Data Modeling of Knowledge Rules: An Oracle Prototype

Rajeev Kaula
Computer Information Systems Department, Missouri State University, Springfield, MO 65897 (USA)
E-Mail: RajeevKaula@missouristate.edu

***Abstract*—Knowledge rules are outlined declaratively in a knowledge base repository. Each rule is created independently for storage in the repository. This paper provides an approach to apply the techniques of traditional entity-relationship data modeling to structure the knowledge rules for storage as a database schema in a relational database management system. Utilization of entity relationship model and relational database for modeling knowledge rules provides for a more standardized mechanism for structuring knowledge rules. Storage of knowledge rules in a relational database shall also bring about improved integration with business applications, besides having the availability of services provided for transactional database applications. The paper utilizes the Oracle database for illustrating the application of the concepts through a sample set of knowledge rules. The approach is explained through a prototype in Oracle's PL/SQL Server Pages.**

***Index Terms*—Data Modeling, Knowledge Rules, Expert Systems, Knowledge Base, Entity-Relationship Diagram, Relational model.**

I. Introduction

Knowledge rules (or production rules) are the primary mechanisms to define knowledge in rule based systems like expert systems or knowledge-based systems [4, 5, 10, 11, 12, 13, 17, 21, 25, 27, 28, 32]. Such rules typically express decision-making guidelines. Each knowledge rule is written declaratively in constraint-action terminology represented as IF constraint THEN action statements. A constraint is some condition, while the action clause reflects the decision or advice. Figure 1 shows an example of a knowledge rule that describes a set of constraints applicable for approving a loan application.

```
IF      credit risk is High AND
        debt to income  is less than 40% OR
        loan requested is less than $50,000
THEN    approve with 5% APR
```

Figure 1. Sample Knowledge Rule

Each rule is independently outlined in the knowledge base repository. In general the structuring and storage of knowledge rules is done through special programming languages like Prolog or Lisp [26, 29] or some proprietary development environments like CLIPS [9], and JESS [7]. This paper outlines an approach to structure knowledge rules as a database schema for storage in relational databases using traditional entity relationship (ER) modeling techniques. As relational database and the associated language SQL are widely considered the ANSI/ISO standard for data storage and manipulation (viz. SQL:2008), data modeling of knowledge rules for storage as a relational database schema provides a more standardized structure for knowledge base (repository). Besides, representation of the knowledge rules as a relational database schema enables utilization of SQL for rule definition, maintenance, and manipulation.

Even though there have been attempts toward integration of knowledge base and database, such attempts have traditionally focused on (i) improving the database working in the form of intelligent databases [1, 2, 18, 20, 22, 23, 30, 31], or (ii) using knowledge based techniques to extract meaningful data from databases in the form of knowledge discovery [6, 8, 15, 24]. Whereas intelligent databases deal with the utilization of artificial intelligence techniques to capture the heuristics needed to control data in databases, the knowledge discovery approach involves utilization of artificial intelligence techniques to discover new knowledge in the form of data mining. Modeling of knowledge rules as a knowledge repository in relational database is an alternative approach to structure knowledge rules and enhance its utilization or integration with application development.

The modeling of knowledge rules as relational database schema is now outlined in the following sections. First the entity relationship concepts for modeling knowledge rules are outlined. This is followed by a prototype that illustrates the entity relationship modeling through a sample set of knowledge rules, along with their transformation and retrieval from an Oracle database. The approach is illustrated on an Oracle 11g database through a prototype in PL/SQL Server Pages [3, 14]. PL/SQL server pages is a server-side scripting approach for developing database driven dynamic Web pages. The PL/SQL server page uses Oracle's primary database language PL/SQL as a scripting language along with HTML to generate database driven Web pages. Even though the prototype utilizes Oracle technology, due to the standardization of relational database concepts, such modeling and manipulation can be accomplished through any other relational database product like MySQL, SQL Server, etc.

II. Relational Model Schema For Knowledge Rules

The relational model schema for knowledge rules begins by modeling the structure of knowledge rules through entity relationship modeling followed by their transformation into a relational model. The modeling and transformation process consists of the following elements: (i) subject area schema specification, (ii) entity type structure specification, (iii) entity relationship specification, (iv) relational table representation, and (v) sharing of subject area schemas. Each element builds on one another.

A. Subject Area Schema Specification

Modeling of knowledge rules begins with the concept of categorizing organizational knowledge into subject areas. A subject area is the decision making area of business. Each subject area contains knowledge rules specific to its domain of working. For example, there could be customer loan subject area, sales analysis subject area, and so on.

While the collection of knowledge rules within a subject area provide the action or decision support for that area, from a data modeling perspective such rules define the schema of knowledge belonging to the subject area. In other words, each subject area is a database schema supporting the knowledge as defined through its knowledge rules. So, for instance, there could be a database schema for customer loan subject area knowledge rules, another one for sales analysis subject area knowledge rules, and so on. The specification of entity types within a subject area schema is outlined now.

B. Entity Type Structure

The knowledge rules within a subject area are represented through a collection of entity types. Such entity types essentially follow the abstract structure of a knowledge rule statement as shown in Figure 2. In the figure each "constraint-i operator value" clause is some constraint, the "AND/OR" entries are logical operators joining constraint clauses, and the "subject area action" clause is some action representing the decision when the constraint conditions are true.

IF	constraint-1 operator value AND/OR
	constraint-2 operator value AND/OR ...
THEN	subject area action

Figure 2. Abstract Structure of Knowledge Rule

Each constraint clause is represented through individual constraint entity types. Each constraint entity type will consist of three attributes as shown in Figure 3. The "Constraint Name" entry that defines the name of the constraint clause entity type is the name of the constraint entry in the constraint clause of the knowledge rule statement. The "Constraint ID" attribute is the primary key, the "Operator" attribute is the condition operator in the constraint clause, while the "Constraint Value" attribute is the value assigned to the constraint condition.

