Acta Apuroquia Vol.2 (1):5-13, 2010

ISSN 1836-9056

A MODEL FOR DESIGNING RULE-BASED EXPERT SYSTEMS

Gabriel Carrillo[1]

Universidad Nacional Experimental de los Llanos occidentales Ezequiel Zamora. Vice-rectorado de Planificación y de Desarrollo Regional, Programa de Ciencias Sociales, San Fernando 7001, Estado Apure, Venezuela.
e-mail: gacs2006@gmail.com

ABSTRACT

The aim of this research is to develop a model for designing rule-based expert systems that uses the forward chaining method of inference. The striking aspect of this model is that the inference engine is based on a simple representation of rules and facts in relational database tables. Rules are decomposed and represented in tables at two levels, which allow the developing of expert systems in any programming language that supports SQL. The explanation facility uses tables containing the explanations of the result of each rule. The model proposed in this paper is based on a simple approach to represent facts and rules in relational database tables. The advantage of this model lies in focusing the design of rule-based expert systems toward knowledge representation in a database, reducing effort and programming difficulties.

Key Words: Expert system, rule-based expert system, forward chaining.

RESUMEN

El fin de esta investigación es desarrollar un modelo para diseñar sistemas expertos basados en reglas que usa el método de inferencia de encadenamiento hacia delante. El aspecto resaltante de este modelo es la base de conocimientos, la cual está basada en una representación simple de reglas y hechos en tablas de base de datos relacional. Las reglas son descompuestas y representadas en tablas a dos niveles, lo cual permite el desarrollo de sistemas expertos en cualquier lenguaje de programación que soporte SQL. El mecanismo de explicación utiliza tablas que contienen las explicaciones al resultado de cada regla. El modelo propuesto en esta investigación se basa en un enfoque simple para representar hechos y reglas en tablas de base de datos. La ventaja de este modelo es enfocar el esfuerzo del diseño de sistemas expertos basados en reglas hacia la representación del conocimiento en una base de datos relacional, con lo cual el esfuerzo y dificultad de programación se reducen considerablemente.

Palabras claves: Sistemas expertos basados en reglas, encadenamiento hacia delante.

Recibido: 13/04/2009 Aceptado: 19/09/2009

INTRODUCTION

An Expert System is a knowledge-based information system that uses its knowledge about a specific, complex application area to act as an expert consultant to end users (O'Brien, 2001).

The symbolic reasoning of an expert system enables it to draw conclusion from premises and to provide explanations. Expert system technology is based on the domain knowledge of the problem being addressed. A problem domain defines the objects, properties, tasks and events within which a human expert works, and the heuristics that experienced professionals have learned to use perform better.

The components of an expert system are (O'Brien, op.cit.): Knowledge base and inference engine. Additional there can be a user interface and an explanation facility. The knowledge base contains facts about a specific subject and heuristics that express the reasoning procedures of an expert. The inference engine processes the rules and facts related to a specific problem. Two methods are used for making inferences: forward chaining (reaching a conclusion by applying rules to facts) and backward chaining (justifying a proposed conclusion by determining if it results from applying rules to facts). The user interface allow user to interact with the expert systems. The explanation facility allows the system to explain its reasoning to the user.

Expert systems have been traditionally built using programming languages like LISP, PROLOG, C, C++ and Java, or with development tools, such as expert systems shells, like Clips (Giarratano and Riley, 2004). With those software tools, most expert systems have been developed, considering its rich set of instructions and data structure support. The aim of this research paper is to develop a model for designing rule-based expert systems that uses the forward chaining method of inference. The most important aspect of this model is that the inference engine is based on a simple representation of rules and fact in database tables. Rules are decomposed and represented in tables at two levels, which gives the possibility of developing expert systems in any programming language that supports SQL.

