2009 Fifth International Conference on Autonomic and Autonomous Systems

AutonomousDB: a Tool for Autonomic Propagation of Schema Updates in Heterogeneous Multi-Database Environments

Arlei Calazans Moraes Ana Carolina Salgado Patrícia Azevedo Tedesco

Centro de Informática – CIn
Universidade Federal de Pernambuco – UFPE
Recife, Pernambuco, Brasil
{ajcm, acs, pcart}@cin.ufpe.br

***Abstract*—One of the biggest challenges of building and maintaining applications with long life cycles, is dealing with the inevitable changes in requirements that occur over time. Many of these applications depend on DBMS that most often suffer direct consequences in their schemas due to changes in the reality that they model. In this work, we propose a new alternative to the issue of schema evolution in multi-database environments, which uses concepts of Autonomic Computing in DBMS to propagate updates in schemas replicated within the same environment, thus ensuring their consistency. Our prototype counts with a Multi-Agent System, which executes events for updating schemas in the target DBMS, which may be of different platforms. Repetitive tasks which are performed in different databases to ensure the evolution of replicated schemas are thus eliminated, freeing the DBA to do other tasks that are more important, such as analysis, database design and strategic management of data.**

Keywords-schema evolution; propagated updates; autonomy

I. INTRODUCTION

The advance of information technologies has caused systems to become increasingly complex and interconnected, thus increasing the difficulty to maintain them. This trend leads the systems to a level of complexity so high that tasks as installing, configuring, optimizing, maintaining and integrating them become hard, even for skilled professionals. Hence, current systems administration demands that some of the tasks currently performed by human administrators are delegated or assisted by automatic processes. This would avoid that the accelerated increase in complexity aligned with the diversity of technologies brought about serious problems of management [1] [2].

Computer systems evolve for different reasons, such as expansion of functionalities, modification of requirements or maintenance. Those evolutions affect different aspects of systems, since they need to reflect the new reality modeled. In its turn, this might demand additional evolutions on the database schemas in use. Structural changes on the schemas must be carried out consistently and intelligently. A simple change badly managed can cause serious problems and negative impacts. Database Management Systems (DBMS) have resources to help manage those changes, but many critical issues are still treated manually by Database Administrators (DBA), without any automated support. This increases the probability of inconsistencies on the database (data and schemas) as well as the amount of repetitive work. Some research works that accommodate schema evolution and changes propagation, can be found in the literature (e.g. [3,4]). They focus in supporting schema evolution while maintaining the traceability between the various artifacts involved in the database development process.

In this light, we propose AutonomousDB, a tool that provides autonomy on the task of schema evolution and propagation of changes. AutonomousDB works in heterogeneous multi-database environments, where there are replicated schemas. The tool is capable of autonomously propagating updates in tables of one given schema, no matter where they are replicated whithin the DBMS's environment. So, our tool tries to ensure the consistency, eliminating the need for the DBA to perform various and repetitive updates, manually, on all replicated schemas. This frees the DBA of a hard and repetitive job, saving their time and allowing them to dedicate themselves to the strategic planning of administrative data. Moreover, our tool reduces costs and avoids human failures.

This paper is organized as follows: section 2 discusses autonomic computing and its application in schema evolution; section 3 introduces our approach; section 4 describes the proposed problem about schema evolution and its features, section 5 presents AutonomousDB and details the experiments we have carried out, and section 6 presents our conclusions and suggestions for further work.

II. AUTONOMIC COMPUTING AND DATABASE SCHEMA EVOLUTION

According to Ganek [5], "Autonomic Computing is the logical evolution of past trends to resolve the growing problem of complexity of the systems and distributed computing environments". The intention is to develop applications that are able to free the user of details of operation and maintenance of systems, thus maximizing the performance in general. This idea applies perfectly to the context of DBMS, because they support a diversity of applications with the most varied characteristics and

978-0-7695-3584-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICAS.2009.14

contexts. For this reason, DBMS are quite complex tools, hard and expensive to manage, and thus needing autonomic features [6].