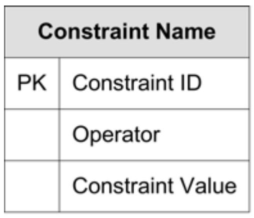


Figure 3. Constraint Entity Type

Instances of the constraint entity type are the individual constraint clauses in an associated knowledge rule statement. For example, consider two knowledge rules pertaining to subject area "customer loan" as shown in Figure 4.

Rule 1:	IF	credit risk is High OR
		loan requested is greater than $40000
	THEN	reject
Rule 2:	IF	credit risk is Low AND
		loan requested is less than $40000
	THEN	approve with 5% APR

Figure 4. Sample Customer Loan Knowledge Rules

The modeling of the constraint clause is shown in Figure 5. In the figure, part (a) shows the structure (definition) of constraint entity type, while part (b) shows the entity instances pertaining to the different clauses for the constraint in the two knowledge rule statements.

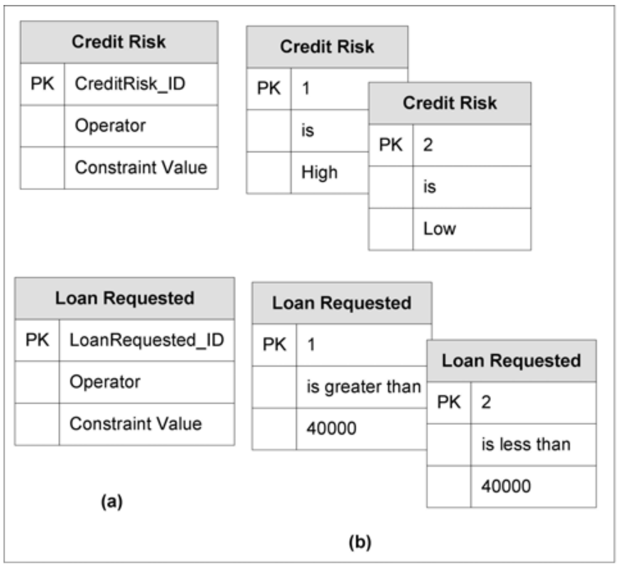
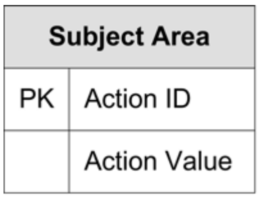


Figure 5. Constraint Entity Instances for Customer Loan Example

The subject area action clause is represented through a subject entity type. This entity type consists of two attributes as shown in Figure 6. To ensure symmetry within the modeling process, the subject entity type is named after the subject area. The "Action ID" attribute is the primary key, while the "Action Value" attribute is the value assigned to the subject area action entry in the associated knowledge rule statement. So, for instance if the knowledge rule belongs to "customer loan" subject area, then the subject entity type would be titled "customer loan."



Subject Area	
PK	Action ID
	Action Value

Figure 6. Action Entity Type

Instances of the subject entity type are the subject area action clauses within different knowledge rule statements. For example, consider the knowledge rules of Figure 4 to outline the details of subject entity type. Since the subject area for the knowledge rules is "customer loan" the subject entity type is also named "customer loan." Figure 7 shows the modeling of the subject area action clauses for these knowledge rules. Part (a) of the figure shows the structure (definition) of subject (Customer Loan) entity type. Part (b) of the figure shows the entity instances pertaining to the different clauses for the subject area action values in the associated knowledge rule statements.

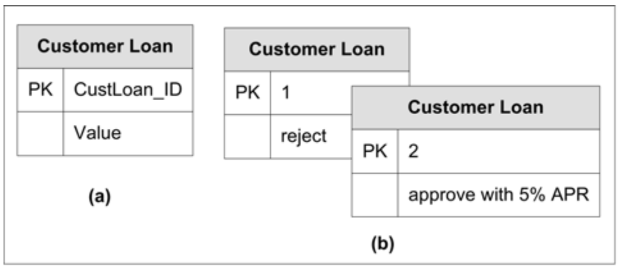


Figure 7. Action Entity Instance for Customer Loan Example

C. Entity Relationship Specification

The various entity types of a knowledge rule will have binary relationship with the subject entity type. The binary relationship will be one-to-many (1:N) as shown in Figure 8. The logical operator binding the constraint clauses within a knowledge rule shall become the relationship attribute of the binary relationship between the constraint entity type and the subject entity type. The minimum cardinality is optional to mandatory from the constraint entity type to subject entity type. Each instance of constraint entity type now will be associated with one or more subject entity instances. On the other hand, the subject entity instances may optionally be associated with different constraint entity instances.

