Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2012, Article ID 247592, 13 pages
doi:10.1155/2012/247592

Research Article

Metadata for Approximate Query Answering Systems

Francesco Di Tria, Ezio Lefons, and Filippo Tangorra

Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro, Via Orabona 4, 70125 Bari, Italy

Correspondence should be addressed to Francesco Di Tria, francescoditria@di.uniba.it

Received 30 January 2012; Revised 5 June 2012; Accepted 23 June 2012

Academic Editor: Gerardo Canfora

In business intelligence systems, data warehouse metadata management and representation are getting more and more attention by vendors and designers. The standard language for the data warehouse metadata representation is the Common Warehouse Metamodel. However, business intelligence systems include also approximate query answering systems, since these software tools provide fast responses for decision making on the basis of approximate query processing. Currently, the standard meta-model does not allow to represent the metadata needed by approximate query answering systems. In this paper, we propose an extension of the standard metamodel, in order to define the metadata to be used in online approximate analytical processing. These metadata have been successfully adopted in ADAP, a web-based approximate query answering system that creates and uses statistical data profiles.

1. Introduction

The concept of metadata—born in the transactional systems literature—has gained the greatest attention in data warehousing environments, where two classes of metadata are distinguishable: back-room and front-room metadata. The first class aims to describe, among other things, the several heterogeneous data sources, the data transformation that must be performed to feed the data warehouse, and the refresh status of the stored data. The front-room metadata aim to represent the conceptual data model of the data warehouse and they are commonly used by the so-called business intelligence platforms in order to automatically generate the analytical queries.

According to [1], the business intelligence can be represented as an information supply chain, to highlight that the information comes from a set of data sources and goes through transformation processes, in order to become useful for decision making. However, this chain is composed of different steps, each of them being both producer and consumer of data and metadata. It is unlikely that a single vendor provides a complete system able to covers all the steps of the chain. For this reason, each step is usually managed by specific software tools, produced by different vendors. Of course, their integration is quite difficult, since each tool aims to improve its own effectiveness and efficiency. It follows that the metadata are commonly stored according to a proprietary format.

In order to overcome this limit, a central management of metadata is always a benefit for enterprises that are divided into departmental areas not directly linked among them. In this case, the most effective choice consists of using a central metadata repository [2] that must be *generic*, as it must allow to store metadata according to topics areas but not to specific tools; *integrated*, for it has to store metadata describing heterogeneous business assets (such as processes, documents, products, software tools, databases, and so on); *updated*, in order to be consistent with the business reality; *historical*, for all the insert-update-delete operations on the repository must be traced for data recovery and rollback.

Moreover, the central repository must rely on standard representations of metadata, since standard metadata allow different business intelligence ence tools to effectively communicate with each other (*i.e.*, tools integration) and provide a uniform management of data of various type and format (*i.e.*, data integration).

The current standard for data warehouse metadata definition is the Common Warehouse Metamodel (CWM), that provides a common model of metadata for both tools and data integration [3].

The OMG has also defined the XML Metadata Interchange (XMI) [4] as the XML-based physical layer to best allow the interoperability and the portability of metadata among the single components involved in business intelligence.

However, data warehouses are widely used also as data sources for specific analytical tools based on approximate query processing [5]. Such tools, or the so-called approximate query answering systems [6], are decision support systems that help decision makers by providing them with fast answers.

These software tools usually integrate traditional OLAP systems in order to return both approximate and/or exact answers, according to the user settings. Whereas the exact answers can be obtained by accessing real data, the approximate answers are obtainable only if preelaborated data are available. Therefore, the role and importance of metadata for approximate query answering systems relate to the need of tracing whether and which data have been transformed and prepared for the approximate query processing. In other words, based on these metadata, users are allowed (a) to know which data are available for approximate query processing and (b) to define queries that return (approximate) answers very quickly.

Currently, the CWM does not include a metamodel able to represent the metadata used by these systems and the aim of this paper is to extend the existing standard metamodel in order to define a novel metamodel that covers the issues related to approximate query processing in the scope of business intelligence.

This paper has the following structure. Section 2 presents the architecture of approximate query answering systems. Section 3 provides an overview of the CWM. Section 4 contains our proposal for the representation of metadata for approximate query answering systems, while Section 5 introduces our system and reports a case study. At last, Section 6 concludes the paper reporting final remarks.

2. Approximate Query Answering System

At the present time, approximate query answering systems are used in decision support systems, since they allow business users to obtain fast responses to queries whenever they do not need total precision or exact values [7]. Accuracy estimation is also provided along with the approximate answer.

Figure 1 depicts the high-level architecture of the approximate query answering system. Datasets stored in data sources (viz., the data warehouse DW) are preelaborated by a data reduction process in order to compute synopses of the data (DS). Data reduction is a process that reduces the cardinality of the DW and stores a small set of data solely. Data synopses are then used in the approximate query processing, whose aim is to perform traditional OLAP based on approximate answers. Approximate processing is able to provide fast answers to complex (and usually aggregate) queries that would normally require high computational time to produce the exact answers.

There are several consolidated methodologies that can be implemented in approximate query answering systems.

If the system adopts *summary tables*, the DS consists of materialized tables representing precomputed aggregate queries [8]. Of course, the creation of tables corresponding to all the possible user queries is an impracticable solution due to the explorative nature of analytical elaborations.

When using *wavelet-based* methodologies, the system computes a set of values, called *wavelet coefficients*, to be used in SQL instructions based on redefined SQL operators [9]. In monodimensional Haar wavelet decomposition, the DS is given by storing the average between a pair of values along with their difference. This computation is recursively repeated and, at each recursion, data with a lower resolution are obtained. So, given the vector $V_0 = \langle 2, 2, 4, 6\rangle$, $V_1 = \langle 2, 5\rangle$ is the vector with resolution 1, while $V_2 = \langle 3.5\rangle$ is the vector with resolution 0. The differences are used to reconstruct original data and they are, respectively, $\langle 0, -1\rangle$ for V_1 and $\langle -1.5\rangle$ for V_2.

In *histogram-based* methodologies, the DS is represented by a set of histograms. The queries are translated into equivalent queries on these histograms, allowing the same expressivity of SQL operators [10]. Histograms store the frequency of data falling into a set of intervals, the so-called buckets, used to split the domain. The higher the number of buckets, the higher the accuracy of the approximate answers.

Sampling consists in collecting random samples of large volumes of data and in executing queries on these samples, in order to derive information on the original set of DW data. In this case, the size of the sample must be decided *a priori* and it depends on the DW cardinality [11]. So, in many cases, also querying the sample may require a high answering time. As an example, if the table cardinality is 100,000,000 of rows and the sample size is only 1% of the original data, then the sample cardinality is 1,000,000 of rows.