According to Rahm [7], "Schema Evolution is the ability to change deployed schemas". With the growing diversity and complexity of applications that DBMS have to support, Schema Evolution has become a critical issue. This is mainly due to changes in the application requirements. Schema evolution is a very costly operation, because the adjustments resulting from a change in requirements may be propagated in all levels of Database project (conceptual, logical and physical) and data instances, if necessary. As these changes are being propagated to the different project levels, the corresponding documentation must be updated. The changes propagation in data instances must be made very carefully, because an error can insert inconsistencies in the database, causing serious problems for applications that use such data.

Providing effective support to schema evolution is a challenge, taking into account that changes must be propagated correctly and efficiently for the schemas, data instances, views and others. Ideally, dealing with these changes should require the minimum of human intervention and system downtime.

Currently most tasks for schema evolution and change propagation in DBMS are performed mannually by DBA. The tools to support this process are few, cover specific aspects and are not integrated with each other. In a multi-database environment with replicated schemas, the DBA has to perform the same tasks several times, which is extremely unproductive. It is exactly this process that we propose to automate using autonomic computing, through AutonomousDB tool.

III. AUTONOMOUSDB

AutonomousDB is a software tool that supports the task of schema evolution in heterogeneous multi-database environments where there are replicated schemas. This avoids that the DBA have to perform the same manual task repeatedly on different platforms to ensure the changes.

The novelty of AutonomousDB is the capacity to automatically propagate an update performed in a certain schema wherever it is replicated within a specific environment. The environment can be composed of distributed and heterogeneous databases implemented in different DBMS (eg. Oracle, MySQL, Postgres, DB2). Another distinction is that AutonomousDB automatically generates the source code of BIU (Before Insert Update) triggers and update log packages of tables affected by the change operations. AutonomousDB uses an internal repository to store all data related to the environment configuration. Any change in the configuration is done directly in the internal repository through a web interface, avoiding updates in the application source code. AutonomousDB uses concepts of Multi-Agent Systems (MAS) [8], where a set of Intelligent Actuator Agents perform specific actions in the databases to guarantee the schema evolution. Sensor Agents are also used for the verification of new events provided by DBA. Coordinator Agents mediate the actions and comunications of all agents.

A. *Events in AutonomousDB*

For a better understanding of our work it is important to present the concept of event in the system. Any request made by the DBA for AutonomousDB to make the evolution of a particular schema is called **event**. These events are of four types: *add, delete, rename, modify* attributes of a particular table. The number of events that can be provided as input to the system is unlimited. They can also be combined to perform a certain execution plan for a schema evolution. For example, if the DBA need to create a new attribute in a schema and then delete an existing one in the same schema, they can provide the order of execution to AutonomousDB. Although the solution has been designed to treat the four events described above, the tool provides enough flexibility to be extended and include new events. AutonomousDB provides the option of event scheduling in accordance to the DBA's need.

B. *System Architecture*

The AutonomousDB's architecture (shown in Fig. 1) is composed of three layers, organized in a modular form, that isolate the functionalities of the application. This facilitates its maintenance, increasing the flexibility, scalability and extensibility. The architecture can be considered to be hybrid, because it uses concepts of object orientation in the Front End and Back End layers, and agent orientation in the Agents Layer. Below we describe the three layers.

Front End: This layer is responsible for treating all events provided by the DBA. All data related to the events are stored in an internal repository, for exclusive use by the system. This repository also stores all the information about the environment. This layer is composed by the following components: GUI, which serves as boundary between the system and the DBA, and is the point of entry of required events. The Controller makes the communication between the internal repository and the rest of the system. It executes SQL queries to answer requests originated either from the GUI or from the agents. Some request examples are: insert new events in the repository, see if a particular event is an error, check if there are scheduled events for a given time. Finally, this layer also has an Internal Repository, which stores all events, environment and application information. All communication with the internal repository is done exclusively through the Controller.

