International Journal of Latest Research in Science and Technology **ISSN (Online):2278-5299**
Volume 2, Issue 5: Page No.1 -8 ,September-October 2013
https://www.mnkpublication.com/journal/ijlrst/index.php

FILTERING THE INPUT DATA BY PRODUCTION RULES ON OPEN DATABASE

[1]Tonahtiu Ramírez, [2]Miguel Patiño, [3]Benina Velázquez
[1]Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, IPN, D.F. México
[2] Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, IPN, D.F. México
[3] Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, IPN, D.F. México

Abstract- **This paper presents a proposal for filtering input data to the system through a comparison of the data entered for each of the conditions or rules, to be accepted. We propose to use a system for performing this process relying on a database manager where they will be stored rules to assess.**

Keywords - Data bases, Rules, Systems

INTRODUCTION

It has long been working on the filtering of the input data to various computer systems; this will have raised many solutions that have evolved over the years, derived from the global technological development. At the same time the organizations have become more dynamic, forcing updated with the same speed. Therefore software systems have to be developed to become more flexible and can update more easily in operating rules of organization.

Although much depends on the country and organization, computer systems that receive data, filter the received differently, but many, perhaps even most entry conditions integrated directly in the program code, and for certain types of applications is good because these conditions often change little, but in places where such conditions are constantly changing them less competitive, because it is necessary to integrate code changes and often involves the recompilation of the same, it is also necessary turn the changes to software development area into the organization or request changes to the developer company and in both cases are usually slow.Therefore in this paper proposes that the filtering is execute by a computer program to read a list of rules**[5]** containing the conditions to meet in order to enter the data, similar to a traditional expert system[8] in order to give a greater degree of efficiency is proposed that the filtering rules are in a table inside a relational database, at this proposes the use of a database manager such as MySQL 5.

The purpose of doing this is so that another application is possible to make direct changes to the database in terms of rules and apply the change or insert a new rule immediately applicable to the system. The proposal is detailed in the next section.

DEVELOPMENT

In order to follow a product development methodology is the subject of this article the following steps:

1. Analysis (state of the art, processing algorithms [10] proposed)
2. Design (database, design subsystems)
3. Development (Implementation of proposed implementation)
4. Implementation (They begin pilot testing, after running in normal operating conditions)
5. Settings (Derived testing adjustments are made)
6. New developments (I have seen new possibilities to be implemented)

For practical purposes in this article will show part of the analysis and design as representative. It began with a comprehensive explanation and then the details.

By mental map is presented in Figure 1 shows the overall view of the filtering system and the modules involved.

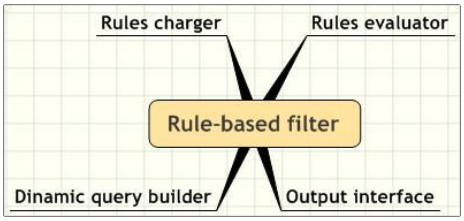

Figure 1. General diagram of the filter system

From Figure 1 it can be seen that the rule-based filter comprises four main modules being the rule evaluator and auxiliary rules charger, Dynamic query builder[9], and standard interface output for testing.

Below briefly describes the function of the auxiliary modules of each in order to have a global vision, and after explain in detail the rules evaluator, the purpose of this article

Rules charger has the function of add rules to the database, while maintaining his integrity, and every time you enter a rule this will be reviewed in terms of syntax, there are elements to run as well as it can be evaluated,

Engine Query Builder will generate valid code dynamic SQL which will be transferred to the database manager.

Publication History
Manuscript Received : 23 October 2013
Manuscript Accepted : 27 October 2013
Revision Received : 28 October 2013
Manuscript Published : 31 October 2013

Output Interface relates to see the results of the tests.

Now proceed to explain Rules evaluator. Before explaining the details of the rule evaluator will display the context where acts, for this purpose see figure 2.

