A RULE REPOSITORY FOR ACTIVE DATABASE SYSTEMS

Sidney Viana
UNIFIEO - Centro Universitário FIEO, Computer Science Department
Osasco, Brazil, CEP 06020-190
viana@unifieo.br

and

Jorge Rady de Almeida Junior
University of São Paulo, Department of Computer Engineering
São Paulo, Brazil, CEP 005508-900
jorge.almeida@poli.usp.br

and

Judith Pavón
Anhembi-Morumbi University, Computer Science Department
São Paulo, Brazil, CEP 04546-000
jpavon@anhembi.br

Abstract:

Active Database Systems (ADBSs) provides a good infrastructure to define and execute active rules. Nevertheless, this infrastructure offered by ADBSs does not completely satisfy the necessities of rules management that demands current business applications. Rules also need to be stored in appropriate structures to facilitate their management, as the existing structures for data in these systems. This work proposes a rule repository, composed by structures that allow the storage and organization of rules, in order to facilitate their management. For this purpose, a rule classification with the main rule types existing in the literature is presented, and then, it represents the characteristics and anatomy of each type in a meta-model, with the goal of analyzing the data that must be stored about rules. The rule repository, proposed in this paper, has been built based on this meta-model.

Keywords: Business Rules, Active Database Systems, Rule Repository

1. INTRODUCTION

Today's fast changing and global environment dictates that a successful enterprise has a rich decision-making process. This means not only gathering and processing data using information systems, but also making decisions with the support of business rules. Business rules are statements about how a business is done, i.e. about guidelines and restrictions concerning to states and processes of an organization. The business rules representation, storage and management are crucial in all information systems, because this knowledge becomes the main assets of all enterprise.

The business analyst can make decisions faster when rules are automated. This automation requires an environment that provides resources for the rules storage and management. For its appropriate storage, it is necessary to have a proper understanding of the concept of rule and its parts or elements. Rules should be specified in an executable rule language, i.e. a SQL standard language, in order to be automated in an information system (IS) [3]. The activity of rules management consists of the definition, query, elimination and modification of rules [14].

Generally, business rules are specified in a programming language, like Cobol or Java, or in an Active Database Systems (ADBS). An ADBS is a conventional database system extended with the capability of reactive behavior [1]. This means that the system can perform certain operations automatically, in response to certain situations that have occurred in the database. The ADBS is composed by an Active Database Management System (ADBMS) and a database. An ADBMS support the definition, management and

execution of Event-Condition-Action (ECA) rules[11, 14] or variations thereof that specify reactive behavior. ECA rules consist of events, conditions and actions. That means: "when an event occurs, check the condition and if it holds, execute the action". ECA rules are also known as active rules or triggers in ADBMSs.

When business rules are implemented in information system, using a programming language, the code that implements a rule is generally scattered across several application programs. If business requirements changes, developers must read through application code to locate the corresponding rules, make the changes, recompile them and test them. Modifying rules in this way is an arduous and delayed task. In the other hand, when a rule is implemented in an ADBS, rules are defined once, centrally managed by the ADBMS, stored in a rule repository and shared by all application programs that access the database. Business rules are implemented as constraints when they are very simple (i.e. domain constraints), and they are implemented as triggers when they are more complex (i.e. workflow rules). The importance of triggers is that they can express a lot of the semantic that is normally encoded in every application program.

There is a large number of business rules in information systems, and they are of various types, scopes and levels of complexity. This complexity is due to the relationship between rules and its relationship with meta-data. They have several types relationships, the same way there are different types of relationships between data. An example of rules relationships is when a rule is implicitly fired by the occurrence of an event, and execution of this rule cause again the firing of other rules, giving continuity to this sequence of firings, until a mechanism be activated automatically for avoiding an indefinite triggering. Therefore, all of these relationships can generate this way a complex association net. A way of avoiding this infinite chaining is to store rule information and their relationships in a repository, in an ADBMS. As consequence, rules are treated as database objects in a manageable and controlled environment, and they can be shared by all applications that access to the database.

In spite of ADBS advantages for implementing business rules, there is a lack of these systems in rule definition languages, because they do not allow specifying more complex business rules due to the limited expressiveness power of these rule languages, considering only rules that can be represented by constraints or triggers, omitting complex rules like the rules represented within programs code as chains of *if-then-else-statements* or *case-statements*. Moreover, rule repositories of these systems do not contain complete information about rules defined in the ADBS, because it stores only the rule definition and the rule relationships with meta-data, leaving out considering relationships between rules. In this way, there is a lack of rule management operations due to the inappropriate infrastructure of the rule repository.