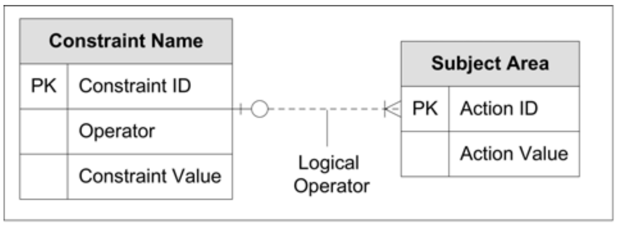
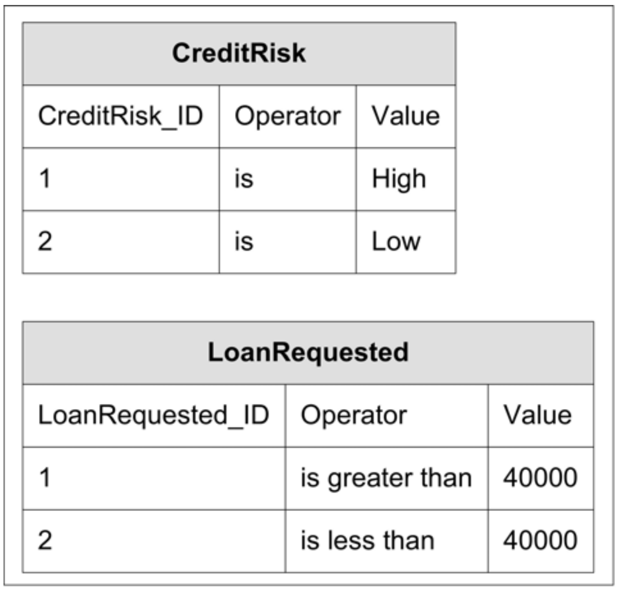


Figure 8. Knowledge Rule Entity-Relationship Model

The 1:N relationship between the constraint and subject entity types binds the constraint entity instances with the subject entity instances to represent a complete knowledge rule statement. Further, as knowledge rules are structured into distinct entity types, the entity relationship model of a subject area represents a database schema of entity types. For example, there can be a schema of entity types for the customer loan subject area representing its various knowledge rules. The transformation of entity relationship model of a subject area into a relational model is outlined now.

D. Relational Table Representation

Each constraint entity type is represented as a separate table in a relational database. For example, Figure 9 which is an extension of Figure 5 shows the table structure of the credit risk and loan requested constraint entity types.



CreditRisk

CreditRisk_ID	Operator	Value
1	is	High
2	is	Low

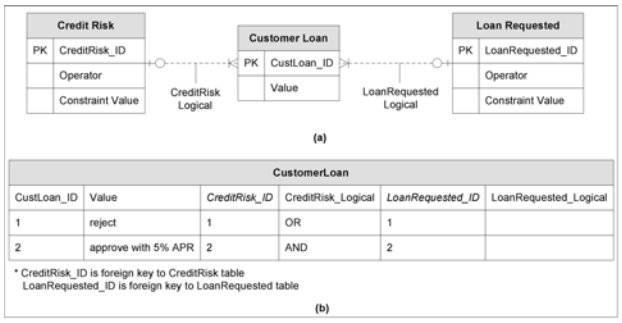
LoanRequested

LoanRequested_ID	Operator	Value
1	is greater than	40000
2	is less than	40000

Figure 9. Database Tables for Customer Loan Example Constraints

Similarly the subject entity type will be represented as a separate table in the relational database. For example, Figure 10 extends Figure 7 through the table structure of the Customer Loan subject entity type. The CreditRisk_Logical attribute represents the logical operator value that binds credit risk constraint with loan requested constraint for this rule. The loanrequested_logical attribute being associated with the last constraint clause within the rule structure will be null. Part (a) of the figure shows the 1:N relationship between the subject (Customer Loan) entity type with the constraint Credit Risk and Loan Requested, while part (b) shows the table structure of the Customer Loan entity type. The foreign key CreditRisk_ID and LoanRequested_ID in Customer Loan table represents the 1:N relationship with the CreditRisk and LoanRequested tables respectively. Included in the CustomerLoan table is also the value of the logical operators.

The database schema of constraint entity types tables and subject entity type table represents a collection of knowledge rules for the subject area in a relational database.



CustomerLoan					
CustLoan_ID	Value	*CreditRisk_ID*	CreditRisk_Logical	*LoanRequested_ID*	LoanRequested_Logical
1	reject	1	OR	1	
2	approve with 5% APR	2	AND	2	

* CreditRisk_ID is foreign key to CreditRisk table
LoanRequested_ID is foreign key to LoanRequested table

Figure 10. Database Tables for Customer Loan Example

E. Sharing of Subject Area Schemas

Subject area database schemas' can share entity types among each other. Such sharing represents the chaining of knowledge among knowledge rules belonging to different subject areas. First type of sharing occurs when the constraint in one subject area set of knowledge rules is also a constraint in another subject area knowledge rules. For example, consider two knowledge rules in two subject areas. The first rule belongs to the credit risk subject area with three constraint clauses as shown below:

IF	credit score is less than 600 AND
	debt to income is greater than 40% OR
	house is investment
THEN	Credit Risk is High

The second rule belongs to the customer loan subject area with three constraint clauses as shown below.

IF	credit score is less than 600 AND
	loan to value is less than 80% OR
	loan requested is less than $40,000
THEN	Approve with 5% APR

The first constraint "credit score is less than 600" in both rules is the same. From a modeling perspective, the entity type representing this constraint is defined once in either of the two subject area schema, and then shared between the two subject area schemas. Such sharing of constraint entity types across subject area schemas can be viewed symbolically in Figure 11.

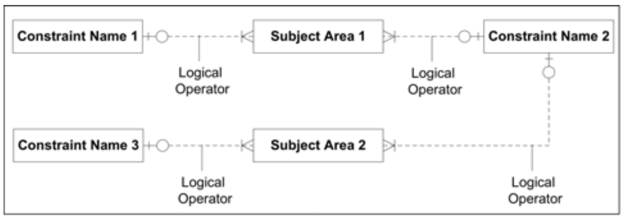


Figure 11. Sharing Constraints Entity Types among Subject Areas

Second type of sharing occurs when action value in one subject area set of knowledge rules serves as a constraint clause in another subject area set of knowledge rules. For example, consider two knowledge rules in two subject areas. The first rule belongs to the credit risk subject area with three constraint clauses as shown below:

IF	credit score is less than 600 AND
	debt to income is greater than 40% OR
	house is investment
THEN	Credit Risk is High

The second rule belongs to the customer loan subject area with three constraint clauses as shown below.