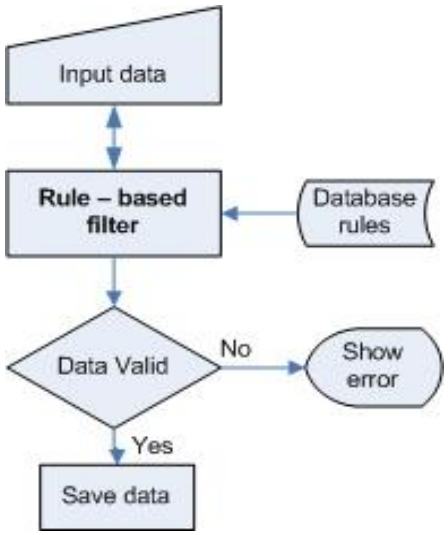

Fig. 2 Location scheme rules evaluator

Figure 2 shows in bold the rule-based filter or rules evaluator for the element to be present, First the user enters a data that is received and transmitted to the filter, this determines the nature of the application and database query data rules to see what the rule to apply, and depending on the result, if true, save it or if not, indicates that failed.

The following explains the operation of the filter based on rules stored in a relational database.

Usually for entering data into a system, you must meet conditions and as already mentioned these are defined somewhere in the system, these conditions usually are of the type:

1. Comparisons of the type $a < b$; $a > b$; $a<=b$; $a>=b$ or $a=b$. such that a variable or constant, and b constant if a is variable or b variable if a constant.
2. Verification of existence of x data in a database
3. Comparison of a real number with the generated result of the application of a function of the group in SQL database as for example: $cuenta(x) < b$; $suma(x) > b$
4. Comparison between data entered with the result stirred by a database query

Surely to apply this filter to various systems exist in quantity a variable number of rules, and with different degrees of complexity, but the vast majority can be summarized in such comparisons <; >; <=; >=; = , order to obtain a true or false and thus easily determine whether to continue or stop the evaluation of other rules.

Note there are rules that are combination of the 4 types mentioned above, even with nesting or rules made by those guys.

Now we proceed to explain how the rules will be stored in a relational database. [1]. For this it is proposed to store in a normal table but modeling a tree structure [2] of knowledge, as shown in Figure 3.

In the tree of knowledge rules and facts can reside on multiple levels, the way that it can provide certainty to rule only in certain environment and even in a certain time. For this article, the proposal is that the rules can contain information to process the rule type comparative facts and functions as shown in Figure 3.

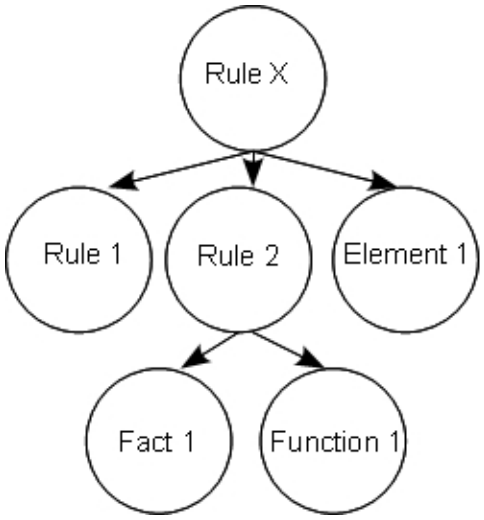

Figure 3. Example tree structure of knowledge

To explain the logical operation on the tree handler should look at figure 3, in the model presented there are basically two elements that contain circles, called nodes and lines connecting calls branches, it is important to note that the three objects shown: rules, roles and events **[3,4]**.

The order in the tree is important, because it determines the area where some knowledge is true and where not, for example, if you wanted to know if X is true Rule would have to first meet the rules of this down, be true in this particular configuration, ie they must be certain Rules 1 and 2. The functions are called facts and aids in the rules for determining the nature of the elements of the rule, in this particular case, the Fact 2 Function 1 and are only valid for Rule 1.

The system first evaluates whether they are true or false rules descendants, through a scanning depth of the tree, and if they are true, will assess the main rule or pivot.

Now we proceed with some definitions needed

The following definitions denote theory KC (Knowledge Base) and its components, for subsequent use in the proposed rule evaluator to be used in filtering data.

Definition 1. A Knowledge Base *Basec* is formally defined as:

BaseC = Reg U Con U func, where *Reg* representing the rules in the table; *Con* represents concepts or facts in the same table in the knowledge base and *func* predefined functions for various operations with database or outsiders.

