An Architecture for Monitoring and Recommending Active Database DDLs

Paulo H. dos Santos[1], **Nádia P. Kozievitch**[1], **Roberto C. Betini**[1]

[1]Departamento de Informática, Universidade Federal Tecnológica do Paraná, Curitiba, PR, Brazil

phscuritiba@gmail.com, nadiap@utfpr.edu.br, betini@utfpr.edu.br

Abstract. *The integrated monitoring within heterogeneous database environments can become complex, due to the particularities in the language syntax and available tools. In particular, active databases allow developing mechanisms and automation of processes involving data or objects. This work proposes the development of an architecture to identify and monitor DDL, exploring the active databases and recommendations approach. This architecture could be explored in several ways: simple monitoring of DDL events in one or more databases, or recommendation of future DDLs or settings within the monitored database. In this context, this paper proposes: (i) an integrated architecture of active databases and recommendation; (ii) the adaptation of a recommendation algorithm, and (iii) the validation of concepts through a prototype.*

1. Introduction

Many researches and practical work related to Database Management Systems (DBMSs) have emerged, in particular the monitoring of changes within databases. The development of mechanisms for process the automation and handling of this information is also extensively explored. However, monitoring the changes within the object structures (tables, indexes, etc.) is somewhat overlooked, due to the complexity within the integration and treatment of the available DBMSs [Jun 2010].

In order to develop mechanisms that allow monitoring of such changes dynamically, Active Databases (AD) have been explored. The basic idea relies on the combination of DBMS technology programming techniques with rule-based and event-driven [Firoozy-Najafabadi and Navin 2012]. Examples include executive information systems (maintaining derived data), cancer clustering (knowledge discovery in databases) and software process management (process management and control) [Appelrath et al. 1996].

This type of system supports the definition of rules, along with system monitoring and execution of actions in response to defined rules. Basically, rules are used to monitor and respond to events that happen inside the database. Rules can reduce the amount of code that is required in applications, and can also prevent redundant code that would otherwise be required when multiple programs perform the same operations on the database. A trigger is one type of rules in an active database system which is used to detect and then react to some conditions.

For monitoring events related to object changes from the database (Data Definition Language - DDL events), the level of complexity increases, since different DBMSs

have specific characteristics within syntax. Another important and under explored context relies on the integration of AD with Recommendation Systems (RS). Here the objective is suggest changes or settings to the user, based on the history of the available system.

This paper proposes a conceptual architecture which includes: i) the development of mechanisms which enable the creation, execution, and storage of active rules; ii) the monitoring of changes made by DDL events; iii) the recommendation of possible configurations or changes in the structure (objects) of the monitored database; and iv) a preliminary evaluation. This document has the following organization: section 2 presents the related work; section 3 shows the conceptual architecture; section 4 presents the experimental evaluation; and finally section 5 presents conclusions and a discussion of future work.

2. Related Work

2.1. Active Databases

Active databases support the preceding application by moving the reactive behavior from the application (or polling mechanism) into the DBMS. Active databases are thus able to monitor and react to specific circumstances of relevance to an application. The reactive semantics is both centralized and handled in a timely manner [Paton and Díaz 1999]. An active database system must provide a knowledge model (i.e., a description mechanism) and an execution model (i.e., a runtime strategy) for supporting this reactive behavior. Nevertheless, ADs still provide a paradigm for new research approaches from combining Artificial Intelligence and DBMS technologies [Morgenstern 1983].

The reactive behavior resides on rules which integrates cause with an expected effect [Montesi and Torlone 1995]. These rules (mainly **active rules**), allow the monitoring and reaction to specific events. Such rules allow to express in a natural way, important database concepts, such as scenarios, constraints and methods. An active rule is a generalization of this schema language [Cardoso 2004]. Active rules applications support conventional DBMSs within different contexts, such as relational integrity, data maintenance, replication, monitoring data updates, etc. [Zaniolo et al. 1997].

Active rules are composed of three main components: an Event, a Condition and an Action (known as ECA rule). The basic semantics of an ECA rule can be summarized in the following sentence: when an event of type E occurs, if the condition C is true, then the action A is executed (as shown in Figure 1). The knowledge model for active rule sets defines the behavior of a set of rules at runtime [Paton and Díaz 1999], so the system monitors and reacts to certain circumstances. Within Figure 1, the symbol $\subset$ is used to indicate that the particular dimension can take on more than one of the values given, whereas $\in$ indicates a list of alternatives. The ECA knowledge model represented by [Paton and Díaz 1999] is based around a number of dimensions of active behavior, making explicit the decision space within which the designers of active rule system works.

