GENERATING RESTRICTION RULES AUTOMATICALLY WITH AN INFORMATION SYSTEM

M.Sc. Martha Beatriz Boggiano Castillo, Lic. Alaín Pérez Alonso, M.Sc. María Elena Martínez del Busto, Dr. Ramiro Pérez Vázquez, Dra. Luisa González González

mbeatriz@uclv.edu.cu;apa@uclv.edu.cu; mmbusto@uclv.edu.cu; rperez@uclv.edu.cu

Universidad Central "Marta Abreu de las Villas". Cuba

ABSTRACT

Information systems design problems due to informal treatment of requirements have led to consider the way requirements are handled as essential factors, currently assembled as business rules. The purpose of this work is to generate restriction rules automatically with an information system, specifically those that may reside in the database, gaining flexibility, transparency and cost reduction in business. This paper also describes some benefits of using business rules. A pattern was used for rules, analyzing resources that made possible their implementation in a database and determine their form of storage. This pattern of rules was modified, adapting it to new rule requirements. Database resources were chosen to implement them, and a compiler that generates this code was built.

KEYWORDS Business rules, automatic code generation, restriction rules

RESUMEN.

Los problemas en la concepción de Sistemas de Información, debido al tratamiento informal de los requerimientos, han permitido considerar como factor de esencial importancia la forma en que se manejan los requisitos, enfocados actualmente como reglas de negocio.
El propósito de este trabajo es generar automáticamente las reglas de restricción en un sistema de información, específicamente aquellas que puedan residir en la base de datos ganando con esto agilidad, transparencia y reducción del costo en un negocio. Se describen algunos beneficios de usar reglas de negocio. Se usa un patrón para estas reglas, analizando los recursos que posibiliten su implementación en una base de datos y determinar su forma de almacenamiento. Se modifica un patrón de reglas adaptándolo a las nuevas exigencias de las reglas, se eligen los recursos de bases de datos para implementarlas, y se construye un compilador que genere este código

PALABRAS CLAVES
Reglas de Negocio, generar código automáticamente, reglas de negocio de restricción

1. INTRODUCTION.

Nowadays, many information system specialists consider that the starting point of requirement identification is business rule design. These rules can be considered as sentences that allow the expert users to define policies, knowledge and conditions of the business in small, isolated units. Part of the requirements, that are currently focused as business rules, have been traditionally implemented by the developer by hard-coding them in the source code of the program, in database mechanisms or both.

By automating this process, failures are avoided and in this way time and cost are significantly reduced. In order to automate these "requirements" a system is needed to give them functionality so that they are implemented in a given language and stored for safe handling.

Current studies tend to store the rules in a repository where they are handled by an independent engine rule. Therefore, restrictions are not codified in the system and this can only make necessary accesses to the repository. The idea of business rules, as a definition of the desired policy, would provide a solution for its possible treatment in databases and the benefits it provides.

There are several potential advantages of using Business Rules. Of all, according to Lowenthal (Lowenthal, 2005), the three most important are:

- Agility: simple and rapid response to the dynamic requirements
- Cost Reduction: low cost to create or update the parts of applications that implement the business. Business policies
- Transparency: the rules easily allow the auditing that the software services carried out in their respective business policies

2. IMPLEMENTATION OF BUSINESS RULES

There are many types of business rules. Due to their complexity and diversity, authors tend to group classify them according to different points of view. One of these classifications is was given by Solivares, Lowenthal, Morgan (Perez, Boggiano 2007). These classifications were analyzed and Morgan’s classification was finally adopted in this paper since it takes into account a more elaborate classification of rule patterns, and therefore it was considered the most suitable of all.

According to Tony Morgan (Morgan, 2002), the most convenient way to create rule sentences is by selecting appropriate patters from a short available list. For example: <sentence>: <*Subject*> *should* < *restriction*>. Morgan adds that there is no standard about how to do business rules but it is possible to give some recommendations. The pattern rules could reflect the forms and types of problems that an automated system intends to negotiate. Morgan provides several kinds of rules such as Restriction, Classification, Calculation and Numbering. Of all these, the present paper focuses on the restriction type.