The concepts used here are definitions of the types of elements used within the rules.

Definition 2. A rule base *Reg* is a set of expressions $R_1;R_2;:::;R_n$ of the form:
$A => S_1; S_2;...; S_n$

Which are defined for two functions:

$ant(R) = A$
$suc(R) = S_1; S_2; :::; S_n$

where:

- *A* is the background refers to the environment within which rules have veracity $R_1; R_2;...;R_n$.
- $S_1; S_2; :::; S_n$ successors or children (objects-facts) represents other possible rules, concepts, events or functions necessary to activate *A*.

The rule base is what establishes the categories that acquire the concepts according to their position in the rules. For its part, the inference engine uses the knowledge stored in the KB for reasoning and determines how to solve a particular problem that is, presenting the results. This interpreter determines the control scheme and contacts solving strategy to use and is independent of the rule base.

Definition 3. Let $R \epsilon Reg$, a nonempty set of rules, we say that a sequence $R_1;R_2;:::;R_n$ R element is a string or line of reasoning, if it holds that for each $Ri(i = 1; ...; n\text{-}1)$ any of their successors objects also appears as part of the antecedent of the rule R_{i+1}. formally, $R = R_1;R_2;:::;R_n$, is a line of reasoning if it meets **[4]** con:

$$\forall R_i(1 \leq i \geq n) \in Reg,$$
$$\exists Ce Con | C \subset suc(R_i) \wedge ant(R_{i+1})$$

Basis of these definitions, hereafter consider working on a knowledge base *Basec*, and that in each evaluation of a condition or in some data filtering was used one or more line of reasoning.

It is noteworthy that in this scheme the purpose of rules is that the rules are as simple for easy maintenance and instead of extensive rules, these should be broken down into several simple rules, as saying that phrase divide and conquer.

Proposes the use of two types of rules:

- Forced Compliance (Y- AND)
- Compliance optional (in a set of rules at least one of the group must meet) (O - OR)

To explain the operation of rules enforced (Y) observe Table 1.

Result rule 1	Result rule n	Ultimate result
1	1	**1**
1	0	**0**
0	1	**0**
0	0	**0**

Table 1. Evaluation logical condition Y(and)

To explain the operation of rules not enforced (O) observe Table 2.

Result rule 1	Result rule n	Ultimate result
1	1	**1**
1	0	**1**
0	1	**1**
0	0	**0**

Table 2. Evaluation logical condition O(or)

From Table 1 and Table 2, which explains the result is evaluated for the rule for example Table 1 considers all enforceability certain rules must be fulfilled to consider that, for Table 2 shows optional compliance it is sufficient that a rule is certain to be considered as a group that is accepted rules.

To show how the rules differ enforceability (Y) compliance and optional (O) observe Table 3.

Rule	Id group
Rule 1	
Rule 2	1
Rule 3	1
Rule 4	

Table 3. Example rules *Y* & *O*

Derived from Table 3, observe two columns, the first titled Rule denotes individual conditions or rules applicable to certain application, ID brings together the second call is important to this case, because if you notice Rule 1 and Rule 4 does not have assigned value, but Rule 2 and Rule 3 if they have the same identifier being for both and which in this case is 1. Indicating that belong to a group to evaluate overall evaluation as a whole and which should give affirmative, this means that at least one rule must be certain or more than one or all for any of the three cases is affirmative the group 1.

Now proceed to explain how save it the rules in a table in a database.

Derived from the need to create a container to accommodate a tree data structure, and given that the proposal in this paper is that the rules are in the database, we propose the creation of a database table and in hereinafter referred to rules which have the attributes that are shown in Figure 4.

Figure 4. Description of rules table

In the first instance be explained as a tree structure stored in a list or table which only grows in number of records dynamically.

First we define a root node, which is called the initiation and it will derive the first general rules or concepts which are evaluated or derive other as shown in Figure 5.

Then each rule represent a record in the attribute Father who derives positioned to root for this field empty anger in value will contain the rule in terms of power benchmark so that you can evaluate meets or does not meet .