Considering the fact that rules should be stored in proper structures to facilitate their management, as the existing structures for data in databases, and that current ADBS still have inappropriate rule repository for storing them. It is concluded that it is necessary to evaluate the current rule repository in order to discover what kind of relevant rule information is not stored in it, and also to discover if their structures are enough to support the information that lacks in them, in order to enriching this repository.

This work presents a rule repository for relational ADBSs and it considers the main rule types frequently represented in information systems. For identifying the main rule types, many rule taxonomies proposed in the literature were analyzed, and finally a rule taxonomy that consolidates the rule types represented in the main classifications was selected [15, 19]. The resultant rule meta-model of this taxonomy is studied, because it represents in a simple way the characteristics and relationships of rule types. The rule repository is built based on this meta-model.

The structure of this paper is as follows: Section 2 briefly describes the rule taxonomy used as base for the rule repository. In Section 3, the rule meta-model is presented, considering all rule types of the rule taxonomy used in this work. In Section 4, some rule repositories of ADBSs are analyzed, and then, in Section 5, a rule repository is proposed, based on the rule meta-model described in Section 3. Finally, Section 6 presents conclusions and discusses some issues for possible extensions of this work.

2. RULE TAXONOMY

In literature there are numerous proposals of business rule classifications [4, 20], considering the business context or the rule functionality. When it is considered the business context [12, 17], the rule types are defined in accordance to the particular characteristics of an organization or enterprise. Thus, there are different rule types depending on the business branch or each orgaization's policy of each organization. For instance, rules can be classified according to the area, such as, marketing rules, sales rules, etc. In the other hand, when the criterion to classify rules is their functionality, the classification is independent of the business context that rules express; it considers only the functionality of the rule, i.e. rules that define integrity constraints on business data.

In this work, the main rule types are considered according to the rule functionality. Unfortunately, there is not a consensus about business rule classifications in this context. Most of classifications used different names for the same rule type, and in some cases, there are rule types that are considered in some classification but not in the others. A consolidation of rule classifications is proposed in [15], in order to reorganize and present all

the main rule types in a unique proposal, considering a systemic view. In Table 1 the rule types proposed by the main rule classifications are presented in the first column in groups. Each group is formed by similar rule types and in the second column is assigned a name proposed for each group.

Table 1. Rule Taxonomy [15]

Rule types (Main classifications)	Proposed Rule types
- Structural assertion [5]	Structural rules
- Consistency rules [9] - Action assertion [5] - Integrity rules [18] - Integrity rules [20]	Assertions
- Derivation [5] - Derivation [18] - Inference rules [20]	Derivation
- Activity rules [9]	Operational rules
- Reaction rules [18] - Active rules [20]	Stimulus/Response rules

The rule taxonomy proposed does not define new rule types, just puts together all rule types presented in the main rule classifications as it can be seen in Table 1, obtaining this way a more complete and consistent rule classification. To illustrate the principal characteristics of these rule types, the definition of each rule type and some examples in natural language are presented.

- **Structural rules:**

These rules are sentences that define the business concepts or some feature related to the structure of the organization. Some examples are shown below.

R1: Each project should be associated to a specific department.
R2: Each department should have a manager.

- **Assertions:**

They are integrity constraints about business rules; therefore, any action that tries to violate them will be rejected. The following sentences present some examples of this rule type.

R3: The budget of any project can not exceed 300,000 dollars.
R4: The deadline to finish any project should be equal or less than 24 months.

- **Derivation:**

This rule type allows generating new information, based on known information. Some examples are shown below.

R5: If the employee's category is A1, then his bonus is U$ 3,000.00 per month
If the category is A2, then his bonus is U$ 2,000.00 per month

This rule type has the Condition-Action (CA) format, that is, each rule is composed by a condition and an action.

- **Stimulus/Response rules:**

They define the action should be executed as response to the occurrence of the predefined event. Some examples are presented in the following sentences.

R7: Each sold product, must have its stock updated.
R8: Every time that the stock value is updated; it must be checked if this value is equal or less than the minimal stock.

This rule type is characterized by having the ECA format. The condition is optional; therefore, sometimes these rules have the EA format.

- **Operational rules:**

They define an action set or a sequence of operations that must be performed to reach some goal. The R9 rule, specified below, shows an example of this rule type.

R9: The final grade of a student in a subject named "Data Structures" must be calculated in the following way:
- Add the grades of the two partial evaluation and divide the result by two; this grade is considered Grade1.
- Add the grade of the practical work with the grade of the final exam; this grade is considered like Grade2.