IF	credit risk is High AND
	loan to value is less than 80% OR
	loan requested is less than $40,000
THEN	Approve with 5% APR

The action clause in credit risk subject area value is referred as a constraint "credit risk is High" in customer loan subject area. This reference in the customer loan subject area helps in the validation of the constraint value associated with a separate set of knowledge rules in credit risk subject area. From a modeling perspective, such reference is represented through the relationship between the subject entity types between the involved subject areas as shown symbolically in Figure 12.

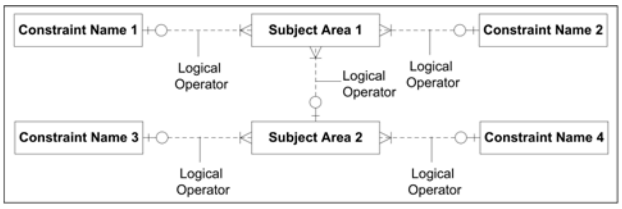


Figure 12. Sharing Action Entity Types among Subject Areas

III. Knowledge Rules Modeling Prototype

A prototype for modeling knowledge rules based on two subject areas belonging to finance discipline is outlined in this section. The proposed entity relationship model is transformed for storage in an Oracle database through the SQL language. The prototype also shows the retrieval of database stored knowledge rules in declarative format through a database procedure using the Oracle's PL/SQL database language.

A. Modeling Subject Area Knowledge Rules

The subject areas for the prototype are customer loan and credit risk. The credit risk knowledge rules are listed first, followed by the entity relationship diagram for the credit risk subject area as shown in Figure 13.

<u>Credit Risk</u>
Rule 1:

IF	credit score is less than 600 AND
	debt to income is greater than 40% OR
	house is investment
THEN	High

Rule 2:

IF	customer credit score is more than 600 AND

	debt to income is less than 40% OR
	house is primary
THEN	Low

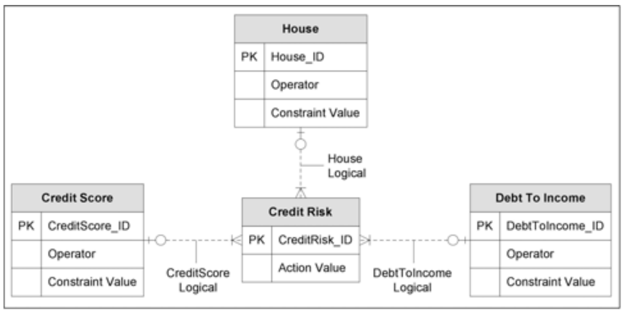


Figure 13. Credit Risk Knowledge Rules Entity-Relationship Model

The customer loan knowledge rules are listed now, followed by the entity relationship diagram for the customer loan subject area as shown in Figure 14.

Customer Loan

Rule 1:

IF	credit risk is High AND
	loan to value is greater than 80% OR
	loan requested is greater than $40,000
THEN	Reject

Rule 2:

IF	credit risk is High AND
	loan to value is less than 80% OR
	loan requested is less than $40,000
THEN	Approve with 5% APR

Rule 3:

IF	credit risk is High AND
	loan to value is 100% OR
	loan requested is greater than $75,000
THEN	Reject

Rule 4:

IF	credit risk is Low AND
	loan to value is less than 80% OR
	loan requested is greater than $150,000
THEN	Approve with 4.5% APR

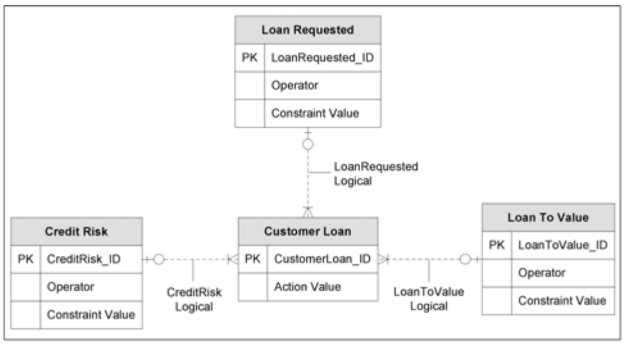


Figure 14. Customer Loan Knowledge Rules Entity-Relationship Model

The prototype also illustrates the second type of sharing between the two subject areas. The *credit risk* constraint entity type within the *customer loan* schema is itself a separate subject area. Consequently, there is sharing of credit risk subject entity type with customer loan subject entity type. The composite entity relationship model for the two subject area schemas is shown in Figure 15.

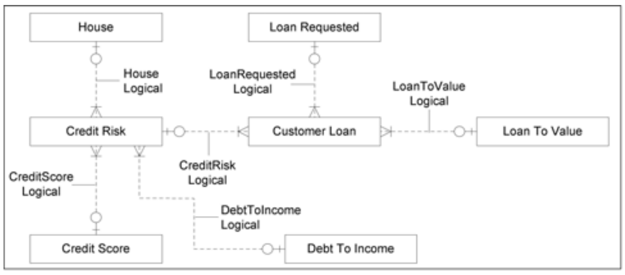


Figure 15. Sharing among Customer Loan and Credit Risk Subject Areas

B. *Relational Model Representation*

The entity relationship model of the two subject area schemas is transformed into a relational model for storage in a relational database. The table structure along with the row values for the various entity types is now outlined. To facilitate understanding of the concepts, the *credit risk* schema tables are outlined first, followed by the *customer loan* schema tables.