Child or *desc(R)* is placed in the name of the descendant or child if there otherwise the field will be empty, so that linear growth response fields or rules can create a node with n number of child nodes or descendants, to exemplify this see figure 5

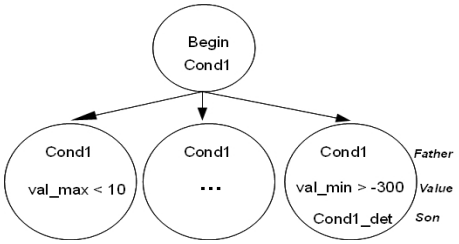

Figure 5. Example placement rules in tree

A tree as Figure 5 can be represented as shown in Table 4.

Father	Value	Son
	begin	Cond1
Cond1	val_max < 10	Cond1_det
…	…	…
Cond1	val_min > -300	Cond1_det

Table 4. Example of creating knowledge tree

With this structure n rules can be derived from the same father and n levels of rules, only determined by the ability of computer equipment capacity handler + databases.

It is noteworthy that both the field and the field Father and Son are identifiers that are used to create a dynamic structure, but the field value will contain the rule, and the Type field is empty in the case of a rule, but the nature of the element contain otherwise, both the value and the type will be stored in the table rules, but in order, if this rule is only constructed of elements that will be used here only then can be derived as children of these, but if other components of the same level will use it may be at the same level of regulation, even if more general at higher levels may be such that all the rules of that level and below can use those components.

Now they talk about what components of a rule, they can be of type:

- Numbers
- Data base field's
- Numeric variables
- Functions or procedures.
- Comparators
- Database functions

Thus it becomes essential to know which one will have each element in order to take proper care. Since this rule is becoming text instructions that will be served by the system manager rules in question.

To assign the nature of this element is stored in the attribute type and will in the following way:

V – Variable
R – Relational operator
E – Integer
F- Function
T – Table
C – Operator sets
A- Arithmetic

Now we proceed to show through a mind map as an example of a line of reasoning would store in figure 5.1

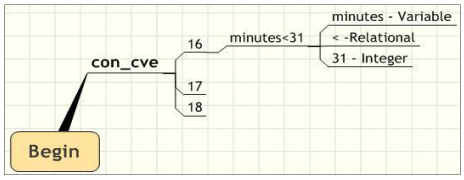

Figure 6. Example of representation

Figure 6 shows an object identifier 16 in this case represents a delay of a ticketing system, here is presented only one rule is: min <31, this results in a delay that must meet the rule, the rule stems having 3 elements proposed algorithm considers the separator elements is the space, the first element plus minutes it is considered to be a variable named, so that when this variable will be replaced, also reaches other element is <, operator and eventually lower than said integer 31 is finally summarized in the value assuming the variable minutes should be less than 31, so that it is met or not. In this way one any object within the catalog can have n rules to comply.

In Table 5 are examples of more complex rules with access to databases, which can be processed by the rules evaluator:

With reference to Table 5, each record represents a line or rule to be evaluated

Where:

The first rule, already operates a database query and has to do with finding some information in a table in the database of the organization and that having the! C indicates that it should be contained in the table. More specifically it can be interpreted as the search for a record with employee ID

(emp), date (fec_aplic), the key submitted (CVE) and the field and to be rejected by some not to be found.

Father	Value	Son
cve16	emp & fec_aplic & cve & rechazado !C base.falta	cve16_det
cve16	SUMA (dias E emp & anio_aplic & con_cve & !procesado & tpo_categ C base.falta) < 7	cve16_det
cve16	dias_entrantes + SUMA (dias E emp & anio_aplic & con_cve & !procesado & tpo_categ C base.falta) < 7	cve16_det

Table 5. Examples of rules to databases

The second rule has two parts: The first is located before the logical operator less than (<), and the second part after this operator, which in this case is the number 7. The first part of the rule says: Add the field days identified as belonging to the scheme E - base.falta table that meet the search conditions similar to rule 1. In evaluating the rule synthesizing is one simple terms a number that is less than 7, and therefore true or false 1 0.

The third rule in addition to the result given by the sum, which represents what is already in the database, do the arithmetic with days to count the incoming grand total is less than 7. This last rule is the most complex, because it is a single rule, allows the addition of an incoming value, more the result of the database to a logical comparison.