In *orthonormal series*, the probability density function of multidimensional relations is approximated by orthonormal polynomials. As a result, a set of coefficients are computed and stored in the DS database [12]. The number of the coefficients depends on the polynomial approximation degree and not on the DW relation cardinality. Therefore, a constant response time is observed in these systems. As a counterpart, the approximation degree affects the accuracy of the approximate answers.

Since the advantages of approximate query processing have been widely accepted, further methodologies have been defined in last years. Emerging methodologies are based on graphs [13], genetic algorithms [14], and probabilistic models [15].

Once the data reduction has been performed, it is important to trace which fields and relations have been actually reduced, as some data stored in the database can be ignored for decision making based on approximate answers. Therefore, tracing which data have been reduced provides decision makers with the knowledge about which data are effectively available for approximate query processing. This

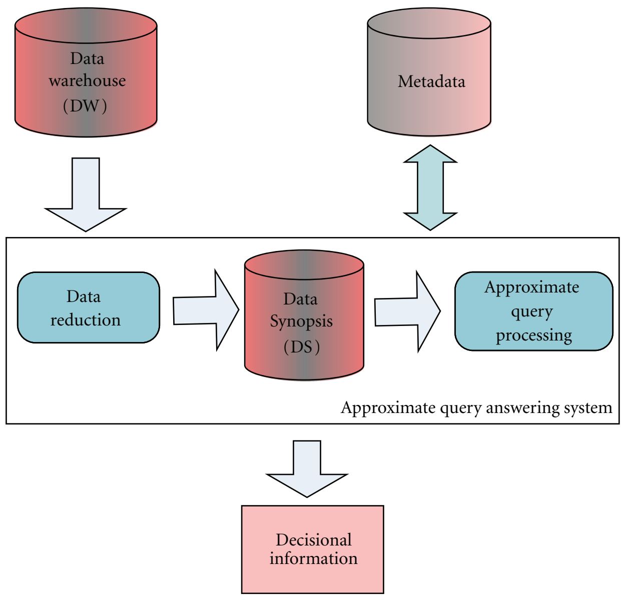


Figure 1: Approximate query answering system architecture.

knowledge is usually managed by metadata, that is, data providing a semantic level.

Currently, the metadata are stored in a central repository in standard format, in order to be shared and used by the several business intelligence tools produced by various vendors. As an example, these metadata are used by the business intelligence platforms to represent the multidimensional model of the data warehouse and to generate SQL instructions automatically without writing lines of code [2]. Moreover, the standard metadata are usually exported in XMI [4], as this XML-based layer provides interoperability among business intelligence tools.

However, the CWM does not include a metamodel that covers approximate query processing. So, there is no standard representation of metadata useful for approximate query answering systems. As already stated, these systems need to compute synopses of the data stored in a data warehouse. Then, the synopsis is used for the computation of the analytical queries. Therefore, this kind of systems need to know which data have been effectively reduced and, consequently, usable in the approximate query processing. Of course, this information must be provided by metadata. For this reason, a central research topic related to approximate query processing is strictly related to the representation of metadata.

2.1. Towards Metadata Identification. Here we focus on approximate query answering systems which adopt methodologies performing a data reduction and whose general architecture accords with the one depicted in Figure 1. For these methodologies, we present the metadata to be generated and used in approximate query processing in order to satisfy both user needs and system requirements.

In general, we distinguish between user-oriented metadata and system-oriented metadata. User-oriented metadata are those corresponding to the user choices, for example, as previously seen, which tables and columns have been selected for the data reduction. However, some of these strongly depend on the adopted methodology for the data reduction and, therefore, they are *specific* to the context or system implementation.

System-oriented metadata are those generated by the system on the basis of the user choices. These metadata provide further and immediate information about the compressed data, such as the refresh date. Nonetheless, some of these metadata—the column domain, for example—may be used by the algorithms performing the approximate query processing.

Table 1 reports the metadata identification for each methodology.

2.2. Metadata Requirements. From Table 1, we observe that the metadata are mainly used to know which elements of a relational database have been selected by the user for the approximate query processing. These elements are effectively reduced and stored in another database (viz. the DS database). Therefore, the Metadata database (cf. Figure 1) stores the following items:

(i) the name of the data warehouse along with the connection information,

(ii) the names of the reduced tables of a given data warehouse,

(iii) the names of the reduced columns, for each selected table.

In reference to the underlying system methodology, users perform usually further choices, which can affect the accuracy of approximate answers. Using orthonormal series, the approximation degree must be chosen, for example.

Table 1: Metadata identification.

Methodology	User-oriented metadata		System-oriented metadata
	Common	Specific	
Summary table			Refresh date
Wavelet		Resolution level	Table cardinality Column domain Refresh date
Histogram	Data warehouse name Reduced table name Reduced column name	Bucket width	Table cardinality Column domain Number of buckets Refresh date
Sampling		Sample size	Table cardinality Refresh date
Orthonormal series		Approximation degree	Table cardinality Column domain Refresh date

Also these values must be preserved as metadata. However, since these parameters are specific, a general solution is provided by using an opportune descriptor—consisting of the pair (*name*, *value*)—to represent data whose semantics depends on the context. This solution can be also adopted for system-oriented metadata.

Then, using metadata, users are allowed to formulate queries that will be automatically traduced into SQL statements, in the case they are exact queries, or into *ad hoc* plans for accessing data in the DS database, whenever they are approximate queries. In fact, the latter must be suitably transformed in order to define a data access plan whose features strongly depend on the adopted methodology.

As an example, in orthonormal series, a *sum* aggregate query is traduced into a call to the function *sum* (*d*, *tableStruct*, *columnsStruct*, *intervalsStruct*) where:

(i) *d* is the approximation degree set by the user;

(ii) *tableStruct* is the data structure storing several information—the cardinality, for example—about the reduced table involved in the query;

(iii) *columnsStruct* is the data structure storing information about the reduced columns involved in the query;

(iv) *intervalsStruct* is the data structure storing information about the query intervals set by the user.

These parameters are used to load the coefficients up to degree *d* relative to the reduction of the required columns of the table.

3. Overview of the Standard Metamodel

The current standard for the definition of the metadata to be used in business intelligence and data warehousing is the CWM [3].

The CWM is composed of a set of modular metamodels, structured in layers, where each metamodel depends only on the metamodels of the underlying layers.