The restriction rules are those restricting the data in the system and they may overlap with the rules of the data model, since those also prevent the introduction of erroneous data. The difference lies on the fact that the restriction rules determine the value attributes or properties of an entity beyond the basic restrictions.

The rules can be implemented in different ways and even there could be different techniques for the same rule. Morgan (Morgan, 2002) shows some of them: Programming Languages, Scripts, Database. We have found that many of the business rules fit more naturally within the database, where they can have more direct contact with the data (Mota, 2005). The technology types most commonly used are Oracle, SQL

Server and Postgres. Other correlative database products have widely similar capabilities, so it is not difficult to apply this research to other languages.

In the case of SQL Server, programming can be made in a dialect called Transact-SQL. The most usual procedure is to use business rules in respect to functional words around the basics (CREATE, READ, UPDATE and DELETE). These mechanisms are common to all data services: restrictions (Constraints) CHECK, trigger, views, user-defined functions and assertions. Here we are working with triggers and functions in order to implement the restrictions rules on the business database.

The pattern suggested by Morgan for the basic restricting rule is the following:

<determinant> <subject> [no] (must | have) <characteristics> [(If | unless) <fact>].

This is the most common pattern for business rules; it provides a restriction on the subject of a rule. It was decided to make changes for improving it and adapting it to the needs so as to obtain tangible effects. The resulting pattern of restriction is given below (holding its conventions):

<determinant> <subject> (can not have <characteristics>) |(<characteristics> can have <characteristics> only if <facts>).

Element	**Meaning**
<determinant>	It is the determinant for each subject, for example: One, Two, She, He, Every, All, according to the best meaning of the written text
<subject>	It is an element of the business database such as entities and objects. The entity can be qualified by other attributes descriptors, such as the existence in a particular state or linked to a specific application of the rule.
<characteristics>	It describes the characteristics of the subject in the business, both internal and others related with other entities.
<facts>	They are facts related to the state or behavior of the business database, including or not including the subject

Table.1 Elements of Pattern.

The conventions used for the patterns in this work are as follows:

- Parentheses () They hold a group of items.
- Brackets [] They enclose optional elements.
- Vertical Bar | It separates alternative terms.
- Angle brackets <> They hold special terms as those shown in the table below.
-

2.1. A case of basic restriction and list of restrictions.

We present a rule of business for a kidney transplant.
Natural Language:

A patient can not take potential donors with more than 4 physical exams or have in his evolution a higher maximum temperature of 42 degrees.

To achieve simple rules and easy to maintain, trying to separate them into smaller rules as possible, it is Natural Language:

A patient can not take potential donors with more than 4 Physical Exams.

A patient can not have a maximum temperature of 42 degrees.

This way of expressing this rule very close to the customer is called natural language, taking into account the pattern adopted. In order to achieve an SQL implementation, an intermediate representation is considered (called the technical language); to represent the rule in technical language, the representation of the rule is used, where the navigational point is used (Morgan, 2002). This style is used in the technical language to access the attributes of the entities and browse the relationships between them.

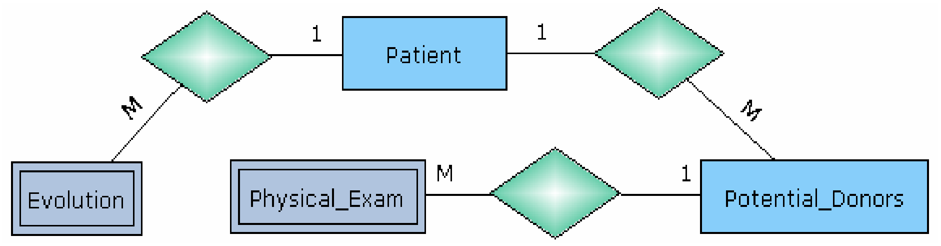
Simple access to an attribute:

Entity_1.Attribute.