This rule can be seen that complex rules can be processed, due to the progressively reduces symbols priorities, taking the following order:

1. Solve operations on databases
2. Solve arithmetic
3. Resolves logical comparison

The reduction is treated, processed as part of the rule, the value obtained is immediately replaced by the substring that originated the transaction and remains a smaller chain and so on until 1 or 0.

Importantly, treatment to achieve reductionism, recursion is not used, because the recursive processes consume more resources than a single cycle, the cycle being less expensive resource, and thus leaves more resources to meet longest rules .

In the long version of the rule is given by the length of the field value in the rules table, but this is not a limitation, because the more complex and lengthy rules usually decomposed into several rules, and in this point could be extended to millions of records.

RESULTS

Derived from the use rules evaluator implementation in a production environment in the human resources area in an institution of secondary education, undergraduate and graduate, the following results were obtained.

514293 executions were made or system testing rules evaluator having an average total time tests 0.0421688208588 seconds, as can be seen is an adequate amount of executions or tests, from which it can be seen that the average time is less the second.

There have been executions of the system with different types of rules in order to meet the production needs of the environment where they tested the rules evaluator implemented in PHP libraries. In Table 6 shows these types of rules.

Id rule	Rule´s type
1	Comparison of <,> or == between a constant and a variable.
2	Comparison of <,> or == data obtained from a user-defined function
3	Search for existence of a data in database
4	Suma de un campo de una tabla, dadas ciertas condiciones
5	Incoming data more sum of a field in a table, given certain conditions
6	Search data between dates in a database table

Table 6. Types of rules

Tests have been performed with the 6 types of rules, however are shown in Table 7, an example of each representative type to see the time in seconds.

Evaluated rule	Type	Num tests	Time secs
minutos < 31	1	21630	0.00286
fecha_aplic <= fecha_actual	2	46214	0.00838
num_emp C test.nom_persona	3	46027	0.02212
SUMA (horas E num_emp & fecha_aplic & con_cve2 & !reint C incidencias.nom_falta) < 8	4	21049	0.05757
dias_entrantes + SUMA (dias E num_emp & anio_aplic & con_cve & !procesado & tpo_categ C incidencias.web_falta) < 366	5	1786	0.33856
num_emp & ENTRE (fecha_aplic ; fecha_ini ; fecha_fin) & con_cve4 !C incidencias.nom_falta	6	42920	0.06902

Table 7. Test statistics

As can be seen the average execution time in table 7 run much faster under rules that are simple, and are increasing the average time for the type of rule being exposed type 5 the more time consuming, because they perform queries made to database, refer to the *nom_falta* table contains 1726537 records and table *web_falta* contains 1046198 records, to the date of analysis.

Such executions have been carried out in a normal production system starting this on: 09/05/2013 13:06:48 and last execution was carried out: 19/06/2013 15:39:47

In figure 7. Show a scatter plot for the May 30, 2013 with 7453 executions shown on the x-axis the hour that happened the execution of one or more rules in the Y axis shows the time in seconds implementation with respect to the time.

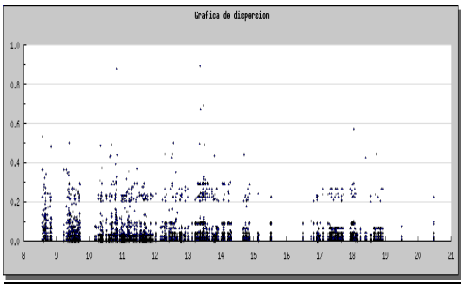

Figure 7. executions on May 30, 2013.

In figure 8. Show the chart corresponds to June 4, 2013 and performed executions of rules 45.758

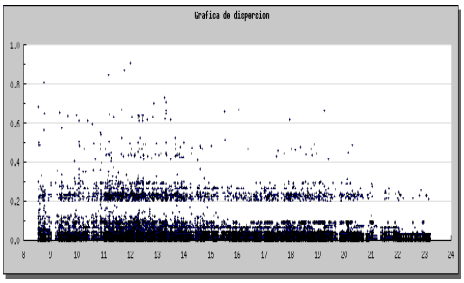

Figure 8. executions on Jun 4, 2013.