TABLE I.
CREDITSCORE TABLE

CreditScore_ID	Operator	Value
1	is less than	600
2	is more than	600

TABLE II.
DEBTTOINCOME TABLE

DebtToIncome_ID	Operator	Value
1	is less than	40%
2	is greater than	40%

TABLE III.
HOUSE TABLE

House_ID	Operator	Value
1	is	Investment
2	is	Primary

TABLE IV.
CREDITRISK TABLE (PART 1)

CreditRisk_ID	Value	*CreditScore_ID*	CreditScore_Logical	*DebtToIncome_ID*
1	High	1	AND	2
2	Low	2	AND	1

TABLE V.
CREDITRISK TABLE (PART 2)

CreditRisk_ID	DebtToIncome_Logical	*House_ID*	House_Logical

TABLE V.
CREDITRISK TABLE (PART 2)

CreditRisk_ID	DebtToIncome_Logical	*House_ID*	House_Logical
1	OR	1	
2	OR	2	

where CreditScore_ID is foreign key to CreditScore table; DebtToIncome_ID is foreign key to DebtToIncome table; and, House_ID is foreign key to House table.

TABLE VI.
LOANTOVALUE TABLE

LoanToValue_ID	Operator	Value
1	is greater than	80%
2	is	100%
3	is less than	80%

TABLE VII.
LOANREQUESTED TABLE

LoanRequested_ID	Operator	Value
1	is greater than	40000
2	is less than	40000
3	is greater than	75000
4	is greater than	150000

TABLE VIII.
CUSTOMERLOAN TABLE (PART 1)

CustomerLoan_ID	Value	*CreditRisk_ID*	CreditRisk_Logical	*LoanToValue_ID*
1	Reject	1	AND	1
2	Approve with 5% APR	1	AND	3
3	Reject	1	AND	2
4	Approve with 5% APR	2	AND	3

TABLE IX.
CUSTOMERLOAN TABLE (PART 2)

CustomerLoan_ID	LoanToValue_Logical	*LoanRequested_ID*	LoanRequested_Logical
1	OR	1	
2	OR	2	
3	OR	3	
4	OR	4	

whereCreditRisk_ID is foreign key to CreditRisk table; LoanToValue_ID is foreign key to LoanToValue table; and, LoanRequested_ID is foreign key to LoanRequested table.

C. *Web Prototype*

The relational schema tables of the two subject areas are installed in an Oracle 11g database. Once the subject area schema is in the database, it can be queried for decision support. The prototype at this stage performs a simple retrieval of knowledge rules. The results of the retrieval in the form of selected knowledge rules is displayed in declarative format. The prototype consists of two Web pages. The interaction of the two Web pages within the Web architecture is shown in Figure 16.

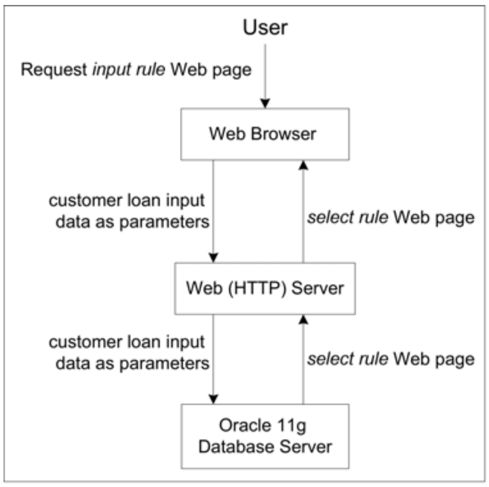
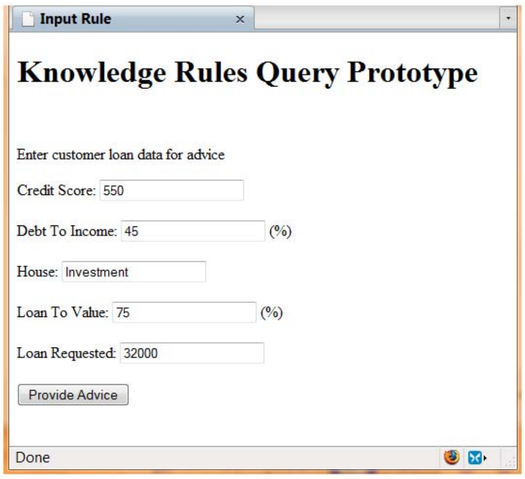

User
Request *input rule* Web page
Web Browser
customer loan input data as parameters
select rule Web page
Web (HTTP) Server
customer loan input data as parameters
select rule Web page
Oracle 11g Database Server

Figure 16. Prototype Web Architecture

The user requests for the first Web page titled “input rule.” This page displays a Web form with text boxes to input data needed to search for valid knowledge rule in the database (knowledge) repository. Figure 17 shows a view of the input rule Web page.



Knowledge Rules Query Prototype

Enter customer loan data for advice

Credit Score: 550

Debt To Income: 45 (%)

House: Investment

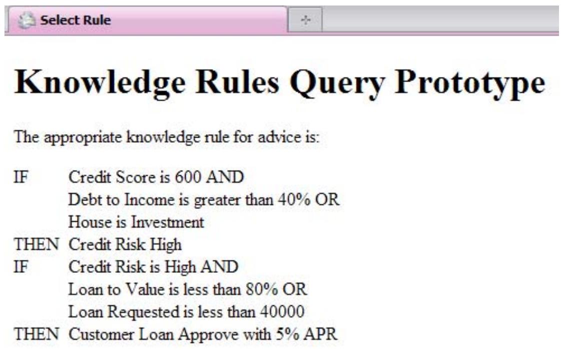
Loan To Value: 75 (%)

Loan Requested: 32000

Figure 17. input_rule Web Form

The *input rule* Web page is generated through a Web procedure titled "input_rule_web." Once the user completes the Web form, the “Provide Advice” button in clicked which enables the browser to forward the form input data to the second Web page in the database through the Web (HTTP) Server.