- The final grade is obtained as it follows: (Grade1 + Grade2) / 2

R9 specifies all of operations that are necessary for obtaining the final grade of a specific subject. This rule type is characterized for having the Action (A) format, the rule is composed by an operation or operations set.

Some case studies [16] that implements each rule type, described above, in the SQL3 language, revealed that the simple rules (structural rules and assertions) have an appropriate support in ADBMSs, but the complex rules (derivation, action rules and stimulus/response rules) do not always have support in these systems. We only focus the complex rules, because the other rule types already have appropriate structures in the rule repository of ADBMSs for their storage.

The anatomy of each complex rule type is shown in Table 2. Two kinds of action are proposed for rules that have the condition element, the primary action that is executed when the condition is satisfied (A), and the secondary action that is executed the opposite (A_2). This format facilitates the specification of rules that are represented like chains of *if-then-else-statements* or *case-statements*.

Table 2. Rule Types Anatomy.

Rule Types	Anatomy
Derivation rules	Condition-Action (CA)* Condition-PrimaryAction-SecondaryAction (CA| A_2)*
Action Rules	Action (A)*
Stimulus/Response	Event-Condition-Action (ECA) Event-Action (EA) Event-Condition-PrimaryAction-SecondaryAction (ECA| A_2)

* In executable rule expressions, the event that fires these rule types are not specified in their definitions, the user fired these rules in an explicit way, using a special command (i.e. FIRE).

3. RULE META-MODEL

The goal of a rule meta-model is to represent graphically the rules anatomy, the relationships between rules and between rules and meta-data. Figure 1 presents the rule meta-model with all relevant characteristics of these rule structures in a UML class diagram.

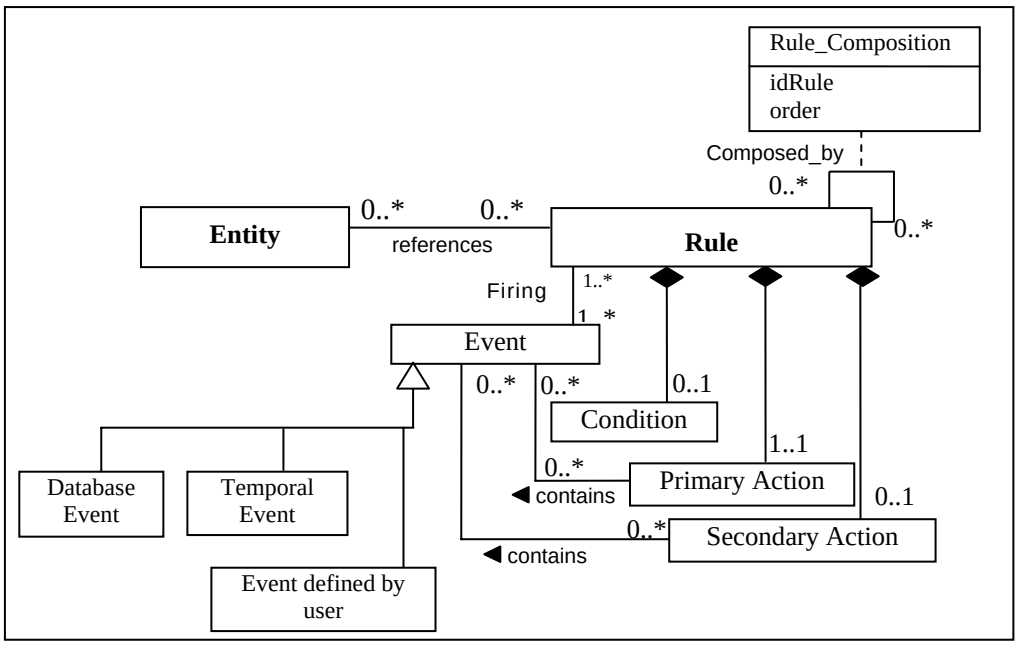

Figure 1. Rule Meta-model [15].

This meta-model is based on the rule taxonomy described in Section 2 and considers the following elements of the ECA paradigm and their variations: Event (E), Condition (C), PrimaryAction or Action (A) and SecondaryAction (A_2). The unique element that is common to all structures is the Primary Action or Action. The elements Action and Primary Action are synonymous.

