ONDERZOEKSRAPPORT NR 9312

Integration of a Decision Table System with a Relational Database Environment

by

Jan VANTHIENEN
Geert WETS
Integration of a Decision Table System with a Relational Database Environment

J. VANTHIENEN & G. WETS
Katholieke Universiteit Leuven
Department of Applied Economic Sciences
Dekenstraat 2, B-3000 Leuven (Belgium)
Phone: (0)16-28.58.09, Fax: (0)16-28.57.99, E-mail: fdban06@blekul11

Abstract

Recent years have shown a lot of research about the integration of knowledge based systems and databases, e.g. adding knowledge based facilities to traditional database management systems, or storing and manipulating knowledge using proven database techniques.

It has been reported earlier that knowledge, in the form of clauses or decision tables, can be stored in a relational database. In this paper it is examined how the relational approach may be used, not only in storing, but also in constructing, filling in, validating and executing decision tables, with special emphasis on building tables using the decision grid chart and on advanced features, such as: table contraction, validation and verification, rule based knowledge acquisition, etc.

This integration of a decision table system and the relational concept is studied in the context of an existing decision table engineering workbench, PROLOGA (PROCedural LOGic Analyzer), an interactive rule-based design tool for decision table construction and manipulation. Based on the correspondence between a decision table and a table in a relational database system, the major modelling issues of decision table construction and validation are examined and a possible integration of a decision table system and a relational database management system is suggested and evaluated.

Keywords

Knowledge based systems, decision tables, relational databases, knowledge acquisition, verification & validation, decision table construction, decision grid chart
1. Introduction

Recent years have shown a lot of research about the integration of knowledge based systems and databases, e.g. adding knowledge based facilities to traditional database management systems, or storing and manipulating knowledge using proven database techniques. It has been reported earlier that knowledge, in the form of clauses ([6]), or in the form of decision tables ([5], [7]), can be stored in a relational database. The decision table then embodies a set of clauses representing various combinations of possible condition values.

Storing and evaluating clauses (whether or not decision table based), however, is only one aspect. In this paper it is examined how the relational approach [2] may be used, not only in storing, but also in constructing, filling in, validating and executing decision tables. This integration of a decision table system and the relational concept is elaborated in the context of an existing decision table engineering workbench, with special emphasis on building decision tables using the decision grid chart and on advanced features, such as: table contraction, validation and verification, rule based knowledge acquisition, etc.

In section 2 the decision table representation is briefly described. Section 3 analyzes the correspondence between a decision table and a table in a relational database system. Based on these concepts, section 4 deals with the major modelling issues of decision table construction, validation and consultation. In section 5 a possible integration of a decision table system and a relational database system is presented. Finally our major findings are summarized.

2. Knowledge Representation Using Decision Tables

"A decision table is a table, representing the exhaustive set of mutual exclusive conditional expressions, within a predefined problem area." [11]

The tabular representation of the decision situation is characterized by the separation between conditions and actions, on one hand, and between subjects and conditional expressions (states), on the other hand. Every table column (decision column) indicates which actions should (or should not) be executed for a specific combination of condition states.
The entries part consists of columns (with condition states and action values) separated by a vertical line from the first different condition state. A column then contains a state for each condition or a contraction of states which yield the same result (possibly "irrelevant" ("-")) if this is the case for all states), followed by the resulting value for each action.

In this definition, the decision table concept is deliberately restricted to the so called single-hit table, where columns are mutually exclusive. Only this type of table allows easy checking for consistency and completeness. Each possible combination of conditions has to be included in one and only one table column. Some combinations of conditions, however, may be impossible and can be deleted from the table.

Fig. 1 shows an example of a decision table

<table>
<thead>
<tr>
<th>Type of book</th>
<th>hard cover</th>
<th>normal</th>
<th>pocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wholesaler</td>
<td>Y</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Quantity ordered</td>
<td>&lt; 5</td>
<td>&gt;=5</td>
<td>-</td>
</tr>
<tr>
<td>Special discount</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Free delivery</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
</tbody>
</table>

In order to make a meaningful use of decision tables possible, the decision table has to be defined clearly and must meet the important requirements of consistency and completeness. For these purposes, the decision table is defined as function.