When an DDL event (drop, alter, create) occur, and any changes are made to the database, the execution condition of the rule is verified, and if it matches the conditions set, the action is performed by recording the occurrence in a given location. Figure 2 illustrates the major steps used in processing an active set of rules: 1) the detection of the event; 2) the "Triggered Rules" box rule is triggered according to the detected event;

Event	**Source** $\subset$ {Structure Operation, Behavior Invocation, Transaction, Abstract, Exception, Clock, External} **Granularity** $\subset$ {Member, Subset, Set} **Type** $\subset$ {Primitive, Composite } **Operators** $\subset$ {or, and, seq, closure, times, not } **Consumption mode** $\subset$ {Recent, Chronicle, Cumulative, Continuous } **Role** $\in$ {Mandatory, Optional, None}
Condition	**Role** $\in$ {Mandatory, Optional, None} **Context** $\subset$ {DB_T, $Bind_E$, DB_E, DB_C}
Action	**Options** $\subset$ {Structure Operation, Behavior Invocation, Update-Rules, Abort Inform, External, Do Instead } **Context** $\subset$ {DB_T, $Bind_E$, $Bind_C$, DB_E, DB_C, DB_A}

Figure 1. Active Rule components.

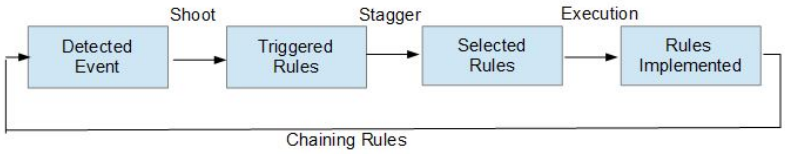


Figure 2. Steps of Event handling [Paton and Díaz 1999].

Table 1. Comparison among the available related work and this proposal: 1) [Badia 2003]; 2) [Qiao et al. 2007]; 3) [Lu and Miklau 2008]; 4) [Simon et al. 2008]; 5) [Huang et al. 2009]; 6) [Jin 2009]; 7) [Medina-Marin et al. 2009]; 8) [Jun 2010].

Characteristics \ Papers	1	2	3	4	5	6	7	8	Proposal
Active rules implemented.	+	+	+	+	+	+	+	+	+
Storage of records changes.	-	-	+	+	-	-	-	-	+
Platform independent.	+	-	+	-	+	+	+	+	+
Based on events.	+	+	+	+	+	+	+	+	+
Monitoring changing data.	+	+	+	+	-	+	+	+	-
Monitoring change in the structure, events DDL (database objects).	-	-	-	-	-	-	-	-	+

3) in the "Selected Rules" box the condition is evaluated; and finally, 4) if the condition evaluated is true, the rule is executed. Note that in Figure 2 there is a loop for selection rules - this occurs when there are more rules to be executed depending on the type of event detected.

Considering several proposals (such as [Badia 2003, Qiao et al. 2007, Lu and Miklau 2008, Simon et al. 2008, Huang et al. 2009, Jin 2009, Medina-Marin et al. 2009, Jun 2010]), Table 1 presents a comparative analysis. The first column lists the characteristics, the identifier "+" indicates that the work covers such a feature, and "-" in case it does not cover.

2.2. Recommendation Systems

Modern systems, especially those available at Internet, use mechanisms to explore the preferences of its users. Others are based largely on the recommendations of these sys-

tems when making decisions. These systems are examples of Recommendation Systems (RS) [Cazella et al. 2010].

RSs often can be considered as a particular type of personalization, learning about the needs of an individual or of a community and then proactively identifies and recommends information that meets these needs [Smeaton and Callan 2005]. These systems are based on personalization of information. This customization is related to the way in which information and services can be tailored to the specific needs of a user or a community. So, one can say that RSs are used to help users identify products or services of interest to them, within a lot of options.

A recommendation is a mapping function of interest of the customer to obtain one or more products [Pimentel and Fuks 2012]. A recommendation R for the choice "p" can be defined as:

$$R(c, p) = \max F(c, pi)$$

where:

- c represents the customer which uses the RS;
- p a set of products available to evaluation;
- $pi \in p$;
- F is a function which verifies the relevance of pi considering c.

