International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:5, 2010

Flexible, Adaptable and Scaleable Business Rules Management System for Data Validation

Kashif Kamran, Farooque Azam

***Abstract*—**The policies governing the business of any organization are well reflected in her business rules. The business rules are implemented by data validation techniques, coded during the software development process. Any change in business policies results in change in the code written for data validation used to enforce the business policies. Implementing the change in business rules without changing the code is the objective of this paper. The proposed approach enables users to create rule sets at run time once the software has been developed. The newly defined rule sets by end users are associated with the data variables for which the validation is required. The proposed approach facilitates the users to define business rules using all the comparison operators and Boolean operators. Multithreading is used to validate the data entered by end user against the business rules applied. The evaluation of the data is performed by a newly created thread using an enhanced form of the RPN (Reverse Polish Notation) algorithm.

***Keywords*—**Business Rules, data validation, multithreading, Reverse Polish Notation

I. Introduction

MOST of the organizations define their business rules according to their terms and conditions decided by the stake holder considering multiple factors. These business rules are applied on the data of the organization to enforce the policies by data validation [1].

Whenever a company decides to implement the information systems, it is imperative to define the business rules to be modeled and imposed by the system analyst. It is practiced by well structured organization to hire a data administrator to define all such rules comprehendible by the system analyst. Initially, it has been observed that companies demand to impose strict rules to maintain the data integrity. System analyst is bound for conformance of the organizational requirements for data validity.

Once the system analyst has modeled the strict business rules as per the requirements of the organization, the modeled rules are hard coded by the developer using any development tool. As a matter of fact, after the system's "Go Live" phase, organization starts realizing to lose the strict control imposed for data entry. The requirement for changes in the business rules might be result of the change in the terms and conditions or change in the business processes. To avoid the change in the business rules, it has also been suggested that business rules must be well structured and modeled at the design time in order to avoid the change after the software deployment [2].

Data validation rules are usually implemented in form of triggers and procedures defined in the database and handled by the DBMS [3]. These rules are again hard coded, if there is any change in the validation rules, triggers or procedures are required to be changed.

Change in business rules is only possible by either change in the database design or change in the code. Such change after deployment affects organization by exponential increase in cost for software maintenance and "After Sales Support".

The solution to the aforesaid problem has been given in this paper. It is proposed to devise such a scheme and semantics to define and configure the validation rules at run time. Thus any change required by the organization will easily be managed by change in the configuration of business rules. Configuration at runtime will result in "No Change" in the code and the database.

It is also helpful to briefly go through the different business rules management systems/engines briefly.

A. A generic business rules validation system for Oracle applications.

This business rule validation system has been designed to implement the validation of complex business rules for any ORACLE database application - without incurring any modification to the user interface [4]. The customized business rules may be configured for any application already running in oracle and may validate the historical data already entered in the database.

The validation system has following disadvantages:

- The rule engine performs the data validation once the data has been entered. It allows invalid data to be entered.
- The business rules validation engine only works with ORACLE data base.
- The rules engine demands to write pl/sql procedures and sql statements for data validation.

B. Decision Support System for Data validation

DSS (Decision Support System) has a different mechanism to operate and defines data objects for data validation. Data object can be considered as an abstract

F. A. Author is with the College of Electrical and Mechanical Engineering, National University of Sciences and Technology Rawalpindi Pakistan (phone: 092-333-5761981; e-mail: kamran.kashif@gmail.com)

S. B. Author, is with the College of Electrical and Mechanical Engineering, National University of Sciences and Technology Rawalpindi Pakistan (phone: 092-51-9278056; e-mail: farooq@ceme.nust.edu.pk)

object defined by a four tuple (A, N, V, T) where A represents the address of an object in the address space of the computer system, N represents the name of an object in the name space of the software system, V represents the value of an object, and T represents the type of an object [5].

The DSS validates the data considering its data type. The system has been divided into four modules: 1) User Interface Module 2) Model Processing Module 3) Data Base Module and 4) Knowledge Base Module.

Although the prototype of the DSS has been implemented and tested, but it has following deficiencies:

- It only validates the data that is properly organized in the text files with data types specified.
- The data validation is totally dependent upon the data type.
- It does not support the extraction and validation of data from any DBMS.

