4. DESIGN OF THE CREDIT ANALYSIS SHELL (CAS)

The Shell is aimed at supporting the credit granting process in an intelligent manner. The major objective in the design of the shell is to ensure that the system allows the developer of ES to incorporate his/her expertise in this field in an easy way and, therefore, be able to easily customize the knowledge base. CAS relies on a Relational Data base Manager for data storage as well as knowledge representation. The design of CAS very much differs from conventional ES shells like RuleMaster, M1 etc. in different aspects.

CAS uses ORACLE as the Relational Data Base Manager which is a popular RDBMS available in wide range of platforms. CAS uses Pro*C which is a module of ORACLE. Pro*C facilitates to use the power of Fourth generation language SQL and third generation language C.

The general structure of an Expert System consists of a Knowledgebase, an inference engine and user interface. The knowledge base can be represented as in the forms of IF-THEN rules, semantic nets, frames etc. But in the case of credit rating analysis IF-THEN rule structure is more suitable. The knowledge base of
CAS is represented in the form of IF-THEN rules only. The general form of IF-THEN rule will be as

IF condition THEN action

This is quite similar to the IF-THEN statements in conventional high level languages like C, Pascal. Here action may be giving the decision of the ES i.e finding the goal variable or updating another variable in ES work area or modifying other databases.

The inference engine is responsible for the reasoning of the knowledge base thus far delivering its decisions. The knowledge base in the CAS is represented as tables using Relational Data Base Manager. Also the facts Customer Data Base in this case are stored as tables. Interface engine is designed using PL/SQL and C language.

4.1. Rules Representation As Tables In The Shell

A rule is a meta object i.e. it can be thought of as a composite of smaller objects. It can be created, updated and deleted just like any entity in database. A scheme of successive and stepwise refinement to decompose the rule into smaller
objects down to the atomic level is employed. A rule essentially consists of two components, condition component and result component. In this system the condition component consists of a simple condition or a number of simple conditions connected with AND operator. The result component a result, a weight or a probability is atomic. The action component is simple to implement. It consists of rule identifier SCORE and experts comments on that rule. The condition component, if it consists of multiple conditions, it can be broken into smaller or simple conditions.

These simple conditions share a common structure. They define a range for a continuous state variable or a point value for a discrete state variable. These state variables are stored in the databases. The range is defined by Upper Limit (UL) and Lower limit (LL) parameters which will have the same values a point for discrete variables.

e.g : credit ratio is a continuous variable which has a lower value and upper value

creditrate is between 1.5 and 2.0

payment record of the customer can be described as poor, average, good etc.
if payment is between good and good

Fig. 4.1. Representation of rules as tables
Three tables are essentially needed to hold a set of rules. e.g. to calculate the financial strength of a customer. In the same way three tables are needed to store rules for other attributes like geographical location of the business, Payment record of the customer etc.

CAS supports the use of the fuzzy logic theory of knowledge representation developed by Lofti Zadeh (1965) of the university of California. The Fuzzy logic deals with the problem of uncertainty, an important issue in expert systems. The developer of the expert system has given the option to use fuzzy logic in knowledge base.

CAS prompts the user to enter details of the state variable that are being used in the rule base. For each state variable it stores, information as to how many regions (high, very high, low, very low etc.) it can be divided. Also it stores lower limits and upper limits for each region.

e.g. Credit ratio is a state variable used to calculate the financial strength of the customer. This state variable can be broadly divided into five regions Very High(VH), High(H), Medium(M), Low(L), Very Low(VL).
The ranges of different regions are

VL 0.0 to 0.8
L 0.6 to 1.2
M 1.6 to 1.8
H 1.6 to 2.4
VH 2.2 to 3.1

Fig. 4.1 Dividing a state variable into regions

CAS stores a membership function for each state variable. The details of how to select a membership function for different variables based on the density distribution function is discussed in the next chapter. It takes the information of membership functions used for different state variables. All this information is stored in a relational table called FACTOR.
The first relational table FACTOR consists of the details of different factors used to calculate particular attribute's score. The fields in the table used are factor_name, factor_type, description of the factor, a relational table name in the customer database from which the value of the factor can be accessed, a field name by which the value of the factor can be retrieved and a flag to show whether that factor is selected or not. Valid types of factors in CAS are string type and number type and other field is current value of the factor at runtime. The cur_val is accessed from customer database using the table name and field name and customer code entered by the user. The other fields used in the Factor table are used to find the fuzzy value of the state variable. Those are number of regions that the state variables can be divided into the lower limits and upper limits for each region. The maximum no. of regions that the state variable can be divided into in CAS are 9. And other columns in the table holds the membership function for different regions.