The two main classes of the meta-model are **Rule** and **Entity**, as shown in Figure 1. The relationship between rule and entity depicts that the rule elements could reference attributes or tables of the database (meta-data). The relationship "firing" represents two situations: when an implicit firing occurs, that is, when a database or temporal event occurs and a rule that contain this event in its definition is fired. The second situation is when an explicit firing occurs, that is, when a rule is firing with a command FIRE (event defined by user). In this case the event is not defined in the rule definition, the command FIRE is specified indicating explicitly the rule to be fired (i.e. FIRE R8). A rule is fired when an event occurs (database event, temporal event or event defined by user). The next state to the rule firing is the rule execution, that implies the condition evaluation, and if the condition is satisfied, the action is executed. Nevertheless, if the rule has a secondary action, this action is executed when the condition is not satisfied. The relationship named "composed_by" implies that a rule could be composed by others rules following a predefined order, that is, a rule can fire other rules explicitly with the command FIRE in its action element.

In summary, the rule meta-model shown in Figure 1 presents two types of rules: rules that are fired by implicit solicitation (database events or temporal events) and they must contain the event in their definition (stimulus/response rules); and rules that are fired by explicit solicitation using the FIRE command (derivation and operational rules). The concepts presented in the rule-meta-model can be mapped to a rule model, specifically in a rule language. The syntax of the rule language is as follows:

CREATE RULE <rule_name>
[BEFORE|AFTER <event> {OR <event>}]
[IF <condition>]
DO <Primaryaction>
[ELSEDO <Secondaryaction>]

Note that the only mandatory element is the primary action. All information about rule definition should be stored in a rule repository for allowing an efficient rule management. For this purpose, it is necessary to define proper structures to store them.

4. THE STATE OF THE ART OF RULE REPOSITORIES

There are many proposals of rule repositories [4, 8, 10, 13], but most of these proposals do not consider all rule types represented in the meta-model rule.

In this section, some current rule repositories are analyzed to find out if these repositories have appropriate structures to store information about rules. First, we present and analyze the rule repository proposed by SQL3, the current standard query language for object-relational ADBMS. Next, we present the structures that compose the rule repository of Oracle, a commercial object-relational ADBMS, and discuss about the content of rule repositories of SAMOS, an academic ADBMS. Finally, we remark some conclusions about the state of the art of relational ADBMS rule repositories.

4.1 Rule Repository proposed by SQL3

SQL3 [10] proposes four tables for the rule repository of ADBMSs. The Triggers table is responsible for maintaining information about the elements of rules, specifically about event, condition and action.

Trigger_Table_Usage and Trigger_Column_Usage store information about the tables and columns respectively that are referenced in the condition or action of rules. Finally, the Trigger-Update-Columns table is responsible of storing information about the columns that are referenced in the update event of the rule. The structure of each system table is described below.

- **TRIGGERS**: this table has one row for each trigger defined in the database. The content of this system table is shown in Table 3.

Table 3. TRIGGERS system table.

Column	Description
Trigger_name	Trigger's name
Event_manipulation	Database operation - DML (insert, update or delete)
Event_object_table	Table's name on which the event is defined.
Condition	Condition of the trigger

Action	Action of the trigger
Action-Orientation	Trigger granularity (*row* or *statement*)
Activation-Time	Activation time of the trigger (*before* or *after*)
Timestamp	Date and hour of creation of the trigger

- **TRIGGER_TABLE_USAGE**: this table has one row for each base table referenced in the trigger, specifically in the condition or action. The content of this system table is shown in Table 4.

Table 4. TRIGGER_TABLE_USAGE system table.

Column	Description
Trigger_name	Trigger's name
Table_name	Table's name referenced in the trigger

- **TRIGGER_COLUMN_USAGE**: this table has one row for each column referenced in the trigger, specifically in the condition or action. The content of this system table is shown in Table 5.

Table 5. TRIGGER_COLUMN_USAGE system table.

Column	Description
Trigger_name	Trigger's name
Column_name	Column's name referenced in the trigger
Table_name	Table's name to which belongs the column referenced in the trigger

- **TRIGGER_UPDATE_COLUMNS:** this table contains a row for each column referenced in the update event of the trigger. The content of this system table is shown in Table 6.

Table 6. TRIGGER_UPDATE_USAGE system table.

Column	Description
Trigger_name	Trigger's name
Event_object_column	Column's name referenced in the definition of trigger event (when the event is the operation *update*)
Event_object_table	Table's name on which the event is defined

These structures allow storing the rule definition and the relationships of rules with metadata (columns/tables). However, they do not have support to store the relationships between rules. Moreover, it considers only the ECA and EA rules, omitting the others rule types. Another issue is that SQL3 considers only database operations as event, and more specifically DML (database manipulation language). SQL3 suggests these structures for the rule repository of object-relational ADBMSs, but they do not have the same structures due to the particular features that each system implements in its query language, despite using the SQL3 language.

