then enriched/transformed in order to facilitate subsequent mapping into a logical multidimensional schema. To do so, the authors define four transformations applied on the UML conceptual model: (1) determination of identifying attributes for the classes; (2) manual determination of attributes representing measures to distinguish them from qualitative attributes (i.e., descriptive); (3) migration of attributes of 1-1 and $N$-1 associations into one of the participating classes; and finally, (4) transformation of the generalizations into aggregations to enable their automatic mapping into multidimensional hierarchies. After these four transformations, the logical multidimensional schema is generated, from the enriched UML conceptual model, through the semi-automatic transformations detailed in Prat et al.\textsuperscript{50} “These transformations are semi-automated, more specifically in the logical phases human interaction is required to validate the step-by-step application of the transformations or to provide information”.\textsuperscript{50} The result of this step is a UML class diagram extended with the multidimensional concepts. This diagram can be mapped semi-automatically into a physical multidimensional schema.

Also, Zribi et al.\textsuperscript{61} proposed a method for the construction of DM schemas, starting from a UML class diagram, in five semi-automatic steps: (1) identification of transaction entities on which they build facts; (2) construction of decisional UML-packages each containing a transaction entity and its associated classes (directly or indirectly linked to the transaction entity); (3) graphical annotation of packages to identify multidimensional concepts resulting from the previous stage — the annotation uses a set of stereotypes to represent the multidimensional concepts; (4) validation of the annotation by the decisional designer; and (5) automatic generation of a DM star schema modeled according to the DFM formalism,\textsuperscript{25} from each annotated package.

On the other hand, Zepeda et al.\textsuperscript{62} proposed a DW design method based on the UML class diagram and user requirements. This method is divided into two phases.
The first phase starts with facts identification; its goal is to identify the entities that are candidates to become facts. Once facts are identified, a recursive algorithm is applied to find their dimensions. This algorithm accepts the UML class diagram with the set of candidate facts, and then produces a snowflake schema for each candidate fact. The second phase gets a set of metrics computed on the bases of user requirements. These metrics help selecting snowflake schemas from the candidate ones previously generated.

In summary, we notice the six following shortages in the so far proposed DW/DM design methods:

1. Most DW/DM design methods relying on an object data-source start from a UML class diagram. However, in practice, such diagram either does not exist within an organization, or it may be obsolete so that it does not reflect the modifications due to the evolution of the OLTP system.

2. The method starting the design from an ODB goes through one unnecessarily intermediate modeling level and, manually performs the transformations.

3. Current methods consider that the built candidate DMs are equally important whereas some of them may be insignificant for the decisional process.

4. Except for the method of Prat et al., the proposed methods try to represent the main DM properties at a conceptual level by abstracting away details of an envisaged implementation platform. “Unfortunately, none of these approaches defines a set of formal transformations in order to: (i) univocally and automatically derive every possible logical representation from the conceptual model; or (ii) help the designer to obtain the most suitable logical representation of the developed conceptual model”.

5. The existing methods for the design of DW/DM based on object sources require a manual human intervention. However, they necessitate a high expertise both in OLAP domain and in the ODB domain.

6. In the DW/DM design process, current works neglect some specifics of the object data-sources, like complex attributes (i.e., collections, structures) and method definitions.

In this paper, we propose a DM design method that overcomes these limits. To reach this objective, first our method relies on the recent version of the object data source that we directly extract from the ODBMS repository; this latter contains the OBD meta-data (i.e., structures of objects, methods). Second, it automates the main design steps and assists the designer in the choice of relevant multidimensional concepts among those extracted; this assistance is ensured by assigning to each concept a relevance level reflecting its analytical potential for the decision making. Third, it keeps track of the origin of each component in the generated DM schema. This traceability is fundamental both to automatically derive logical representations and to prepare the generation of ETL (Extract Transform and Load) procedures.
4. Multidimensional Models from ODB

Our bottom-up DM design approach starts from an ODB compliant to the ODMG model. It is composed of three steps namely ODB schema retrieval, multidimensional concepts identification and DM construction, and DM schemas display and adjustment. Among these steps only the third is manually conducted by decision makers/designers where they adapt the automatically constructed DM schemas to their particular OLAP analytical requirements.

The ODB schema retrieval step extracts the ODB schema (e.g., object names, attribute names and types, operation names and types, key constraints) from the ODBMS repository, and constructs the set of objects strongly linked by functional association. The multidimensional concepts identification and DM construction step extracts the multidimensional concepts (facts and their measures, dimensions and their attributes organized into hierarchies) from the ODB schema. It produces potential DM schemas. To automate this step, we defined for each multidimensional concept, a set of extraction rules. Our rules are domain independent because they rely on the structural properties of the data source independently of their semantics. In addition, they have the merit of keeping track of the origin of each multidimensional concept in the generated DM schemas. This traceability is fundamental during the definition of ETL procedures. In the third step, DM schemas display and adjustment, decision makers/designers are presented with a set of potential DM schemas that they can adapt to meet their particular analytical needs. During this adaptation, they can add derived measures, remove and/or rename DM schema components (i.e., dimension, attribute, hierarchy) using a set of operations.32 The application of these adaptation operations is constrained to ensure that the resulting schemas are syntactically well formed.6,54 As examples of simple constraints, we cite the following: a fact must be linked to at least two dimensions; a dimension must have at least one hierarchy; all hierarchies of a dimension \(d\) must start from the identifier of \(d\).

