Tạp Chí Tin Học và Điều Khiển Học
Tập VIII(1992), Số 2(21-31)

Coupling of Knowledge and Data in TESOR System

Ho Tu Bao & Dao Nam Anh
Institute of Informatics
Hanoi, Vietnam

1. Introduction

Knowledge engineering - the art and science of building knowledge - based systems - has been emerged as a discipline in its own right in the last decade. The expert system shells of 10 years ago have grown up into hybrid shells. These days, a "real" shell is an integrated set of tools that provides facilities for building a knowledge base as well as for various inference paradigms.

In recent years knowledge and data integration has been day by day widely acknowledged and becoming one of the main subjects of knowledge engineering [3].[4],...

Data refer to factual information suitable for computer processing which in general can be measured or observed directly. Databases involve relatively simple representational structures for holding voluminous amounts of data. Knowledge refer to condition of knowing something which has been perceived, discovered, or learned. Knowledge ranges from simple facts stored in a conventional database to complex statements about the real or a modeled world. Knowledge bases usually involve relatively complex representational structures for holding a large number of relations (rules, frames, etc.).

The comlementary nature of processing styles and methods for database and knowledge bases has been recognized. Database inference is usually restricted to either precise boolean combination of modest complexity or simple numerical comparisions. Knowledge bases often employ complicated reasoning procedures that based on formal logic. Intelligent information systems must therefore provide the capability to process both these kinds of information in which factual knowledge that describes relations among entities and their attributes lied in conventional databases and knowledge bases.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

There are several architectural paradigms for coupling databases to knowledge bases that differ in the nature of the DBMS calls and in the DBMS facilities which are used in the knowledge bases. The simplest paradigm is to keep data in the internal form of the knowledge base by converting them prior to their use. In this case, we can not use the facility of the query evaluator of the database systems during the running phase of the knowledge base systems and the database systems are off line from the knowledge base. Another paradigm is to tightly couple database and knowledge base. In the case, the database is internal to the knowledge base, i.e. the database component is an integral part of knowledge base system. A third paradigm is to loosely couple these two kinds of information. In the case, the database is external to the knowledge system and is acts as a server for the knowledge base [2],[3].

In our current paradigm,the TESOR knowledge processing system is integrated with databases, hypertext systems and computation processes. In particular, a loose coupling of databases and TESOR system is carried out.

2. TESOR system

2.1 Main features of TESOR.

TESOR is an evolutionary environment for building knowledge based systems that combines artificial intelligence techniques, graphics tools, and object-oriented programming. TESOR includes the following features [5]:

Knowledge representation and Reasoning

• Combination of structured knowledge (objects) and unstructured knowledge (rules),

• Flexible inference system that offers the hybrid features of several different styles or paradigms of reasoning,

• Interfaces to spreadsheets and database systems,

• Combination of inference engine power with the flexibility of hypertext search and retrieval,

• Interface with computational processes.

Graphical editor

• Interactive environment for creating and editing knowledge bases,

• Describe graphically the structure of objects, their relationships and characteristics with context-sensitive helps,

• Compile and test knowledge bases incrementally,

• Detect some kinds of incoherence in knowledge declaration.

Open Architecture:

• Embeddable,

• External programs and user's functions interfaces,

• Dynamic links to other data sources, and C programs allow to extend existing applications,

• Library of graphic objects and its interpreter which allow to produce animations and.