In CAS the following SQL DDL statements are used to create factor table.

EXEC SQL CREATE TABLE FACTOR
(
  factor_name varchar(32) primary key,
  factor_type char(5) not null,
)
Here factor name is taken as primary key so it is unique and it should not be null. Next table has simple conditions of different rules. It stores the ranges of factors and a range id_value or address of the simple condition. It consists of the fields factor_name which refers in FACTOR table, lower value, upper value and address or range_id. The first three fields forms the simple rule as IF factor_name
IN BETWEEN low_val AND upper_val. Here, for continuous factor variables, lower and upper vaules can be defined.

To indicate infinite lower value or infinite upper value a special character "$" is used.

e.g. : if bankbalance 1000000

it can be represented as

range_id factor_name low_val upper_val

1020 bankbalance 1000000

Address field or range_id maintains identification number for each simple condition. This field is primary key for this relation table. For discreate values like name, station etc. Lower and upper values are the same. The following statement creates this table.

EXEC SQL CREATE TABLE RANGES

( range_id char(5) primary key, factor_name varchar(32) references FACTOR.factor_name, low_val varchar(32) not null, high_val varchar(32) not null );
Here both low_val and high_val are represented as strings only. But while reasoning is being done they will be interpreted as numbers if the type of the factor is num, else they will be treated as strings only.

RESULT table is to store the action component of the rule. It consists of the fields Rule number, factor_name, comments of financial experts and addresses of the simple conditions of the rule, and a flag to indicate validity of the rule. CAS uses the following SQL statement to create RESULT table

EXEC SQL CREATE TABLE RESULT

(
    ruleno number(6,0) primary key,  
    factor_name varchar32 not null,  
    result_val varchar80 not null,  
    expert_text varchar500,  
    sc1   number(6,0) references RANGES_range_id,  
    sc2   number(6,0) references RANGES_range_id,  
    sc3   number(6,0) references RANGES_range_id,  
    ...  
    sc12  number(6,0) references RANGES_range_id,  
);
Here sc1, sc2 ... sc12 are simple condition address. This value gives the address of simple condition in the RANGES table. This allows fast and simple way to access simple conditions of the rule.

And other table CONST.DAT consists number of rules and next possible address of simple condition in RANGES table. This allows easy modifications to the knowledge base. In CAS the final credit rating is given by calculating the different attributes which effect credit worthiness of the customer. It can be synthesized in terms of 5 broad categories. Within each category different factors are to be assessed.

An illustrated list of such categories

1. Financial Strength
2. Customer Background
3. Payment Record
4. Business potential and Frequency
5. Geographical Location
CAS calculates scores for each category and based on these different category score values the final credit rating is decided. To calculate final credit rating CAS maintains another set of rules.

4.2 Exceptional rules

CAS maintains a separate set of relational tables to handle exceptional rules. Exceptional rules deal with some special cases to decide credit rating of the customers. Irrespective of the normal conditions and normal state variables values based on some values like high rank of the company, high assets of the customer etc., credit rating for those customers can be judged. Exceptional rule-base deals with such cases. All exceptional rules are kept separately in the following three different tables EX_FACTOR, EX_RANGES, EX_RESULT.

The inference engine first reasons on the exceptional rule-base which is as described above.

e.g. of some exceptional rules

IF rank_of_firm 3 THEN credit_rat 100

IF current_ratio 100 THEN credit_rat 100
IF cust_name "TELCO" credit_rat 100

**Structure of knowledge base**

![Diagram of knowledge base](image)

CAS at the starting of the designing of the expert system for credit rating analysis, prompts the user to select built-in categories or it allows the developer to define his/her own category names. And CAS also maintains an exceptional rulebase which consists of rule related to different variables or factors in different categories or attributes. The relational tables for these rules will be same as above explained tables, except additional field attribute_name will be included.
4.3. Representing rules with disjunctive conditions

If there is an OR disjunctive in the rule then that rule must be represented as two rules. Here MQ stands for Market Quality.

e.g.:

bankbalance from 100000 to 1000000

OR experience from 5 to 10

AND MQ is excellent

If bankbalance from 100000 to 1000000

AND MQ is excellent

If experience from 5 to 10

AND MQ is excellent

4.4. Customer Database

It consists of several relational tables which consists of all available data pertaining to the firm’s customers. This data base has access not only to ES but
also to any other application. Generally customer database consists of the tables like customer basic data.