4.2 Rule Repositories of ADBSs

In this section, the structures of the rule repositories of some ADBSs are described. First, the rule repository of Oracle [13] is analyzed and to continue then with repository of SAMOS [8], an academic system. The version of Oracle is 10g release 2 (10.2) and uses the SQL3 as rule language. Oracle only uses two tables in its rule repository: Triggers and Triggers_Col, which are presented in Table 7 and Table 8 accordingly.

- **TRIGGERS**: this table has one row for each trigger defined in the database. The possible events for triggers defined in Oracle are: Data Definition Language (DDL) operations like CREATE TABLE, DML operations (insert, update or delete) and system operations like STARTUP DATABASE or LOGON USER. Triggers have a status in Oracle, which can be enabled or disabled. When a user creates a trigger, the status of this object is enabled by default. Its status can be changed to disabled using a command DISABLE and specifying the name of the trigger. A trigger with the status disabled is not fired when its correspondent event occurs. Generally, only the database administrator has privileges to change the status of a trigger. Another particular characteristic is that triggers can have two ways to express its action. The user can specify the

operations directly in the action or can call a stored procedure, where the operations to be executed are. Most of commercial ADBMSs has the same characteristics related to trigger, therefore, the rule repositories of these systems are similar. The content of the Triggers table is shown in Table 7.

Table 7. TRIGGERS system table.

Column	Description
Trigger_name	Trigger's name
Trigger_type	Activation time of the trigger (before or after) and trigger granularity (row or statement) Ex. before statement, after statement, before each row, etc.
Triggering_Event	Database event (DDL, DML or system operations).
Base_Object_Type	Object type on which the trigger is defined (ex. table, view, etc.)
Table_name	Name of the table or view on which the event is defined
Referencing_names	Names used for referencing OLD and New column values from within the trigger
When_clause	Condition of the trigger
Status	Status of the trigger (Enabled or Disabled)
Action-Type	The action type of the triggers body (CALL or PL/SQL)
Trigger-Body	Action of the trigger

- ***TRIGGER_COLS***: this table has one row for each column used in the trigger, in the event, condition or action definition. The content of this system table is shown in Table 8.

Table 8. TRIGGER_COLS system table.

Column	Description
Trigger_name	Trigger's name
Column _name	Column's name used in the trigger
Table_name	Table's name to which belongs the column used in the trigger
Column_list	Y or N (it indicates if the column is used in the trigger event *update*)

Note that the rule repository of Oracle stores the relationships between rules and metadata, but not the relationships between rules. Furthermore, it does not consider all rule types, just ECA and EA rules. Oracle allows using more variety of events than SQL3 in the trigger definition, but it does not include temporal events and events defined by user. Nevertheless, most of the relational ADBMSs incorporate some interesting characteristics like the status of triggers (enabled or disabled) and the possibility of calling a store procedure in its action element.

As for the Object Oriented ADBMSs, the rule repository of SAMOS [8] was analyzed as follows. SAMOS allows specifying events defined by user using the command CREATE EVENT and when the user wants to fire a rule associated to this event, he uses the command RAISE EVENT and specify the name of the event, that is equivalent to the concept of implicit firing presented in the rule meta-model described in Section 3.

A rule definition in the SAMOS rule language specifies an event type (primitive event or composite event), a condition, an action and the execution constraint (priority). Primitive events describe elementary occurrences, like a database operation. Composite events are specified by applying operators to so-called component events, which can be primitive or composite events as well.

The structures that compose the rule repository of SAMOS is represented in an UML class diagram in Figure 2.

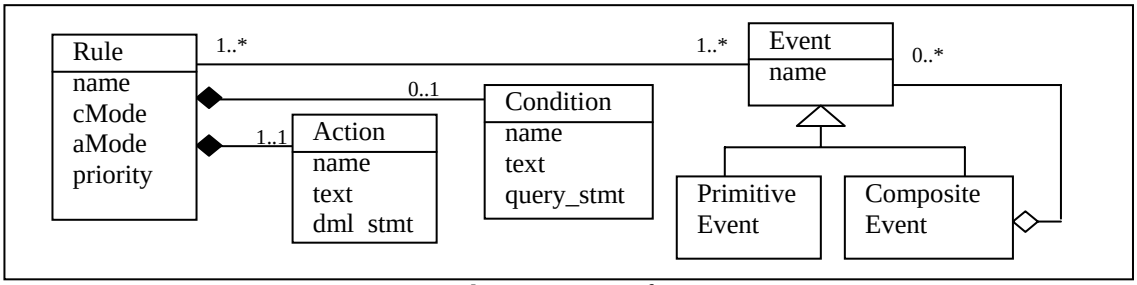

Figure 2. Rule repository of SAMOS.