- CS = \{CS_i\} (i=1..cnum) is the set of condition subjects;
- CD = \{CD_i\} (i=1..cnum) is the set of condition domains, with CD_i the domain of condition i, i.e. the set of all possible values of condition subject CS_i;
- CT = \{CT_i\} (i=1..cnum) is the set of condition state sets, with CT_i = \{S_{i,k}\} (k=1..n_i) an ordered set of n_i condition states S_{i,k}. Each condition state S_{i,k} is a logical expression concerning the elements of CD_i, that determines a subset of CD_i, such that the set of all these subsets constitutes a partition of CD_i (completeness and exclusivity of the condition states);
- \( \text{AS} = \{ \text{AS}_j \} \) (j=1..anum) is the set of action subjects;
- \( \text{AV} = \{ \text{AV}_j \} \) (j=1..anum) is the set of action value sets,
with \( \text{AV}_j = \{ \text{true (x), false (-), nil (.)} \} \) the set of action values, which is, in first instance, equal for every action subject, for reasons of consistency checking.

The (expanded) decision table is a function from the cartesian product of the condition states \( \text{CT}_1 \times \ldots \times \text{CT}_{\text{anum}} \) to the cartesian product of the action values \( \text{AV}_1 \times \ldots \times \text{AV}_{\text{anum}} \).

If each column only contains simple states (no contractions or irrelevant conditions), the table is called an expanded decision table (canonical form), in the other case the table is called a contracted decision table (consolidated form). The translation from one form to the other is defined as expansion (rule expansion) and contraction (consolidation) respectively [1], [9].

**3. The Decision Table as Database Relation**

The decision table can be seen as a set of ordered n-tuples \((\text{ct}_1, \ldots, \text{ct}_{\text{anum}}, \text{av}_1, \ldots, \text{av}_{\text{anum}})\), with \( \text{ct}_i \in \text{CT}_i \) and \( \text{av}_j \in \text{AV}_j \), that can be represented as a relational table. This relational table, as representation of a decision table, has the following characteristics:

- each row represents a column of the decision table;
- the rows do not have any particular order (but some orderings are more useful);
- all rows are distinct (exclusivity);
- the order of the columns (conditions and actions) is not important to the description of the problem at the logical level (unless a certain order has to be respected at execution time because of side effects, for instance an ordering of the actions);
- the meaning of each column is explained through a named domain as heading (condition or action subject);
- on each row position of the table, an attribute value (a condition state or an action value, possibly "nil") is found, and not a set of values.

It is clear that such relational table is identical to the transposed expanded decision table, so that the rows correspond with the columns of the decision table and vice versa. The
identity is formal and does not refer to the utility of both representation methods. This becomes important when the dimensions increase and the table must be split.

Since every condition combination occurs precisely once in the relation, the condition attribute values uniquely identify the n-tuples in the relation (candidate key). It is, indeed, the intention of the decision table to indicate which actions should be executed for a given combination of conditions. So the set of condition attributes is defined as primary key. The action attributes can then be indicated as the non-key attributes.

A combination of non-key attributes (actions), that is part of the primary key of another table and thereby refers to that table (foreign key), corresponds with a condition assignment as action in a condition subtable (called condition reference).

4. Modelling Decision Tables using SQL

Since the decision table is equivalent to the relational table, the relational technique (in the form of a relational DBMS) can be used to construct the decision table. This means that both the physical storage and the construction and manipulation of the decision table can be partly executed through the relational structure and operators (relational algebra).

In the following, the construction of a decision table in a relational environment is illustrated. The full construction process is formulated through the use of SQL-statements. When the construction process is finished the decision tables can be consulted using SQL-queries.

Although a decision table can be constructed in a relational environment some advanced features of decision table construction are not possible. E.g. decision table contraction and splitting up decision tables. In addition the query facilities offered by SQL are rather basic.

Those advanced features together with a flexible consultation manager are offered by Prologa (PROcedural LOGic Analyzer), an interactive rule-based design tool for decision table construction and manipulation cf. infra.
4.1. THE BASIC CONSTRUCTION PROCESS

The basic construction process proceeds analogous to the "direct method based on simple rules" [11]. The following stages can be distinguished:

- make a list of conditions, condition states and actions;
- draw up the empty table;
- determine and introduce the logical expressions;
- check for completeness, contradictions and correctness.

