A Framework for Active Data Warehouses

Sonia Bouattour¹, Riadh Ben Messaoud¹, Omar Boussaid¹, Hanene Ben Abdallah² and Jamel Feki²

¹ Laboratory ERIC, University Lumière Lyon 2, France.
{sonia.bouattour, rbenmessaoud, omar.boussaid}@eric.univ-lyon2.fr
² Laboratory MIRACL, University of Sfax, Tunisia.
{jamel.feki, hanene.benabdallah}@fsegs.rnu.tn

Abstract. Data warehousing and OLAP technology have known an important progress toward an efficient support for decision making. Nevertheless, traditional decision systems are not equipped with automatic analysis mechanisms and can not maintain dynamic OLAP reporting. Therefore, we propose an approach of an active data warehouse (ADW) where we exploit the notion of active rules (ECA rules) already known in active databases. We integrate analysis scenarios into the decision system regarding events, conditions and actions that can express usual analysis needs of enterprises. In this paper, we particulary define our scenarios by using an XML modeling of analysis rules. Here, XML is used to model the logical level as well as the physical level of analysis rules. We also develop a prototype software in order to validate our proposition.

1 Introduction

Data warehouses were introduced in the 90s. Since then, they have known a widespread acceptance and an important progress toward an efficient support for decision making. Traditionally, a data warehouse is defined as a huge collection of subject-oriented, integrated, consolidated, time-varying and non-volatile data [1, 2]. A data warehouse can be viewed as a large repository of historical data organized in a pertinent way in order to perform analysis and to extract interesting information through the use of the OLAP technology. In fact, data are usually designed according to multidimensional structures, commonly called data cubes. Each data cube expresses a specific analysis context that may be visualized, explored and summarized thanks to OLAP operators [3]. Analytical reports can therefore be generated from multidimensional data according to the needs defined by the business decision makers.

Nevertheless, traditional data warehouse and OLAP systems do not manage automatic generation of analytical reports. Yet, in a competitive environment, enterprises need reactive decision systems able to automate reporting throughput, maintain dynamic analysis, and improve the user control over the analytical processing. In order to equip data warehouses and OLAP with these capabilities, we propose an approach of an active data warehouse (ADW). We particularly exploit the notion of active rules already known in active databases. An active
database is a system which provides the functionality of a traditional database and additionally is capable of reacting automatically to state changes, both internal and external, without user intervention [4]. An active rule consists traditionally of events, conditions and actions. This triplet can be used to fully define analysis rules that express scenarios used for OLAP reporting. Thus, our proposition consists in integrating these analysis rules into the decision system. In this paper, we provide a conceptual formalization of analysis rules. We define the notion of events, conditions and actions in OLAP scenarios. We also propose an XML modeling of analysis rules. Therefore, the XML formalism is used to model the logical level as well as the physical level of OLAP scenarios. Hence, analysis rules are stored in XML documents. Finally, in order to validate our approach, we propose a case study and a prototype software.

This paper is organized as follows. In section 2, we expose a state of the art about ADWs. Section 3 presents notations and the terminology adopted in our formalization of analysis rules introduced in section 4. In the following section, we expose our XML modeling of analysis rules. In section 6, we propose to validate our approach. First, we introduce our ADW architecture and its prototype software then we present a validation example. Finally, in section 7, we conclude and present some future research directions.

2 Related work

A lot of proposals have designed frameworks for ADWs. The concept of ADWs generally consists in combining active mechanisms based on Event, Condition, Action rules (ECA rules) with the analysis power of data warehouses. These proposals aim at extending passive decision systems and provide them with reactive capabilities [5].