The rule priority specifies the execution order of a rule in relation to other simultaneously triggered rules. The rule repository of SAMOS does not have support to store derivation rules or operational rules, only stimulus/response rules. In addition, these structures do not maintain information about relationships between rules.

The amount of semantics specified in rules depends directly on the language used. The expressive power of a rule language is related to the variety of supported constructs, like types of events, types of rules, types of firing, etc. Most of the academic ADBMSs [2, 6; 7, 8, 15] provide a rule language with greater expressive power than the commercial ADBMSs, but even so they do not have a rule language that supports all rule types represented in the rule meta-model in Section 3.

5 THE PROPOSED RULE REPOSITORY

Considering that current ADBMSs have rule repositories that support in a limited way the storing of rule information, we propose a new rule repository in order to define appropriate structures for storing all information represented in the rule meta-model presented in Figure 1. This repository is an extension of the rule repository proposed by SQL3. The proposed rule repository is represented in an Entity-Relationship diagram in Figure 3 and in the following corresponding tables.

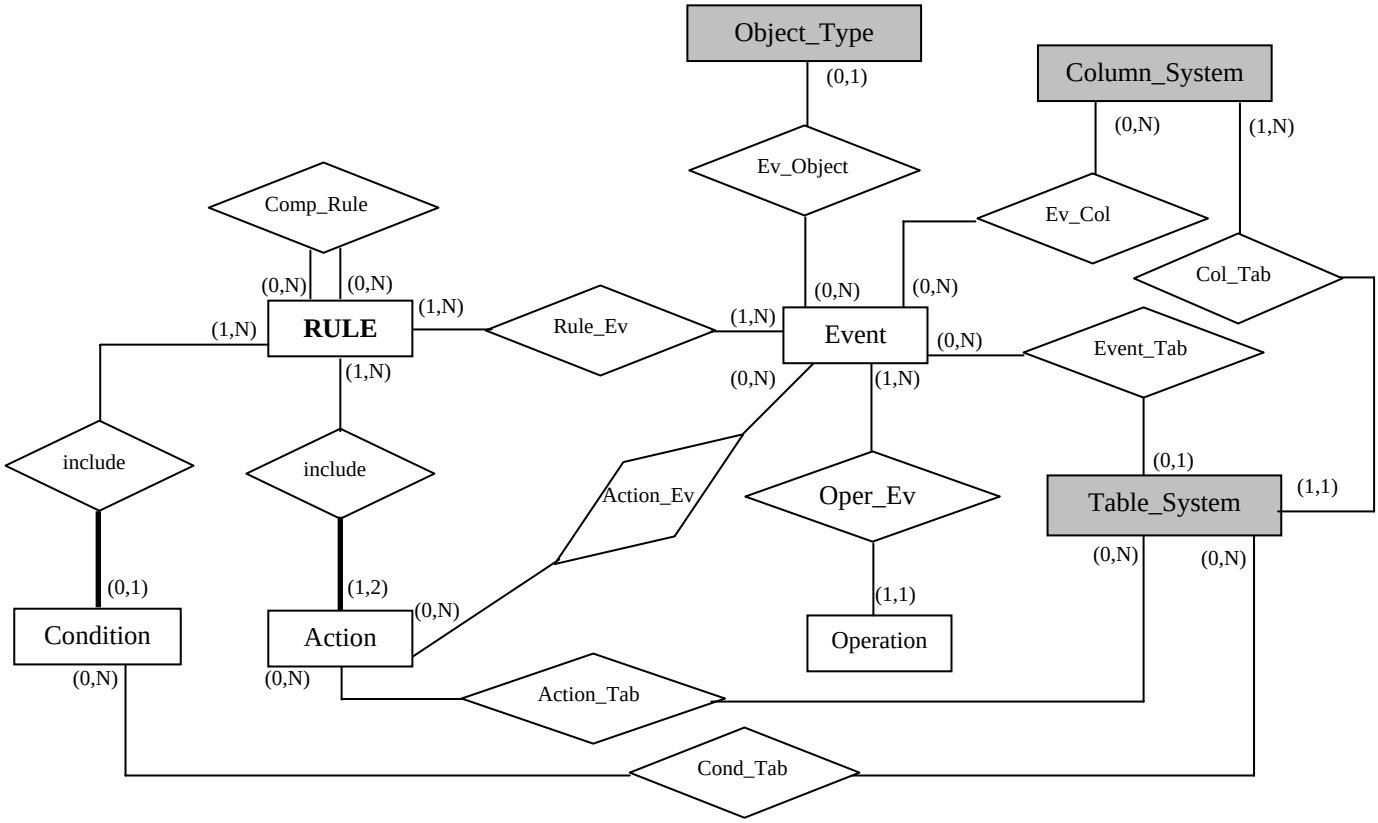

Figure 3. The Proposed Rule Repository.