The method will be illustrated with the table of fig. 1.

a. Representing conditions, condition states and actions

When obtaining the list of conditions, condition states and actions, a table is created for each condition. This table contains one column (the condition name) and as many rows as there are different condition states. Because of the demands of completeness and exclusivity, the states have to be unique and should not be null\(^1\) (unicity in SQL is only guaranteed through the creation of an index). Later on, these tables will give rise to the expanded decision table (by executing a join operation).

```
CREATE TABLE C_Book (Book CHAR(15) NOT NULL)
INSERT INTO C_Book VALUES ('hard cover')
INSERT INTO C_Book VALUES ('normal')
INSERT INTO C_Book VALUES ('pocket')
[CREATE UNIQUE INDEX X_Book ON C_Book (Book)]

CREATE TABLE C_Wholesaler (Wholesaler CHAR(1) NOT NULL)
INSERT INTO C_Wholesaler VALUES ('Y')
INSERT INTO C_Wholesaler VALUES ('N')
[CREATE UNIQUE INDEX X_WHOLESALER ON C_WHOLESALER (WHOLESALER)]

CREATE TABLE C_Quantity (Quantity CHAR(3) NOT NULL)
INSERT INTO C_Quantity VALUES ('<5')
INSERT INTO C_Quantity VALUES ('>=5')
[CREATE UNIQUE INDEX X_QUANTITY ON C_QUANTITY (QUANTITY)]
```

\(^1\)A NULL-value in the condition part means irrelevant. A NULL-value in the action part means don't execute.
b. Draw up the empty table

Next, the empty, expanded decision table can be created and the condition part can be filled (fig. 2). The condition part consists of the cartesian product of the various condition states and can thus be formulated as a join of the already constructed condition tables (C_Book, C_Wholesaler and C_Quantity). According to the definition of the decision table, the condition combinations must be unique and the condition states must not be null.

CREATE TABLE Bookstore
(  Book CHAR(15) NOT NULL,
  Wholesaler CHAR(1) NOT NULL,
  Quantity CHAR(3) NOT NULL,
  Discount CHAR(1),
  Delivery CHAR(1))

[CREATE UNIQUE INDEX X_Bookstore ON Bookstore (Book, Wholesaler, Quantity)]

INSERT INTO Bookstore (Book, Wholesaler, Quantity)
SELECT * FROM C_Book, C_Wholesaler, C_Quantity

<table>
<thead>
<tr>
<th>Book</th>
<th>Wholesaler</th>
<th>Quantity</th>
<th>Discount</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>hard cover</td>
<td>n</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>hard cover</td>
<td>n</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>y</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>y</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>n</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>n</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>y</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>y</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>n</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>n</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

figure 2: Empty expanded decision table
c. Adding decision rules

The next step is adding the decision rules (these rules can be obtained from e.g. a text, interview with the user, ...). The implementation of those rules can be realized by a number of update statements, with in the where clause the logical expression selecting the condition combinations for which the action(s) must be executed.

**UPDATE Bookstore**
SET Discount = 'x'
WHERE Book = 'hard cover' OR (Book = 'normal' AND Quantity = '>=5')

**UPDATE Bookstore**
SET Delivery = 'x'
WHERE Book = 'hard cover' AND Wholesaler = 'y' AND Quantity = '>=5'

Finally, this results in the completed decision table that can be checked and adapted.

<table>
<thead>
<tr>
<th>Book</th>
<th>Wholesaler</th>
<th>Quantity</th>
<th>Discount</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&lt;5</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&gt;=5</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>hard cover</td>
<td>n</td>
<td>&lt;5</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>hard cover</td>
<td>n</td>
<td>&gt;=5</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>y</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>y</td>
<td>&gt;=5</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>n</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>normal</td>
<td>n</td>
<td>&gt;=5</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>y</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>y</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>n</td>
<td>&lt;5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pocket</td>
<td>n</td>
<td>&gt;=5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

figure 3: Table bookstore

d. Check for completeness, exclusivity and consistency

It is the responsibility of the user to guarantee the semantic exhaustivity. When this constraint is fulfilled, the join instruction generates all possible combinations of condition states. As a consequence, the generated table will be complete with respect to the conditions. To check if at least one action corresponds with each combination of
condition values, the following query can be executed. If at least one row is selected, the table is not complete.

```
SELECT Book, Wholesaler, Quantity
FROM Bookstore
WHERE Discount IS NULL AND Delivery IS NULL
```

The exclusivity is guaranteed by creating a unique index when creating the table.