Browse between entities relationships:

Entity_1.Entity_2. (...). Entity_n.Attribute .

For example, if one needs to access the patient's name, it can be reached by writing Patient.Name. However, it is also possible to browse it in the relationships of the following schedule:



Ilustration 1 Portion of E / R Schedule for Kidney Transplant

When using the notation point navigation to establish the relationships between entities and accessing their attributes, two main types of results could be obtained, due to the cardinality between the business objects. This is because it can be specified either as a single member of the business or referred to a set of them and therefore it is necessary to discern between both cases.

Multiple elements: By stating "The Patient's evolutions ..." (Patient.Evolution) several evolutions related to a unique patient are designed as shown in the cardinality of the diagram. Note again that the order of the notation is very important.

An expression of a rule in technical language would be:

A patient can not take sizeof (Evolucion.idEvolucion)> 30

<Subject>: Patient

<Characteristics>: Sizeof (Evolucion.idEvolucion)> 30

In natural language the corresponding expression to this rule would be:
A patient can not have more than 30 evolutions.

2.2 The Triggers and functions in implementing the rule.

It is significant to reach a consensus on when you should use the various resources of SQL99, according to the peculiarities of the rule and how such restrictions of the business will be implemented. This will provide a tool to select the form of generating the rule.

Applying rules of restriction patterns in a database for kidney transplant.
For Example 2
A nephrologist can assist more than 12 patients only if the number of patients is greater than 50.
With technical language:
<Subject> Patient
<characteristics> sizeof(Patient.idPatient)>12
<fact>: sizeof(Patient.idPatient)>50

The code generator will produce in the SQL Server database of the business of kidney transplant the following function and trigger:

The function:

```
CREATE FUNCTION FRN#0 (@idDoctor Integer) RETURNS BIT AS
BEGIN
DECLARE @result BIT
SET @result = 0
IF ( (SELECT COUNT (b.idPatient)
    FROM Nephrologist a, Patient b
    WHERE (@idDoctor = a.idNephrologist)
    AND (a.idNephrologist = b.idNephrologist)) > 12)
 IF NOT ( (SELECT COUNT (a.idPatient)
    FROM Patient a) > 50)
    SET @result = 1
RETURN @result
END
```

The trigger:

```
CREATE Trigger TRN#00 ON Patient AFTER INSERT, UPDATE AS
DECLARE @subject INT
SET @subject = (SELECT a.idNephrologist
    FROM Nephrologist a, Inserted b
    WHERE (a.idNephrologist = b.idNephrologist))
IF dbo.FRN#0(@subject) = 1
BEGIN
 RAISERROR('RN#0',16,1);
 ROLLBACK TRANSACTION;
END
```

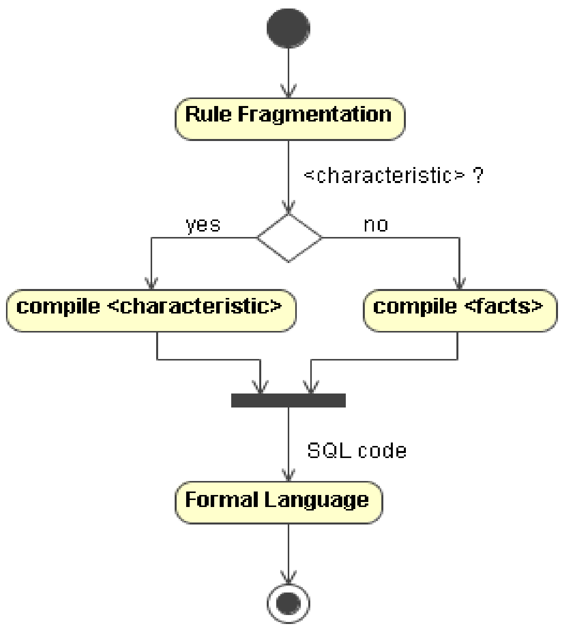
These triggers, as can be noted, state a @subject that is the conditioned <subject> to the rule, which is extracted through a query defined by the point notation. Afterwards, in a conditional sentence it is questioned whether the rule is shattered or not to the specified subject on whom a relational modification was done; if so, then suitable measures are adopted.