According to [Pimentel and Fuks 2012], the RS process represented as the definition above evaluates the usefulness of a product for a particular customer. Generally, function F takes into account the similarity between the customer profiles.

3. The Conceptual Architecture

The objective of this paper is to explore the applicability of the automatic monitoring DDL events for active databases and, therefore, develop an architecture by integrating the necessary mechanisms to provide such automation. Figure 3 presents the proposed architecture. The database layer can comprise different technologies, and their DDL can be monitored through the applied rules (active rules), transformed into triggers. The triggers will represent the main mechanisms for identifying and registering how, when and who made the change or creation of an object in the database structure monitored.

The centralized layer includes AD repositories and mechanisms for recommendation generation and application of rules. The change repository stores information related to changes within the monitored database. The recommen-dation parameters repository stores information related to the parameters requested by the user when the recommen-dation is requested. The rules repository stores the active rules registered to apply to the monitored database. The syntax and semantics repository is used by the rules module, storing appropriate syntax and semantics for rules in different DBMS. The rule enforcement mechanism is responsible for applying the rules in the monitored databases. And finally, the mechanism of recomme-ndation generation is responsible for generating recommendations for a database and a chosen server.

The interface modules are set to interact with features such as: registration and application of rules; querying and recommendations; change records of view; and syntax and semantics of inclusion.

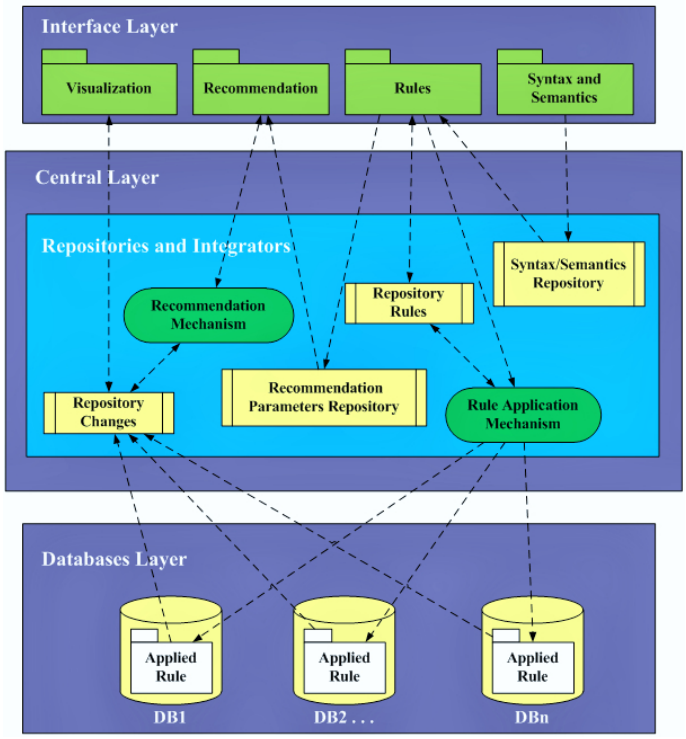


Figure 3. The conceptual architecture.

In this work we use the tuple-based recommendation approach proposed by [Eirinaki et al. 2013]. The original algorithm is part of a framework which generates recommendation queries from the current session of a user, combining the current session query with a predictive model generated from queries made by other users in past sessions. The proposal has been updated within two directions: (i) instead of data, the algorithm was adapted to work with DDL, such as tables creation; and (ii) the session summaries have been updated to each user, server and instances.

The framework consists of three components: (a) a model of session summaries, (b) a method to compute the session summaries $S_0, ..., S_n$, the construction of a session summary for each user *i* based on the **DDL commands** in Qi, (c) the computation of S^{pred}, and (d) a method to recommend queries based on S^{pred}, resulting as top ranked queries returned as the recommendation.