In the remainder of this section, we focus on the second step and present our rules for the identification of multidimensional concepts from an ODB. For this, we adopt the following notation:

- Given a relationship \(R\) between two objects \(O_1\) and \(O_2\) with multiplicity \((m_1, M_1)\) on the side of \(O_1\) and \((m_2, M_2)\) on the side of \(O_2\), the \(\text{Max}(O_1, O_2)\) function returns the two maximum multiplicities \((M_1, M_2)\).
- The transitive closure of an object \(O\), noted \(O^+\), denotes the set containing \(O\) and all objects \(O'\) directly or transitively linked to \(O\) by relationships with \(\text{Max}(O, O') = (1, 1)\). Note that if an object \(O'\) belongs to \(O^+\) then, \(O'^+ = O^+\); this avoids the computation of \(O'^+\) and therefore optimizes the approach. In our running example (Fig. 3), Model\(^+\) = \{Model, Study\} and Model\(^+\) = Study\(^+\).

In our work, we consider that the set \(O^+\) can be semantically seen as a single complex real-world object; indeed, all objects of \(O^+\) are linked by strong functional dependencies.
Note that, in the remaining of this paper, ODB, fact, dimension and hierarchy denote their schemas; and objects indicate object classes. When we refer to instances of these concepts, we will explicit this.

Now we detail the identification step; it starts with facts identification and then continues with measures, dimensions and hierarchies identification.

4.1. Fact identification

A fact is composed of numerical factual data on which analyses are focused. In an ODB, we identify facts either from relationships (rule RF1) or from objects (rules RF2, RF2.1 and RF2.2).

In the data warehousing field, many-to-many relationships are commonly used to build facts.20,24,37,42,50,55,61

RF1. Each relationship between two objects $O_1$ and $O_2$ such that $\text{Max}(O_1, O_2) = (N, M)$ with $N > 1$ and $M > 1$ is identified as a fact.
We conventionally name such a fact $F-O_{1\text{gen}}-R-O_{2\text{gen}}$ where $R$ is the name of the relationship transversal path (Fig. 1) and $O_{1\text{gen}}$ (respectively $O_{2\text{gen}}$) is the concatenation of the names of $O_1$ (respectively $O_2$) and all its generalizing objects if any.

Note also that because a relationship between two objects in an ODB does not contain attributes, then rule RF1 generates empty facts, i.e., facts without measures (cf. Sec. 2.2).

The application of rule RF1 on the example of Fig. 3 finds out that Max(Automatic, Machine) satisfies its condition and produces the fact $F$-AutomaticTask-executed by-Machine, called according to the above naming convention. Similarly, the fact $F$-ManualTask-carried out by-Personnel is produced.

As previously mentioned, we also extract facts from objects. To do so, we first compute the transitive closure of each object. Indeed, the set of objects forming a transitive closure may be semantically seen as a single object of the real world, since a strong functional dependency exists between them.

RF2. Each $O^+$ containing an object with a nonkey numeric attribute or with an operation returning numeric value(s) is identified as a fact.
The name of such a fact is the concatenation of “F-” with the names of the objects belonging to $O^+$.

We note that:

- The term numeric attribute covers ODB attributes that are atomic, collection or belonging to a structured attribute (Fig. 2).
- Given a fact $F$ built on $O^+$, each object in $O^+$ is useful to extract measures and dimensions for $F$. Thus, considering all the objects in $O^+$ provides for building facts covering a large variety of analyses.
- If an object does not have a key attribute, then the designer can intervene to select one. Such an optional intervention improves the result of our identification method.
Applying rule RF2 on our running example, first we identify from $Model^+ = \{Model, Study\}$ the fact $F-\text{Study-Model}$ since the object $Model$ contains the nonkey numeric attribute $Price$. In addition, we identify seven other facts: $F-\text{estimate_detail}$, $F-\text{estimate}$, $F-\text{Automatic}$, $F-\text{Manual}$, $F-\text{semiautomatic}$, $F-\text{Machine}$ and $F-\text{Personnel}$.

Note that rule RF2 can produce facts built either on specialized or generalized objects. We next examine whether it is better to group each such fact with its generalizing/specialized objects into a single fact. For this, we need to examine the instances of these objects according to the next two sub-rules.

**RF2.1.** If $S$ is a specialized object of $n$ generalized objects $G_i (n \geq 1)$ and $S$ produced a fact $F-S$ (via rule RF2), then replace the fact $F-S$ with a new fact built on $S^+ \cup \bigcup_{i=1}^{n} G_i^+$. The name of the new fact is $F-S$ concatenated with all its generalized objects.

The main intuition behind this rule is that a specialized object cannot exist without its generalizing objects: there is a strong functional dependency from the specialized object to its generalizing object. Hence, by grouping the specialized object with its generalizing objects, we construct a richer fact offering more analysis potentials. As illustrated in Fig. 6, every instance of the specialized object (SO) has to be necessarily linked to an instance of its generalizing object (GO).

For our example, rule RF2.1 replaces $F-\text{Automatic}$ and $F-\text{Manual}$ by $F-\text{Automatic-Task}$ and $F-\text{Manual-Task}$, respectively, by including the generalizing object Task. Similarly, $F-\text{Semiautomatic}$ is replaced by $F-\text{Semiautomatic-Automatic-Manual-Task}$ which includes three generalizing objects Automatic, Manual and Task.

**RF2.2.** If a generalizing object $G$ produced a fact $F-G$ (on $G^+$ through RF2), and if $G$ has not all its instances linked to any instance of its specialized objects, then maintain $F-G$ as a fact independent of its specialized objects.

Rule RF2.2 guarantees that instances of $G$ not associated with instances of its subclasses will be analyzed alone through the fact $F-G$. Figure 6 case 2 depicts this situation where GO-instance$_2$ has no specialized instance (SO).

For our running example, rule RF2.2 maintains the two facts $F-\text{Manual-Task}$ and $F-\text{Automatic-Task}$ without the specialized object semiautomatic. Indeed, in our database, there are instances of Manual Tasks and Automatic Tasks not linked to instances of Semiautomatic.
The application of RF2.1 and RF2.2 on objects S(pecialized) of G(eneralized) can identify simultaneously two facts F-G and F-G-S with their transitive closures. Conceptually, the fact F-G is included in the fact F-G-S. However, at the logical level, F-G is loaded with instances from G not attached to any instance of S, and F-G-S with those of G linked to S. Hence, in our example, first the fact F-Automatic-Task (resp. F-Manual-Task) will be loaded with objects Automatic and Task (resp. Manual and Task) which are not attached to any instance of Semiautomatic. Second, the F-Semiautomatic-Automatic-Manual-Task will be loaded with those linked to Semiautomatic.