The function is the body of the rule. Triggers will refer to a function. This is the one responsible to perform the body of the rule where it is verified if the subject breaks the conditions imposed by the restriction. Its objective is to reply if any of the prohibitions is infringed by using a Boolean value. The function FRN#0 is called from the trigger.

2.3 Implementation, operators and functions.

The process of compiling business rules, restriction type, used in the current version of an editor of rules in technical language is performed since the first moment you execute. During this process part of the resulting code is being directly generated and the rest is formalized in the corresponding application.

This compiler is composed of two other compilers, one of these is engaged in compiling the <characteristics>, and the other is engaged in the <facts>. The compilation process is shown in this activity diagram:



llustration 2. Diagram in the Collection Activity or translation.

The compiler was made in the software Parser Generator v1.12 with Lex and Yacc for Delphi. Patterns of coincidence in codes are created With Lex. The patterns are specified using regular expressions, each of these has a corresponding action associated to them. When a string that matches the regular expression is found, the corresponding action is done.

With Yacc, given the BNF (Backus Normal Form) of grammar specification, the corresponding parser can be generated. Yacc does not accept every grammar presented; however, the kind of grammar that it accepts is generally powerful and enough for the needs of programming (Bumble-Bee, 2000).

In Lex it is mainly defined the data types, the operators and the functions that are accepted in the technical language; they are useful for shaping the grammar of the rule, restriction type. Only triggers and functions are generated. Efforts are being made to improve this compiler, to represent CHECK restrictions, to do a rigorous check type and to evaluate the repercussion of the business rule transformation in the business database.

Currently, the software is being validated to implement a set of business rules to the problem of kidney transplant in an SQL Server database and work is being done on a version of the MySQL 5.x software database

3. CONCLUSION

This article has addressed the need to deal with the requirements of the system using the approach of business rules, showing their dominance in the process of managing the organization in order to have a potential for a drastic reduction in the number of errors. Business rules, classified as restriction types, are formalized in a pattern of rules of business, based on the pattern proposed by Morgan. A program that compiles these rules was built and used starting from entries of the rules in technical language. Using the navigation point in which the rule entries in the pattern were made considerations were adopted for each part of this pattern and a trigger was automatically created for insertion and update which implements a restriction rule to a information system SQL Server 2000 database. Also examples are presented for a database of kidney transplants. This compiler will be valid for any business using a business database in SQL Server 2000. Automation to generate business rules helps to avoid failures, achieving significant reductions of time and cost.

REFERENCES

REFERENCES

Book

MORGAN, T. (2002) Business Rules and Information Systems: aligning IT with Business Goals. Addison Wesley.Chapter 3. Defining Business, 2002.

DEMUTH, B., HUSSMANN, H. & LOECHER, S. (2001) OCL as a Specification Language for Business Rules in Database Applications. UML'01 4th Intl. Conf Unified Modeling Language,. Toronto, Ontario, Canada, October 2001.

Journal

MOTA, S. A. (2005) Active Databases.

BESEMBEL, I. M. & CHACÓN, E. Objetos y reglas de negocios en la integración y automatización de procesos de producción continua. Universidad de Los Andes. GIDyC. Departamento de Computación. pags 12. Mérida. Venezuela. 5101. Electronic Article.

PEREZ, BOGGIANO (2007) Aplicación para reglas de restricción en negocios. Universidad Central de Las Villas. Facultad de Ciencias de la Computación .Trabajo de Diploma. Año 2008. p. 18-21

BUMBLE-BEE (2000) Parser Generator 1.12 ed. Software. http://*www.bumblebeesoftware.com/release.doc*