The first and second phase of the framework comprises the model for session summary along with the computation of the session summaries $S_0, ..., S_n$. The session summary S_i is represented as a weighted vector that corresponds to a DDL monitored. We assume that the total number of tables in the database, and as a consequence, the length of the vector is T. The weight $S_i\,[\tau]$ represents the importance of a given DDL $\tau \in T$ in session S_i, and is non zero if τ is a witness for at least one DDL in the session. The intuition is the same of the binary weighting scheme (from Eirinaki), capturing the tables in the base that are touched by the DDL in the user´s session. Hence, sessions that contain equivalent DDLs will map to the same summary. We assume that the vector S_Q represents a single DDL Q.

$$S_Q\lceil\tau\rceil = \begin{cases} 1 & if\ \tau \text{ is a witness;} \\ 0 & if \tau \text{ is not a witness.} \end{cases}$$

Given the vectors S_Q for each DDL Q posed by user i, the session summary S_i is defined as:

$$S_i = \sum_{Q \in Q_i} S_Q$$

The session summaries of past users are used to construct the $(n \times T)$ session-DDL matrix which will serve as input to the recommendation algorithm.

The third phase comprises the vector of DDL weights S^{pred}, where each weight signifies the predicted importance of the corresponding tuple for the active user. We assume the existence of a function $sim(S_i, S_j)$ that measures the similarity between two session summaries and takes value in [0,1]. The similarity function employed was the cosine similarity. In resume, two users are similar if their queries imply similar weights for the database DDLs. The S_0^{pred} summary is defined as a function of the current user´s summary S_0 and the normalized weighted sum of existing summaries, for each user, server and instance:

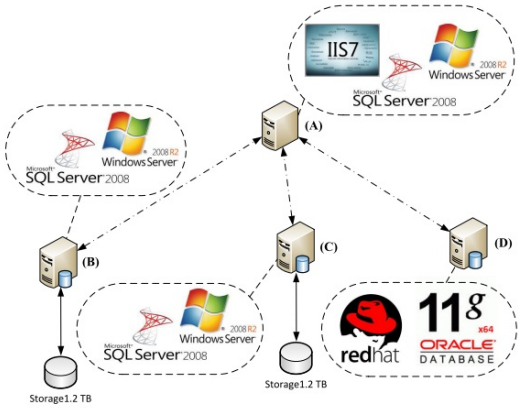
$$\mathrm{S}_0^{pred}(vuser, vserv, vinst) = \alpha * S_0 + (1-\alpha) * \frac{\sum_{1 \le i \le h} sim(S_0, S_i) \cdot S_i}{\sum_{1 \le i \le h} sim(S_0, S_i)}$$

The value of $\alpha \in [0, 1]$ determines which user´s traces will taken into consideration. If $\alpha = 0$, then it takes into account only the user´s traces. When $\alpha = 1$, only the active user´s session summary is taken into account when generation recommendations. Any value in between allow us to assign more "'importance" to either side. Within this paper, we assume $\alpha = 0,5$ to assign importance to both user´s traces. Overall, S^{pred} yields a weight per that corresponds to the importance of the DDL within the user´s exploration. Having computed S^{pred}, the algorithm indicates the top 5 ranked queries as the recommendation.

4. Experimental Evaluation

We concentrated our efforts within a tuple-based approach (as indicated by [Eirinaki et al. 2013]). The DDL commands are relaxed towards the operated table in order to increase the cardinality. The intuition is that if two users create the same table, using slightly different conditions (such as the column type as varchar for user A and integer for user B), the algorithm will consider them similar. Basically, the user creates a rule which is later derived to a specific semantic (to an oracle database, for example). This rule is translated to a trigger which is applied to a database, starting the monitoring phase. Later, the logs of monitored events are then used to create recommendations.

The rest of this section discusses the preliminary experimental results with a real database, along with examples of DDLs and generated reco-mmendations.



(A)

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
alter trigger [trg_audit_ddl] on create table
as  begin set nocount on
        declare @erro nvarchar(4000)
        declare @evento xml
        declare @instancia nvarchar(250)

        set @erro = (select error_message());
        set @evento = eventdata()
        set @instancia = convert(nvarchar(250), (select serverproperty('InstanceName')))

        if (@instancia is null) set @instancia = 'MSSQLSERVER'