The second Web page is titled “select rule.” This Web page receives the form input data, completes the database processing for searching the valid knowledge rule, and returns the outcome to the Web server, which in turn forwards the page to the Web browser. Figure 18 shows a view of the output using the inputs entered in the first Web page.



Knowledge Rules Query Prototype

The appropriate knowledge rule for advice is:

```
IF      Credit Score is 600 AND
        Debt to Income is greater than 40% OR
        House is Investment
THEN    Credit Risk High
IF      Credit Risk is High AND
        Loan to Value is less than 80% OR
        Loan Requested is less than 40000
THEN    Customer Loan Approve with 5% APR
```

Figure 18. Select rule Web output

The *select rule* Web page is generated through two Web procedures. The first Web procedure titled “select_rule_web” searches for the valid knowledge rule, while the second Web procedure titled “cust_loan_output_web” formats the selected knowledge rule in declarative format for display in the Web browser. Figure 19 shows the pseudocode logic for knowledge rule search strategy. The select_rule_web Web procedure is listed in Appendix-A, while the cust_loan_output_web Web procedure is listed in Appendix-B.

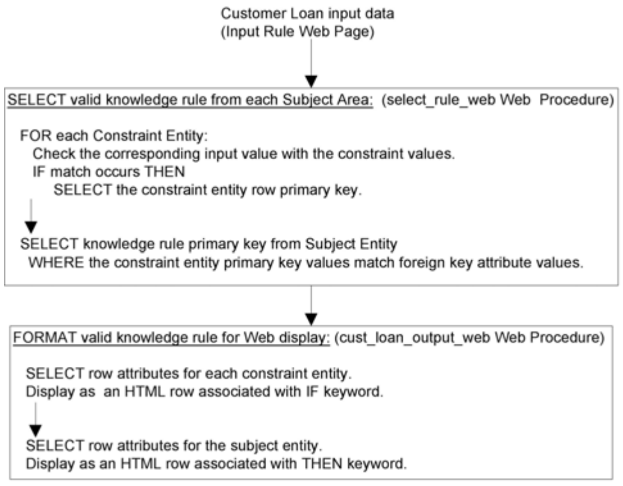

Customer Loan input data
(Input Rule Web Page)
SELECT valid knowledge rule from each Subject Area: (select_rule_web Web Procedure)
FOR each Constraint Entity:
Check the corresponding input value with the constraint values.
IF match occurs THEN
SELECT the constraint entity row primary key.
SELECT knowledge rule primary key from Subject Entity
WHERE the constraint entity primary key values match foreign key attribute values.
FORMAT valid knowledge rule for Web display: (cust_loan_output_web Web Procedure)
SELECT row attributes for each constraint entity.
Display as an HTML row associated with IF keyword.
SELECT row attributes for the subject entity.
Display as an HTML row associated with THEN keyword.

Figure 19. Select rule Web page logic

IV. Conclusion

Entity relationship model is generally utilized to model data in a transactional database or data warehouse [13, 16]. However, modeling of knowledge rules through an entity relationship model and storing them in a relational database provides for a standard based mechanism for structuring knowledge rules. Storage of knowledge rules as a knowledge base in a relational DBMS allows for the utilization of services similar to those provided for transactional database like conceptually centralized management, access optimization, recovery and concurrency controls, and so on.

The knowledge rules representation as a relational database schema can facilitate some additional features like:

1. Any access to knowledge can be restricted to only those rules that are pertinent to that user. This is similar to how access is restricted to data in a transactional database.
2. Rules shall be queryable and updatable through widely known SQL language in the form of (i) new rules can be added or dropped to keep the nature of knowledge rules current (ii) new constraints can be added or existing constraints can be dropped or modified, and (iii) even individual attributes can be modified since now the knowledge rules are stored as relational tables.
3. Rule consistency can be maintained with regard to its format and relationships.
4. Rules may be integrated with business or enterprise applications, wherein such applications shall always get current knowledge from the database.

As the relational database SQL concepts are standardized since SQL was adopted as a standard by the American National Standards Institute (ANSI) in 1986 and International Organization for Standardization (ISO) in 1987 (the current standard is SQL:2008), the data model (relational database schema) can be easily ported to all major enterprise DBMS like Oracle, SQL Server, DB2, MySQL, and so on. The manipulation of the schema through a database language as illustrated through the prototype will vary, even though the nature of such manipulations will be conceptually similar.

Further research is in progress to extend the modeling of knowledge rules. This involves (i) developing rule engine mechanisms to query rules for specific constraints, (ii) incorporate additional complexity in rule specification, (iii) techniques to link constraint values with transactional database, and (iv) exploring the notion of inferencing rule chains.