To focus on the most important extracted facts, we consider that facts issued from relationships (using rule RF1) are more relevant for decision makers than those built on objects or transitive closure of objects (using rule RF2). In fact, a relationship generally contains data that describe a business activity; these data result from transactions of the OLTP system and evolve more quickly than objects.

4.2. Measure identification

Measures serve to compute summarized results by means of aggregate functions. This imposes that measures have numeric values. In our method, we extract measures from properties (i.e., attributes and operations) of any object O belonging to the transitive closure O+ which produced a fact.

Remember that in general \( O^+ = \{ O, O_1, \ldots, O_n \} \); \( O^+ \) could be restricted to O if there is no \( O_i \in O^+ \) with \( \text{Max}(O, O_i) = (1, 1) \). The following measure identification rules take into account this situation.

RM1. Given a transitive closure \( O^+ \) on which a fact F is built. Every operation (i.e., method) in an object \( O_i \) that belongs to \( O^+ \) and returns a numeric value is a candidate measure for F.

In the object paradigm, an operation is a service that can return a value directly, or indirectly through a parameter. If the returned value is numeric then rule RM1 considers it as a candidate measure.

RM2. Nonkey numerical attributes in an object that belongs to a transitive closure \( O^+ \) identified as a fact F are candidate measures for F.

Note that rule RM2 excludes key-attributes because they are generally artificial and redundant data; in addition, they do not record/trace a business activity.

In addition, our method has also the advantage to determine measures from structures; this enriches the set of identified measures as follows:

RM3. Numerical atomic attributes in a simple Structure of an object that belongs to a transitive closure \( O^+ \) identified as a fact F are candidate measures for F.

This rule considers only simple structures and ignores collections of structures. In fact, a simple structure S in an object O can be seen as a relationship between O and S, where \( \text{Max}(O, S) = (1, 1) \). Furthermore, as shown in the ODB meta-model
(Fig. 2), a numeric attribute can be atomic or a collection (e.g., Set, Bag, List, Array). A collection is multi-valued. However, fact measures are simple values. Therefore, a numeric function to compute a single value, from the numeric collection of attributes identified as measures, should be added by the designer. Such a function is obviously semantic and domain dependent.

Rules RM1 and RM2 identify measures, issued from attributes and operations, that are directly belonging to object(s) identified as fact whereas, RM3 extracts measures from Structure within object(s) identified as a fact. Since, the relevance of measure decrease when we move away from the initial fact object then, we consider that measures obtained with RM1 and RM2 are more relevant than those obtained with RM3. In addition, we notice that measures defined on operations (RM1) are more relevant than those defined on attributes (RM2) because these latter are likely to be keys or descriptive attributes. Hence, we opt for three relevance levels for measures in the descending order: RM1 then RM2 then RM3.

For our running example, the application of these three rules produces, for each identified fact, the measures shown in Table 1.

### 4.3. Dimension identification

Recall that a dimension represents a business analysis axis. Also, dimensional attributes are organized into one or more hierarchies of levels, which correspond to different ways to aggregate fact measures. To complete our DM design method, we determine for each identified fact, a set of candidate dimensions either from objects or from attributes.

For dimensions built on objects, we define the following two rules:

**RD1.** Given an empty fact \( F \) built, with rule RF1, on a relationship between \( O_1 \) and \( O_2 \). The transitive closure of each of \( O_1 \) and \( O_2 \) is a dimension for \( F \).

Conventionally, the name of this dimension is the concatenation of the object names in \( O_1^+ \).

This rule is graphically explained in Fig. 7.

**RD2.** Let \( O_1 \) be an object directly linked to an object \( O_2 \) with \( \text{Max}(O_1, O_2) = (1, *) \), and \( O_2 \) belongs to a transitive closure identified as a fact \( F \). Then, \( O_1^+ \) builds a dimension for \( F \).

Conventionally, this dimension name is the concatenation of the object names in \( O_1^+ \).

Rule RD2 identifies as a dimension \( d \), every object linked maximally with \( (1, *) \) because every fact instance is one-to-one linked to an instance of its dimension. In addition, considering \( O_1^+ \) as a dimension (as opposed to \( O_1 \) only) enables us to extract all dimensional attributes for \( d \).

In order to complete the set of dimensional attributes obtained using rules RD1 and RD2, we add to each specialized object \( SO \) belonging to the transitive closure \( O_1^+ \) on which a dimension \( d \) is built, all generalizing objects \( GO \) (of \( SO \)) together with their transitive closures (\( GO^+ \)). (The name of \( d \) will include names of \( GO^+ \)). For our running example, the impact of this on the results obtained so far is that we replace...
Table 1. Multidimensional concepts identified from the ODB of Fig. 3.