The corresponding tables in relational model are:

RULE (idRule, creation_datetime, status, rule_type, granularity)
EVENT (idEvent, activation_time, *idObject*, *idTable*, *idOperation*, conector)
RULE_EV (*idRule*, *idEvent*)
CONDITION (idCondition, cond_statement)
COND_TAB (*idCondition*, *idTable*)
ACTION (*idAction*, category, action_statement)
ACTION_TAB (*action_id*, *tab_id*)
ACTION_EV (*idAction*, *idEvent*)
COMP_RULE (*idRule*, *idFiredRule*, order)
TABLE_SYSTEM (idTable, name)
COLUMN_SYSTEM (idCol, name, *idTable*)
EV_COL (*idEvent*, *idColumn*)
OBJECT_TYPE (idObject, name)
OPERATION (idOperation, description)

Obs.: The primary key of each table is underlined and the foreign keys are in italics.

The main goal of the proposed rule repository is to support the storing of the definition of all rule types described in the rule meta-model and all semantic information associated to each rule, in order to facilitate the rule management. The main table is RULE, where the definition of rules is stored. The information about the elements of rules is stored in EVENT, CONDITION and ACTION accordingly. Each rule can be fired by one or more events and each event can fire one or more rules, this relationship is represented by RULE_EV. Each rule must have an action, but the condition is an optional element.

The structures presented in grey are existent tables in the data dictionary of ADBMSs that uses SQL3, which relationship with the elements of a rule (event, condition and action) represents the relationship between rules and metadata. These relationships are stored in the following structures: EVENT_TAB, COND_TAB, ACTION_TAB and EV_COL. These tables store information about tables and/or columns referenced by rules.

The table EVENT stores information about the three types of events defined in the meta-model: database event, temporal event and event defined by user. An event can reference a table when it is a DDL or DML operation, and this information is stored in the attribute idTable in the EVENT structure. However, an event not always reference a table, it can reference a view, a database or other object type, depending on the event type. The attribute idObjectType, presented in the EVENT table, stores the identifier of this referenced object (view, database, etc.) and the information about these object type is stored in the table TYPE_OBJECT. Each event is associated to one operation, and is represented by the attribute idOperation specified in the table EVENT.

The relationship between rules is represented in two tables: ACTION_EV and COMP_RULE. The table ACTION_EV is related to those rules that have an event in its definition (stimulus/response rules) and have an implicit firing. In this case, one rule can contain an event that fires another rule in its action, and that information is stored in ACTION_EV with the purpose of identifying the relationships of this type of rules. The second table (COMP_RULE) stores information about rules that contains the explicit firing of other rules in their action element, defined by the event FIRE. The sequence of firing of rules is represented by the attribute order. This table is equivalent to the entity Rule_Composition presented in the rule meta-model in Figure 1.

This repository was checked with a rule set that represents a simple business process. This rule set includes all rule types considered in the repository and all semantic information that are necessary to evaluate in a complete way the scope and flexibility of the rule repository. Every time that some information about rules is updated in the structures of the rule repository, the metarules must be checked. Metarules are rules of integrity, which goal is to maintain the consistency of the rule repository. For example, a rule can not fire itself.

Due to the different characteristics presented in the infrastructures of ADBMSs, it could be necessary to implement some modifications to the rule repository of then in order to adapt this repository to a specific ADBMS.

6 CONCLUSIONS

The rule management has been a subject of many research studies under different perspectives, proposing several rule classifications and some solutions for the rule storage. However, the current rule repositories do not consider some main rule types used in information systems and do not store all relevant information about rules, like relationships between rules. This work proposes a rule repository as an extension of the rule repository proposed by SQL3, incorporating all the rule types presented in main rule classifications. This work uses the rule taxonomy proposed in [15, 19], which consolidates all the main rule types in a unique proposal.

The features that are considered in the proposed rule repository are: the rule types (derivation rules, stimulus/response rules and operational rules), the different event types, the two kinds of firing (implicit and explicit firing), the relationships between rules and metadata and the relationships between rules and the rule composition. When the proposed repository was tested using a rule set example, the necessity of defining a metarule set to guarantee the consistency of the rule repository was identified. Currently, we are working in the refinement of these metarules along with the definition of rule management operations.