On the one hand, most proposed frameworks have particularly focused on the ETL (Extracting, Transforming, Loading) steps and treated the problem of data refreshments. For instance, in order to speed the refreshment process, Karakasidis et al. manage ETL activities over queue networks [6]. They employ the queue theory to predict the tuning of the operation during the overall refreshment process. In [7], Bruckner and Tjoa propose to treat the problems of time consistency in ADW. Their approach introduce a data model that deals with timely delays about refreshments of detailed data of a warehouse. Mohania and Inderpal note that an enterprise should adapt changes that occurs on their information sources in a timely manner [8]. These changes might detected automatically or semi-automatically by monitoring and analyzing the information sources. In order to build a reactive business processes, the authors propose in [8] a policy for a reactive integration of information in a data warehouse. This policy aims at correlating events occurring in information sources so that timely business decision can be taken. In [9], Polyzotis et al. consider that active data warehousing has basically emerged as an alternative to conventional warehousing practices in order to meet the high demand of applications for up-to-date information. Thus, ADWs should achieve an efficient consistency between the
stored information and the latest on-line data updates. Therefore, the authors propose a join algorithm (MeshJoin) that performs the join of a fast stream of source updates with a disk-based relation. Meanwhile, Araque defines the notion of a real-time data warehouse (RTDW) as a historical and analytical component of an enterprise level data stream [10]. In order to capture data from web sources and feed a RTDW, Araque proposes to reduce the delay between the changes of a web page and the time these changes are detected by the data refreshment system. Nguyen and Min has also defined a framework for a zero latency data warehousing (ZLDWH), which attempts to reduce the integration time of data into a warehouse [11]. Their framework combines the continuous data integration and active rules based on ECA techniques (Event – Condition – Action).

On the other hand, some proposals treated the active warehousing from an analytical point of view. They aim at reaching forward a reactive decision system capable to perform automatic analytical tasks. According to Thalhammer et al., ADWs belong to a new category of decision support systems that offer automatic routine for decision tasks [12]. In that sense, Thalhammer et al. define an active data warehouse cycle which realizes a closed-loop decision approach. This approach is able to export analysis rules – expressing decisions – back to OLTP systems. Therefore, according to the decision needs of users, the data warehouse can extract, transform, and load adequate data from OLTP systems autonomously without user interaction. As defined in [12, 13], analysis rules utilize the idea of ECA rules. An event is used to specify the time points at which analysis rules should be carried out. Whereas a condition is a boolean predicate or a query, which permits the rule to execute the action. Note here that the action is a directive to execute a transaction – such as insert, update, or delete statement – for some entities in the OLTP database.

Unlike the proposal of Thalhammer et al., ours treat the issue of ADWs from an analytical point of view. We aim at attending an automatic analytical processing. Thus, we use the concepts of analysis rules and analysis graphs – already proposed in [12] – in order to model analysis scenarios. We express these scenarios in a general and a simple way with XML, which will allow us to store them in XML documents that can easily be embedded into the decision system.

3 Notations and terminology

We particularly use a data cube model and an OLAP algebra proposed in [14] in order to model our analysis rules.

3.1 Data cube

We assume that $\mathcal{U}$ is a set of data cubes. A cube $U \in \mathcal{U}$ is defined according to the 5-tuples $\langle \mathcal{C}, \mathcal{A}, \mathcal{M}, \mathcal{L}, \mathcal{F} \rangle$, where: (i) $\mathcal{C} = \langle C, d \rangle$ is a space of characteristics representing the dimensions and the measures of the cube; (ii) $\mathcal{A} = \langle A, f, O_A \rangle$ is the attribute space of $U$; (iii) $\mathcal{M} = \langle M, g, O_M, h \rangle$ is the modality space of $U$; (iv) $\mathcal{L}$ is the set of cells of $U$; and (v) $\mathcal{F}$ is a set of aggregation functions. For a more detailed talk on this model, we refer the reader to [14].
3.2  OLAP operators

In order to model OLAP operations that will be used in our analysis rules, we refer to the OLAP algebra proposed in [14]. This algebra basically consists of two families of operators.