<table>
<thead>
<tr>
<th>Fact</th>
<th>Measure</th>
<th>Dimension</th>
<th>Identifier</th>
<th>Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-AutomaticTask-executed-by-Machine</td>
<td>Ø</td>
<td>Automatic-Task</td>
<td>O_id</td>
<td>MO_id → M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD1)</td>
<td>(RDII)</td>
<td>MO_id → P_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S_id → M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S_id → P_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine</td>
<td>MC_id</td>
<td>A_Duration → Second → Minute → Hour → Day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD1)</td>
<td>(RDII)</td>
<td>Purchase_date → Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD1)</td>
<td>(RDII)</td>
<td>MO_id → P_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S_id → M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S_id → P_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Personnel</td>
<td>P_id</td>
<td>M_Duration → Second → Minute → Hour → Day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD1)</td>
<td>(RDII)</td>
<td>Affectation_date → Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SeqIdqualification → Date → Day → Month → Year</td>
</tr>
<tr>
<td>F-Model-Study</td>
<td>Price</td>
<td>Article</td>
<td>P_id</td>
<td>Ø</td>
</tr>
<tr>
<td></td>
<td>(RM2)</td>
<td>(RD2)</td>
<td>(RDII)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Material</td>
<td>M_id</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD2)</td>
<td>(RDII)</td>
<td></td>
</tr>
<tr>
<td>F-Estimate-detail</td>
<td>Price</td>
<td>Estimate</td>
<td>E_id</td>
<td>C_id → Address</td>
</tr>
<tr>
<td></td>
<td>(RM2)</td>
<td>(RD2)</td>
<td>(RDII)</td>
<td>E_date → Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quantities</td>
<td>D_id</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD2)</td>
<td>(RDII)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Model-Study</td>
<td>S_id</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD2)</td>
<td>(RDII)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total_detail()</td>
<td>T_id</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RM1)</td>
<td>(RDII)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total_detail_TI()</td>
<td>M_id</td>
<td></td>
</tr>
<tr>
<td>Fact</td>
<td>Measure</td>
<td>Dimension</td>
<td>Identifier</td>
<td>Hierarchy</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>F-Estimate (RF2)</td>
<td>Total_Price()</td>
<td>Client</td>
<td>C_id</td>
<td>Address</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD2)</td>
<td>(RD1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total_Price_TI()</td>
<td>E_date</td>
<td>E_date</td>
<td>Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD5)</td>
<td>(RDB)</td>
<td></td>
</tr>
<tr>
<td>F-Automatic-Task (RF2, RF21 &amp; RF2.2)</td>
<td>AO_cost()</td>
<td>Model-Study</td>
<td>S_id</td>
<td>P_id</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD2)</td>
<td>(RD2)</td>
<td>M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A_Duration</td>
<td>A_Duration</td>
<td>Second → Minute →</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>Hour → Day</td>
</tr>
<tr>
<td>F-Manual-Task (RF2, RF21 &amp; RF2.2)</td>
<td>MO_cost()</td>
<td>Model-Study</td>
<td>S_id</td>
<td>P_id</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD2)</td>
<td>(RD2)</td>
<td>M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M_Duration</td>
<td>M_Duration</td>
<td>Second → Minute →</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>Hour → Day</td>
</tr>
<tr>
<td>F-semiautomatic-Automatic-Manual-Task (RF2 &amp; RF2.1)</td>
<td>SA_cost()</td>
<td>Model-Study</td>
<td>S_id</td>
<td>P_id</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD2)</td>
<td>(RD2)</td>
<td>M_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M_Duration</td>
<td>M_Duration</td>
<td>Second → Minute →</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>Hour → Day</td>
</tr>
<tr>
<td></td>
<td>AO_cost()</td>
<td>A_Duration</td>
<td>A_Duration</td>
<td>Second → Minute →</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>Hour → Day</td>
</tr>
<tr>
<td></td>
<td>MO_cost()</td>
<td>SA_Duration</td>
<td>SA_Duration</td>
<td>Second → Minute →</td>
</tr>
<tr>
<td></td>
<td>(RM1)</td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>Hour → Day</td>
</tr>
<tr>
<td>F-Machine (RF2)</td>
<td>Hour_cost</td>
<td>State</td>
<td>State</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(RM2)</td>
<td>(RD4)</td>
<td>(RDB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Purchase_date</td>
<td>purchase_date</td>
<td>purchase_date</td>
<td>Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td>(RD5)</td>
<td>(RDB)</td>
<td>(RDB)</td>
<td></td>
</tr>
<tr>
<td>F-Personnel (RF2)</td>
<td>Wage</td>
<td>Affectation_date</td>
<td>Affectation_date</td>
<td>Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td>(RM2)</td>
<td>(RD5)</td>
<td>(RDB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qualification</td>
<td>SeqId_qualification</td>
<td>SeqId_qualification</td>
<td>Day → Day → Month → Year</td>
</tr>
<tr>
<td></td>
<td>(RD3)</td>
<td>(RD4)</td>
<td>(RDB)</td>
<td></td>
</tr>
</tbody>
</table>
the dimensions Automatic and Manual by Automatic-task and Manual-task (Table 1, column 3) since Automatic and Manual have the generalizing object Task.

Besides rule RD2, we can identify a dimension from an attribute of a particular data type as well as from some collections of attributes. Such a dimension is known as a degenerate dimension. Rules RD3, RD4 and RD5 treat degenerate dimensions.

RD3. Every attribute of a structure collection in an object that belongs to a transitive closure identified as a fact F is a degenerate dimension for F.

A structure attribute can be assimilated to an object composed of a set of attributes. Thus, a collection of structure S (i.e., Set <S>, List <S>) in an object O simulates a link between O and S. In addition, in ODB, two objects cannot share the same instance of an attribute. This guarantees that the link between S and O has Max(S, O) = (1, *) and, therefore, we identify the collection of S as a dimension for the fact built on O.

Degenerate dimensions come also from attributes of special data types, as stated in the next two rules.

RD4. An atomic Boolean attribute in an object that belongs to a transitive closure identified as a fact F is a dimension for F.

Naturally, a Boolean attribute splits its object instances into two subsets; thus, it is a candidate analysis axis.

Finally, as commonly claimed the DW is a chronological collection of data. Consequently, the time dimension appears in all DW. We take into account this claim to define the temporal dimension rule.

RD5. A temporal attribute (Date, Time, Timestamp, or Interval) in an object that belongs to a transitive closure identified as fact F is a dimension for F.

Table 1 (column 3) shows the complete set of dimensions identified with these rules.