The Figure 1 shows the layered architecture of TESOR. Central to this architecture is the TESOR knowledge base and inference engine. TESOR knowledge representation language combines two forms of knowledge representation of objects (structured knowledge) and production rules (unstructured knowledge). TESOR inference engine supports a high - level management of knowledge with integrated forward and backward chaining, multiple inheritance, etc. TESOR interface modules play an important role in improving the performance of the system. First, TESOR interface to

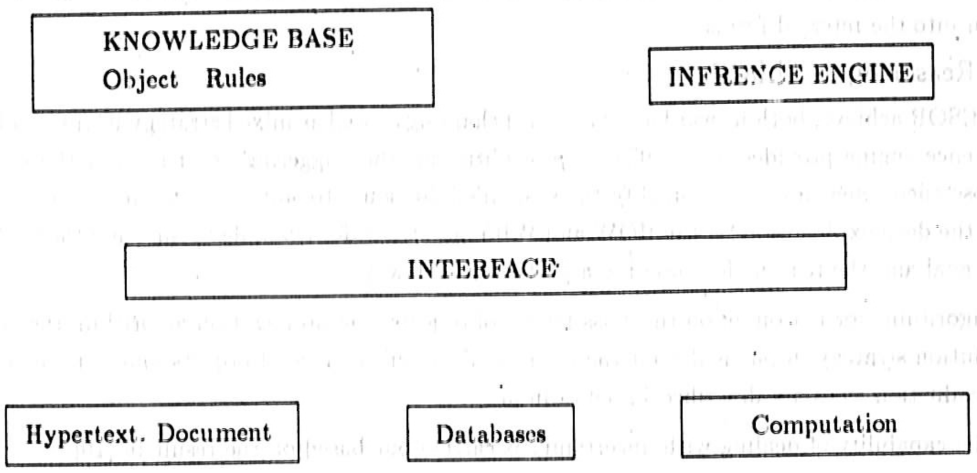

Figure 1. Conceptual architecture of TESOR system

hypertext document database allows accesses by a non linear manner to hepertext systems that store different kinds of information such as text, image, graphics created by TESOR's authoring. Second, database interface allows make use of data in the database during the inference processing. Third, TESOR allows to connect with computational procedures externally to the system. The purpose of this paper is to describe how to design and implement the interface of TESOR with databases.

2.2 Knowledge representation

The principal knowledge representation techniques used in knowledge based systems are: production rules, frames, semantic networks, and objects [1]. It is clear that no single knowledge representation technique is vertisale enough to all problems in knowledge processing and the trend of combinating advantages of different techniques has been attracted great attention. In most early knowledge based systems, knowledge is expressed in the form of production rules. The object oriented programming in the last few years has been adapted to databases as well as to knowledge bases systems. The object-rule marriage in TESOR offers one of the most sophisticated ways to represent and manipulate knowledge. There is no sharp distinction about the use of these two representation kinds, but in general we have:

• knowledge represented by objects are principally static, structurable so that one can find a hierarchical relation between its entities,

• knowledge represented by rules are principally dynamic, which express causal relations, actions depending conditions, or which are unstructurable chunks of knowledge, ...

TESOR allows the user to represent a knowledge base as either objects or rules. In a general case, an TESOR knowledge base is a set of objects and rules [6].

TESOR knowledge representation language is based on this combination and its external syntax is described in the appendix. TESOR graphical editor allows to declare knowledge and encodes them into the internal forms.

2.3 Reasoning in TESOR

TESOR achieves both forward and backward chaining as well as mixed strategy flexibly. TESOR inference engine provides some different possiblities for the triggering order so that the user can choose these ones in order to modify the system's bihaviours to suit concrete situations. It gives also the deep explanation for the HOW and WHY questions, for example by showing the pathways to a goal and the reasons for choosing a particular pathway.

Algorithms for reasoning on the class lattice of objects, the propagation algorithm, the conflict resolution strategy in particular for the relation of the inheritance of objects and non inheritance of production rules are described in detail in [5].

The capability of dealing with uncertainty is carried out based on the result in [10].