C. Data Driven Validation Rules, Custom Data validation without custom programming

The system discussed in this paper was developed for internal use at Ursa Logic Corporation. The goal of the development project was to simplify, streamline, and standardize the process of creating customized validation rules for large collections of data stored in SAS datasets [6]. The validation system will be referred as data checker hereafter.

Data checker can handle all the basic to advance checks in the form of business rules. It may handle range check, valid value check and missing value check.

The system has the following drawback.

- All the rule sets are stored in Microsoft Excel. This leads to the following problems.
- The data file for maintaining the rule sets is vulnerable to security threats and data loss as compared to the data stored in DBMS.
- The system is fully dependent on Microsoft Excel and the implementation of the data checker is questionable in other operating systems.

II. System Defined

An overall view of the system architecture has been presented in Figure 1. The detailed elaboration of each component in the system follows:

A. Rule Set

Rule set is the set of conditions required to be satisfied by the valid data. Rule sets are categorized in two categories, 1) Range Check, 2) Domain Check.

1. Range Check

Range check is the rule/s that is applied to the numeric data only including the date. Range check ensures that the valid data must fall within the range of values specified. The examples of range check are:

- Age must be between 18 and 35 years inclusive.
- Joining date must be lesser than or equal to the current date.

2. Domain Check

Domain check value states that the data must be equal to one of the values specified. The examples of domain check are:

- Blood Group must be equal to any one of the values listed: "A^+", "A^-","B^+","B^-","AB^+","AB^-","O^+","O^-".
- Marital Status must be equal to "Single" "Married", or "Divorced".

B. Rule Set Creation

The rule set can easily be created at run time by using the Comparison and Boolean operators. The user may easily create the rule set using the graphical user interface. User is provided the option to select the comparison and Boolean operators along with parenthesis to specify the precedence. The rule set created once may be applied to as many data variables as required. A data variable may have many rule sets for data validation. The symbols used for Rule Set definitions are:

TABLE I Comparison operators used for rule creation

S.No.	Operator	Notation
1.	Greater Than	>
2.	Lesser Than	<
3.	Greater Than or Equal To	>=
4.	Lesser Than or Equal To	<=
5.	Equal To	=
6.	Not Equal To	!=