A degenerate dimension is a dimension reduced to one attribute stored as part of the fact, and is not in a separate dimension.
Since we envisage assisting the decisional designer, we affect relevance levels to dimensions. We consider that dimensions identified by RD1, RD2, RD4 or RD5 are the most relevant as they are extracted from objects linked to facts, or from atomic attributes of specific data type (Temporal/Boolean). Whereas dimensions identified by RD3 are less relevant because they are extracted from complex attributes sometimes difficult to aggregate.10

The star schema is the keystone construction in multidimensional modeling.55 In this paper, we have shown how to build star schemas. In addition, we can automatically build constellation schemas in two different manners: by merging star schemas, those have common dimensions, or even by looking to correlated facts within the OLTP data model. This matter is studied in Ref. 19.

Once dimensions are identified, we continue the construction of multidimensional schemas by identifying dimensional attributes and organize them into hierarchies.

4.4. Hierarchies identification

Recall that a dimension hierarchy is made up of discrete dimensional attributes organized from the finest to the highest granularity. The dimension identifier attribute is the finest aggregation granularity; the remaining attributes define progressively coarser granularities. In addition to these organized attributes, a hierarchy may include weak attributes that are descriptive information for dimensional attributes. Since all hierarchies of a dimension \( d \) start from the identifier of \( d \), we first extract the dimension identifier.

4.4.1. Dimension identifier identification

Based on the data structure from which a dimension \( d \) is built (i.e., object, attribute), we define four rules to extract the identifier. The first rule extracts an identifier built on single attribute key of an object.

RDI1. The key attribute in an object \( O \) identified as a dimension \( d \) is the identifier of \( d \).

As an example, the identifier of the dimension Machine is \( MC_id \).

RDI2. The identifier of a dimension built on the transitive closure \( O^+ \) of an object \( O \) is any key attribute belonging to one of the objects in \( O^+ \); we consider the key attributes of the remaining objects in \( O^+ \) as weak attributes for the selected identifier.

For example, the Model-Study dimension can be identified by one of the two keys attributes \( S_id \) (identifier of Study) or \( Mo_id \) (identifier of Model). In Table 1 (column 4), we have arbitrarily chosen the identifier \( S_id \) and, therefore, consider \( Mo_id \) as a weak attribute for \( S_id \) (cf. RND1).

Note that if none of the objects being considered by rules RDI1 and RDI2 has a key attribute, then the identifier of the dimension may be the OID of any of these objects.

As we can build dimensions on attributes, the next rule defines identifiers for such dimensions.
**RDI3.** The identifier of a dimension built on an atomic attribute $A$ is $A$.

For instance, the identifier of the dimension $A_{Duration}$ is the attribute $A_{Duration}$. Finally, rule **RDI4**, sets how to define an identifier for a dimension built on a structured attribute.

**RDI4.** The identifier of a dimension built on a structured attribute $A$ is a surrogate (i.e., sequential artificial) identifier named $SeqId_A$.

Table 1 (column 4) gives the identifier of every identified dimension.

We continue the construction of hierarchies by identifying the remaining dimensional attributes; i.e., those located after the identifier.

### 4.4.2. Dimensional attributes identification

First, we extract the dimensional attributes located immediately after the dimension identifier (those of the second level of the hierarchy); second, for each one, we extract its successors. For this, we defined four rules: The first one for extraction from objects, and the three other rules for extraction from attributes.

**RDA1.** Let $O_1$ be an object directly related to $O_2$ with $\text{Max}(O_1, O_2) = (1, *)$ and $O_1$ belongs to a transitive closure that produced a dimension $d$. Then, the key attribute of every object in $O_1^+$ is a dimensional attribute of level two for $d$.

Similarly to the dimension identifiers, if an object does not have a key attribute, then the OID (assigned by the ODBMS) is assumed as a default dimensional attribute of level two.

**RDA2.** Every attribute with a structure collection type in an object belonging to a transitive closure identified as a dimension $d$ is a dimensional attribute of level two for $d$.

As we have proceeded for dimension identification, attributes of a specific data type can be identified as dimensional attributes.

**RDA3.** Every atomic Boolean attribute in an object belonging to a transitive closure that produced a dimension $d$ is a dimensional attribute of level two for $d$.

**RDA4.** A temporal structured attribute (Date, Time, Timestamp, or Interval) in an object belonging to a transitive closure that produced a dimension $d$ is a dimensional attribute of level two for $d$.

Note that rules from **RDA1** to **RDA4** extract dimensional attributes of level two. To obtain attributes at higher levels, we apply recursively either the above four rules on objects from which a dimensional attribute is extracted by **RDA1**, or the rules **RDA2** to **RDA4** on attributes identified by **RDA2**. In the second assertion we do not apply **RDA1** because an attribute cannot be linked to an object by a relationship and **RDA1** is based on relationships between objects.

Note that Temporal attributes (Date, Interval, Time, and Timestamp by interfaces) resulting from rule **RDA4** are defined in the ODMG object model as
structured objects. We have studied the factory of ODMG operations for creating such objects and we have defined for each one its corresponding hierarchy. Table 2 gives the most used constructors for temporal attributes and their corresponding hierarchies. In Table 2, Attribute1 → Attribute2 means that Attribute1 has lower granularity than Attribute2.

Similar to the classification of dimensions, we consider that dimensional attributes produced by RDA1, RDA3 and RDA4 are more relevant for decision making than those obtained with RDA2. The last ones are issued from complex attributes. For our running example, Table 1 (column 5) lists the hierarchies for each dimension.

We continue the construction of hierarchies by identifying for each dimensional attribute its weak attributes (i.e., nondimensional attributes), if any.

4.4.3. Weak attributes identification

To identify weak attributes, we defined the following four rules.

**RND1.** Key attributes of objects belonging to O⁺ that produced a dimension d, other than those extracted through RDI2 as dimension identifier for d, are weak attributes for d identifier.