Agents Layer: This layer is responsible for executing actions that will affect the events in the target DBMS (specific schema). It is formed by a Multi-Agents System (MAS). The MAS is composed by a sensor agent, which checks the existence of new or scheduled events to be applied; a actuator agents (Adder, Deleter, Renamer and Modifier), which perform specific actions in the target DBMS according to the type of event (there is a specific agent to treat each event). Actuator Agents are scalar, i.e., they can be added to or removed from the system. These agents are capable of generating the new source code for BIU triggers and update log packages, to the table that was

affected by an event, already considering the new changes. This is done through the components Triggers Generator and Packages Generator respectively, which read the new attributes in the target DBMS data dictionary for the table affected, and generate the new corresponding source codes. Finally the MAS has a coordinator agent which manages the division of events according to the type of actuator agent, the communication between the agents, the status of applied events and the log generation.

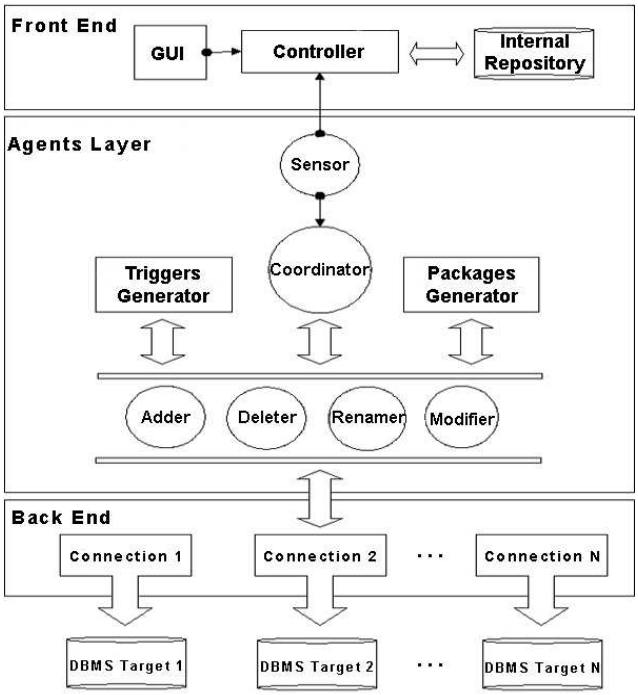

Figure 1. AutonomousDB's Architecture

For security reasons, the source code for the new triggers and packages is not applied automatically to the target DBMS, as it was done for the events. They are generated and submitted to a DBA review, that approves or not the execution. The DBA can perform adjustments in the triggers or packages source code before approving them execution in the target DBMS. This layer is based on intelligent agents, allowing AutonomousDB to perform the operations in distributed database environments. This is the feature that provides scalability and flexibility to work with several heterogeneous DBMS.

Back End: This layer is responsible for establishing the connections with the various target DBMS, which can be of different platforms, have several databases and users. This layer, more specifically through the existing connections, allows that actuator agents to carry out their actions to perform schema evolution.

In case of failure in the action application, the connection returns a code and an error message to the responsible agent. It passes the error information to the coordinator agent, which records the error in the log, in the internal repository of AutonomousDB, and suspends this event, placing it as pending for further analysis and possible reexecution by the DBA.

IV. A REAL APPLICATION OF AUTONOMOUSDB: THE CASE OF SERPRO

To validate AutonomousDB, a problem that occurs quite frequently in SERPRO[1] was chosen: the repetitive work of DBA to make schema evolution in a heterogeneous multi-database environment, where there are replicated schemas. To better understand this problem, it is important to understand first how the database environment is structured in SERPRO.