TABLE II Boolean Operators Used for rule set creation

S.No.	Operator
1.	AND
2.	OR

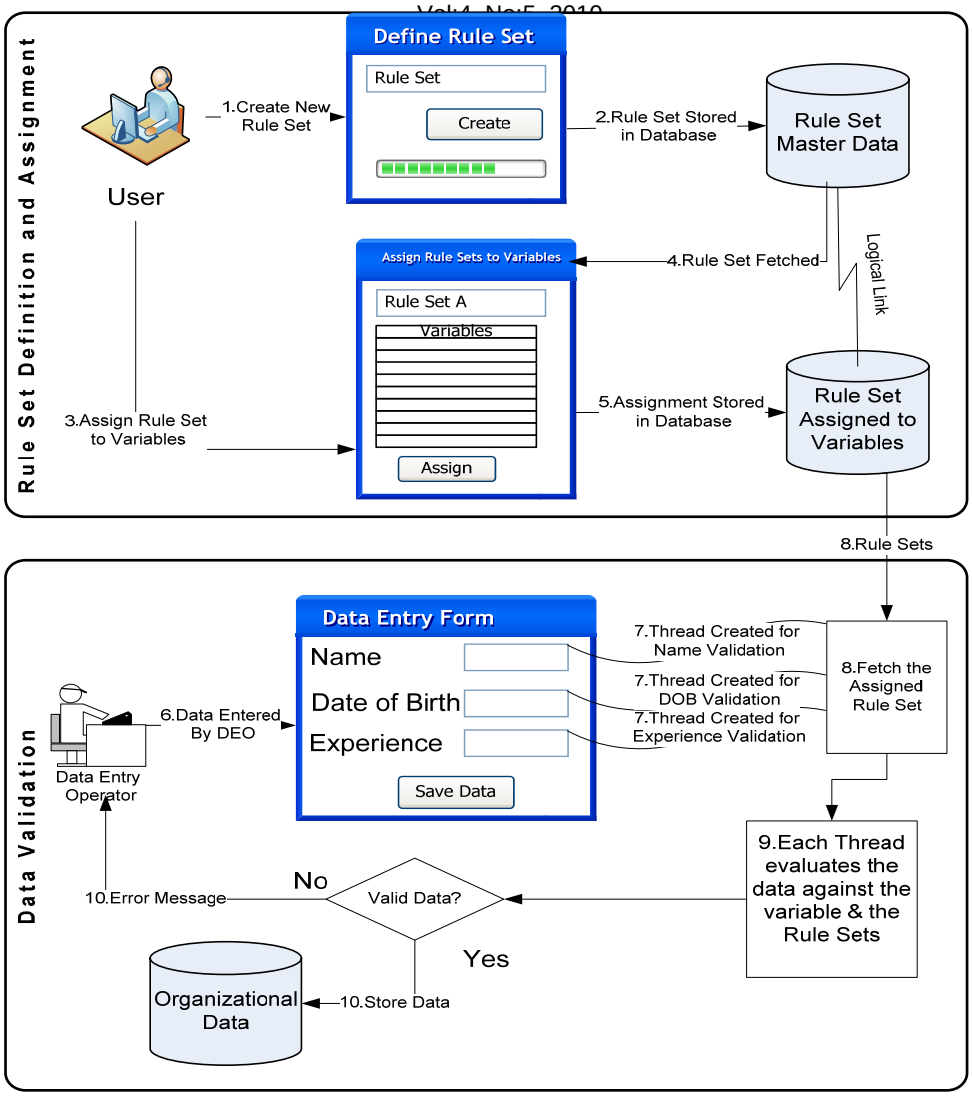

Fig. 1. System Architecture of the Proposed Technique

While designing the semantics used to define the rule set, it was emphasized for the following:

- The rule set definition process must be simple enough to be easily used by end users
- The scheme for rule set definition must be dynamic enough to define all types of rule sets

Let us create the rule set for the following conditions to be met. These examples created here will be considered in the subsequent sections of this paper. Condition I and Condition II represents the Range Check and Condition III is an example of Domain Check.

Condition I: The age of an employee must be between 18 and 35 years (Both ends are inclusive).
Rule Set I: RVE >= 18 AND RVE <= 35

Rule sets are created independently initially and later on they may be associated to as many variables as desired. RVE means Run Time Value Entered by the data entry operator. Here the name of the variable for example age is not used because this Rule Set I has not been associated with age yet. It will be later on bound with age.

Condition II: The Date of Joining must be less than or equal to the current date.
Rule Set II: RVE<= sysdate

Condition III: The marital status of an employee must be "Single" or "Married"
Rule Set III: RVE= "Single" OR RVE="Married"

C. Rule Set Database

The proposed approach differentiates itself from other techniques by creating the validation rules after software

development process. Therefore it is mandatory to store the user defined rule sets in the database. The Rule set defined becomes part of the master data. The database architecture used has been defined as:

Referring to the
Rule_Sets: This table stores only the identity of the rule set.
Rule_Set_Defined: It is child table of Rule_Sets. It stores the actual conditions of any rule set.
All_Tabs_Cols: It is an implicit view created by DBMS (Oracle 10g and later releases). This view stores every column of each table.[5]
Applied_Rule_Sets: This table is the main table used to store the information of which rule sets have been applied on which columns.
Table1, Table2, Table3 and Table4: These tables are generic tables to represent the database. Since the module is dynamic, so it may contain as any number of tables as required.

D. Binding Rule Sets with Variables

Considering the above database and the examples already defined, we enter the data values into the above tables according to the condition I, II and Condition III.

TABLE III RULE_SETS

Rule_Set_ID	*Description*
RS_Age	Employee Age must be between 18 and 35 years inclusive.
RS_DOJ	Date of Joining must be lesser than or equal to the System entry date.
RS_MS	Rule Set for Marital Status, the valid data must be “Single” or “Married”

Two Rule Sets have been defined in Table III. The first Rule Set having ID as RS_Age ensures that age of an employee must be between 18 and 35 years. RS_Age represents the Condition I, RS_DOJ represents the Condition II and RS_MS represents Condition III already stated in Section II.B.