METHODS

Rule-based Expert Systems

Rule-based programming is one of the most commonly used techniques for developing expert systems. A rule-based expert system consists of a set of rules that can be repeatedly applied to a collection of facts. The following concepts are essential to rule-based systems (Clancey, 1981):

- Facts represent circumstances that describe a certain situation in the real world.
- Rules represent heuristics that define a set of actions to be executed in a given situation.

Production rules have the pattern if <condition> then <action>. It consists of producing practical consequences from certain conditions. Rules are composed of an if portion and a then portion. The if portion of a rule the left hand side (LHS), is called predicates or premises. The LHS consists of an expression, which can be a single expression (an individual fact that must be true for applying the rule) or a series of expressions (composite expression). In the literature of rule-based languages, a single expression is usually called pattern. A composite expression consists of several single expressions connected together by using the conditional elements "and, or, not" in order to create complex rules.

Inference engines consist of all the procedures that handle the knowledge base in order to achieve a conclusion (Nilsson, 1998). Inference engines are usually coded in such languages as LISP, PROLOG, C, C++, Java.

Architecture of Rule-based Expert Systems

The general architecture for rule-based expert system is depicted in Fig 1. The main elements of a rule-based system are facts, rules, and the engine that acts on them. The core of the architecture shown in Fig 1. consists of the working memory (fact base), the rule base (knowledge base) and the inference engine (rule engine).

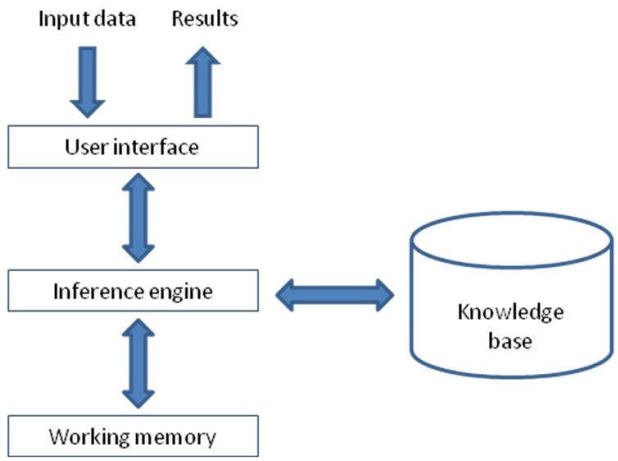

Figure 1. General architecture for a rule-based expert system.

- The working memory contains facts that are the smallest piece of information supported by the rule engine.

- The rule base contains rules in the form of if-then statements, which represent the knowledge provided by the user and/or an expert of the problem domain;

-The inference engine matches facts in the working memory against rules in the rule base, and it determines which rules are applicable according to the reasoning method adopted by the engine.

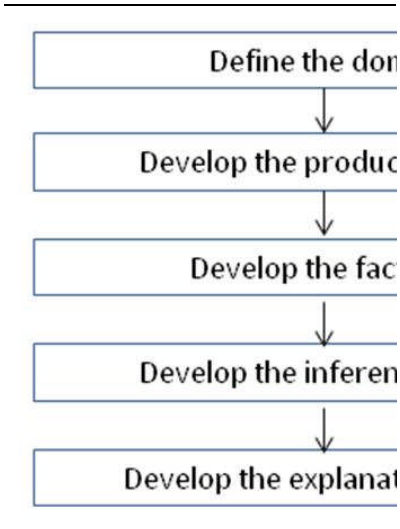

Figure 2. Procedure to develop a rule-based expert system.

Production rules

Knowledge in an expert system can be represented with a Production rules (Lewis, 2003). A production system consists of 3 items: A set of production rules, a working memory and an interpreter.

The structure of production rules can be formally stated as follows:

if <condition> then <conclusion>
or
if <condition> then <action>

Productions have two parts
Sensory precondition ("IF" part)
Action ("THEN" part)

When the state of the 'world' matches the IF part, the production is fired, meaning the action is executed. So production rules link facts ("IF" parts, also called antecedents) to conclusions ("THEN" parts, also called consequents).