The first one concerns operators that manage the structure of a data cube. It includes: (i) adding of a characteristic (AddCharacteristic) ; (ii) deleting a characteristic (DltCharacteristic) ; (iii) nesting an attribute (Nest) ; (iv) pushing a dimension (Push) ; (v) pulling a measure (Pull) ; and (vi) adding a modality into an attribute (AddModality). The second family of operators allows navigation into a data cube. It includes: (i) rolling up a dimension (RollUp) ; (ii) drilling down a dimension (DrillDown) ; (iii) switching a modality (Switch) ; (iv) selecting a modality (Slice) ; and (v) computing an aggregate (Aggregate).

We note that these two families are limited to unary operators. In fact, all the previous operators accept a single input data cube. This algebra is incomplete regarding the modeling of our analysis rules since it does not include nary operators. In order to meet a more general formalisation, we complete this algebra by adding a third family of binary operators. This family includes the union (Union) ; the intersection (Intersect) ; and the difference (Difference) of two data cubes having the same structure.

(i) Union allows to unify two data cubes. It can be achieved by successive adding of modalities (AddModality) in the common attributes cubes.

(ii) Intersect computes the common parts of two data cubes. It can be achieved by applying successive selections (Slice) of common modalities in the cubes.

(iii) Difference tells the difference of two cubes. It can be computed according to successive selections (Slice) of non-common modalities in the data cubes.

4  Analysis rule formalization

We formalize an analysis rule according to the following definition.

Definition (Analysis rule)
An analysis rule is a triplet of the form \( \langle E, N, G \rangle \), where: (1) \( E \) is a conjunction of events ; (2) \( N \) is a set of conditions ; and (3) \( G \) is an analysis graph.

4.1  Conjunction of events

The conjunction of events \( \mathcal{E} \) indicates the moment when one analysis rule will be executed. It is written according to a conjunction of disjunctions of a set of elementary events \( \{ E_1, E_2, \ldots \} \). An elementary event \( E_i \) may represent a fixed point in the calendar or one notable change in the OLTP database. For instance, in a sales data cube, we may consider the events \( \{ E_1, E_2 \} \) where: \( E_1 = \langle \text{Time} = \text{EOM} \rangle \) represents the end of each month ; and \( E_2 = \langle ST < 1000 \rangle \) corresponds to a sales turnover lower than $1000. Thus, we build the conjunction \( \mathcal{E} = E_1 \land E_2 \) which will allow the achievement of an analysis scenario when the sales turnover of the enterprise is less than $1000 at the end of each month.
4.2 Set of conditions

\( N \) is a set of elementary conditions \( \{N_1, N_2, \ldots \} \) that should be satisfied in analysis scenario execution. These conditions express analysis needs and represent the front request stages of a business decision. For example, in a sales data cube, we can consider the set of conditions \( N = \{N_1, N_2, N_3\} \), where: \( N_1 = \langle \text{Product Family} = \text{Food} \rangle \) is a selection of food products; \( N_2 = \langle \text{Country} = \text{Canada} \rangle \) restricts the investigation in Canadian stores; and \( N_3 = \langle \text{Sales Unit} < 300 \rangle \) comes out with sale facts less than 300 units.

4.3 Analysis graph

The analysis graph \( G \) is a conceptual representation of an analysis scenario. It is of form \( \langle U, P \rangle \), where: (i) \( U \) is a non-empty set of cubes and (ii) \( P \) is a non-empty set of analysis paths. We define what we call an analysis path as follows.

**Definition (Analysis path)**

An analysis path, denoted \( P \), is a function from \( U^n \) to \( U \) where \( 1 \leq n \leq 2 \). It associates to one or two input data cubes an output data cube \( P(U) \), which results from the input cube(s) according to an OLAP operation.

According to that definition, an analysis path represents an OLAP operation. It can represent both unary (\( n = 1 \)) or binary (\( n = 2 \)) operations. As shows the analysis graph of Figure 1(a), we represent an analysis path as a direct links that start from the input cubes to the output cube \( P(U) \).