        if ((is_srvrolemember('sysadmin',system_user) = 1) or
                    (system_user = 'lgFPw') or
                    (system_user = 'jira6dbuser') or
                    (system_user = 'GSFinancingUser'))
                    begin set ansi_padding on
                    set ansi_padding off end else begin
                    rollback
                    begin transaction BicBic
                    set ansi_padding on
                    insert  master.dbo.Alteracoes
                                (dataalteracao,bdusuario,login,nomehost,nomeinstancia,
                                 bancodados,identddl,nomeEsquema,tipoobjeto,nomeobjeto,
                                scriptalteracao,msgerro)
                    values  (getdate()
                                ,convert(nvarchar(250), current_user)
                                ,convert(nvarchar(250), system_user)
```

(B)

Figure 4. (A) The environment for the experimental evaluation and (B) the trigger for monitoring the create tables.

Table 2. Data Set Statistics.

Description	Server B	Server C	Server D
Database Size	5GB	5GB	2.9 GB
# Sessions	49	81	35
# DDLs	459	435	209
Avg # distinct DDLs	51	48	23
Min # distinct DDLs per session	9	5	6

4.1. Monitored Dataset and the Database for the Central Layer

Three different servers (B,C,D) (illustrated in Figure 4-A), with 3 months of historical data (Jun-Aug 2014) were monitored during the experiments by another server (A) (details are listed in Table 2). Server A presented 400GB of financial data, stored under Microsoft SQL Server 2008 R2. Server B had 1.2 TB storage, with 10 databases, comprising ERP, CRM and financial data, stored under Microsoft SQL Server 2008 R2. Server C had 1.2 TB, with 8 databases, comprising data from HR, Clients and company tickets, stored under Microsoft SQL Server 2008 R2. Server D had 1 database, comprising financial data, stored under Oracle 11G.

To store the complete information regarding the architecture and the central layer (located at server A), 6 groups of information are stored in the schema presented in Figure 5: the users which access the system, the rules, the rules applied to the SGBD instances, the semantics of the applied rule, the databases which are monitored, and the DDL changes (along with information such as user which executed the DDL, the schema used, the script for the update, etc.).

4.2. Hardware

The conceptual architecture was implemented as shown by Figure 4-A. Server A was used to implement the interface and the centralized layer. Servers B,C,D represent the monitored databases.

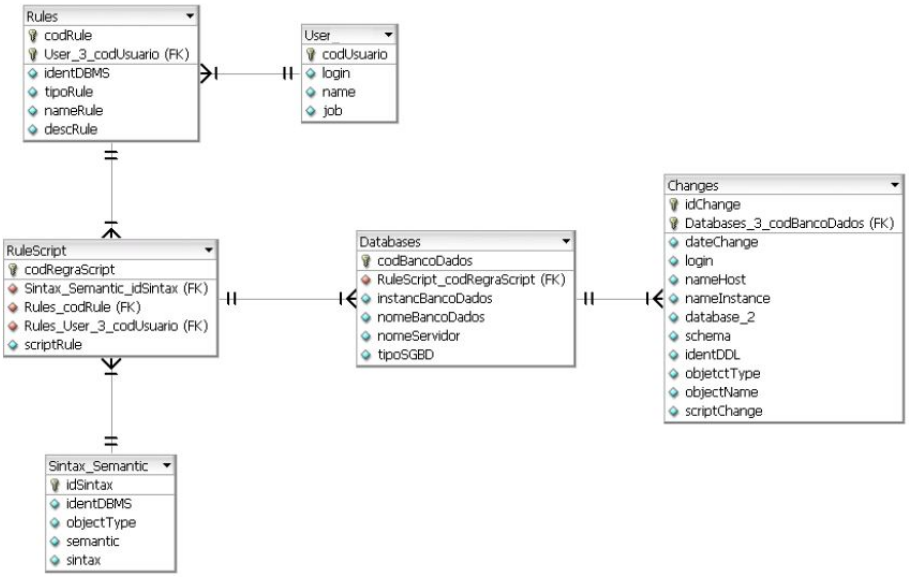


Figure 5. The database schema for storing the architecture information.

Server A (with the interface and centralized layer) was a HP ProLiant BL460c G7, with 2 processors Six-Core Intel Xeon with 2667 MHz, 16 GB of RAM, 600 GB of disk SAS, with 15 rpm in RAID 5, along with 4 network interfaces of 1 Gbps each one.

Server B was a HP ProLiant BL460c G6, with 2 processors Six-Core Intel Xeon with 2667 MHz, 72 GB of RAM, 1,2 TB of disk SAS, with 10 available at a IBM DS3512 storage, with 2 network interfaces of 1Gbps each one, along with 1 Fibre Channel interface of 8 Gbps connecting this server to the storage.

Server C was a HP ProLiant BL460c G7, with 2 processors Six-Core Intel Xeon of 2667 MHz, 96 GB of RAM, 1,2 TB of disks SAS with RAID 10 using storage IBM DS3512, with 2 network interfaces of 1 Gbps each one, and 1 interface Fibre Channel of 8 Gbps.

Server D was a HP ProLiant BL460c G6, with 1 processor Six-Core Intel Xeon of 2667 MHz, 8 GB of RAM, 600 GB of storage SAS with 15 rpm in RAID 5, and 2 network of 1 Gbps each one.