RESULTS

Rule-based expert systems use forward chaining or backward chaining as inference method to achieve a conclusion. A simple model of forward chaining is the following:

If A and B or C then X;

If D or E then Y;

If X and Y then Z;

The proposed model (Gabing model) represents rules to which the forward chaining method will be applied, according to two aspects: Representation and decomposition.

Representation

Rules are represented according to the following schema:

- Each rule is made of elements related by logical connectors (and, or).
- Each rule is represented by a row in a relational table.
- A rule made of n elements connected by 'or' is represented by n rows in a relational table, each row containing a single element.

- A rule made of n elements connected by 'and' is presented by n row in a relational table, each row containing a single element and a connecting field with a 'AND' value.
- Elements in a rule belong are selected from a 'attributes' relational table.
- Each row in the 'attributes' table represents a attributes associated to a rule.
- The rules are applied to subjects. A subject that meets at least a rule, is selected and included in the conclusions.
- A subject represents real or abstract objects.
- Attributes associated to a subject are represented in a table.
- The data types used in rule representation are: varchar and float.

Decomposition

Decomposition is a technique whereby you can decompose a rule into simple pieces. The set of pieces are equivalent to the original rule. The decomposition procedure is the following:

(i) A rule is represented by a set of rule pieces.

(ii) A rule and the equivalent set of rule pieces have the same identification, i.e., a field identifying the rule.

(iii) Each rule piece contains only and only one logical expression, and a logical connector

(iv) The allowed values of the logical connector are: 'AND' and 'NULL'.

(v) An 'AND' logical connector in the original rule is mapped to 'AND' in the rule piece.

(vi) An 'OR' logical connector in the original rule is mapped to a new rule piece.

Creating a set of rule pieces

In table 1, examples of rule are represented by a new set of pieces .

Table 1: Equivalent set of rules

Id	Expression	Id	Expression	Operator	Link	
Rule1	If a	Rule1		A	NULL	LR1
Rule2	If a AND b	Rule2		A	AND	NULL
		Rule2		B	NULL	LR2
Rule3	If a OR b	Rule3		A	NULL	LR3A
		Rule3		B	NULL	LR3B
Rule4	If a AND b AND c	Rule4		A	AND	NULL
		Rule4		B	AND	NULL
		Rule4		C	NULL	LR4
		Rule5		A	AND	NULL
		Rule5		B	NULL	LR5A
		Rule5		C	NULL	LR5B
Rule6	If a OR b AND c	Rule6		A	NULL	LR6A
		Rule6		B	AND	NULL
		Rule6		C	NULL	LR6B

A complex rule must be decomposed into smaller and simpler rules. For example:

Rule 1: If (a AND b) or (c AND d) is equivalent to:

Rule1 : a AND

Rule1: b NULL LinkA

Rule1: c AND

Rule1: d NULL LinkB

Architecture for the Gabing model

The Gabing model for rule-based expert systems is made of the following elements:

- A database containing the rules and data about objects.
- An inference engine for manipulating rules and data about objects to achieve a conclusion.
- An explanation mechanism.

The database contains the following tables:

- Attributes: definition of attributes used in rules and facts (Table 3).
- Rules: definition of each rule (Table 4)..
- Rule_detail: list of rule details associated to a rule. Each detail entry contains a logical connector to the following detail entry. The data item is associated to an attribute (Table 5)..
- Rule_link: For combining intermediate results of a rule, with logical operators AND, OR (Table 6)..
- Explain: explanation associated to each rule that has been fired (Table 7 and 8)..
- Object: Contain data about facts, persons, object that are manipulated in the expert system. Conclusions are bound to objects (Tables 9, 10,11,12 and 13)..
- Object_detail: list of details associated to an object. Each detail entry contains a logical connector to the following detail entry. The data item is associated to an attribute(Table 10)..

The data base schema for the Gabing model is shown in Annex A.