3. Coupling TESOR with databases

3.1. Conceptual schemas and approach

In this article we use the keyword "link" to mean dynamic relationships between attributes in knowledgebases and extrenal data files. In section 2.1 the TESOR conceptional scheme has been introduced briefly. The below figure 2 outlines our approach to loosely couple TESOR to existing databases. First of all, the TESOR knowledge representation language will be extended to that the knowledge engineer could write links in declaring a knowledge base depending on what data and when it will be done in the knowledge processing system. In compiling and transforming the knowledge base into internal form as inference network [5], links will be converted into suitable intermediate form. The interpreter module plays a central role in the integration. It receives messages about links from TESOR inference engine, analyses the links under intermediate form, realizes them and returns required data to working memory or on screen.

The essence in knowledge processing of TESOR is to match its knowledge base to factual information about some objects stored in the working memory. These factual information can be provided either by the user in conversation with the system or from external environment and sometimes through a computation process (reading from databases, from measured devices, etc.). These information will be transmited into the working memory and according to attributes which describes knowledge entities and their values.

We study three cases where TESOR can provided factual information from existing databases for knowledge processing by its intergration mechanism.

Case 1: Factual information from databases is directly returned to the working memory and will be treated by the infrence engine. In the case, the attributes about considering domain are used simultaneously in the knowledge base and the databases. Their values are stored and updated in databases.

Case 2: Factual information required by the knowledge processing may be obtained by some external computational procedure whose input is exstracted from databases. In the case, the interpterter returns extracted in a suitable form for the computation.

Case 3: Data extracted from database will be displayed on the screen as a support for the user in choosing an answer to a question posed by the system in reasoning.

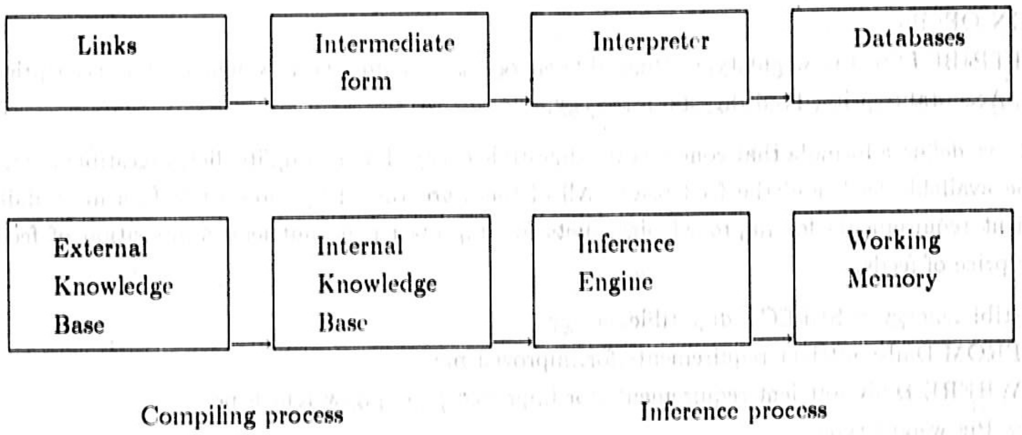

Figure 2. Loose coupling of TESOR and databases

3.2. Extension of TESOR's knowledge representation language for coupling it with databases

In order to couple TESOR to databases, its knowledge representation language have to be extended. The language must satistify the following requirement:

- During knowledge processing, TESOR can access to databases and use the facilities of DBMS,
- It is possible to manipulate simultaneously attributes in knowledge bases and databases.

In considering these requirements, some SQL language's statements have been selected and improved as basic link commands in TESOR [7]. It is chosen because SQL is standard for database query languages and it is powerful for databases manipulation. These link commands can treat attributes which appear in the knowledge base and not appear in the linked databases. The SQL statements chosen are:

- SQL statement,
- SELECT UNION statement,
- SELECT statement,

These are improved in order to represent the necessary information to be declared in the links:

- Name and type of data files, by the argument FROM in statements SELECT and SELECT UNION,
- Relationships between attributes in knowledge base and databases, by the arguments SELECT,
- Conditions for choosing required information from data files, by the argument WHERE, HAVING, UNION,
- Order for returning extracted data to the working memory, by the argument ORDER BY.