We have already introduced this rule in RDI2. The idea is to associate for the identifier of dimension d a descriptive attribute from each object that participates to d construction: the Mo_id attribute becomes a descriptive attribute for the Model-Study dimension (Fig. 8).
**RND2.** Every nonkey (textual or numerical) atomic attribute belonging to an object of a transitive closure providing a dimensional attribute \( p \) (through rules RDI1, RDI2 or RDA1) is a weak attribute for \( p \).

**RND3.** If an object provides a dimensional attribute \( p \), through rules RDI1 or RDI2 or RDA1, and contains a simple structure attribute \( S \) (not a collection) then textual and numerical atomic attributes of \( S \) are weak attributes for \( p \).

**RND4.** Every nonkey textual or numerical atomic attribute belonging to a collection of structured attribute providing a dimensional attribute \( p \) (through rules RDA4 or RDA2) is a weak attribute for \( p \).

In rules **RND2** to **RND4**, we have considered both textual and numerical attributes. Because weak attributes are descriptive, then we consider that textual attributes are more significant than numerical ones. Practically, numerical attributes may be insignificant as descriptive.

Figure 8 shows graphically according to the DFM model the two facts **F-Automatic Task_executedby_Machine** and **F-Estimate-detail** among those extracted and listed in Table 2. In this figure, most relevant multidimensional concepts are in bold font.

In Appendix A, we give an algorithm for the identification of multidimensional concepts.

To examine experimentally that our proposed rules extract all pertinent multidimensional concepts starting from an ODB and produce significant/useful
multidimensional schemas, we have developed the CAME-BDO toolset that support our proposed method.

5. CAME-BDO: A Toolset for DM Construction

CAME-BDO extends the CAME software tool that carries out the design of conceptual DM schemas starting from either the relational database, or from a set of XML documents compliant to a given DTD. CAME-BDO is an assistance toolset for the decisional designer: its main functions cover our DM design method steps starting from a MATISSE ODB. More accurately, it enables the construction of multidimensional schemas. We have implemented our tool using SQL to access the repository and compute some results as the transitive closure of an object, and java as a programming language.

MATISSE is an ODBMS compliant to the ODMG standard. Its strength lies mainly in its meta-schema (Fig. 9). In MATISSE, everything is an object: a class is an instance of the meta-class Mt-Class, relationships are instances of the meta-class Mt-Relationship, attributes are instances of the meta-class Mt-Attribute, and methods (i.e., operations) are instances of the meta-class Mt-Method. For more details about MATISSE the reader can refer to the Matisse release notes.

The remainder of this section is devoted to the description of the CAME-BDO features and to its GUI, through the source object database Media planning (Fig. 10). For this ODB source, Prat et al have semi-automatically identified multidimensional concepts. In Sec. 6, we will compare our results to those obtained by Prat et al in order to evaluate CAME-BDO and highlight the benefits of our method.

![Fig. 9. The ODBMS MATISSE Meta-schema.](image-url)
CAME-BDO features cover the three steps of our design method (Fig. 11):

- **Object database schema retrieval.** This step, first displays the list of databases implemented under the MATISSE ODBMS. After the selection of one of these ODB sources, the DM schemas construction process starts. CAME-BDO accesses the MATISSE meta-model (i.e., repository), extracts the objects of the selected data source and displays them in a tabular format as depicted in the interface of Fig. 12. In this GUI, the designer can see attributes, methods (operations), relationships, and the type (generalized, specialized or normal) of every pointed object. Here, the designer can keep all/some objects for the DM construction process (i.e., Select/deselect objects) and, finally launches the identification of multidimensional concepts in order to obtain all DM schemas.
Design Method Steps

CAME-BDO Functions

1. Object Database schema retrieval
2. Multidimensional concepts identification and star schema construction
3. DM Schemas display and adjustment

CAME-BDO Repositories

- ODBMS MATISSE Repository
- Extracted schemas repository
- Multidimensional schemas repository

Fig. 11. CAME-BDO functional architecture.

Fig. 12. The schema of the ODB of Fig. 10 as extracted from MATISSE repository.
Multidimensional concepts identification and star schema construction.

This identification applies automatically our identification rules on the elected objects of the chosen ODB. It extracts facts and, for each one it identifies its measures, dimensions with their dimensional attributes organized into hierarchies and weak attributes, if any. These identified concepts are stored in the multidimensional schemas repository together with their corresponding elements (i.e., names of objects, attributes, methods) in the ODB source. This stored traceability has a twofold benefit; first, it is fundamental to derive automatically logical representations and, second, it helps to generate ETL procedures.

Applied on the example of Fig. 10, our identification rules construct 10 multidimensional schemas (Fig. 13). For example, the obtainment of the schema of the fact called F-Main_shareholder follows steps below:

— RF1 identifies F-Main_shareholder as a fact; indeed, Main_shareholder+ = \{Main_shareholder\} and Main_shareholder contains the nonkey numeric attribute Percentage_of_shareholder.

— RM2 identifies the nonkey numeric attribute Percentage_of_shareholder as a measure for F-Main_shareholder.

— RD2 identifies each of the objects Shareholder, Media and Date as a dimension for F-Main_shareholder: each one is linked to Main_shareholder with maximum
multiplicities (1, *). Applying **RDI1**, the identifier of each of these dimensions is the key attribute.

— **Date** and **Media** hierarchies are constructed using **RDA1** (applied twice for **Date**).

— The hierarchy of the **Shareholder** dimension is constructed by **RDA4**; the ODMG **Calendar_date** constructor (Table 2) gives its last three parameters.

**DM Schemas display and adjustment.** To display the DM star schemas built in the previous step, CAME-BDO offers two formats: tabular and graphical. The tabular format (Fig. 13) displays the DM schemas extracted with CAME-BDO for the ODB of Fig. 10. For the selected fact **F-Main_shareholder**, this interface displays its measures and dimensions classified by relevance level (dark blue for the highest relevance). Once a dimension is selected, the interface visualizes its attributes and hierarchies, ordered by relevance level (i.e., high or low).