![Analysis Graph](image)

**Fig. 1.** (a) Example of analysis paths. (b) General architecture of our ADW.

We adopt the three previous families of OLAP operators to use them within our framework on analysis graphs. Thus, an analysis graph defines a conceptual formalism to represent an analysis process in order to produce scenarios...
of automatic analytical reporting. These automatic scenarios give to the data warehouse an active character. From a graphical point of view, an analysis graph $G$ is schematized by a set of nodes $U$ (data cubes) and a set of paths $P$ (OLAP operations).

In order to meet the needs of our XML logical modeling, we also introduce the notion of an analysis sub-graph. An analysis sub-graph, denoted $SG$, is an part of an analysis graph $G$ which satisfies the two following conditions: (i) $SG$ starts with an input cube in $G$, or two input cubes of a binary operation ; and (ii) $SG$ ends by an output cube in $G$, or an output cube of a binary operation. For example, in the analysis graph of Figure 1(a), we distinguish two sub-graphs delimited by the dotted frames.

## 5 XML modeling of analysis rules

We propose to rewrite an analysis rule according to the XML formalism at a logical level. An XML analysis rule consists of three principal tags names: Events, Conditions and AnalysisGraph. An analysis graph has one or several analysis sub-graphs represented by the SubGraph tag. For example, Figure 2 shows the
XML modeling of the two sub-graphs of Figure 1(a). We associate to each sub-graph an InputCubes tag, to indicate its input cube(s), and an AnalysisPaths tag, which represents its successive analysis paths. Each analysis path is modeled with a Path tag, and each Path tag contains one OLAP operation written in XML according to Table 1.

<table>
<thead>
<tr>
<th>OLAP Operation</th>
<th>XML Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCaracteristic</td>
<td><code>&lt;AddCaracteristic caracteristic=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>DelCaracteristic</td>
<td><code>&lt;DelCaracteristic caracteristic=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Nest</td>
<td><code>&lt;Nest caracteristic=&quot;..&quot; attribute_new=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Push</td>
<td><code>&lt;Push attribute_new=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Pull</td>
<td><code>&lt;Pull attributes=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Nest</td>
<td><code>&lt;Nest caracteristic=&quot;..&quot; attribute_new=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Rollup</td>
<td><code>&lt;Rollup caracteristic=&quot;..&quot; attribute_old=&quot;..&quot; attribute_new=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>DrillDown</td>
<td><code>&lt;DrillDown caracteristic=&quot;..&quot; attribute_old=&quot;..&quot; attribute_new=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Switch</td>
<td><code>&lt;Switch attribute=&quot;..&quot; modality_1=&quot;..&quot; modality_2=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Slice</td>
<td><code>&lt;Slice caracteristic=&quot;..&quot; Predicate=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Aggregate</td>
<td><code>&lt;Aggregate caracteristic=&quot;..&quot; modality=&quot;..&quot; /&gt;</code></td>
</tr>
<tr>
<td>Union</td>
<td><code>&lt;Union /&gt;</code></td>
</tr>
<tr>
<td>Intersect</td>
<td><code>&lt;Intersect /&gt;</code></td>
</tr>
<tr>
<td>Difference</td>
<td><code>&lt;Difference /&gt;</code></td>
</tr>
</tbody>
</table>

Table 1. XML modeling of OLAP operators.

Table 1 presents an XML modeling of OLAP operators. Each operator models an XML tag name and its parameters model attributes of the tag. However, binary operators do not have parameters in their tag name. In fact, input cubes of a binary operation are represented in the InputCubes tag of the corresponding sub-graph. Notice here that sub-graphs allow a coherent XML modeling. Indeed, when we deal with a binary operation in an analysis graph, we face the problem of sequencing the input cubes taking part in this operation. The concept of sub-graph resolves this modeling problem by cutting out the analysis graph in several connected parts. Thus, the output cube of a sub-graph may represent the input cube of another sub-graph. In our XML modeling, we use the specific value `ResultOf:[SubGraph_Name]` to indicate the output cube of sub-graph [SubGraph_Name]. This value can thus be used as an input cube in another analysis sub-graph.