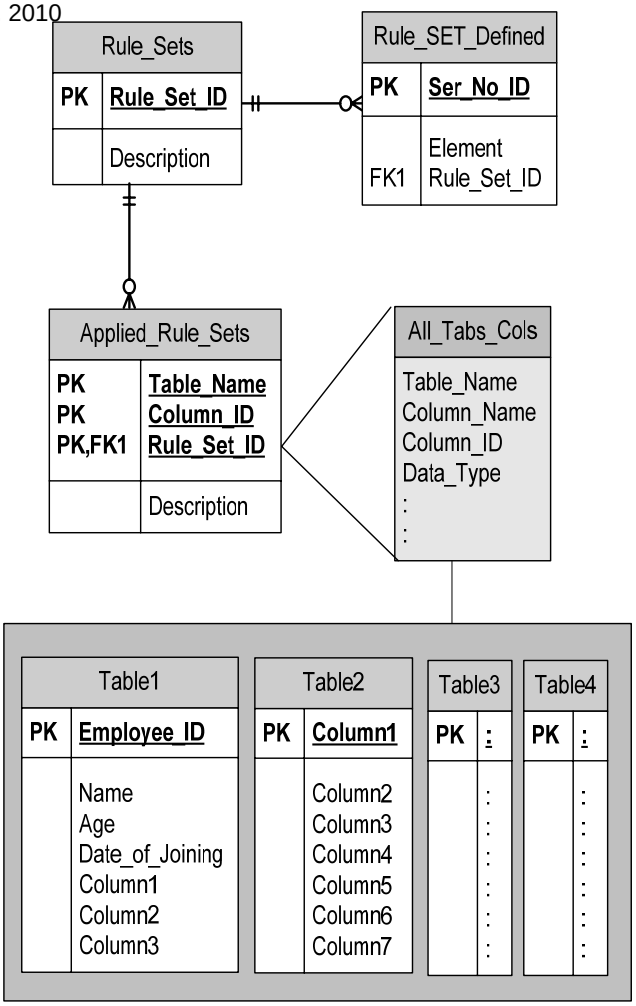

Fig. 2. Database Design of the Proposed Technique.

TABLE IV RULE_SET_DEFINED

Ser_No_ID	*Element*	*Rule_Set_ID (FK)*
7.	RVE	RS_Age
8.	>=	RS_Age
9.	18	RS_Age
10.	AND	RS_Age
11.	RVE	RS_Age
12.	<=	RS_Age
13.	35	RS_Age
14.	RVE	RS_DOJ
15.	<=	RS_DOJ
16.	sysdate	RS_DOJ
17.	RVE	RS_MS
18.	=	RS_MS
19.	“Single”	RS_MS

20.	OR	RS_MS
21.	RVE	RS_MS
22.	=	RS_MS
23.	"Married"	RS_MS

Table IV represents all the three Rule Sets Created in SectionII.B. Rows with Serial_No_ID from 1 to 7 represent Rule Set I defined as RVE >= 18 AND RVE <= 35. Rows with Serial_No_ID from 8 to 10 represent Rule Set II defined as RVE<= sysdate and Rows with Serial_No_ID from 11 to 17 represents Rule Set III RVE= "Single" OR RVE="Married".

TABLE V ALL_TAB_COLS

Table_Name	*Column_Name*	*Column_ID*
Table1	Employee_ID	1
Table1	Name	2
Table1	Age	3
Table1	Date_of_Joining	4
Table1	Marital_Status	5
Table1	Blood_Group	6
Table1	Column1	7
Table1	Column2	8
Table1	Column3	9

Table V is basically a view created automatically by DBMS oracle 10g and later releases. This table contains the Table name, Column Name and Column ID of each table created in the database. Table1 has been created as an example for applying the Rule Set I,II and III on it for data validation.