Since the CWM is based on the object model, a metamodel is logically represented by a set of classes (the so-called package) that are related to each other via associations. The classes and associations are specified according to the Meta Object Facility (MOF) [16], which is an extension of the object model based on the Unified Modelling Language (UML). In this context, the MOF acts as a meta-metamodel able to represent CWM metamodels. Therefore, every instance of a class of the CWM metamodel is a metaobject, representing an element of the target model.

As an example, the relational metamodel (see Section 3.1) allows to describe the metadata of relational databases and each class instance represents an element of the relational database being modelled (*i.e.,* a table, column, constraint, or data type, and so on).

The layers and related metamodels are as follows.

Object Model. The layer that groups all the metamodels that provide the basic constructs for creating and describing the other metamodels.

(i) *Core* is the metamodel containing basic classes and associations used by all other packages.

(ii) *Behavioural* is the metamodel collecting together classes and associations that describe the behaviour of objects, such as operations and methods.

(iii) *Relationships* is the metamodel collecting together classes and associations that describe the relationships between objects, each one being an association (*i.e.,* is-part-of relationship) or a generalization (*i.e.,* is-type-of relationship).

(iv) *Instance* is the metamodel that allows the inclusion of data instances in the metadata.

Foundation. The layer that groups all the metamodels that are devoted to represent the concepts and structures shared by overlaying metamodels.

(i) *Business Information* is the metamodel supporting business-oriented services, such as name and description of the elements of the target model.

(ii) *Data Types* is the metamodel supporting the definition of constructs useful to create specific data types in the target model.

(iii) *Expressions* is the metamodel that provides basic support for defining expression trees, in order to allow objects to record shared expressions in common form.

(iv) *Key Index* is the metamodel used for specifying instances and for identifying alternate keys of instance sortings, such that they can be shared among the various data models that employ them.

(v) *Type Mapping* is the metamodel that supports the mapping of data types between different systems.

(vi) *Software Development* is the metamodel containing classes devoted to record how the software is used in the data warehouse.

Resource. The layer that groups all the metamodels that allow to represent different resource types.

(i) *Object* metamodel contains classes and associations that allow to represent metadata of objects (*i.e.,* the object model itself).

(ii) *Relational* contains classes and associations that allow to represent metadata of relational databases.

(iii) *Record* contains classes and associations that allow to represent metadata of record data resources.

(iv) *Multidimensional* contains classes and associations that allow to represent metadata of multidimensional databases.

(v) *XML* contains classes and associations that allow to represent metadata of XML documents.

Analysis. The layer that groups all the metamodels that must be implemented by business intelligence tools.

(i) *Transformation* contains classes and associations that represent metadata used by data transformation tools.

(ii) *OLAP* contains classes and associations that represent metadata used by OLAP tools.

(iii) *Data Mining* contains classes and associations that represent metadata used by data mining tools.

(iv) *Information Visualization* contains classes and associations that represent metadata used by tools devoted to support the graphical visualization of information.

(v) *Business Nomenclature* contains classes and associations that represent metadata about business taxonomy and glossary.

Management. The layer that groups all the metamodels that represent high-level tasks.

(i) *Warehouse process* is the metamodel that documents the process flows used to execute transformations.

(ii) *Warehouse operation* is the metamodel that contains classes recording the day-to-day operations of the warehouse processes.

In the next sections, we briefly discuss the Relational, Transformation, and OLAP metamodels, in order to show the context of data warehousing, and we consider only the relational metamodel in order to define an extension of the CWM. This extension enables representing the standard metadata for approximate query answering systems that perform reductions of the data stored in data warehouses [17]. These systems are usually built on the relational model (*i.e.,* they are ROLAP systems).

3.1. Relational Metamodel. Figure 2 depicts a simplified but essential version of the relational metamodel, that states that each schema of a relational database is composed of a set of tables, whereas each table is composed of a set of columns. In fact, the main classes of this metamodel (viz. Schema, Table, and Columns) inherit from the basic classes of the Core metamodel (viz. Package, Class, Attribute, Namespace, Classifier, and Feature).

On the basis of the associations defined in the Core metamodel, it is possible to state that a table owns a set of columns, in the same way as a class owns a set of attributes, whereas both Table and Class classes are specialization of the Classifier class.

Finally, a table is owned by a relational schema. A further class, named Catalog, represents a physical database that includes one or more relational schemas, each of them representing an independent logical database.

This metamodel allows representing also the usual constraints that must be defined in relational databases, such as primary key and foreign key constraints. According to this metamodel, it is possible to create a set of metaobjects (instances of classes of this metamodel), in order to represent the several elements of a relational database.

3.2. Transformation Metamodel. This metamodel includes classes and associations useful to represent metadata related to the data transformation occurring in a business intelligence system.

The transformations are thought as a set of rules to change data format. Each rule is composed of (a) a transformation function, (b) a data source, and (c) a data target.

3.3. OLAP Metamodel. The data warehouse conceptual schema is designed according to the multidimensional model [18, 19], that can be graphically represented using the metaphor of the cube. According to this metaphor, a fact related to an occurring event of interest can be represented as a cube. Moreover, a fact can be described by numeric attributes that provide quantitative information. These

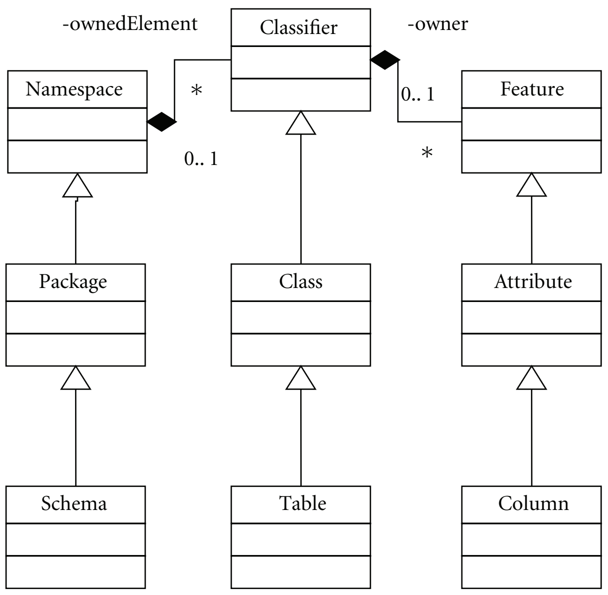


Figure 2: Part of the relational metamodel.