SERPRO's development environment is composed by DBMS from various platforms (i.e. Adabas, Oracle 10g, MySQL) that may run on different servers (distinct machines). Each DBMS can have more than one database and each database can have various users. The data schema for a particular application can be replicated on multiple users. These schemas are called **key schemas**, needed for other applications to work correctly in a specific user. For example, in SERPRO there is a critical application called PER/DCOMP's Load. The correct operation of 7 other applications depend on the functioning of PER/DCOMP's Load. The development team of each of the 8 applications uses 6 specific users (logins), created in two development databases, totaling 96 users. There are also databases for approval, testing and production, but here we consider only the developments databases. As all applications need PER/DCOMP's Load to work, its schema must be replicated in all users. A change in the PER/DCOMP's Load schema, causes a big workload for DBA, because they have to manually propagated the change in all users, modifying tables, triggers and update log packages 96 times to guarantee consistency. This increases significantly the probability of error and consumes much time of DBA.

Because of the size and complexity of the whole SERPRO's environment and due to security questions, we chose a part of it for our validation. This part contemplates only two DBMS, but it expresses the general problem quite well. Fig. 2 illustrates the structure of the environment. The part we have chosen is composed by two distinct DBMS: MySQL and Oracle 10g, executing on two different servers: Server 1 and Server 2. Note that the MySQL server has one available database (Database 1) with two users (Users 1 and 2), while the Oracle 10g server has two available databases (Databases 2 and 3) with one user each (Users 3 and 4). The key application data schema (Schema B highlighted in Fig. 2) is replicated on multiple users (Users 1, 2, 3 and 4). If Schema B needs to be updated, all its copies must also be updated no matter where they are in the environment. This will ensure the consistency of schemas for different users. In addition, all BIU triggers and update log packages that refer to the updated table, also need to be updated. In SERPRO, this task is done manually by the DBA. Many times a schema in a specific user can not be updated immediately,

[1] SERPRO is the Brazilian Federal Data Processing Sercices, a public enterprise from the Brazilian government.

which leads the DBA to postpone the change in this case. As there is not an adopted tool to automate the control of postponed changes, the DBA may forget to apply the changes in correct time, leaving schemas inconsistent.

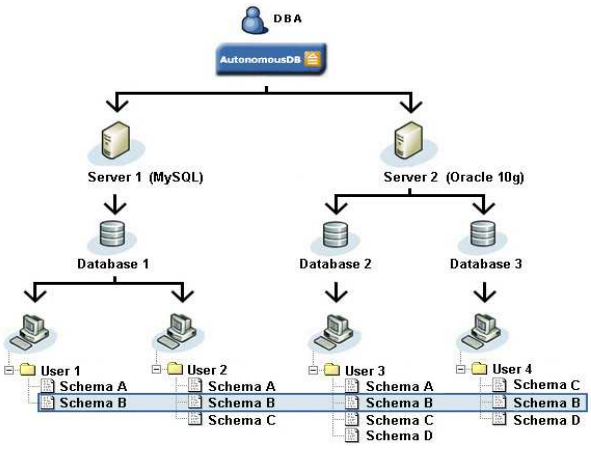

Figure 2. An excerpt of SERPRO's Environment

The difference between DBMS platforms also makes the work difficult, because the DBA needs to know the physical schema of all involved databases. AutonomousDB proposes to solve the problems described, freeing the DBA of repetitive tasks and ensuring the consistency of all schemas in the environment.

V. IMPLEMENTATION ISSUES

For the implementation of AutonomousDB, the main requirements chosen, among several raised from SERPRO's DBA, were: i) create, delete, rename and change attribute types; ii) generate source code for BIU triggers and update log packages; iii) schedule events; and iv) generate reports (logs). All these requirements were implemented in the first prototype version.

Since its conception, there was a concern to develop AutonomousDB as a multi-platform and scalable system. The Java [9] language was chosen for implementation, because it provides resources for object orientation, which are important for modularity, scalability and extensibility. Java also supports a variety of frameworks for developing and has drivers available for connection with various DBMS.

In the Agents Layer, to support the development of intelligent agents and management of communication between them, we used JADE (Java Agent Development Framework) [10], a software framework fully implemented in Java. It simplifies the implementation of multi-agent systems through its middleware and a set of graphical tools that supports the debugging and deployment phases. JADE gives all support to the agents management in distributed and heterogeneous environments, which is completely aligned with the objectives of AutonomousDB.