The interface used Microsoft .Net (Visual Studio 2012), located at Server A. The recommendation algorithm was implemented through a SQL procedure, and the cosine similarity was implemented by a SQL function (more details are available at [dos Santos 2014]).

4.3. Prototype Limitations

Toward a prototype implementation, we explored a simplified version of the architecture proposed within the previous section. Basically these limitations include:

- the available tests explored only the create table DDL: all the create DDL commands (from all users) from three servers (B,C,D) were monitored and logged within server A. The trigger representing the rule for monitoring the table creation is presented in Figure 4-B;

- the prototype was evaluated through two different SGBDs (SQL Server and Oracle);
- the prototype was tested only within preliminary phases;
- the available tests explored only one recommendation algorithm (DDL-Based Query Recommendation), and only one similarity function (cosine). Note that the architecture and modules could be easily extended to other recommendation algorithms and similarity functions;
- three months of historical data from a real company was monitored.

4.4. The Prototype

From the interface perspective, once the user accesses the system, four modules are available: the visualization of he monitored dataset logs, the search for recommendations, the creation of rules, and the verification of syntax and semantics.

To startup the monitored system, the user should register a rule through the interface, as shown in Figure 6-A, using information such as DBMS, event name, the semantic of the ECA events, and the body trigger for the rule (storing the information within the rules repository.

The second step includes the application of the rule to a specific server instance (Figure 6-B). This step includes the description of the server name, the database, the DBMS type and the trigger itself adapted to the database (and the respective information is stored at the syntax/semantics repository). Each instance will create a trigger which will monitor the respective database (through the rule application mechanism module).

The application of rule in Figure 6-A starts logging all the create table DDL (started by users or applications), along with the information such as the login of the user and the host, as shown in Figure 7-A. Note that within the same session, an user can create two or more tables at the same host, within the same date.

For each session, user, server and instance, a summary of accessed tables is created (phase 1), serving as input to the recommendation algorithm. Phase 2 comprises the same information is summarized by user, server and instance, as showed in Figure 7-B. Finally, phase 3 calculate S^{pred}. The 5 top ranked queries are returned as the recommendation. The interface for searching recommendations is presented within Figure 6-C.

Table 3. The top-5 ranked queries returned as recommendation.

Table	Weight
Tbl_C	1.03
Tbl_B	1.03
Tbl_D	1.00
Tbl_A	0.23

Two preliminary experiments were executed within the prototype. At the first experiment, a rule for create table was created. The objective was to monitor the production environment of a real enterprise environment, where several tables were created without DBA's permissions. Figure 8 shows the logs created by the rule. Note that two log entries (in red) were created by an application in another server (login and host names were

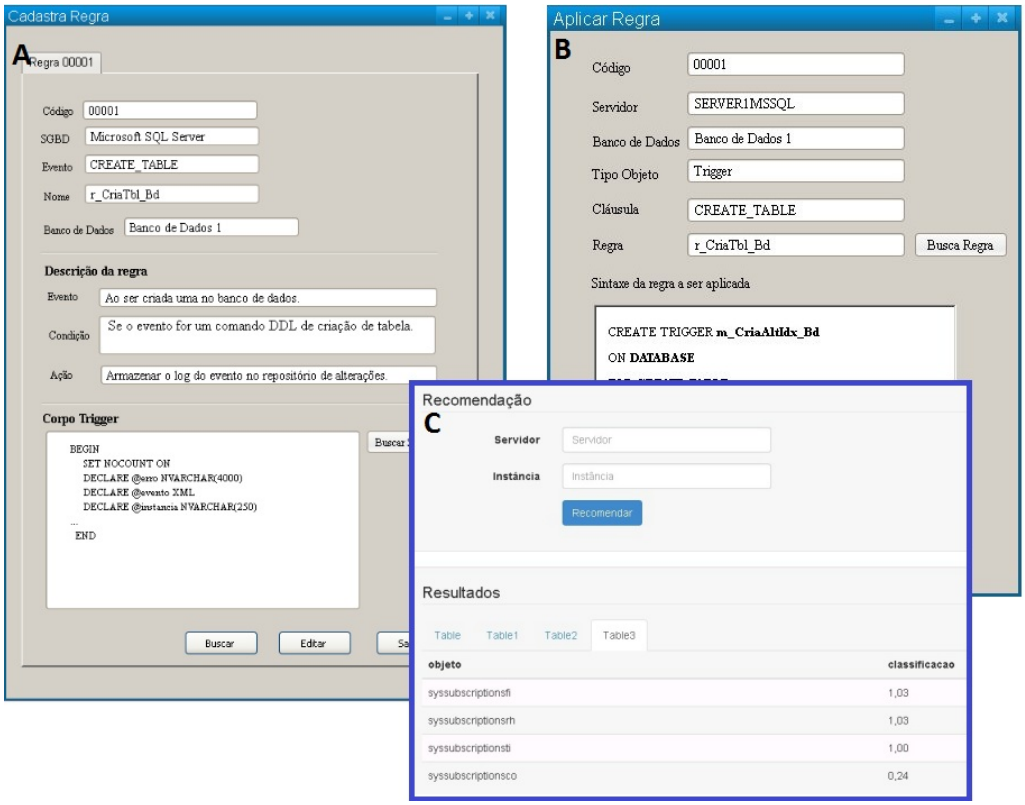


Figure 6. The interface for (A) creating the rule, (B) applying the rule to a server and (C) recommending a table creation.

Session	Date Change	User	Server	Instance	Database	Object Name