TABLE VI APPLIED_RULE_SETS

Table_ Name	*Column ID*	*Rule_Set_ ID*	*Description*
Table1	3	RS_A ge	Age Validation
Table1	4	RS_D OJ	Date of Joining validation
Table1	5	RS_ MS	Marital Status Validation

Table VI is the composite table which elaborates the relationship between database tables' columns and Rule Set. The data entered in this table shows that Column 3 of Table1 has Rule Set RS_Age associated, Column 4 is bound to pass the data validation check for Date of Joining (RS_DOJ) and similarly Column 5 data is bound to pass the Domain Check of Marital Status (RS_MS).

TABLE VII TABLE I DATA

Employee_ID	*Name*	*Age*	*Date_Of_Joining*	*Blood Group*
E-001	Imran Haseeb	23	12-03-2009	A^+
E-002	Junaid Iqbal	32	07-09-2008	O^+

Table VII shows the actual data entered into the Table1.Data of two employees has been entered. It is obvious from the data entered that both the employees' data conforms to the Rule Set I, II and III. The validation process has been elaborated in the subsequent sections.

III. DATA VALIDATION PROCESS

Section II described the basic architecture of the proposed technique. In this section the data validation process with its core components is elaborated. The main steps involved during the validation process are:

A. Multithreading

Data validation rules may have higher degree of complexity depending upon the user requirements. User may create complex Rule Set involving multiple conditions joined by AND or OR operator and prioritized by parenthesis [12].

To efficiently perform the data validation, multithreading has been used for the prompt response to end user during the data entry process [13].

Whenever user enters the data on the graphical user interface provided, on leaving the user control, a thread is automatically created. The newly created thread scans the Applied_Rule_Set table to identify that any Rule Set exists for data validation of data variable under consideration. If there is any Rule Set defined, the definition of the Rule Set is fetched from Rule_Set_Defined table. The Rule Set is stored in the infix notation in the database [11]. The thread is responsible to convert the Rule Set in Post Fix notation using the algorithm defined below. Once the Rule Set has been converted to the postfix notation, it is evaluated by another algorithm.

B. Rule Set Conversion to Postfix Notation

The Rule Set fetched against the age variable is RS_Age and is described as RVE >= 18 AND RVE <= 35, once it has been fetched from the database by the thread created for evaluating it. The evaluation algorithm first converts it to the postfix notation and then evaluates it [12]. The postfix conversion algorithm is:

Step 1: Initialize an empty stack.
Step 2: Read a Row one by one from the Table Rule_Set_Defined against a rule set identified (RS_Age in this case).

- If Left parenthesis is found then Push it onto the stack
- If Right parenthesis is found then perform the following steps:
  - ✓ Pop and display all the stack elements until a left parenthesis is found
  - ✓ Pop the top left parenthesis and do not display it.
  - ✓ Pop the top stack element if it's a comparison or logical operator and display it.

- If Boolean/Comparison Operator is found then:
  if Boolean Operator ("AND" or "OR") then
  - ✓ Pop the top element from the stack if it is any operator (Comparison or Logical)
  - ✓ Pop the top element again if it is a logical operator
  - ✓ Push the logical operator on stack read from expression.
  - If Comparison Operator is found then

  Push the element on stack if one of the following conditions is true
  - ✓ The stack is empty
  - ✓ The top element on the stack is Left Parenthesis
  - ✓ The top element on the stack is Boolean operator

  Otherwise pop up and display the "top" stack elements

If Operand (e.g RVE, 18, 35, Single) is found then
Display it.

Step 3: When End of rules set is reached, Pop and display stack items until stack is empty.
When the Rule Set RS_Age has been traversed through the postfix notation conversion algorithm, it has been transformed as below.
RVE 18 >= RVE 35 <= AND

Rule Set Evalutation after Conversion to Postfix Notation

Let us suppose that user has entered value 23 for age of an employee. Thus the above postfix statement will be transformed after substitution of user value as

23 18 >= 23 35 <= AND

The above postfix notation expression will be evaluated by the following algorithm

Step 1: Initialize an empty stack.
Step 2: Read the token one by one until no token is found.
If there is no token to read, then the top element and the only element on the stack is the result
Else go to step3
Step 3: Evaluate the token and perform the following step.