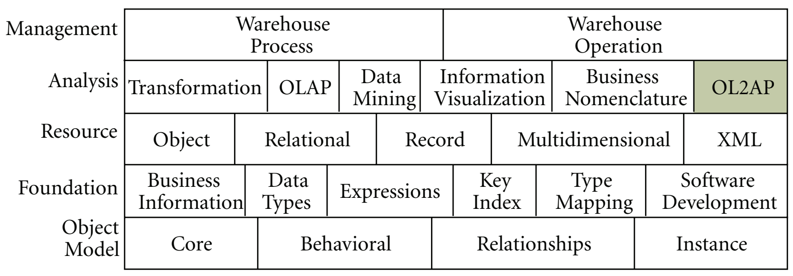


Figure 3: Extended CWM block diagram.

numeric attributes are the so-called measures. Therefore, each cell of the cube stores a single numeric value, pointed out by a set of dimensions representing levels of analysis. The first-level dimensions define the minimum granularity of the data stored in the cube. The set of the first-level dimensions forms the primary aggregation pattern. Further levels define how the data can be aggregated. The levels, which are in one-to-many relationship among themselves, form a hierarchy that represents an aggregation path.

The logical models to describe a data warehouse conceptual schema are MOLAP and ROLAP. In MOLAP, data warehouses are built on multidimensional databases, whose metadata are defined by the Multidimensional metamodel. On the other hand, in ROLAP, data warehouses are built on relational databases, whose metadata are defined by the relational metamodel.

The OLAP metamodel allows to define metadata used by OLAP tools, in order to represent the metadata of a data warehouse, built on both the ROLAP and the MOLAP technologies.

For the sake of simplicity, Relational and Multidimensional metamodels represent the data warehouse metadata at the logical level, while the OLAP metamodel describes the data warehouse metadata at the conceptual level.

4. Extension of the Standard Metamodel

As widely explained in [20, 21], the importance of metadata derives from the fact that they represent the only way to provide further knowledge about a business process or a component of an information system. In our context, the metadata are used to describe data reduction processes for supporting approximate query answering systems in multidimensional analyses. Since the CWM does not include a metamodel to define metadata for approximate query answering systems, we extended the CWM with a further metamodel, namely *OL2AP*, which stands for *OnLine Approximate Analytical Processing*.

We included the *OL2AP* metamodel in the *Analysis* layer of the CWM (cf. Figure 3) that groups all the metamodels that must be implemented by business intelligence tools performing approximate query processing.

The aim of this metamodel consists of tracing:

(i) which fact tables have been chosen for data reduction among all the tables of a data warehouse;

(ii) which dimension tables have been involved in the data reduction;

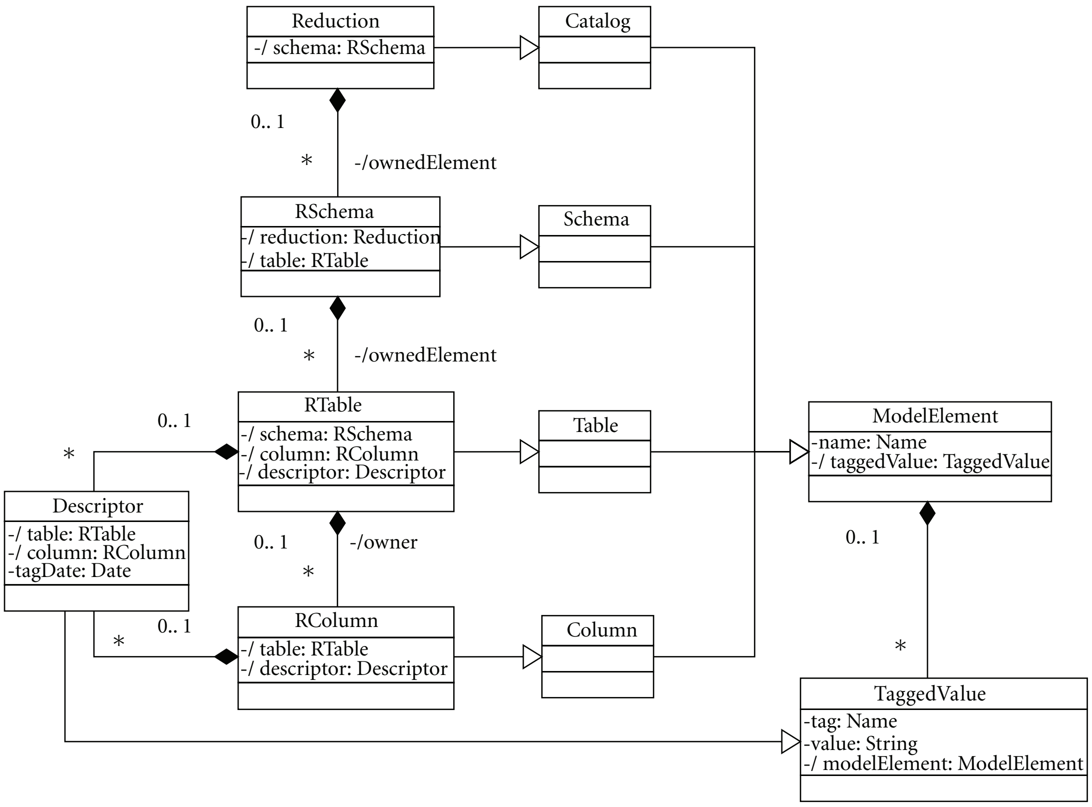


Figure 4: *OL2AP* Metamodel.

(iii) which attributes of the selected tables have been reduced;

(iv) the possible parameters of the data reduction.

The dimensions and measures of cubes that have been reduced are effectively available for the approximate query processing. Therefore, the decision maker is able to express analytical queries involving the reduced attributes in order to obtain fast responses.

4.1. OL2AP Metamodel. Figure 4 shows the main classes and associations that allow creating standard metadata that can be effectively used by approximate query answering system in order to trace the data reduction process and to generate analytical queries based on approximate answers. Notice that this metamodel depends on (a) the *relational* metamodel, that provides an approximate query answering system with the metadata describing the data warehouse logical model (usually, ROLAP), and (b) the *Core* metamodel, that allows to attach a descriptor to any model element (*i.e.,* the element of the database being modelled).

The main classes of the *OL2AP* metamodel are summarized in Table 2. Each instance of class is called metaobject and it represents a construct of the system to be modelled.

As an example, to create the metadata of a relational database, we have to use the *relational* metamodel, which establishes to create a metaobject of the *Table* class for each table of the database.

The steps for the creation of the metaobjects of the *OL2AP* metamodel are as follows:

(i) a metaobject of the class *Reduction* is created to represent the physical database, and the name of this object is the data source name (DSN) used for the physical connection to the database;

(ii) for each relational schema chosen for the data reduction, a metaobject of the class *RSchema* is created, having the name of the schema (*i.e.,* the name of the relational database) and a reference to the physical database it belongs to;

(iii) for each table of a relational schema chosen for the data reduction, a metaobject of the class *RTable* is created, having the name of the table and a reference to the schema it belongs to;

(iv) for each column of a table chosen for the data reduction, a metaobject of the class *RColumn* is created, having the name of the table and a reference to the table it belongs to;

(v) if a table or a column must be tagged, a metaobject of the class *Descriptor* is created, having an arbitrary name and a reference to the model element it belongs to.