Appendix A Web Procedure to Search Valid Knowledge Rule

```
<%@ page language="PL/SQL"%>
<%@ plsql procedure="select_rule_web"%>
<%@ plsql parameter="cs_in" default="null"%>
<%@ plsql parameter="dti_in" default="null"%>
<%@ plsql parameter="h_in" default="null"%>
<%@ plsql parameter="ltv_in" default="null"%>
<%@ plsql parameter="lr_in" default="null"%>
<%@ plsql parameter="formsbutton1" default="null"%>
<%! cs_key integer; dti_key integer; h_key integer;
    cr_key integer; cr_value creditrisk.value%type;
    ltv_key integer; lr_key integer;
    cl_key integer; cl_value customerloan.value%type; %>
    <%
if cs_in < 600 then
    select creditscore_id into cs_key
    from creditscore
    where operator = 'is less than' and value = 600;
    else
    select creditscore_id into cs_key
    from creditscore
    where operator = 'is more than' and value = 600;
end if;
```

```
if dti_in < 40 then
    select debttoincome_id into dti_key
    from debttoincome
    where operator = 'is less than' and value = '40%';
else
    select debttoincome_id into dti_key
    from debttoincome
    where operator = 'is greater than' and value = '40%';
end if;
if h_in = 'Investment' then
    select house_id into h_key
    from house
    where value = 'Investment';
else
    select house_id into h_key
    from house
    where value = 'Primary';
end if;
select creditrisk_id, value into cr_key, cr_value
from creditrisk
where creditscore_id = cs_key and debttoincome_id = dti_key
and house_id = h_key;
if (ltv_in < 80) then
    select loantovalue_id into ltv_key
    from loantovalue
    where operator = 'is less than' and value = '80%';
end if;
if ((ltv_in > 80) and (ltv_in < 100)) then
    select loantovalue_id into ltv_key
    from loantovalue
    where operator = 'is greater than' and value = '80%';
end if;
if (ltv_in = 100) then
    select loantovalue_id into ltv_key
    from loantovalue
    where operator = 'is' and value = '100%';
end if;
if (lr_in < 40000) then
    select loanrequested_id into lr_key
    from loanrequested
    where operator = 'is less than' and value = 40000;
end if;
if ((lr_in > 40000) and (lr_in < 75000)) then
    select loanrequested_id into lr_key
    from loanrequested
    where operator = 'is greater than' and value = 40000;
end if;
if ((lr_in > 75000) and (lr_in < 150000)) then
    select loanrequested_id into lr_key
    from loanrequested
    where operator = 'is greater than' and value = 75000;
end if;
if (lr_in > 150000) then
    select loanrequested_id into lr_key
    from loanrequested
    where operator = 'is greater than' and value = 150000;
end if;
select customerloan_id, value into cl_key, cl_value
from customerloan
where creditrisk_id = cr_key and loantovalue_id = ltv_key
and loanrequested_id = lr_key;
cust_loan_output_web(cr_key, cl_key); %>
```

APPENDIX B WEB PROCEDURE TO DISPLAY IN DECLARATIVE FORMAT

```
<%@ page language="PL/SQL"%>
<%@ plsql procedure="cust_loan_output_web"%>
<%@ plsql parameter="cr_key" default="null"%>
<%@ plsql parameter="cl_key" default="null"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Select Rule</title>
</head>
<body>
<div align="left"><p><h1>Knowledge Rules Query
Prototype</h1></p>
<%!
    cursor cr_curs is
    select * from creditrisk
    where creditrisk_id = cr_key;
    cr_row cr_curs%rowtype;
    cursor cl_curs is
    select * from customerloan
    where customerloan_id = cl_key;
    cl_row cl_curs%rowtype;
    creditrisk_val creditrisk.value%type;
    loantovalue_op loantovalue.operator%type;
    loantovalue_val loantovalue.value%type;
    debttoincome_op debttoincome.operator%type;
    debttoincome_val debttoincome.value%type;
    loanrequested_op loanrequested.operator%type;
    loanrequested_val loanrequested.value%type;
    creditscore_val creditscore.value%type;
    house_val house.value%type; %>
<p>The appropriate knowledge rule for advice is:</p>
<table border="0" cellpadding="0" cellspacing="0">
<% for cr_row in cr_curs
    loop %>
<tr><td>IF</td>
<% if cr_row.creditscore_id is not null then
    select value into creditscore_val
    from creditscore
    where creditscore_id = cr_row.creditscore_id; %>
<td>Credit Score is <%=creditscore_val%> <%= '
'||cr_row.creditscore_logical %></td></tr>
<% end if;
    if cr_row.debttoincome_id is not null then
    select operator, value into debttoincome_op,
debttoincome_val from debttoincome
    where debttoincome.debttoincome_id =
cr_row.debttoincome_id; %>
<tr><td></td><td>Debt to Income <%=debttoincome_op%>
<%=debttoincome_val%> <%= '
'||cr_row.debttoincome_logical %></td></tr>
    <% end if;
    if cr_row.house_id is not null then
    select value into house_val
    from house
    where house_id = cr_row.house_id; %>
<tr><td></td><td>House is <%=house_val%></td></tr>
    <% end if; %>
<tr><td>THEN &nbsp</td><td>Credit Risk
<%=cr_row.value%></td></tr>
    <% end loop; %>
<tr><td></td><td></td></tr>
    <% for cl_row in cl_curs
    loop %>
<tr><td>IF</td>
    <%     if cl_row.creditrisk_id is not null then
    select value into creditrisk_val from creditrisk
    where creditrisk_id = cl_row.creditrisk_id; %>
<td>Credit Risk is <%=creditrisk_val %> <%= '
'||cl_row.creditrisk_logical %> </td></tr>
    <% end if;
if cl_row.loantovalue_id is not null then
    select operator, value into loantovalue_op, loantovalue_val
from loantovalue where loantovalue.loantovalue_id =
cl_row.loantovalue_id; %>
<tr><td></td><td>Loan to Value <%= loantovalue_op %>
<%=loantovalue_val%>
 <%= ' '||cl_row.loantovalue_logical %></td></tr>
<% end if;
if cl_row.loanrequested_id is not null then
    select operator, value into loanrequested_op,
    loanrequested_val  from loanrequested
    where loanrequested.loanrequested_id =
cl_row.loanrequested_id; %>
<tr><td></td><td>Loan Requested <%=loanrequested_op%>
<%=loanrequested_val%></td></tr>
    <%     end if; %>
    <tr><td>THEN &nbsp</td><td>Customer Loan
<%=cl_row.value%></td></tr>
    <% end loop; %>
    </table>
    </body>
</html>
```

REFERENCES

[1] S. Antony, D. Batra, and R. Santhanam, "The use of a knowledge-based system in conceptual data modeling," *Decision Support Systems*, vol. 41, pp. 176 - 188, 2005.