In the graph of figure 7 was a day with few processes and can appreciate that most assessments are rules morning from 8:00 to 15:00 hrs and then another time between 17:00 and 19:00 hrs, and can see an accumulation of less than 0.2 seconds and over the lesser of 0.2 and 0.8 seconds.

In the graph of figure 8 was a day with a lot of processes and can appreciate that most assessments are rules morning from 8:30 to 21:00 hrs, and can see an accumulation of less than 0.2 seconds and over the lesser of 0.2 and 0.8 seconds.

Results of the July 3, 2013

In order to determine the trend of the samples on June 4, 2013 is interpolated by the least squares method with the use of a line and 45758 executions and applying the least squares formulas exposed then we have:

$$m = \frac{n\sum_{i=1}^{n} f(x_i)x_i - \left(\sum_{i=1}^{n} f(x_i)\right)\left(\sum_{i=1}^{n} x_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$$b = \frac{\left(\sum_{i=1}^{n} f(x_i)\right)\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} f(x_i)x_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Where n is the number of samples considered

From this we determine the equation of the line as shown below:

$$y = mx + b$$

$$\sum_{i=1}^{n} x_i = 685152.62$$

$$\sum_{i=1}^{n} f(x_i) = 1487.5759$$

$$\sum_{i=1}^{n} x_i^2 x_i = 10852785.6098$$

$$\sum_{i=1}^{n} f(x_i)x_i = 45758$$

m = -0.0014859453440868

b = 0.05475928462078

Leaving an equation of the line like this:
y = -0.0014859453440868x + 0.05475928462078

Tabulating with spaces of 0.5 in the x-axis we have the following values:

$datox[1] = 8.5;
$datoy[1] = 0.0428;
$datox[2] = 9;
$datoy[2] = 0.0421;
$datox[3] = 9.5;
$datoy[3] = 0.0413;
$datox[4] = 10;
$datoy[4] = 0.04064;
$datox[5] = 10.5;
$datoy[5] = 0.0398;
$datox[6] = 11;
$datoy[6] = 0.0391;
$datox[7] = 11.5;
$datoy[7] = 0.0384;
$datox[8] = 12;
$datoy[8] = 0.0376;
$datox[9] = 12.5;
$datoy[9] = 0.0369;
$datox[10] = 13;
$datoy[10] = 0.0361;
$datox[11] = 13.5;
$datoy[11] = 0.0354;
$datox[12] = 14;
$datoy[12] = 0.0346;
$datox[13] = 14.5;
$datoy[13] = 0.0339;
$datox[14] = 15;
$datoy[14] = 0.0332;
$datox[15] = 15.5;

```
$datoy[15] = 0.0324;
$datox[16] = 16;
$datoy[16] = 0.0317;
$datox[16] = 16.5;
$dato7[16] = 0.0309;
$datox[17] = 17;
$datoy[17] = 0.0302;
$datox[18] = 17.5;
$datoy[18] = 0.0294;
$datox[19] = 18;
$datoy[19] = 0.0287;
$datox[20] = 18.5;
$datoy[20] = 0.0280;
$datox[21] = 19;
$datoy[21] = 0.0272;
$datox[22] = 19.5;
$datoy[22] = 0.0265;
$datox[23] = 20;
$datoy[23] = 0.0257;
$datox[24] = 20.5;
$datoy[24] = 0.0250;
$datox[25] = 21;
$datoy[25] = 0.0242;
$datox[26] = 21.5;
$datoy[26] = 0.0235;
$datox[27] = 22;
$datoy[27] = 0.0228;
$datox[28] = 22.5;
$datoy[28] = 0.0220;
$datox[29] = 23;
$datoy[29] = 0.0213;
```

Now we proceed to present a scatter plot on the scale presented in Figure 7 and 8. see Figure 9.