Table 2: *OL2AP* metamodel classes.

Metamodel	Class	CWM Description
Core	ModelElement	A model element is an element that is an abstraction drawn from the system being modelled.
	TaggedValue	A tagged value allows information to be attached to any model element in the form of a “tagged value” pair; that is, *name* = *value*.
Relational	Catalog	A catalog is the unit of logon and identification. It also identifies the scope of SQL statements; the tables contained in a catalog can be used in a single SQL statement.
	Schema	A schema is a named collection of tables and collects all the tables of the same relational schema (i.e., a logical database).
	Table	Table is a data structure representing a relation.
	Column	Column is a data structure representing the field of a relation.
OL2AP	Reduction	This class represents a process of reduction of the data stored in a relational database.
	RSchema	This class represents a relational schema chosen for the data reduction.
	RTable	This class represents a table chosen for the data reduction.
	RColumn	This class represents a column whose data have been reduced.
	Descriptor	A descriptor is a tag that can be attached to any element of the model.

In the modelling process, in order to obtain identifiers of the created metaobjects and to ensure correct referencing among these objects, techniques derived from object-oriented database management must be used (cf. [22] for instance).

4.2. Metadata Representation. The metadata are stored into a relational database (cf., the architecture in Figure 1). The logical schema of the relational metadata database is shown in Figure 5. This schema expresses the concept that a data reduction requires a physical connection to a database, which may be composed of several and independent logical schemas. On the turn, a schema is composed of a set of tables, and a table is composed of one or more columns. Each table and each column can be tagged by several descriptors, which are pairs (*name*, *value*) useful to refer to context-dependent data.

However, such metadata must be exported in web-based environments to be accessed by decision makers since they need to know which data are available for approximate query processing and how to formulate analytical queries.

The business intelligence platform capability matrix [23] has been defined by the Gartner Group in order to establish standard criteria to evaluate systems used by business companies to develop applications supporting decision makers. A set of criteria is related to the information delivery issue. According to these criteria, a platform must provide the ability to publish dashboards to a web-based interface and each user must be allowed to build personal indicators.

To this end, the OMG has defined the XMI format as the physical layer used to store the metadata that are obtained through a serialization process of the created metaobjects. Since it is XML-based, XMI allows high interoperability among business intelligence tools. Therefore, an XMI-compliant file can be suitably transferred via a web service.

5. Case Study

In this section, we illustrate the experimentation executed with an approximate query answering system. The tool, named *Analytical Data Profiling* (ADAP) system, is based on polynomial series approximation and produces sets of coefficients that permit to have knowledge and management of the multivariate data distribution of data warehouse cubes. We used this system as the testing environment of the *OL2AP* Metamodel.

5.1. System Architecture. According to the general architecture shown in Figure 1, the architecture of ADAP is presented. In detail, ADAP is an OLAP tool, whose features are to collect, to organize, and to process large volumes of data stored in a data warehouse, in order to obtain statistical data profiles to be used in approximate query processing.

The main macroactivities supported by ADAP are as follows.

(1) Data Reduction. In this first step, the system calculates the synopses of data. The calculated data are stored in the system database, which represents the main repository accessed in the next analytical processing. ADAP performs read-only accesses to the data warehouse, whenever it is necessary to (re)calculate the data synopses DS.

(2) Approximate Query Processing. In this step, the system performs the computation of aggregate functions in approximate way, by only using the data synopses. The output of the processing consists of scalar values that represent the approximation of aggregate values. However, the system also provides a method to execute queries directly on the data warehouse, every time the user deals with critical factors or when the maximum precision is needed.

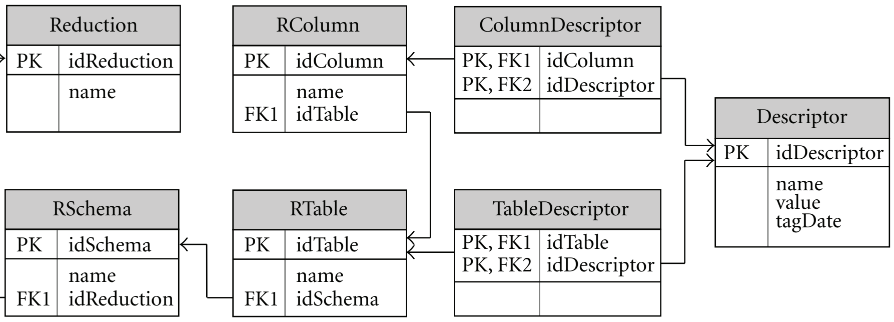


Figure 5: Relational schema of the metadata repository.

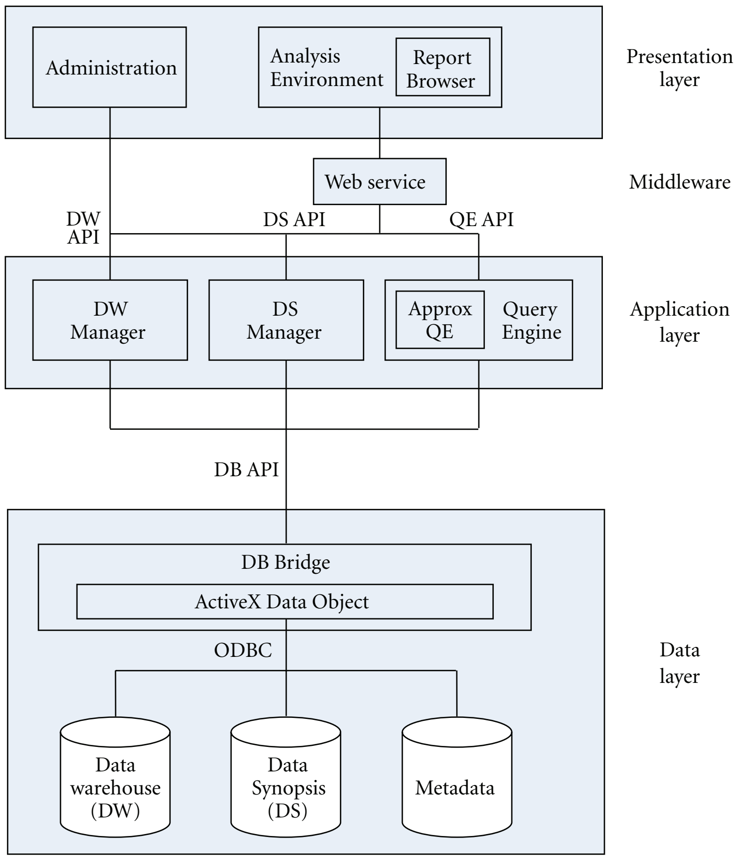


Figure 6: ADAP four-level architecture.