If it is not allowed for the actions Discount and Wholesaler to occur together (as can be specified in Prolog cf. infra), then the following query will discover this error:

```
SELECT Book, Wholesaler, Quantity
FROM Bookstore
WHERE Discount = 'x' AND Delivery = 'x'
```

### 4.2. Construction of Decision Tables Using the Decision Grid Chart

Formulating the decision rules as update statements, has the disadvantage that these rules are not preserved in the decision description, although they constitute an essential part of the decision situation. Therefore it is advised to construct a separate table corresponding with the so called decision grid chart and to fill up the decision table using this added table. The decision grid chart is a tabular representation of a set of rules, containing one column for each rule, and corresponds with the rule base of a KBS. The grid table is constructed in a similar way as the decision table. There are, however, some differences: it is not necessary to mark conditions as not null attributes because don't care entries are allowed, and an index is not defined, because a combination of condition values may occur more than once in the grid chart.

**Creation of the table**

```
CREATE TABLE Grid
( Book CHAR(15) ,
  Wholesaler CHAR(1) ,
  Quantity CHAR(3) ,
  Discount CHAR(1) ,
  Delivery CHAR(1) )
```
Adding the decision rules

INSERT INTO Grid
VALUES('hard cover', null, null, 'x', '-')

INSERT INTO Grid
VALUES('normal', null, '>=5', 'x', '-')

INSERT INTO Grid
VALUES('hard cover', 'y', '>=5', '-', 'x')

Filling the decision table from the grid chart

UPDATE Bookstore T
SET Discount = 'x'
WHERE EXISTS
  (SELECT * FROM Grid
   WHERE (T.Book = Book OR Book IS NULL)
       AND (T.Wholesaler = Wholesaler OR Wholesaler IS NULL)
       AND (T.Quantity = Quantity OR Quantity IS NULL)
       AND (Discount = 'x'))

UPDATE Bookstore T
SET Delivery = 'x'
WHERE EXISTS 'x'
WHERE EXISTS
  (SELECT * FROM Grid
   WHERE (T.Book = Book OR Book IS NULL)
       AND (T.Wholesaler = Wholesaler OR Wholesaler IS NULL)
       AND (T.Quantity = Quantity OR Quantity IS NULL)
       AND (Delivery = 'x'))

These update queries are independent of the decision logic and remain unchanged when the decision logic is changed. Whenever the grid is updated, they have to be triggered.

In this simple case there are only three decision rules and is not possible to reduce these rules. However, in more complicated situations it might be useful to minimize the decision grid chart [4]. Minimizing the decision grid chart minimizes the number of columns in the grid chart. As a consequence it minimizes the number of insert instructions.
4.3. CONSTRUCTION OF A SYSTEM OF DECISION TABLES

In this section a system of decision tables of fig. 4 will be constructed using the decision grid chart method cf. supra. A condition or an action which is worked out in detail in a subtable is preceded by the subtable sign ‘\(^{\text{\textquoteleft}}\).’

ORDER

<table>
<thead>
<tr>
<th>1. Credit Limit ?</th>
<th>Ok</th>
<th>Not Ok</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. ‘CustomerGood’</th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Stock Sufficient ?</th>
<th>Y</th>
<th>N</th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1. ‘ExecuteOrder’</th>
<th>X</th>
<th>-</th>
<th>X</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. ‘Refuse Order’</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>3. ‘Put On Waiting List’</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

CUSTOMERGOOD

<table>
<thead>
<tr>
<th>1. Age Of Account ?</th>
<th>&lt; 1 Year</th>
<th>&gt;= 1 Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Turnover ?</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

| 1. CustomerGood:=NotGood | X | - | X | - |
| 2. CustomerGood:=Good   | - | X | - | X |