Now, let see an example concerng a SELECT links while declaring a knowledge base. Suppose that we are tallking about the object "Improved pig":

OBJECT: Improved pig;

SON_OF: Pig;

ATTRIBUTES: Pig_weght_type, Digestible_energy, Feed_name, Feed_weight_in_day, Feed_price, Diet, Acceptable_price, Feed_digestable_energy;

Next, we define a formula that concerns the digestible energy for the pig, its diet, digestible energy of the available feed, and the feed price. All of them are valued by links with data files: daily nutrient requirements for improved pigs, diets for improved pigs, nutrient composition of feed, daily price of feeds.

```
Digestible_energy = SELECT digestible_energy
    FROM Daily_nutrient_requirements_for_improved_pigs
    WHERE Daily_nutrient_requirements_for_improved_pigs. pig_weight_type =
    = Pig_weight_type;
Diet = SELECT diet
    FROM Diets_for_improved_pigs
    WHERE Diet_improved_pigs.pig_weight_type = = Pig_weight_type;
Feed_digestible_energy = SELECT digestible_energy
    FROM Nutrient_composition_of_feed
    WHERE Nutrient_composition_of_feed.feed.name = = Feed_Name;
Feed_price = SELECT price
    FROM Daily_price_of_feeds
    WHERE Daily_price_of_feeds.name = = Feed_name;
```

And the conditions of decision for the feed in the diet are declared as follows.

```
IF (Diet * Feed_digestible_energy > = Digestible_energy) .AND.
(Diet * Feed_price < = Acceptable price);
```

THEN The diet is acceptable;

These clauses mean that if the diet with the feed contains digestible energy enough for the pig and the price of the diet is acceptable for the farmer than this diet is adequate.

In order to help the user to describe link commands, TESOR editor has been added a special form that displays the fields for the SQL statements to fill up. The user can also reference to the database structure as well as to the information about the knowledge base such as the attribute table, the object table, the link list, etc.

3.3 Intermediate form and interpreter

Links to external data files are analyzed and compiled by the knowledge editor into intermediate form. This intermdiate form is suitable and ready for the interpreter. Thwy are the reverse Polish form and their components are paire of type and handle. The type is corresponding to a processing procedure.

When the interpreter module receives message about link from TESOR inference engine, it will analyze the corresponding component of this link in the intermediate form. The interpreter then executes a link operation according to the type of the component. Result of the links is required data from databases and it is returned to working memory or on screen depending on requirement of the links. The link operations are classified into four following groupes:

(i) operations for the management of attributes in knowledge bases,
(ii) operations for the management of attributes in databases,
(iii) operations for the exchange of attribute's in knowledge bases and databases,
(iv) operations for the transformation of types of databases,

The groupe (ii) is the most important because it permits to manage data files of a concrete type, for example, the *.DBF files or PARADOX files. Because all types of data files can be converted to others, one needs only a way of accesing to data files of one defined type, and data files of other types will be converted into the acsesible type.

Hence, the PARADOX ENGINE [8] allows to manipulate data files in paradox format. It provides application programming interface that lets create, read, and write paradox tables, records, and fields. Futher, TESOR uses paradox analogous link functions that allow to read, write, update data in *.DBF files. A motivation of this is our coupling application don't use all paradox engine's facilities and a paradox engine's use in TESOR inquires its installation with TESOR. Among link functions of groupe (iv) exist functions with converts paradox data files to *.DBF files.

4. An example of use of knowledge and data in TESOR. Diets for pigs feeding

Pigs can kept throughout Southesast Asia and the Pacific and in fact the bulk of the world's pigs are in this region. The below table shows for example the number of pigs given in 1000s and as % of the total number in the region (Source: FAO Production Yearbook, Vol. 39,1986) [8].