- If the token is comparison Operator Push it on the stack.
- If the token is comparison/Boolean operator then Pop the top two elements from the stack and perform the operations as the first element popped must be considered as the second operand and the second element popped must be considered as the first operand during the operation.
- Push the result on to the stack (True or False)

The Last element on the stack is the final result.

In our case for 23 18 >= 23 35 <= AND, the result is true and the data entered by user has passed validation. It will be saved in the database.

IV. Experimental Results

Different Rule sets were created at run time after software development process and were applied to data variables. The same Rule Sets were also hard coded for the same set of variables available on the user interface. The tool used to hard code the Rule Sets was Visula C#.Net. The results calculated in terms of time are:

TABLE VIII Experimental Results I

Rule Set Creation and Application Results	
No. of Rule Sets	23
Time Consumed to hard Code	13 Hours
Time Consumed to dynamically configure (Proposed approach)	6.5 Hours

The Rule Sets created in the above scenario were changed. The changes made in the Rule Sets were also experimented by both the approaches for comparisons the results are:

TABLE IX Experimental Results II

Changing the Rule Sets Results	
No. of Rule Sets changed	7
Time Consumed to change the Code	9 Hours
Time Consumed to dynamically configure (Proposed approach)	2 Hours

V. Conclusion

A dynamic approach for Rule Sets creation used for data validation has been proposed in this paper. The system initially creates the Rule Sets after the software has been developed. The Rule Sets may be applied to as many number of data variables (fields, table columns) as required. Similarly a data variable may have many Rule Sets applied on it. Rule Sets are evaluated using multithreading technique, thus enhancing the data validation time. If there is any change in the data validation rules, it may easily be accommodated through user interface with changing the code. The only limitation to the proposed approach is that thread consumes some time to search for the rule set applied on data variable. This can easily be overcome by indexing the table used for keeping the information for Rule Sets and data variables.

REFERENCES

[1] Tony Morgan, Business Rules and Information System, Aligning IT with Business Goals Page 59, chapter 3 Defining Business Rules

[2] O. Vasilecas, E. Lebedys, Application Of Business Rules For Data Validation, Information Technology And Control, 2007, Vol.36, No.3

[3] J. Laucius, E. Lebedys, O. Vasilecas. Realisation of ECA rules by ADBVS triggers. Information sciences, Vilnius University, 2003, 129-133.

[4] *A Generic business rules validation system for ORACLE Applications*. Olivier Francis MARTIN. *System* analyst. European Laboratory for Particle Physics – CERN, Jean Francois PERRIN,Consultant,AUSY,Lyon – FRANCE

[5] Decision support system for data validation, Ramesh M. Choudhari, South Carolina State University, Orangeburg, SC 29117, Shobha R. Choudhari, South Carolina State University, Orangeburg, SC 29117

[6] *Data-driven Validation Rules*: *Custom Data Validation Without Custom Programming*. Don Hopkins, Ursa Logic Corporation, Durham, NC

[7] H. Herbst, G. Knolmayer. The Specification Of Business Rules: A Comparison Of Selected Methodo-logies. Methods and Associated Tools for the Informa-tion Systems Life Cycle, Maastricht, The Netherlands, 1994, 29-46.

[8] http://www.mcs.csueastbay.edu/support/oracle/doc/10.2/server.102/b14237/statviews_2093.htm

[9] E. Ugboma. Assuring Information Systems' Effective-ness Through Data Integrity: Essential Guidelines For Information Systems Databases. In The Proceedings of ISECON 2004, Vol.21 (Newport): §3252, 2004.

[10] Chin-Kuang Shene, Multithreaded programming in an introduction to operating systems course, Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, p.242-246, February 26-March 01, 1998, Atlanta, Georgia, United States

[11] http://en.wikipedia.org/wiki/Infix_notation

[12] http://en.wikipedia.org/wiki/Reverse_Polish_notation

[13] J M Cherry, An experimental evaluation of prefix and postfix notation in command language sytax**,** Volume 24 , Issue 4 April 1986,365 - 374

.