EXECUTEORDER

<table>
<thead>
<tr>
<th>1. Quantity Ordered ?</th>
<th>Q&lt;10</th>
<th>10&lt;=Q&lt;15</th>
<th>Q&gt;=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Travel Distance ?</td>
<td>D&lt;50</td>
<td>50&lt;=D&lt;100</td>
<td>D&gt;100</td>
</tr>
</tbody>
</table>

| 1. No Discount   | X | - | - | - |
| 2. Discount 2 %  | X | - | - | - |
| 3. Discount 5 %  | - | X | - | - |
| 4. Discount 10 % | - | - | X | - |
| 5. Railway Transport | - | X | X | - |
| 6. Road Transport  | - | X | X | - |
| 7. Bill Type A    | - | X | X | - |
| 8. Bill Type B    | - | X | X | - |

CREATE TABLE Grid

\[
\begin{align*}
\text{CREATE TABLE } & \text{Grid} \\
& \left( \text{CreditLimit CHAR(6),} \\
& \text{CustomerGood CHAR(1),} \\
& \text{StockSufficient CHAR(1),} \\
& \text{ExecuteOrder CHAR(1),} \\
& \text{RefuseOrder CHAR(1),} \\
& \text{PutOnWait CHAR(1)} \right)
\end{align*}
\]
INSERT INTO Grid
VALUES ('OK', 'null', 'Y', 'x', '-', '-')
INSERT INTO Grid
VALUES ('OK', null, 'N', 'x', '-', '-')
INSERT INTO Grid
VALUES ('NotOK', 'Y', 'x', 'x', '-', '-')
INSERT INTO Grid
VALUES ('NotOK', 'Y', 'N', '-', 'x', '-')
INSERT INTO Grid
VALUES ('NotOK', 'N', 'null', '-', 'x', '-')

UPDATE Orders T
SET ExecuteOrder = 'x'
WHERE EXISTS
  (SELECT *
   FROM GRID
   WHERE (T.CreditLimit = CreditLimit OR CreditLimit IS NULL)
     AND (T."CustomerGood" = "CustomerGood" OR "CustomerGood" IS NULL)
     AND (T.StockSufficient = StockSufficient OR StockSufficient IS NULL)
     AND (^ExecuteOrder= 'x'))

The construction of the tables Customer and ExecuteOrder are analogous.

Table Order

<table>
<thead>
<tr>
<th>CreditLimit</th>
<th>&quot;CustomerGood&quot;</th>
<th>StockSufficient</th>
<th>&quot;ExecuteOrder&quot;</th>
<th>RefuseOrder</th>
<th>PutOnWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>y</td>
<td>y</td>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ok</td>
<td>y</td>
<td>n</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>ok</td>
<td>n</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ok</td>
<td>n</td>
<td>n</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>not ok</td>
<td>y</td>
<td>y</td>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>not ok</td>
<td>y</td>
<td>n</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>not ok</td>
<td>n</td>
<td>y</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>not ok</td>
<td>n</td>
<td>n</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
</tbody>
</table>
Table CustomerGood

<table>
<thead>
<tr>
<th>AgeofAccount</th>
<th>Turnover</th>
<th>✕CustomerGood</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1</td>
<td>&lt;50</td>
<td>n</td>
</tr>
<tr>
<td>&lt;1</td>
<td>50-100</td>
<td>y</td>
</tr>
<tr>
<td>&lt;1</td>
<td>&gt;100</td>
<td>y</td>
</tr>
<tr>
<td>≥1</td>
<td>&lt;50</td>
<td>n</td>
</tr>
<tr>
<td>≥1</td>
<td>50-100</td>
<td>n</td>
</tr>
<tr>
<td>≥1</td>
<td>&gt;100</td>
<td>y</td>
</tr>
</tbody>
</table>

Table ExecuteOrder

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Distance</th>
<th>0%</th>
<th>2%</th>
<th>5%</th>
<th>10%</th>
<th>Railway</th>
<th>Road</th>
<th>Bill A</th>
<th>Bill B</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10</td>
<td>&lt;50</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>&lt;10</td>
<td>50-100</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>&lt;10</td>
<td>&gt;100</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>10-15</td>
<td>&lt;50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>10-15</td>
<td>50-100</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>10-15</td>
<td>&gt;100</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>&gt;15</td>
<td>&lt;50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>&gt;15</td>
<td>50-100</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>&gt;15</td>
<td>&gt;100</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
</tbody>
</table>

figure 5: The relational tables of the order processing example

4.4. CONSULTING THE DECISION TABLES

Next some basic queries are proposed. More advanced queries require the Prolog-a-system.