For the system's internal repository, we used Oracle 10g Express Edition, which provides an intuitive and easy to operate web interface. Additionaly, Oracle 10g has high performance and other features that facilitate its administration. The internal repository stores the events provided by the DBA, scheduled records for event execution, informations to generate error log, as well as functional data for intelligent agents and environment. This facilitates all types of change, either in the environment or in the agents. In fact, it is necessary only to update the internal repository and the next time the system is started, the changes are already in effect. The target DBMS for the implementation of AutonomousDB and preliminary tests were Oracle 10g and MySQL. The tool is an external module to the target DBMS working as a virtual DBA, performing many tasks that were previously performed by human DBA (Fig. 2).

Four SERPRO's DBA, of a team of 11 professionals, tested AutonomousDB. They used Windows XP Professional, in the environment shown in Fig. 2. Each DBA executed two requested events of a same type, but in distincts schemas, as shown in TABLE I. The schemas A, B, C and D represent schemas of real applications, target of frequent changes. The 4 SERPRO's DBA performed also scheduled execution of events and consulted reports, triggers and packages generated. All the 8 events provided as input to the system correspond to tasks that would be repeated by a DBA, as many times the schema to be changed is replicated. Using Autonomous DB, the DBA only needs to inform the operation, some input data and the schema that will change. All update propagation is done automatically.

A. *Evaluating AutonomousDB*

After the execution of the experiment with SERPRO's DBA, let's now analyze and describe the main results. We verified that all events data provided as input to the system were correctly stored in the internal repository. The events also were executed successfully to the target DBMS, taking into account the propagation of updates in replicated schemas. It is important to emphasize that one event provided by a DBA, does not necessarily correspond to one event executed by the system. For example, an event of creating an attribute in Schema B (see Fig. 2) provided by the DBA, is equivalent to four events executed to the target DBMS, taking into account the replication of this schema on the environment (for all users). TABLE I shows the number of events provided by each DBA and the effective events executed in the target DBMS for the executed experiment.

TABLE I. EXECUTED EVENTS

DBA	Event Name	Requested Events	Executed Events
1	Create	1 (Schema A) 1 (Schema B)	3 (Schema A) 4 (Schema B)
2	Delete	1 (Schema B) 1 (Schema C)	4 (Schema B) 3 (Schema C)
3	Rename	1 (Schema C) 1 (Schema D)	3 (Schema A) 2 (Schema B)
4	Change Type	1 (Schema A) 1 (Schema D)	3 (Schema A) 2 (Schema B)
Total		***8 Events***	***24 Events***

In TABLE I we can see that if the DBA would perform all the requested events manually, as it is currently done in SERPRO, they would have to execute the following actions **24 times**: i) log in; ii) apply DDL commands (to create, delete, rename or change the type of an attribute) in object tables; iii) update the BIU triggers and update log packages, iv) control what was updated or not. This same 8 requested events being executed for whole SERPRO's development environment (described in Section 4) would require the DBA to execute **528 times** the actions described previously. This occurs because the schema of application corresponding to the schema A is replicated 84 times, the schema B 96, the schema C 72 and the schema D 12 times. We can conclude that for a large quantity of actions and BD's users, it is quite unproductive. With AutonomousDB, the DBA only needs to inform the data of the requested event to the GUI, all the rest of the process is automatically done.

The scheduling of events was also implemented successfully. Some events were scheduled to execute at a specific time and this was successfully done. Ocurred failures were treated correctly, the coordinator agent received the error information and recorded it in the log. The whole process of schema evolution and autonomic update propagation, and the artifacts generated by the system, were reviewed and validated. All SERPRO's DBA team (11 professionals) is experts in Oracle 10g, 5 in MySQL, 4 in SQL Server and 1 in Adabas. There are DBA with more than one specialty. The experience average of team is 6 years as SGBD specialist.