In ADAP system, the data reduction process uses the consolidated methodology of orthonormal series approximation (see [24] also) based on Legendre polynomials [12]. Using this method, data synopses are represented by the sets of coefficients of the polynomials, the so-called canonical coefficients.

On the other hand, if using histogram-based methodologies for example, the data synopses are represented by buckets, each of them containing the frequency of values falling within predetermined intervals.

Regardless of the specific methodology, the data synopses carry synthetic information to approximate the multidimensional data distribution of relations. Therefore, the main aggregate functions (such as *count*, *sum*, and *average*) can be computed without accessing the millions-of-records relations of the data warehouse. However, the response may be affected with a small quantity of error.

ADAP has been designed according to a four-level architecture (see Figure 6) and developed according to a modular design, in order to allow the *add-in* of features not yet implemented. The system manages both the data (*i.e.*, the computed coefficients) and metadata (*i.e.*, information on which data in the data warehouse have been reduced).

In the *Presentation* layer, *Administration* is the input component that allows users to select the data warehouse. It receives the metadata of the selected data warehouse

Reduction

idReduction	Name
1	TutorialDSN

RSchema

idSchema	Name	idReduction
1	Tutorial	1

RTable

idTable	Name	idSchema
1	city_ctr_sales	1
2	city_month_sales	1
3	order_detail	1

TableDescriptor

idTable	idDescriptor
1	1
2	2
3	3
1	5
2	6
3	4

Descriptor

idDescriptor	tagDate	Name	Value
1	2012-05-28	degree	27
2	2012-05-30	degree	25
3	2012-05-30	degree	21
4	2012-05-28	cardinality	407529
5	2012-05-30	cardinality	100000
6	2012-05-30	cardinality	200000
7	2012-05-30	min	1
8	2012-05-30	max	100
9	2012-05-30	min	1
10	2012-05-30	max	100
11	2012-05-28	min	1
12	2012-05-28	max	200000
13	2012-05-28	min	1
14	2012-05-28	max	39
15	2012-05-28	min	1
16	2012-05-28	max	1200
17	2012-05-28	min	1
18	2012-05-28	max	10000

RColumn

idColumn	Name	idTable
1	cust_city_id	1
2	cust_city_id	2
3	order_id	3
4	emp_id	3
5	unit_price	3
6	customer_id	3

ColumnDescriptor

idColumn	idDescriptor
1	7
1	8
2	9
2	10
3	11
3	12
4	13
4	14
5	15
5	16
6	17
6	18

Figure 7: Example of standard metadata.

(in particular, names of fields and tables) from the *DW Manager* and presents them to the user. Using the *Administration* component, users define the attributes to be involved in the data reduction. When the user ends the selection process, this component asks the *DS Manager* to start the computation of the data synopses on the basis of the selected attributes.

Analysis Environment is the web-based application that allows users to define business indicators to be published in dashboards [25]. First, it loads ADAP metadata via the web service. These metadata define which data are available for approximate query processing. The approximate analysis is executed by accessing only the *Data Synopsis* repository. This kind of analysis returns very fast query answers and the results are visualized in the *Report Browser*, which is the dashboard container. In detail, it gets from the *Analysis Environment* component the user query and starts the analytical processing by calling a public method provided by the web service. At the end of the computation, it reports the results, that can be approximate or real values, and it also shows the response time (in msecs).

In the *Application* layer, *DW Manager* is the only component deputed to access the data warehouse using the *DB Bridge* component. It extracts data and metadata from the data warehouse and distributes them to other components. For the *Administration* component, it extracts the metadata of the selected data warehouse. For the *DS Manager*, it performs a read access to the data warehouse and passes it the dataset containing the data to be used for data reduction. For the *Query Engine* component, it supports real analyses by executing SQL instructions on the data warehouse.

DS Manager is the basic component with a twofold role: to generate the data synopses (*i.e.,* in our case, the sets of Legendre polynomials' coefficients) and to extract them during the approximate query processing. When storing the coefficients in the *Data Synopsis* database, it also stores further metadata (*i.e.,* which data have been selected by the data warehouse administrator for data reduction).

Query Engine is the basic component that executes OLAP queries. In approximate query processing, its subcomponent, the so-called *Approximate Query Engine*, uses the data synopses and returns the approximate answers. In the other cases, the *Query Engine* translates the query into an SQL instruction to be executed by the *DW manager*.

Finally, the *DW Manager* and the *DS Manager* interact with the *DB Bridge* that is the component that manages the communication with both the DW and DS databases via an ODBC connection, according to metadata.

5.2. Example of Metadata.

In the system, there are two basic roles played by users, namely the one played by the data warehouse administrator and that played by the business users.

First, an ODBC connection must be established. Then, the administrator is able to view all the tables (and the related columns) included into the selected data warehouse. At this

```
<CWMOL2AP:RTable xmi.id="a32" name="order_detail">

    <CWM:ModelElement.taggedValue>
      <CWMOL2AP:Descriptor xmi.id="a48" tag="cardinality" value="407529" tagDate="2012-05-28"/>
    </CWM:ModelElement.taggedValue>

  <CWM:Classifier.feature>

    <CWMOL2AP:RColumn xmi.id="a41" name="emp_id">
        <CWM:ModelElement.taggedValue>
          <CWMOL2AP:Descriptor xmi.id="a49" tag="min" value="1" tagDate="2012-05-28"/>
          <CWMOL2AP:Descriptor xmi.id="a50" tag="max" value="39" tagDate="2012-05-28"/>
        </CWM:ModelElement.taggedValue>
    </CWMOL2AP:RColumn>

    <CWMOL2AP:RColumn xmi.id="a38" name="order_id">
        <CWM:ModelElement.taggedValue>
          <CWMOL2AP:Descriptor xmi.id="a51" tag="min" value="1" tagDate="2012-05-28"/>
          <CWMOL2AP:Descriptor xmi.id="a52" tag="max" value="20000" tagDate="2012-05-28"/>
        </CWM:ModelElement.taggedValue>
    </CWMOL2AP:RColumn>

    <CWMOL2AP:RColumn xmi.id="a35" name="unit_price">
        <CWM:ModelElement.taggedValue>
          <CWMOL2AP:Descriptor xmi.id="a53" tag="min" value="1" tagDate="2012-05-28"/>
          <CWMOL2AP:Descriptor xmi.id="a54" tag="max" value="1200" tagDate="2012-05-28"/>
        </CWM:ModelElement.taggedValue>
    </CWMOL2AP:RColumn>

    <CWMOL2AP:RColumn xmi.id="a39" name="customer_id">
        <CWM:ModelElement.taggedValue>
         <CWMOL2AP:Descriptor xmi.id="a55" tag="min" value="1" tagDate="2012-05-28"/>
          <CWMOL2AP:Descriptor xmi.id="a56" tag="max" value="10000" tagDate="2012-05-28"/>
        </CWM:ModelElement.taggedValue>
    </CWMOL2AP:RColumn>

  </CWM:Classifier.feature>
</CWMOL2AP:RTable>
```

Figure 8: Part of XMI-compliant file used in approximate query processing.

point, the administrator has only to select a table and to define which columns must be considered for data reduction.