	BURMA	INDONESIA	LAOS	MALAYSIA	PNG	PHILIPP	THAILAND
PIG	2750	4050	1450	2100	1476	8007	4300
%	0.78	1.16	0.41	0.06	0.42	2.29	1.23

	SAMOA	CHINA	VIETNAM	SINGAPOR	KAMPUCHIA	SOLOMON
PIG	62	313010	11500	810	1200	49
%	0.02	89.43	3.29	0.23	0.34	0.01

Pig production on main factors: stock and breeding, nutrition and feeding, housing and waste disposal, and health control. The lagest recurrent expense in pig feeding will be for feed; this is usually about 80% of all cost. It is therefore very important for profitability to think of ways in providing nutrients at the lowest cost.

Daily nutrient requirements can be viewed as a function of current phase of the life cycle, growth, breeding and can be presented as a vector of about thirty nutrients:

Y = (digestible energy, digestible protein and essential animo acids, minerals and trace elements, fat soluble vitamins, water soluble vitamins, water).

A database has been done that contains records about nutrient composition of hundreds important feeds in the region. Denote this set by **A**. In fact, the user uses in general a very smaller subset $\mathbf{A}_0$ of **A** that are available in this area.

The problem may be formulated as follows: find vector X that contains necessary quantities of nutrients that satisfied: $<X,c(X)> \rightarrow$ min with the constraint $A_0X >= Y$ where c(X) is the cost of X.

The simplest way of finding a "first approximation" solution of this problem is to use available died formulations. However, it must be recognized that this way is less flexible and is not always feasible. In general this problem is a kind of ill-structured problems and it required to use heuristic in finding solutions. The commercial feed compounder can use computers to formulate least - cost died. A prototype of such a decision - support system has been constructed from TESOR that combines knowledge on pig feeding, the calculation based on diet formulation and information about available feeds from databases. Essentially, in each concrete situation poses, the system questions to user about the breed, current phase of the life cycle, the enviroment conditions, health control, and in particular, about the available sources of feeds.

Based on these information, the system automatically extracted by suitable databases links the matrix Ao from the databases A. A solution is found in certain cases when necessary information in complete. In the case of incomplete information, the system uses some heuristic to find supplementary conditions (substitution of certain available feeds for example) and finds an approximation solution to the problem.

Conclusion

We have represented roughly our approach to coupling TESOR system with databases. The system is implemented in the C language and under MS Window 3.1. TESOR has been successfully used in some application and the integration of it with databases is presently at the stage of converting to a product. The system is being examined through some similar applications.

6. References

1. Ringland G.A. & Duce D.A., Approaches to knowledge representation: An introduction (Eds.), John Wisley & Sons Inc., 1988.
2. Yamauchi H. & Ohsuga S., "Loose coupling of KAUS with existing RDBMSs", Data & Knowledge Engineering 5 (1990), p. 227-251.
3. Risch T.; Rebon R.; Hart P. & Duba R., " A Functional Approach to Integrating Database and Expert Systems", Communication of the ACM, December 31 (1988), p.1424-1437.
4. Albanio A. & Attardi G., "Issues in Data Base and Knowledge base Integration", Foundations of Knowledge Base Management, Schmidt J.W., Thanos C. (eds.), Springer)Verlag, 1989, 283-291.
5. Bao H.T.; Khoi P.N.; Khang P.N.; Mai H.T.; Son N.C.; Dung N.T.; Huyen T.T. & Thanh H.Q., "Development and Applications of the TESOR Expert System Generator", Proceedings of National Centre for Scientific Research of Vietnam, N.2, 1992.
6. Bao H.T; Khoi P.V.;Lien D.N. & Nga D.V., TESOR 1.1 User's manual, Institute of Informatics, Hanoi 1989.
7. Paradox engine version 2.0. User's Guide and Reference Guide, Borland International, USA, 1991.
8. Rick F. Van der Land, Introduction to SQL, Addition - Wesley Publishing Company, 1987.
9. Muller M.F., "Manual on pig feeding in Asia and the Pacific", The Rowett Research Institute Aberdeen, U.K., Rome, 1989.
10. Dieu P.D., "On a theory of interval - valued probabilistic logic", Research report, Institute of Informatics, Hanoi, 1991, 1-64.