The interface of Fig. 14 shows graphically a DM schema constructed with CAME-BDO, it is displayed according to the DFM graphical notation. This interface is obtained with **MPI-Editor**\(^5\) toolset which communicates with CAME-BDO through XML technology. The decisional designer can validate the schema by adding derived measures, removing and/or renaming dimensional elements. These adjustments can be performed either through the CAME-BDO tabular format, or through the **MPI-Editor** GUI.

6. Discussion and Evaluation

Our design method for multidimensional schemas from an object-oriented database relies on transformation rules inspired from those we have defined for the
construction of DM schemas from relational databases. Moreover, these rules deeply profit from the ODB specificities: they consider object operations (rules $RF_2$, $RM_1$), exploit inheritance between objects ($RF_2.1$, $RF_2.2$) and complex attributes ($RM_3$, $RD_3$, $RD_4$, $RDA_2$, $RND_3$, $RND_4$). In addition to these advantages, our method differs from those of the literature because it considers the recent version of the data source (i.e., the version in use, actually extracted from the ODBMD repository). Furthermore, it assigns a relevance level to the extracted multidimensional elements (e.g., measures, dimensions, hierarchies) and traces back the DM schema elements to the data source schema elements.

To evaluate our method, we have experimented our CAME-BDO software prototype on several object databases (some of them are taken from the literature and for which multidimensional schemas were built manually/semi-automatically). A comparative analysis between results obtained with CAME-BDO and schemas constructed by authors of these cases are summarized in Table 3.

Through these evaluations, we concluded the following points:

- CAME-BDO (and thus our method) identifies all the facts that a bottom-up analysis can manually figure out; moreover, it finds out empty facts.
- In most cases, CAME-BDO extracts more measures than those obtained manually; this is because we consider both operations and multi-valued numerical attributes. However, we noticed that CAME-BDO does not identify calculated measures since they are semantics dependent. In practice, these measures can be added manually by the designer in the third step of our method.
- The number of dimensions and dimensional attributes extracted by CAME-BDO is slightly higher than those obtained manually. Thus, CAME-BDO builds schemas that offer a larger panoply of analyses. This variation in numbers is due to the fact that our rules take into account Boolean, temporal and structured attributes, and are not limited to work on objects.

Table 3. CAME-BDO evaluation.

<table>
<thead>
<tr>
<th>Data sources$^a$</th>
<th>Fact</th>
<th>Measure</th>
<th>Dimension</th>
<th>Dimensional attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counseling system for technician admission$^{22}$</td>
<td>2/1</td>
<td>3/3</td>
<td>5/3</td>
<td>2/1</td>
</tr>
<tr>
<td>Media-planning$^{20}$</td>
<td>10/9</td>
<td>9/8</td>
<td>10/8</td>
<td>10/8</td>
</tr>
<tr>
<td>Faculty members load$^{61}$</td>
<td>9/8</td>
<td>11/9</td>
<td>8/8</td>
<td>5/4</td>
</tr>
<tr>
<td>Farmer company$^{62,b}$</td>
<td>6/6</td>
<td>3/3</td>
<td>5/4</td>
<td>9/6</td>
</tr>
</tbody>
</table>

$^a$We have created a MATISSE ODB for each data source.

$^b$We have restricted the comparison of measures, dimensions and dimensional attributes only to facts (two) for which a complete schema is constructed by authors.
7. Conclusion

In this paper, we have tackled DW design issues at the conceptual level. More specifically, we have proposed an automatic DW design method that starts from an ODB compliant to the ODMG standard and generates DM schemas modeled according to the well-known DFM model. In order to generate DM schemas based on the most recent version of the ODB source, our method extracts the ODB logical model directly from the ODBMS repository. Thus, the generated DMs can reflect all of the organization activities.

Furthermore, to automatically generate DM schemas, we have defined a set of rules for the extraction of the DM schema components (i.e., facts, measures, dimensions with their attributes organized into hierarchies). Our rules have the merit to be independent of the semantics of the ODB source domain. To be independent, they take advantage of the structural-semantics offered by the ODMG object model. In an additional attempt to assist the DW/DM designer, our rules assign to each component of the generated DM schemas a relevance level reflecting its analytical potential. This assists the designer to choose those DM concepts that are more interesting for the decision-making process. Moreover, and as a third advantage, our method keeps track of the origin of each component in the generated DM schema. This traceability has a twofold benefit; first, it is fundamental to automatically derive logical representations and, second, it helps to generate ETL procedures.

For this method, we have developed a software prototype, called CAME-BDO, we used to conduct some experimental evaluations. In fact, CAME-BDO extends our CAME\textsuperscript{31} software tool which carries out the design of conceptual DM schemas starting either from a relational database source\textsuperscript{30} or from a set of XML documents compliant to a given DTD\textsuperscript{32}. These preliminary evaluations showed the feasibility of our method in identifying all facts and their measures, dimensions and hierarchies. To generalize these results, we are looking for a more consistent evaluation on a set of ODB.

In addition, as an immediate extension of this work, we are currently developing a software tool for the automatic generation of ETL procedures under OWB (Oracle Warehouse Builder); some preliminary results in this perspective are recently published\textsuperscript{33}. Furthermore, we are examining how to integrate adjusted/validated DM schemas obtained with CAME to build a DW schema loadable from heterogeneous data sources (i.e., relational database, XML data-centric documents and object databases), and how to generate platform-independent ETL procedures according to the MDE (Model Driven Engineering) approach.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.
Appendix A. Algorithm for the Design of Multidimensional Schemas from Object-Oriented Databases

DECLARE

Input

Sch: ODB schema and instances

Notation

$S_{Sch}$: the set of specialized objects in Sch

$G_{Sch}$: the set of generalized objects in Sch

Predefined Functions

$Spec(O_i)$: returns the set of specialized objects of an object $O_i \in S_{Sch}$