After this, the data reduction process generates the set of coefficients, according to the polynomial approximation degree d chosen by the user.

In this step, the system produces also the metadata according to the *OL2AP* metamodel.

The computed coefficients are stored in the database managed by the approximate query answering system, while the metadata are saved in the central repository of the business intelligence system (cf. Figure 1).

The ADAP system needs to trace also:

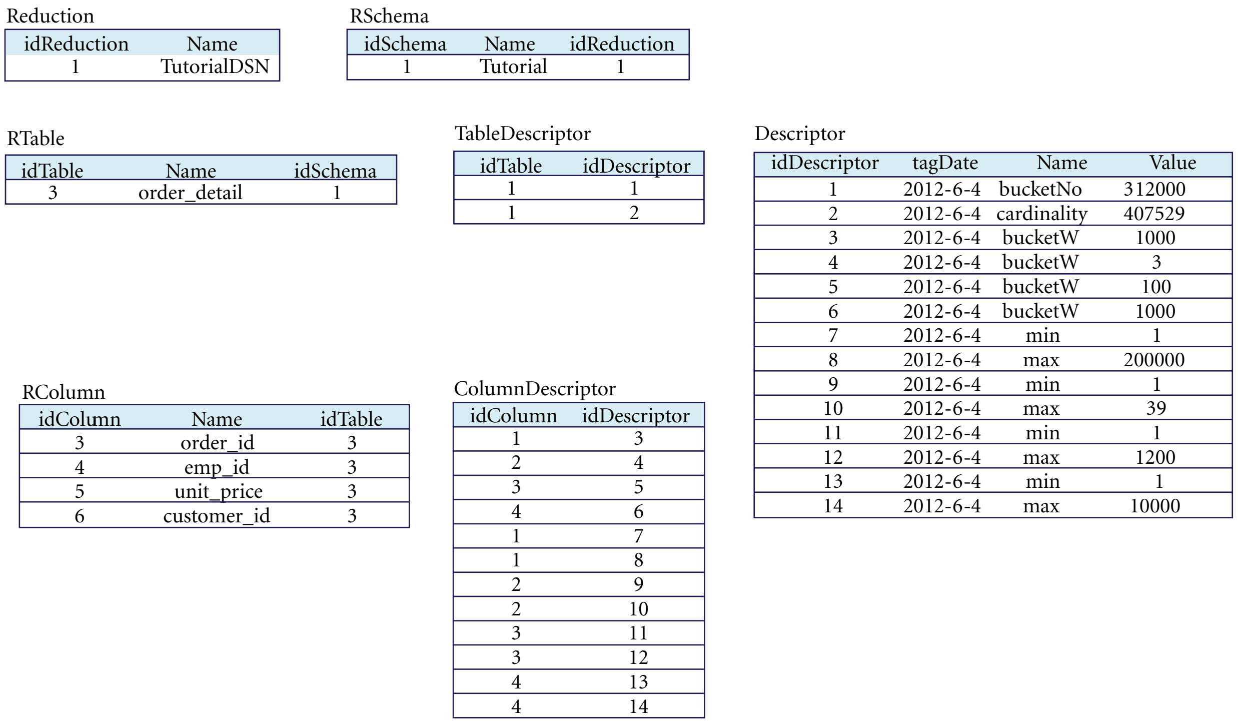
(i) the minimum and maximum of each column,

(ii) the number of rows of each table,

because these data will be used by the algorithms performing analytical processing based on approximate responses.

In Figure 7, there are reported the metadata relative to our case study, automatically generated and stored by the ADAP system (cf. Figure 1 and Data layer of Figure 6) according to the settings and choices made by the users through the administration interface. Using these metadata it is possible to know which table and columns have been compressed and, then, available for approximate query processing.

Finally, the metadata can be exported in XML format and used in the web-based interface that allows users to create a business indicator and to obtain fast approximate answers, by choosing any approximation degree x such that $x \leq d$. Furthermore, the use of XML guarantees a high interoperability among the several systems, ensuring



Reduction

idReduction	Name
1	TutorialDSN

RSchema

idSchema	Name	idReduction
1	Tutorial	1

RTable

idTable	Name	idSchema
3	order_detail	1

TableDescriptor

idTable	idDescriptor
1	1
1	2

Descriptor

idDescriptor	tagDate	Name	Value
1	2012-6-4	bucketNo	312000
2	2012-6-4	cardinality	407529
3	2012-6-4	bucketW	1000
4	2012-6-4	bucketW	3
5	2012-6-4	bucketW	100
6	2012-6-4	bucketW	1000
7	2012-6-4	min	1
8	2012-6-4	max	200000
9	2012-6-4	min	1
10	2012-6-4	max	39
11	2012-6-4	min	1
12	2012-6-4	max	1200
13	2012-6-4	min	1
14	2012-6-4	max	10000

RColumn

idColumn	Name	idTable
3	order_id	3
4	emp_id	3
5	unit_price	3
6	customer_id	3

ColumnDescriptor

idColumn	idDescriptor
1	3
2	4
3	5
4	6
1	7
1	8
2	9
2	10
3	11
3	12
4	13
4	14

FIGURE 9: Example of standard metadata in equal-width histograms.

the interaction via web services and, therefore, obtaining a location independence.

The code in Figure 8 is part of the XML file obtained when exporting the metadata represented in Figure 7.

Using these metadata, it is possible to attach a tag to any model element. As an example, we can associate the descriptor (*cardinality*, 407529) to the *order_detail* table in order to state that the cardinality of this reduced table is 407,529 rows. As a further example, we can associate the descriptors (min, 1) and (max, 100) to a given column in order to trace the minimum and the maximum of its domain, respectively.