DISCUSSION

In order to understand the application of the Gabing model, we present a case where it can be applied. Let us suppose the human resources office of a company needs an expert system to recruitment according to the following profiles:

- Software engineers less than 30 years old, experienced with PHP.
- Software engineers with master degree and research experience.

The rules can be represented as follows:

Table 2: group

id_group	name_group
1	Information Technology

Table 3: attribute

id_attr	name_attr	type_attr
1	Bachelor degree	Varchar
2	Master Degree	Varchar
3	PHP experience	Float
4	Research experience	Float
5	Age	Float
6	English language	Varchar
7	French language	Varchar

Table 4: rules:

id_rule	Id_group	Name_rule	id_explain
RIT01	1	Software engineers less than 30 years old, experienced with PHP	1
RIT02	1	Software engineers with master degree and research experience	1

Table 5: rule_detail

id_ruledetail	Id_rule	name_ruledetail	id_attrib	rel_op	vardata	floatdata	l_op	Rlink
1	RIT01	Software engineer	1	=	Software engineering		AND	NULL
2	RIT01	AGE < 30	5	<		30	AND	NULL
3	RIT01	PHP > 1	3	>		1	NULL	LR1
4	RIT02	Software engineer	1	=	Software engineering		AND	NULL
5	RIT02	Master degree	2	=	M.Sc.		AND	NULL
6	RIT02	Research	3	=	Research		NULL	LR2

Table 6: Table rule_link

id_rulelink	Rulelink	Operator	id_explain
1	LR1	NULL	1
2	LR2	NULL	2

Table 7: Table explain

id_explain	name_explain
1	Successful selection for Software engineers less than 30 years old, experienced with PHP. Rule RIT01.
2	Successful selection for Software engineers with master degree and research experience. Rule Rit02.

The personal data of the applicants can be stored in the object table. The category should be applicants.

Table 8: Table categories

id_cat	name_cat
1	Applicants

Table 9: objects

id_object	Id_cat	name_object
D0058792	1	GABRIEL CARRILLO
D0054865	1	MARKUS KOGAN
D0034986	1	MARIE LAGARD

Table 10: object_detail

id_objectdetail	id_object	id_attrib	rel_operator	Vardata	floatdata
1	D0058792	1	=	Software engineer	
2	D0058792	2	=	Master Degree	
3	D0058792	3	=		3
4	D0058792	4	=		2
5	D0058792	6	=	GOOD	
6	D0054865	1	=	Software engineer	
7	D0054865	5	=	27	
8	D0034986	1	=	Software engineer	
9	D0034986	2	=	Master Degree	

A complex rule can be decomposed and represented with the Gabing model. Let us suppose that we are given the following rule: IF (A AND B OR C) OR (D AND E) By decomposing this rule, we get:

Table 11: rules

id_rule	Id_group	Name_rule	id_explain
R01	1	IF (A AND B OR C) OR (D AND E)	1

Table 12: rule_detail

id_ruledetail	Id_rule	name_ruledetail	id_attrib	rel_op	vardata	floatdata	l_op	rlink
1	R01	A	1	=	Text A		AND	NULL
2	R01	B	1	=	Text B		NULL	LR1A
3	R01	C	1	=	Text C		NULL	LR1B
4	R01	D	1	=	Text D		AND	NULL
5	R01	E	1	=	Text E		NULL	LR1C

Table 13: rule_link

id_rulelink	Rulelink	Operator	id_explain
1	LR1A	OR	1
2	LR1B	OR	1
3	LR1C	NULL	1

The inference engine can be coded in any programming language that supports SQL. The user interface must allow for entry of all data required by the expert system.

Handling data types

The Gabing model supports varchar and float data types (Table 14). Other types must be converted to the most convenient of the supported types. We recommend the following transformations:

Table 14: Data type conversion

Original data type	New data type
Char, Varchar, String	Varchar
Date	Varchar in format: YYYYMMDD
Float m,n	Float m,n
Integer n	Float n,0
Logical	Float 1,0. True= 1, False =0

CONCLUSIONS

Rule-based systems emulate human expertise in well-defined problem domains by using a knowledge base expressed in terms of rules. In this paper we have shown a model for designing rule-based expert systems, with an approach that represents rules in relational tables, with decomposition at two levels.