7. Appendix: The TESOR'syntax description primitives

Objects and Rules are described in TESOR by the following primitives:

Atributes: 1. Attributes, as usualy, are simple properties attached to objects /rules. Properties of the application domain is often defined by pair or a combination of pair of attribute - value. In TESOR, a variable has:

- a name, which is a sequence of characters, digits, spases and "-", beginning with a character.
- a domain, consisting of the attribute type (real, integer, string, proposition), the **domain** (scale, interval, default) and the **weight** of attribute (a number between 0 and 1).

2. Methods: A formula that defines the intrinstic relation between some attribute according to the following syntax:

```
<formula> := <id.> = <num.expr> | <id.> < - <str.expr>
<num.expr> := <id.> | <const.> | <funct.name> (<par.list>)
          | <num.expr> <num.operator> <num.expr> | (<num.expr>)
<num.operator> := + | - | | /
<str.expr> := <string> | <id.> | <funct.name> ( <par.list> ) | (<str.expr>)
<par.list> := < > | <expr> {, <expr>}
<const.>        is a real or an integer constant
<id.>           is the name of an attribute
<string>        is a constant of string type
<funct. name>   is the name of a function
```

Example: TTC_price = unit_price * 1.186;

3. Predicates: Predicates are logical expressions that determine properties of the class being considered according to the folowing syntax:

```
<log.expr> := <comp.expr> | <prop.> <log.expr> .NOT. <log.expr> | (<log. expr>)
<comp.expr> := <num.expr> <comp.operator> <num.expr> <num.expr.>
<comp.operator> := > | < | > = | < = | = =
<log.operator> := .AND. | .OR.
<prop.>       is a proposition
<id.>         is the name of an attribute
```

Example :

```
((Following parameter = = "Maximum time constant".AND.
((T12*_low).OR.(T12*T_high))).OR>
((Following parameter = = "Amplifying factor".AND.
((TAU2TAU_low).OR.(TAU2TAU_high))));
```

4. Conclusions: Conclusions are either propositions or assignments to attributes . The right side of an assignment may be a value, a well-formed expression which probably includes names of external procedures (written in C, Pascal, Fortran.etc.). Assignments has the same syntax of <formula>

Example:

```
CONCL: Capital < - "Paris";
CONCL: Fail the synchronization module,
CONCL: a = f(x,y,z) - 1;
```

These above descriptions are only rough outlines of the representation of knowledge in TESOR.

5.SQL statements:

```
<SELECT statement> := SELECT     <Expression>
            := FROM       <table identifiers>
            :=WHERE       <condition>
            :=GROUP BY    <column identifiers>
            :=HAVING      <having condition>
            := ORDER BY   <orderings>;

<SELECT UNION statement> := SELECT     <Expression>
            := FROM       <table identifiers>
            :=WHERE       <condition>
            :=GROUP BY    <column identifiers>
            :=HAVING      <having condition>
            := UNION SELECT   <expressions>;

                   FROM       <table identifiers>
                   WHERE      <condition>
                   GROUP BY   <column identifiers>
                   HAVING     <having condition>
                   ORDER BY   <orderings>;

Example: SELECT   name, surname, adress
         FROM     person
         WHERE    person.surname = = Kowarski;
```

Abstract

Coupling of knowledge and data in TESOR system

In this paper we shall be concerned with the integration of knowledge and data in TESOR system. In particular, TESOR system, an evolutionary environment for building knowledge based systems is first introduced. Next, the architectural paradigm of the integration and the extension of the TESOR knowledge representation language for this purpose is given. Finally, the implementation and an application of this coupling system are addressed.