1. In which cases does a customer get a discount of 10%?

SELECT CreditLimit, AgeofAccount, Turnover, StockSufficient, Quantity, Distance
FROM Order O, Customer C, ExecuteOrder E
WHERE O.^ExecuteOrder = 'x' AND C.^CustomerGood = O.^CustomerGood
AND E.10% = 'x'
2. When will an order be put on a waiting list?

`SELECT CreditLimit, StockSufficient, AgeofAccount, Turnover
FROM Order O, Customer C
WHERE O.PutOnWait = 'x' AND O.CustomerGood = C.CustomerGood`

5. Integrating the Relational Features in a Decision Table System

5.1. Evaluation of the Basic Approach

Besides its utility in the construction process of decision tables, the relational approach provides report generation, recovery and basic facilities for consulting the information contained in the table. However, the relational environment by itself does not offer enough facilities to support the construction process in a flexible and effective way. The following facilities for instance are not or hardly available:

- contraction of the decision table;
- automatic reconstruction after modifications;
- use of a powerful specification language;
- optimal conversion to computer programs;
- recursive queries
- decision making using what-if analysis.

Therefore we recommend to provide these facilities in a decision table workbench, while the internal storage and manipulation of the data will be performed by the relational database system. This workbench could then generate the SQL-statements to store the tables and to perform some verification and validation.

5.2. Extending the Basic Approach to Additional Prologa Features

A major drawback of the use of decision tables (and many other condition oriented representations) is the complexity of the manual building process. A lot of redrawing work results from small changes like adding a condition, a condition state or an action.
Some manipulations like the reordering of conditions are quite impossible to perform manually. To this end, PROLOGA (PROcedural LOGic Analyzer) has been developed, an interactive rule-based design tool for computer-supported construction and manipulation of decision tables ([7], [8]). Next we will explain some features of Prologa.

- When the table is constructed and verified, it can be contracted in Prologa. Although it is possible to store a simple form of the contracted table (fig. 6) in a relational database, the full advantages of decision table contraction are not available. Only if all states of a condition lead to the same action configuration, contraction is possible (partial contraction). If groups of states with the same action configuration occur, the states may be simply joined by the OR-connector in Prologa (full contraction). Of course this type of contraction cannot be stored in a relational table. Preceding the contraction of a decision table it is possible to perform row order optimization. This determines the condition order which results in the minimum number of columns. It is not the purpose of this paper to explain in detail how the contracted decision table can be obtained. For more information about these issues, see [3], [7].

<table>
<thead>
<tr>
<th>Book</th>
<th>Wholesaler</th>
<th>Quantity</th>
<th>Discount</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&lt;5</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>hard cover</td>
<td>y</td>
<td>&gt;=5</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>hard cover</td>
<td>n</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>normal</td>
<td></td>
<td>&lt;5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>normal</td>
<td></td>
<td>&gt;=5</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>pocket</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

figure 6: Contracted table

- A powerful specification language allows the designer to formulate the decision specification in a straightforward way (with provisions for expressing general rules, exceptions, preliminary results, restrictive causes and consequences). Some example skeletons of decision rules:

Actions [generally] if condition combinations
Not action definitely if condition combinations
Action only possible if condition combinations
Action definitely if and only if condition combination
The modelling process can be simplified considerably by the use of interactive possibilities such as automatic checking for consistency, correctness and completeness or recommendations for a specific construction method.

- The system can be used for optimization purposes, such as optimal contraction, layout, decomposition into subtables or conversion into efficient program code.

- Although SQL offers some basic consulting facilities real decision making is hardly possible. Therefore the hierarchy of decision tables is translated into a question and answer interface. The user however, need not be aware of the existence of the decision tables or any relations between them. A full consultation environment is built together with the decision table application [10]. This includes the following facilities:

  View: a list of all variables with their values
  Footsteps: a chronological survey of the questions asked, with the answer reached by the consultation environment.
  Whatif-mode: offers the possibility in Footsteps or View to change one or more answers and restart the reasoning process.

6. Conclusion

Based on the correspondence between a decision table and a table in a relational database system, the major modelling issues of decision table construction and validation were examined and a possible integration of a decision table system and a relational database management system was suggested and evaluated, with special emphasis on building tables using the decision grid chart. First the basic construction process was examined. Next the Prologa system was introduced to solve some problems which couldn't be handled in the basic approach.

References