[2] E. Babiker, D. Simmons, R. Shannon, and N. Ellis, "A Model for Reengineering Legacy Expert Systems to Object-Oriented Architecture," *Expert Systems with Applications*, vol. 12, pp. 363-371, 1997.

[3] S. Boardman, M. Caffrey, S. Morse, and B. Rosenzweig, *Oracle Web Application Programming for PL/SQL Developers*, Upper Saddle River, NJ: Prentice-Hall, 2003.

[4] R.J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*, San Francisco, CA: Morgan Kaufmann, 2004.

[5] Y. Duan and P. Burrell, "Some issues in developing expert marketing systems," *Journal of Business & Industrial Marketing*, vol. 12, pp. 149-162, 1997.

[6] U. Fayyad and R. Uthurusamy, "Data Mining and Knowledge Discovery in Databases," *Communications of the ACM*, vol. 39, pp. 24-26, 1996.

[7] E. Friedman-Hill, *Jess in Action: Rule Based Systems in Java*, Greenwich, CT: Manning Publications, 2003.

[8] C. Gertosio and A. Dussauchoy, "Knowledge discovery from industrial databases," *Journal of Intelligent Manufacturing*, vol. 15, pp. 29-37, 2004.

[9] J. C. Giarratano and G.D. Riley, *Expert Systems: Principles and Programming*, Boston, MA: Course Technology, 1998.

[10] F. Gomez and C. Segami, "Semantic interpretation and knowledge extraction," *Knowledge-Based Systems*, vol. 20, pp. 51–60, 2007.

[11] Y. Guo, Z. Pan, and J. Heflin, "Choosing the best knowledge base system for large semantic web applications," in *Proceedings of the 13th international World Wide Web conference*, New York, NY, pp. 302 - 303, 2004.

[12] H. Hayes-Roth and N. Jacobstein, "The State of Knowledge-Based Systems," *Communications of the ACM*, vol. 37, pp. 27-39, 1994.

[13] R.D. Hull and F. Gomez, "Automatic acquisition of biographic knowledge from encyclopedic texts," *Expert Systems with Applications*, vol. 16, pp. 261–270, 1999 .

[14] R. Kaula, Oracle 11g: *Developing AJAX Applications with PL/SQL Server Pages*, New York, NY: Mc-Graw-Hill, 2008.

[15] Y. Kim and W.N. Street, "An intelligent system for customer targeting: a data mining approach," *Decision Support Systems*, vol. 37, pp. 215 - 228, 2004.

[16] R. Kimball, *The Data Warehouse Toolkit*, New York, NY: John Wiley & Sons, 1996.

[17] S. Liao, "Expert system methodologies and applications—a decade review from 1995 to 2004," *Expert Systems with Applications*, vol. 28, pp. 93-103, 2005.

[18] B. Lin, "An Overview of Intelligent Database," *Journal of Computer Information Systems*, vol. 33, pp. 8-12, 1993.

[19] M. Mannino, *Database Design, Application Development, and Administration* , New York, NY: McGraw-Hill, 2006.

[20] F. Manola, "Object-Oriented Knowledge Bases," *AI Expert*, vol. 5, pp. 46 - 57, 1990.

[21] Y. Ma, B. Jin, and Y. Feng, "Dynamic evolutions based on ontologies," *Knowledge-Based Systems*, vol. 20, pp. 98–109, 2007.

[22] B. Martin, A. Mitrovic, P. Suraweera, and A. Weerasinghe, "DB-Suite: Experiences with Three Intelligent, Web-Based Database Tutors," *Journal of Interactive Learning Research*, vol. 15, pp. 409-432, 2004.

[23] M.M.O. Owrang and F.H. Grupe, "Database Tools to Acquire Knowledge for Rule-Based Expert Systems," *Information and Software Technology*, vol. 39, pp. 607-616, 1997.

[24] S. K. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining*, Boca Raton, FL: CRC Press, 2004.

[25] J.B. Quinn, *Intelligent Enterprise: A Knowledge and Service Based Paradigm for Industry*, New York, NY: The Free Press, 1992.

[26] P. Seibel, *Practical Common Lisp*, New York, NY: Apress, 2005.

[27] Y.P. Shao, "The Infusion of Expert Systems in Banking: An Exploratory Study," *Expert Systems with Applications*, vol. 12, pp. 429-440, 1997.

[28] J.F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, New York, NY: Brooks/Cole, 2000.

[29] L. Sterling and E. Shapiro, *The Art of Prolog, Second Edition: Advanced Programming Techniques (Logic Programming)*, Boston, MA: The MIT Press, 1994.

[30] P. Suraweera and A. Mitrovic, "An Intelligent Tutoring System for Entity Relationship Modelling," *International Journal of Artificial Intelligence in Education*, vol. 14, pp. 375-417, 2004.

[31] Z. Yuanhui, L. Yuchang, and S. Chunyi, "A Connectionist Approach to Extracting Knowledge from Databases," in X.Liu, P. Cohen, and M. Berthold (ed.), *Advances in Intelligent Data Analysis*, Berlin, Germany: Springer-Verlag, pp. 465-475, 1997.

[32] O.M. Vasil'ev, D. P. Vetrov, and D. A. Kropotov, "Knowledge Representation and Acquisition in Expert Systems for Pattern Recognition," *Computational Mathematics and Mathematical Physics*, Vol. 47, pp. 1373–1397, 2007.