19	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_D
19	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_C
21	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_C
23	23/03/2014	DOMINIO\usuario1	SRVBDDV1	SRVBDDV1	BdSis1	Tbl_B
17	24/03/2014	DOMINIO\usuario1	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_C
22	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_B
22	23/03/2014	DOMINIO\usuario3	SRVBDDV1	SRVBDDV1	BdSis1_H	Tbl_A
20	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_D
20	22/03/2014	DOMINIO\usuario2	SRVBDDV1	SRVBDDV1	BdSis1_T	Tbl_B

(A)

			Tables				
User	Server	Instance	Tbl_A	Tbl_B	Tbl_C	Tbl_D	...
DOMINIO\usuario3	SRVBDDV1	INSTSQL	1	1	1	0	...
DOMINIO\usuario2	SRVBDDV1	INSTSQL	0	1	1	2	...
DOMINIO\usuario1	SRVBDDV1	INSTSQL	0	1	1	0	...

(B)

Figure 7. (A) The log information and (B) the log summarization.

date	login	nameHost	identDDL	...
2014-02-20 17:15:00.823	Domínio\dba1	Desktop1	CREATE_TABLE	...
2014-02-13 04:00:01.677	Domínio\serviço	ServidorExt1	CREATE_ TABLE	...
2014-02-13 04:00:01.570	Domínio\serviço	ServidorExt1	CREATE_ TABLE	...
2014-02-20 01:51:26.557	Domínio\ dba2	Desktop2	CREATE_ TABLE	...
...	...	...	...	...

Figure 8. Logs from the monitored environment.

replaced due to security restrictions). After monitoring the environment for one week, it was verified that the create table DDL triggered by an application in another server was actually created by developers, which discovered a security breach. After identified the event, the developer accesses were removed.

At the second preliminary experiment, three months of data (1103 records) were stored for analyzing the recommendation algorithm. The data between march/2014 and may/2014 resulted in 93 create table events. Exploring a specific user, instance and server, the sessions were summarized, as shown in Figure 7-B. the recommendation algorithm was processed, resulting in the recommendation (previously shown in Figure 3) of Table B and Table C as the most created at the server. Although being preliminary, the two tests states that the implementation of the architecture is viable, using a DDL-tuple based approach along with different databases (SQL server and Oracle).

5. Conclusions and Future Work

Monitoring the changes within the object structures (tables, indexes, etc.), is still overlooked, due to the complexity within the integration and treatment of the available DBMSs. This paper proposed a conceptual architecture which includes: i) the development of mechanisms which enable the creation, execution, and storage of active rules; ii) the monitoring of changes made by DDL events; and iii) the recommendation of possible configurations or changes in the structure (objects) of the monitored database. Finally an experimental prototype was presented, monitoring different databases for a three month period, using the development, testing and production environments of a real business. Preliminary tests concludes that the implementation of the architecture is viable, even using different databases along with distinct DBMS. Future work includes the precision/recall, performance and weighting scheme evaluation, the inclusion of other DDL events (such as alter table, drop table, etc.) along with the fragmented analysis of DDL commands (such as the same create table command which differs by the number of created columns). **Acknowledgments** We thank CNPq, CAPES, and Fundação Araucária for financial support.

References

Appelrath, H.-J., Behrends, H., Jasper, H., and Zukunft, O. (1996). Case studies on active database applications. In Wagner, R. and Thoma, H., editors, *Database and Expert Systems Applications*, volume 1134 of *Lecture Notes in Computer Science*, pages 69–78. Springer Berlin Heidelberg.