- Bank references
- Pay habits
- Management Quality categories
- Trade references
- Balance Sheets
- Income statement etc.

At the time of inference process the current values of the customer are accessed by the shell's inference engine, from this database.

4.5. Knowledge Base and Inference Process

For the above described knowledge base representation scheme the inference process is described below. At runtime all the current values of different selected factors are accessed from the customer database. The table name and the field name are used to find out different factors values. The accessing process can be sped-up by creating indices on customer-code on different customer database tables. Then from the result table each rule is selected and examined for whether it
can be fired or not. To fire a rule all factors involved in that condition of the rule must have a current value. If value is not found then the result component of all rules where that factor name is used is searched for. Then the rule is fired. If it can be fired then keep that value in work area, else continue the process. If one factor rule is not available then go to next rule. Continue the same process.

If we get more than one score value the CAS considers the highest one. While evaluating the rule, first it retrieves the sample condition address in RANGES using that it checks in RANGES. If any sample condition fails then it stops checking for other sample conditions and passes to the next rule. If fuzzy logic is involved to evaluate the rules, then the regions like VH, H etc. under which the values come, along with that the degree of membership within those regions is determined. This degree of membership value is calculated using the membership function provided by the developer. All the information state variable name, cur_value, regional, degree_of_membership1, region2, degree_of_membership2 are stored in a separate data structure state_var_info.

For all the state variables involved in the conditions of the rule, the above data is calculated and stored in state_var_info. If the rule fires successfully then the resultant degree of membership of the action component state variable in that region is calculated as follows.
The general form of the rule is

\[
\text{If current\_ratio is VH}
\]

\[
\text{AND liq is H}
\]

\[
\text{THEN credit\_rat VH}
\]

In this rule credit\_rat comes under region of VH (Very High). According to Lafti Zadeh theory of fuzzy logic the degree of the membership is given by minimum degree of membership value of the state variables in condition component which are connected with AND. All these simple conditions are retrieved from the RANGES table. The resultant component is retrieved from the RESULT table.

State variables in the conditions may fall into more than one region. So more than one rule can be fired for the same set of state variable values. The results of all the rules that are fired for a set of state variable values are stored in a data structure rule\_status\_info. It maintains the data state var name, region name, fuzzy value, crisp value. Crisp value is calculated from inverse membership function.
The fuzzy value of the resultant component is to be transformed into crisp value. This process is called defuzzification. Defuzzification can be done in different methods. CAS uses the average weighted method. According to this method average of all the crisp values is found. If this is an intermediate state variable, information is written into state_var_info table which maintains information about all state variables. Again fuzzy values for this state variable is calculated using membership function provided and written into state_var_info table. If it is the goal variable i.e. credit rating information is written to the output file. For all the rules that are fired successfully CAS maintains a flag to indicate whether that is fired or not. Flag will be assigned ‘Yes’ for all the rules fired successfully. That default value for flag is ‘No’. The process is continued till it reaches a goal variable or no rule fires successfully in a pass.

If all the simple conditions of a rule are true then the factor name in the action component is checked. If it is the goal variable then it outputs the score value. If it is not a goal variable certain factor is updated in the work area and the process of checking the rules starts from the beginning of the rule file. This process is done until the score value is found or no modifications to work area are done. While the inferencing process is going it keeps track of the sequence of the rule numbers fired and the actions taken.
For each category or attribute like customer background, business potential etc., score values are calculated and stored in another table ATTRIB-FACTOR, which stores the defaults of attributes which are similar to FACTOR table previously explained. And other two tables ATTRIB-RANGES, ATTRIB-RESULT store the rules to calculate final credit rate of the customer.

The inferencing process is shown in Fig 4.3.

Inferencing process

Fig. 4.3
4.6. User Interface

To create user interface CAS uses C language. Entering the knowledge base and executing the expert system is done easily in CAS. It provides different menus at each stage. The main menu of CAS consists of two options one is DESIGN and the other is RUNTIME. To DESIGN the system first the developer has to enter all factors and their details for each category. Next step is to enter rules. System prompts the user to enter rules in an easy fashion. In all the steps appropriate instructions to the developer are given using message boxes. Also it gives appropriate error messages. User need not remember the syntax of the rule always.