To evaluate our tool, a questionnaire containing 3 questions about the potential contributions of AutonomousDB to the DBA's work process was executed. The questions were: *i) Did this tool enhance your daily work? ii) Do you think you will gain time using the tool? iii) Would you use the tool?* The questionnaire was answered and sent by e-mail. Analyzing the responses, the issue of reducing considerably the repetitive work to evolve the schemas; ensuring consistency of schemas, since the entire process is done in an automated way; reducing the amount of human failures and gaining of free time to devote to other activities more important than operational were brought up by all. These points were proposed initially as main goals of our tool. All the DBA also answered that they had a gain of time in their daily work and that they would use the tool.

As strengths of AutonomousDB, the DBA also cited: the ease of use (usability), autonomy, ability to work in heterogeneous multi-databases environments, generation of execution log, suspension of events in error and generation of source code. The leader of the DBA team suggested the implementation of the functionality of *apply scripts* for specific schemas. This functionality would make the tool more adherent to the needs of the team. AutonomousDB is being extended and tested to be applied as SERPRO's official tool.

VI. CONCLUSIONS AND FURTHER WORK

The main goal of this work is to build a tool to autonomously propagate updates in replicated schemas in heterogeneous multi-database environments. The idea is to reduce the workload of DBA, avoiding repetitive tasks and human failures. In this sense, we proposed AutonomousDB, a tool that uses concepts of Autonomic Computing applied to DBMS, to help the DBA in the task of schema evolution and update propagation.

In AutonomousDB, we emphasize the capacity to propagate schema updates and generate source code of new BIU triggers and update log packages affected by the schema evolution. The prototype was implemented using Java. To the best of our knowledge there is no tool with the same functionalities, features and the same techniques used in AutonomousDB. The use of an architecture based on intelligent agents, brings autonomy and capacity of distribution to the system. The system was validated in an environment with two different DBMS (Oracle 10g and Mysql). Its first evaluation showed promising results, indicating that this approach is very interesting for DBA.

The next steps of research and development are related to the extension of the tool to support more platforms (i.e. PostGres, DB2, SQL Server), to develop a generator of Triggers and Packages more generic and able to change characteristics of physical schema for each platform (through XML files, for example). This would bring also more independence of the tool with respect to diverse platforms. Another extension is to implement the functionality of scripts to run in particular users. The architecture is already prepared to support new functionalities and efforts in this direction are already being done.

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing". IEEE Computer Society, Vol. 36, Issue 1, pp. 41-50, jan 2003.

[2] Paul Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology". Technical Report, IBM Corporation, October 15, 2001.

[3] Hick, J. M. & Hainaut, J. L."Strategy for Database Application Evolution: the DB-MAIN Approach". University of Namur, Computer Sciences Department, 2002.

[4] Dominguez, E., Lloret, J., Rubio, A. L. & Zapata, M. A. "MeDEA: A database evolution architecture with traceability". In Proceedings of Data Knowledge Engineering, p.419-441, 2008.

[5] A. G. Ganek and T. A. Corbi, "The drawing of the autonomic computing era". In IBM Systems Journal, Vol. 42, No. 1, pp. 5-18, 2003.

[6] S. Elnaffar, W. Powley, D. Benoit and P. Martin, "Today's DBMSs: How Autonomic are they?". Technical Report, Queen's University, School of Computing, 2003.

[7] E. Rahm and P.A. Bernstein "An on-line bibliography on schema evolution". ACM SIGMOD Record, Vol. 35, No. 4, p.30-31, December 2006.

[8] S. J. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach”. Prentice Hall, 2nd Edition, 2003.

[9] Java, “The Source for Java Technology”. Available in: http://java.sun.com. Last Access: 10 out. 2008.

[10] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa and R. Mungenast “JADE Administrator’s Guide”. Available in: http://jade.tilab.com/doc/. Last Access: 5 set. 2008.