Badia, A. (2003). Active database systems for monitoring and surveillance. In Chen, H., Miranda, R., Zeng, D., Demchak, C., Schroeder, J., and Madhusudan, T., editors, *Intelligence and Security Informatics*, volume 2665 of *Lecture Notes in Computer Science*, pages 296–307. Springer Berlin Heidelberg.

Cardoso, V. M. (2004). Uma ferramenta para teste estrutural de regras ativas. Master's thesis, Mestrado em Engenharia Elétrica - Unicamp, São Paulo.

Cazella, S. C., Nunes, M., and Reategui, E. B. (2010). A ciência da opinião: Estado da arte em sistemas de recomendação. *XXX Congresso da SBC Jornada de Atualização da Informática, Porto Alegre, Brasil, 52p*.

dos Santos, P. H. (2014). Uma arquitetura para monitoramento de Banco de Dados e Recomendações utilizando Sistema de Banco de Dados Ativos. Master's thesis, Universidade Tecnológica Federal do Paraná, Curitiba - PR - Brasil.

Eirinaki, M., Abraham, S., Polyzotis, N., and Shaikh, N. (2013). Querie: Collaborative database exploration. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1–14.

Firoozy-Najafabadi, H.-R. and Navin, A. (2012). Rule scheduling methods in active database systems: A brief survey. In *Application of Information and Communication Technologies (AICT), 2012 6th International Conference on*, pages 1–5.

Huang, K., Chen, P., Chen, Y., Song, W., and Chen, X. (2009). The research for embedded active database based on eca rule and implementation in sqlite database. In *Database Technology and Applications, 2009 First International Workshop on*, pages 476–479.

Jin, Y. (2009). Management of composite event for active database rule scheduling. In *IEEE International Conference on Information Reuse Integration (IRI'09), 2009*, pages 300–304.

Jun, C. (2010). Active web database technology and its applications in crm. In *International Conference on Environmental Science and Information Application Technology (ESIAT), 2010*, volume 3, pages 676–678.

Lu, W. and Miklau, G. (2008). Auditguard: A system for database auditing under retention restrictions. *Proc. VLDB Endow.*, 1(2):1484–1487.

Medina-Marin, J., Perez-Lechuga, G., and Li, X. (2009). Eca rule analysis in a distributed active database. In *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, volume 2, pages 113–116.

Montesi, D. and Torlone, R. (1995). A transaction transformation approach to active rule processing. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 109–116.

Morgenstern, M. (1983). Active databases as a paradigm for enhanced computing environments. In *Proceedings of the 9th International Conference on Very Large Data Bases*, VLDB '83, pages 34–42, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Paton, N. W. and Díaz, O. (1999). Active database systems. *ACM Comput. Surv.*, 31(1):63–103.

Pimentel, M. and Fuks, H. (2012). *Sistemas Colaborativos*. Elsevier Editora Ltda. ISBN: 85-7383-260-6.

Qiao, Y., Zhong, K., Wang, H., and Li, X. (2007). Developing event-condition-action rules in real-time active database. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 511–516, New York, NY, USA. ACM.

Simon, F., Dos Santos, L. A., and Hara, S. C. (2008). Um sistema de auditoria baseado na análise de registros de log. *Escola Regional de Banco de Dados (ERBD'2008)*.

Smeaton, A. F. and Callan, J. (2005). Personalisation and recommender systems in digital libraries. *International Journal on Digital Libraries*, 5(4):299–308.

Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R., Subrahmanian, V. S., and Zicari, R. (1997). *Advanced Database Systems*. Morgan Kaufmann Publishers. ISBN: 85-7383-260-6.