With this work, it was obtained a relevant information set about rules, that allows raising rules to citizens of first order in databases, and as consequence, it allows treating the rules with the same importance than data in ADBMSs.

References

1. ACT-NET Consortium. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. ACM SIGMOD Record, v.25, n.3, p.40-49, Sept. 1996.
2. BERNDTSSON, M. Management of Rules in Object-Oriented Databases. Proceedings of the Baltic Workshop on National Infra-structure Databases, vol.1, p.78-85, Vilnius, 1994.
3. BONATTI, P. A; SHAHMEHRI, N.; DUMA, C.; OLMEDILLA, D.; NEJDL, W.; BALDONI, M.; BAROGLIO, C.; MARTELLI, A.; PATTI, V.; CORAGGIO, P.; ANTONIOU, G.; PEER, J.; FUCHS, N. E.; *Rule-based Policy Specification: State of the Art and Future Work*, Project deliverable D1, Working Group I2, EU NoE REWERSE, Sep. 2004.

4. BUTLERIS, R.; KAPOCIUS, K. The Business Rules Repository for Information Systems Design. *Sixth East-European Conference on Advances in Databases and Information Systems* (ADBIS 2002). Bratislava, 2002, p.64-77.

5. BUSINESS RULES GROUP. Rule Track 2000. Available in : http://www.brcommunity.com accessed in: January 7, 2005.

6. CERI, S.; COCHRANE, R. J.; WIDOM, J. Practical Applications of Triggers and Constraints: Successes and Lingering Issues. In: 26th INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASE, Cairo, 2000. Proceedings. 2000. San Francisco: Morgan Kaufmann, 2000. p.254- 262.

7. DIAZ, O; Rule Management in Object Oriented Databases: An Uniform Approach. Proceedings of the 17th International conference on Very Large Data Base, Barcelona, 1991. p.317- 327.

8. DITTRICH, K. R.; FRITSCHI, H.; GATZIU, S.; GEPPERT, A.; VADUVA, A. SAMOS in Hindsight: Experiences in Building an Active Object-Oriented DBMS. Technical Report 2000.05. Department of Computer Science, University of Zurich, 2000.

9. HERBST, H.; MYRACH. T. The Specification of Business Rules: A Comparison of Selected Methodologies. Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle IFIP. v. A-55, 1994. p29-46.

10. ISO/IEC 9075-2:1999. Information Technology – Database Languages – SQL – Part 2: Foundation (SQL/Foundation). New York: American National Standards Institute - ANSI, 1999.

11. JIN, Y., URBAN, S.D., DIETRICH, S.W., A Concurrent Rule Scheduling Algorithm for Active Rules, Source Data & Knowledge Enginnering Archive, vol.60, Issue 3, Amsterdam, Netherlands, March, 2007, p.530-546.

12. MORIARTY, T.; Business Rule Management Facility, DBPD:Data Architect *Local:* Miller Freeman Inc., Sept. 1998. Disponível em: *http://www.dbpd.com/vault/9809darc.html* acesso em 15 de jan. 2005.

13. ORACLE CORPORATION. Oracle10g Database Reference, Release 2(10.2). 2005. Disponível em http://www.oracle.com/technology/documentation/index.html Acesso em 15 de maio de 2006.

14. PATON, N. W.; DÍAZ, O.; Active Database Systems. ACM Computing Surveys, v.31, n.1,p.63-103, 1999.

15. PAVON, J.; Um Modelo de Regras para Sistemas de Bancos de Dados Ativos. 2005. 126p. Tese (Doutorado) – Departamento de Engenharia de Computação e Sistemas Digitais - Escola Politécnica. Universidade de São Paulo, São Paulo

16. PAVON, J.; VIANA, S.; CAMPOS, E.G.L.; Análise da Linguagem SQL3 com Relação à Especificação de Regras de Negócio, Conferência Latino Americana de Informática (CLEI), Santiago do Chile, Chile, 2006.

17. ROSS, R. G.; Principles of the Business Rules Approach. Addison Wesley, 2003.

18. TAVETER, K.; WAGNER, G. Agent-Oriented Business Rules: Deontic Assignments. In: International workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations (OES-SEO2001), Roma, 2001. Proceedings, Roma, Sept, 2001. p.1-10

19. VIANA, S.; PAVON, J.; JUNIOR, R. A., Rule Management in Active Database Systems, In: International Conference on Computing, 15 (CIC2006), IEEE Computer Society (IEEE-CS), cidade do México, pp.315-322. 2006.

20. VON HALLE, B. Business Rules Applied. New York: John Wiley &Sons, 2002.