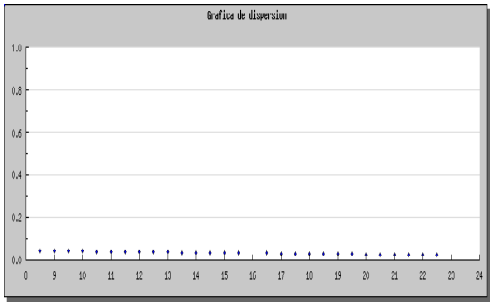

Figure 9. least squares on Jun 4, 2013

In figure 9. You can see that the line is also 0.1seconds below you can see that at the beginning of the day at 8:30 is more time consuming but as the day wore on will decrease over time in each execution of rules and is precisely because the system is less load.

But on the graph 9 is difficult to distinguish this effect so that we proceed to move the scale on the y-axis represents the time in seconds of execution and sets the maximum value of y = 0.06 seconds, and can be seen in figure 10.

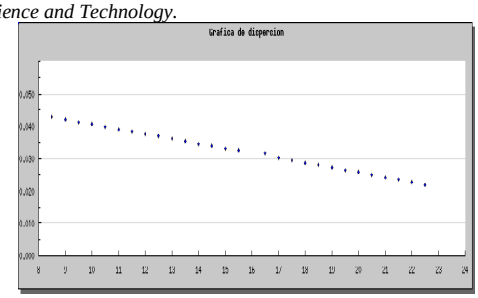

Figure 10. least squares on Jun 4, 2013

In Figure 10 it is clear that in the course of the day use is decreasing

CONCLUSIONS

It can be concluded that the rules put in a database is appropriate from the point of view of maintenance because rules can be recorded quickly and immediately put in use, however to get it running on the production system it was observed that various needs arise not covered yet, which is why it leads to constant research and development. The filter response speed is rapid and depends on the speed of computer equipment that serves as a database server, was used for these tests handler MySQL 5.0 [6] and operated in this computational equipment 1Gbyte of RAM with dual core and speed of 2.2 Ghz. The average time of execution was 0,035secs.

While on the other hand the response time is not linear because one must consider that the evaluation time of each rule depends on its nature. However, the most time consuming and not strictly dependent on the response time of the post itself rule are those rules that involve database queries, and these depending can be fast or very time consuming as the complexity. But this type of validations that have to do with queries to other database schemes [7].

Among some perceived needs and coming developments remain to be seen if the rules for most cases are valid for a period of time and then we lose validity, especially if you need to validate current and previous rules, it will be necessary to validate in time slices to the rules that require it.

ACKNOWLEDGMENT

We like to express sincere appreciation and deep gratitude to all participants in this work

REFERENCES

[1] Ma. Victoria Nevada Cabello, *"Introducción a las bases de datos relacionales",* P.P. 22, Visión Libros, Madrid, España.

[2] Román Martínez, Elda Quiroga., *"Estructura de datos",* Ed. Cengage Learning Editores, P.P.116, Germany:, 2001.

[3] José Hernández, *Introducción a la minería de datos*, Pearson Prentice Hall, 2008, p. 281.

[4] G. Barceló, M.A. Alonso, A.V. de la Cruz, E.A. Cendejas *"Medidas de Complejidad Cuantitativas para Sistemas Expertos Basados en Reglas"* .Inteligencia Artificial 43(2009), 16-31, 2010. http://erevista2.aepia.org/index.php/ia/article/viewFile/1010/1018. doi: 10.1441/ia.v13i43.1010

[5] Este José María Font Fernández. *"Sistemas de representación de conocimiento basados en reglas"*, Tesis de maestría. Universidad Politécnica de Madrid, 2008. http : // oa.upm.es / 1064 / 1 / JOSE_MARIA_FONT_FERNANDEZ.pdf

[6] www.mysql.com

[7] Peter Rob, Carlos Coronel, *"Sistemas de bases de datos"*, pp 213, Thompson editores, México, 2004.

[8] Raúl Pino, Alberto Gómez, Nicolás de Abajo, *"Introducción a la inteligencia artificial: Sistemas expertos, Redes neuronales artificiales y computación evolutiva"*, Universidad de Oviedo, España, 2001.

[9] Alan Beaulieu, *"Learning SQL"*, O'Reilly, USA, 2005

[10] Nivio Ziviani, "Diseño de algoritmos", International Thomson Editores, España, 2007.