$Gen(O_i)$: returns the set of generalized objects of an object $O_i \in S_{Sch}$

$OP(O_i)$: returns the set of operations of an object $O_i$

$Get_Type(OP_i)$: returns the type of data returning by the operation $OP_i$

$ATT(O_i)$: returns the set of attributes of an object $O_i$

$Is\_Key(att_i)$: returns true if $att_i$ is a key attribute

$Data\_Type(att_i)$: returns the data types of the attribute $att_i$

$Instances\_of(O_i)$: returns all the instances of the object $O_i$

$Link\_to\_specialized(ig_i)$: returns true if the object instance $ig_i$ is linked to an instance of its generalized object

$Is\_Simple\_Struct(att_i)$: returns true if the attribute $att_i$ is a simple structure

$Is\_collection\_Struct(att_i)$: returns true if $att_i$ is a collection of structure

$Transitive\_Closure(O_i)$: computes and returns objects of the transitive closure of the object $O_i$

Begin

For each object $O_i \in Sch$

\[ O_i^* \leftarrow Transitive\_Closure(O_i) \]

End For

/* Generating Facts on Relationship */

For each relationship $R$ between $O_i \in Sch$ and $O_j \in Sch$

\[ Max(O_i, O_j) \leftarrow (N,M) \]

If $(N>1)$ and $(M>1)$ Then

\[ Add\_Empty\_Fact(R, O_i, O_j) \] /* Using RF1 */

End If

End For
/* Generating Facts on Transitive Closures */
For each $O_i^+ \in \text{Sch}$
For each $O_j^+ \in O_i^+$
    If ($\exists \ Op_i \in \text{OP}(O_j)$ and $\text{Get_Type}(Op_i)$ is numeric) or ($\exists \ att_i \in \text{ATT}(O_j)$ and $\text{Not Is_Key}(att_i)$ and $\text{Data_Type}(att_i)$ is numeric) Then
        $\text{Add_Fact}(O_i^+)$ /* Using RF2 */
    End If
End For
End For
For each $O_i^+$ building a Fact $F-O_i^+$
For each $O_j^+ \in O_i^+$
    If ($O_j^+ \in S_{\text{sch}}$) then
        For each $G_i \in \text{Gen}(O_j)$
            $(F-O_i^+) \leftarrow (F-O_i^+) \cup G_i^+ /* RF2.1 */$
        /*$(F-O_i^+)$ denotes the set of objects on which the fact $F-O_i^+$ is built */
    End For
End If
End For
End For
For each $O_i^+$ building a Fact $F-O_i^+$ /* $O_i^+$ including those enriched through the loop above */
For each $O_j^+ \in O_i^+$
    If ($O_j^+ \in G_{\text{sch}}$) then
        $\{\text{IG}\} \leftarrow \text{Instances_of}(O_j)$
        $\text{Has_specialized} \leftarrow \text{True}$
        For each $ig_i \in \text{IG}$
            If ($\text{Not Link_to_specialized}(ig_i)$) Then
                $\text{Has_specialized} \leftarrow \text{False};$ break
            End If
        End For
    If ($\text{Has_specialized}$) Then
        For each $S_i \in \text{Spec}(O_j)$
            $(F-O_i^+) \leftarrow (F-O_i^+) \cup S_i^+$
        Else /*RF2.2*/
            $\text{NULL}$ /*Maintain fact without specialized objects*/
    End If
End If
End For
End For
/* Measures identification */
   For each O_i building a Fact F-O_i
      For each O_j ∈ O_i
         For each Op_i ∈ OP(O_i)
            If (Get_Type(Op_i) is numeric) Then
               Add_Measure (Op_i,F-O_i) /*RM1*/
            End If
         End For
         For each att_i∈ ATT(O_i)
            If (Not Is_Key(att_i) and Data_Type(att_i) is numeric) Then
               Add_Measure(att_i,F-O_i) /*RM2*/
            End If
            If (Is_Simple_Struct(att_i)) Then
               For each att_i.attj
                  If (Data_Type(att_i.attj) is numeric) /*RM3*/
                     Add_Measure(att_i.attj,F-O_i)
                  End If
               End For
            End If
         End For
      End For
   End For

/* Dimensions identification */
   For each Fact F identified on a relationship between O_i and O_j
      Add_Dimension (O_i', F) /*RD1*/
      Add_Dimension (O_j', F) /*RD1*/
   End For
   For each O_i' building a Fact F-O_i'
      For each O_j ∈ O_i'
         For each O_k ∈ Sch and linked to O_j
            If(Max(O_j, O_k)=(1,*)) Then
               Add_Dimension(O_k', F-O_i') /*RD2*/
            End If
         End For
         For each att_i ∈ ATT(O_i)
            If (Is_collection_Struct(att_i)) Then
               Add_Dimension(att_i, F-O_i') /*RD3*/
            End If
         End For
   End For
If (Is_atomic(att_1)) Then
    If (Data_Type(att_1) is Boolean) Then
        Add_Dimension(att_1, F-0_1) /*RD4*/
    End If
    If (Data_Type(att_1) is Temporal) Then
        Add_Dimension(att_1, F-0_1) /*RD5*/
    End If
End If
End For
End For

/* Global steps to construct hierarchies */
For each identified dimension d
    Determine_dimension_identifier(d) /* Rules RDI1 to RDI4 */
    Determine_dimensional_attributes(d) /* Rules RDA1 to RDA4 */
End For
For each dimensional_attributes da
    Determine_weak_attributes(da) /* Rules RND2 to RND4 */
End For

END

References
8. M. Böhnlein and A. Ulbrich-vom Ende, Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems, DOLAP, Missouri (1999), pp. 15–21.


27. M. Golfarelli, From user requirements to conceptual design in data warehouse design a survey, in *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction*, Part of the Advances in Data Warehousing and Mining (ADWM) Book Series (IGI Global Publication, 2009).