6 Validation of our approach

We present in this section, a validation of our approach. First, we present our ADW architecture and its software implementation, then we expose a validation example of an analysis rule.

6.1 ADW architecture and implementation

To validate our ADW approach, we propose a software implementation. As shown in Figure 1(b), the architecture of our software consists of three main
modules: (i) a module for connecting to the OLAP server; (ii) a module for setting analysis rules; and (iii) a module for generating analysis rules in XML documents. According to the our ADW architecture, it is up to the decision maker to model analysis rules by respect to his experience and his decision needs. Then, as soon as these analysis scenarios are generated in the XML format, they are stored in the decision system in XML documents. Therefore, the scenarios may evolve in the multidimensional database management system in the same manner as do transactional data, relational tables, query procedures, OLAP queries, etc. Which means that, each time when events and conditions of an analysis rule are satisfied, the decision system become capable of reading and executing – in a reactive manner – its corresponding XML scenario.

We have implemented our approach in a prototype software that starts from the first module (the connection to the OLAP server) and reaches the generation of analytical scenarios modeled in XML according to our ADW architecture. The prototype has been developed on a workstation with the followed configuration: 1024 MB RAM, 100 Gb of Hard disk, frequency 1.52 GHz. The development of the prototype was completed in JBuilder X environment. Its implementation requires the OLAP server with of Oracle 9i database.

The connection to the server from Oracle OLAP is achieved thanks to JDBC (Java DataBase Connectivity). JDBC is a Java API for connection to databases, i.e that JDBC is a set of classes for developing applications that can connect to database servers (DBMS). The communication interface allows the definition of an analysis rule. In fact, a user can set the name of his analysis rule. Then, he is prompted to fill out the parameters of the trigger events, the conditions and the analysis graph. Through the XML modeling module, our prototype is capable to produce the analysis scenario modeled in XML. At this stage, there must be a parser capable of transforming the XML scenario into OLAP queries. This is used to produce analysis reports on each scenario execution.

### 6.2 Validation example

For instance, suppose that a decision maker need to build an analysis scenario that compares the sales of food products in “Victoria” (Canada) and “Beverly Hills” (USA). Thus, the relating conditions are: (i) City = “Victoria”; (ii) City = “Beverly Hills”; and (iii) Family products = “Food”. Suppose that the events relating to this analysis rule are: (i) the end of each month; and (ii) when the turnover exceeds $10000.

Figure 3 shows the analysis graph of the analysis rule constructed by the decision maker. The analysis graph includes four analysis sub-graphs delimited by dotted frames in the figure. We provide in Figure 3 two fragments of the generated XML document corresponding to that analysis rule. The first fragment models the sub-graph SubGraph_1 and the second one models the sub-graph SubGraph_4.
7 Conclusion

Recall that our objective is to propose a simplified and general formalization of an ADW framework. By exploiting the ECA formalism known in active databases, we introduced a new formalism of analysis rules and defined therefore a general framework of an active data warehouse. In this framework, XML is used to model both the logical and the physical level of analysis rules. We also provide an software prototype in order to validate our approach.

Our current approach opens various research directions. We need to propose a model of an ADW which integrates the analysis rules in its ETL process. We also need to add a parser able to read, interpret analysis scenarios and transform them into OLAP queries. In the proposal of Thalhammer et al. [12, 13], three cases were presented in the decision-making process, namely: the routine decision tasks, the semi-routine decision tasks and the non-routine decision tasks. We expect in further work to expand our ADW proposal towards the two latter cases. For instance, we intend to enlarge our study of the ADWs with regard to refreshments and data updates.

Finally, another possible track consists in evolving the ADW to Web services particularly using Active XML. In Active XML, a tag can contain a dynamic procedure which calls upon one or more Web services [15]. Thus, we will be
able to consider the storage of analysis graphs while taking into account this mechanism.

References