5.3. Application to Further Methodologies. The *OL2AP* metamodel has been applied also to systems using methodologies to perform the approximate query processing different from the orthonormal series. In particular, we have tested the metamodel on the histogram-based system (see, [10]).

This methodology needs to trace tables and columns involved in the data reduction process, as it happens in polynomials approximation. But, in histogram-based methodology, the number of buckets to be used for data reduction must be also chosen. More precisely, the equal-width histograms first establish the width of the buckets for each column involved in the reduction process, then the frequency in each bucket is computed. On the other hand, the equal-depth histograms first fix the frequency, then the widths of the buckets are computed so as to obtain the same frequency for all the buckets. All these parameters have to be considered as descriptors to be attached to the model elements.

As a consequence, we observe metadata like those depicted in Figure 9 in case of equal-width histograms. As an example, we have min = 1 and max = 200000 for the domain of *order_id* attribute. Moreover, the fixed bucket width for this attribute is 1000 and, then, we have 200 buckets for *order_id*. In the same way, we compute 13 buckets for *emp_id*, 12 for *unit_price*, and 10 for *customer_id*. Therefore, the total number of buckets, which is a descriptor attached to the *order_detail* table, is $200 \times 13 \times 12 \times 10 = 312000$.

In a similar way, it is trivial to verify the application of the metamodel to other methodologies, since the difference consists only in the creation of specific descriptors as required by each approach.

6. Conclusions

The necessity to decrease the response time in OLAP has led to the exploitation of approximate query answering systems as business intelligence tools able to provide useful information for decision makers, on the basis of fast and approximate answers. However, the current standard for the definition of metadata used by OLAP tools does not include a metamodel to represent *ad hoc* metadata for these systems.

In this paper, we have presented an extension of the standard metamodel that can be used by approximate query answering systems in order to create their own metadata according to the requirements identified by Table 1. The results showed that the metamodel effectively traces which data are available for analytical processing based on approximate methodologies that perform a data reduction. This allows both users to formulate queries based on approximate answers and systems to automatically generate plan for accessing reduced data on the basis of user-defined queries.

References

[1] J. Poole, D. Chang, D. Tolbert, and D. Mellor, *Common Warehouse Metamodel*, John Wiley & Sons, 2002.

[2] A. Sen, "Metadata management: past, present and future," *Decision Support Systems*, vol. 37, no. 1, pp. 151–173, 2004.

[3] Object Management Group, "Common Warehouse Metamodel Specification," vers. 1.1, vol. 1, OMG, Needham, MA, USA, 2003, http://www.omg.org/docs/formal/03-03-02.pdf.

[4] Object Management Group, "XML Metadata Interchange (XMI) Specification," vers. 2.0, OMG, Needham, MA USA, 2003, http://www.omg.org/docs/formal/03-05-02.pdf.

[5] S. Chaudhuri, U. Dayal, and V. Ganti, "Database technology for decision support systems," *Computer*, vol. 34, no. 12, pp. 48–55, 2001.

[6] F. Di Tria, E. Lefons, and F. Tangorra, "Metrics for approximate query engine evaluation," in *Proceedings of the 27th ACM Symposium on Applied Computing (ACM SAC '12)*, pp. 885–887, Riva del Garda, Italy, 2012.

[7] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, "The AQUA approximate query answering system," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 574–576, Philadelphia, PA, USA, 1999.

[8] A. Gupta, V. Harinarayan, and D. Quaas, "Aggregate-query processing in data warehousing environments," in *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)*, pp. 358–369, Zurich, Switzerland, 1995.

[9] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate query processing using wavelets," *The International Journal on Very Large Data Bases*, vol. 10, no. 2-3, pp. 199–223, 2001.

[10] Y. Ioannidis and V. Poosala, "Histogram-based approximation of set-valued query answers," in *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*, pp. 174–185, Edinburgh, Scotland, 1999.

[11] P. B. Gibbons and Y. Matias, "New sampling-based summary statistics for improving approximate query answers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 331–342, Seattle, Wash, USA, 1998.

[12] E. Lefons, A. Merico, and F. Tangorra, "Analytical profile estimation in database systems," *Information Systems*, vol. 20, no. 1, pp. 1–20, 1995.

[13] J. Spiegel and N. Polyzotis, "TuG synopses for approximate query answering," *ACM Transactions on Database Systems*, vol. 34, no. 1, article 3, 2009.

[14] J. B. Peltzer, A. M. Teredesai, and G. Reinard, "AQUAGP: approximate query answers using genetic programming," in *Proceedings of the 9th European Conference (EuroGP '06)*, pp. 49–60, Budapest, Hungary, April 2006.

[15] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra, "Scalable approximate query processing with the DBO engine," *ACM Transactions on Database Systems*, vol. 33, no. 4, article 23, 2008.

[16] Object Management Group, "XML MetaObject Facility Specification," vers. 1.4, OMG, Needham, MA USA, 2002, http://www.omg.org/docs/formal/02-04-03.pdf.

[17] C. dell'Aquila, F. Di Tria, E. Lefons, and F. Tangorra, "Data reduction for data analysis," in *New Aspects on Computing Research*, C. Cepisca, G. A. Kouzaev, and N. E. Mastorakis, Eds., pp. 204–210, 2008.

[18] F. Di Tria, E. Lefons, and F. Tangorra, "Hybrid methodology for data warehouse conceptual design by UML schemas," *Information and Software Technology*, vol. 54, no. 4, pp. 360–379, 2012.

[19] F. Di Tria, E. Lefons, "GrHyMM: a graph-oriented hybrid multidimensional model," in *Proceedings of the Advances in Conceptual Modeling. Recent Developments and New Directions*, pp. 86–97, Springer, Brussels, Belgium, 2011.

[20] D. Marco, *Building and Managing the Meta Data Repository*, Wiley, 2000.

[21] A. Tannenbaum, *Metadata Solutions*, Addison-Wesley, 2002.

[22] E. Bertino and L. Martino, "Object-oriented database management systems: concepts and issues," *Computer*, vol. 24, no. 4, pp. 33–47, 1991.

[23] K. Schlegel and B. Sood, *Business Intelligence Platform Capability Matrix*, Gartner Research, 2007.

[24] F. Yan, W. C. Hou, Z. Jiang, C. Luo, and Q. Zhu, "Selectivity estimation of range queries based on data density approximation via cosine series," *Data & Knowledge Engineering*, vol. 63, no. 3, pp. 855–878, 2007.

[25] T. Palpanas, P. Chowdhary, F. Pinel, and G. Mihaila, "Integrated model-driven dashboard development," *Information Systems Frontiers*, vol. 9, no. 2-3, pp. 195–208, 2007.