This work is the result of an effort to develop a model for designing rule-based expert systems. This approach shifts the load from the algorithm to the knowledge base, with the aim of facilitating expert system design and programming.

REFERENCES

Clancey, W. 1981. The Epistemology of a Rule-based Expert System: A framework for explanation. Department of Computer Science. Stanford University, Stanford. Available: ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/81/896/CS-TR-81-896.pdf.

Giarratano, J. ; Riley, G. 2004. Expert Systems Principles and Programming. Fourth Ed., Course Technology, Boston.

Lewis, P. 2003. Knowledge Representation. Production rules for knowledge representation. [Online document]. Available: http://users.ecs.soton.ac.uk/phl/ctit/ho1/node2.html.

Nilsson, N. 1998. Artificial Intelligence; A new synthesis. Mogan Kaufmann Publishers. San Francisco.

O'Brien, J.A. 2001. Sistemas de Información Gerencial. Irwin MacGraw-Hill.

ANNEX A
Database schema for the Gabing Model

```
-- Database: `gabingesdb`

CREATE TABLE `attributes` (
  `id_attr` int(6) NOT NULL auto_increment,
  `name_attr` varchar(100) NOT NULL,
  `type_attr` varchar(10) NOT NULL,
  PRIMARY KEY  (`id_attr`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

CREATE TABLE `categories` (
  `id_cat` int(6) NOT NULL auto_increment,
  `name_cat` varchar(100) NOT NULL,
  PRIMARY KEY  (`id_cat`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

CREATE TABLE `explain` (
  `id_explain` int(6) NOT NULL auto_increment,
  `name_explain` text NOT NULL,
  PRIMARY KEY  (`id_explain`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;


CREATE TABLE `group` (
  `id_group` int(6) NOT NULL auto_increment,
  `name_group` varchar(100) NOT NULL,
  PRIMARY KEY  (`id_group`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

Acta Apuroquia Vol.2 (1):5-13, 2010

```
CREATE TABLE `object` (
  `id_object` varchar(15) NOT NULL,
  `id_cat` int(6) NOT NULL,
  `name_object` varchar(120) NOT NULL,
  PRIMARY KEY  (`id_object`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `object_detail` (
  `id_objectdetail` int(6) NOT NULL auto_increment,
  `id_object` varchar(15) NOT NULL,
  `id_attrib` int(6) NOT NULL,
  `relation_operator` varchar(3) NOT NULL,
  `vardata` varchar(100) NOT NULL,
  `floatdata` float(15,4) NOT NULL,
  PRIMARY KEY  (`id_objectdetail`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;


CREATE TABLE `rules` (
  `id_rule` varchar(15) NOT NULL,
  `id_group` int(6) NOT NULL,
  `name_rule` varchar(120) NOT NULL,
  `id_explain` int(6) NOT NULL,
  PRIMARY KEY  (`id_rule`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `rule_detail` (
  `id_ruledetail` int(8) NOT NULL auto_increment,
  `id_rule` varchar(15) NOT NULL,
  `name_ruledetail` varchar(100) NOT NULL,
  `id_attrib` int(6) NOT NULL,
  `relation_operator` varchar(3) NOT NULL,
  `vardata` varchar(100),
  `floatdata` float(15,4),
  `logical_link` varchar(5) NOT NULL,
  `rule_link` varchar(15) NOT NULL,
  PRIMARY KEY  (`id_ruledetail`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;


CREATE TABLE `rule_link` (
  `id_rulelink` int(6) NOT NULL auto_increment,
  `id_rule` varchar(15) NOT NULL,
  `logical_link` varchar(5) NOT NULL,
  `id_explain` int(6) NOT NULL,
  PRIMARY KEY  (`id_rulelink`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```