CHAPTER VI

THE EXPERT SYSTEM VMITS

6.1. Chapter Overview

This chapter will discuss the expert system VMITS for logical database design and its turbo prolog implementation. The specific goal of VMITS is to reproduced the database design expert's attitude to exploit experimental knowledge necessary to the mastering of the process of database design by combining it with more formal knowledge. The system VMITS consists of three separate components : VMS, VIS and VTS. The VMS (View Modeling System) takes user requirements of an application as seen by a user or a group of users as input and produces the conceptual schema corresponding to the user view. The methodology described in chapter-III forms the basis of the system VMS. The type of users that can interact with this system VMS are end-users/designers who are familiar with the application domain of the database application and who may or may not be have any knowledge about database concepts. The VIS(View Integration System) integrates the user views designed by VMS by taking a pair of them each time and systematically assists the user (designer/analyst who should be familiar with the database concepts especially

with the EER model) to identify and resolve the conflicts between the two views, and then merges and restructures these two conflict free views into a single view. The final output of the VIS is a single global view that represents the conceptual schema of the database application seen as a whole. The methodology described in chapter-IV forms the basis of this system VIS.

The VTS (View Translation System) takes the conceptual schema produced in the previous stage, as Input and translates it into a set of Fifth Normal Form(5NF) relations, which represents the logical schema of the database. The methodologies described in chapter-III, IV and V are formalized as a set of rules to form the knowledge base of the systems VMS, VIS and VTS respectively. Appendix-A gives a listing of turbo prolog version of these rules Appendix-B contains a chronicle of a typical database design session with the system VMITS.

Section-2 provides the different ways VMITS intents to support the logical database design in an automated manner. This leads to the description of the Functional Architecture of the system.

Section-3 describes the Internal Architecture of the system.

Section-4 simulates a VMITS session in order to illustrate the logical database design phases with examples.

6.2. Logical database design supports in VMITS

6.2.1. Design Engineering and control engineering

As shown in figure 6.1 Logical database design consists of view modeling, view Integration, and View Translation phases.

The objective of the view modeling phase is to capture knowledge about some application domain and represent it in a conceptual schema corresponding the specific view of the database. The tasks carried out during this phase mainly comprising of :

1) **Acquisition** of application domain dependent knowledge. At present this task is carried out interactively through a well defined query and menu driven dialog process.

2) **Conceptualization** and abstractization of this knowledge into conceptual schema (EER model) concepts.

3) of the conceptual specification, and together with a correcting task that aims at proposing to any detected anomaly one or several ways of correcting it.

The view integration phase has the objective of integrating the user views designed in the previous phase into a single global view representing the conceptual

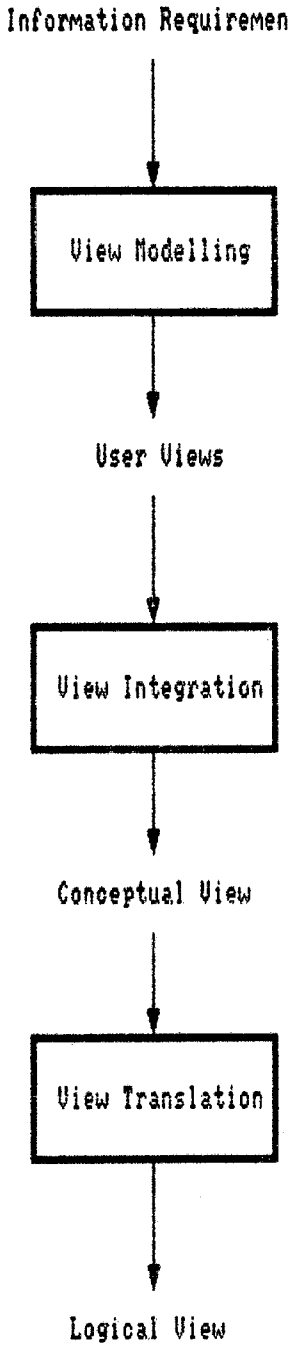

Fig. 6.1 Logical Database Design Phases.

schema of the entire database. This includes the following tasks :

1) **Identification** of conflicts (naming and structural) between pair of views.
2) **Resolution** of these conflicts.
3) **Merging** and restructuring the two conflict views.

The objective of the view translation phase is to translate the conceptual schema designed in the last phase to a logical schema. This includes the task :

Translation of the EER schema into a set of 5NF relations.

Let us call **Design Engineering** to the collection of these knowledge based tasks of the logical database design.

Complementary tasks are required to organize and synchronize the knowledge based tasks. The term **Control Engineering** refers to the collection of tasks which comprises :

1) **Guidance** of the database design process in order to help the user/designer in the process of thinking and conceptualization problem.
2) **Explanation** of achieved results.
3) **Interfacing** with the designer and the end-users.

VMITS aimed at supporting both design engineering and control engineering. This leads to the **function architecture** of the tool depicted in Fig. 6.2.

6.3. Internal Architecture of VMITS.

From the system point of view VMITS is regarded as an expert system consisting of an Inference engine, a rule base, a fact base, and an Interface (fig. 6.3).

VMITS is implemented in (Turbo) PROLOG, thus the inference engine is the PROLOG compiler.

The fact base consists of the data structures used in the prolog database of the system. A list of these data structures are given in Appendix-A in VDOM.PRO file.

The rule base is composed of production rules which allows the system to support both the design engineering and control engineering tasks. The design engineering and the control engineering tasks are automated through these rules. They represent the expert knowledge acquired by formalizing the design methodologies of the three phases, heuristics and other procedures as a set of rule.

Interfaces

In the present version of VMITS the interaction between the system and user are through simple and well

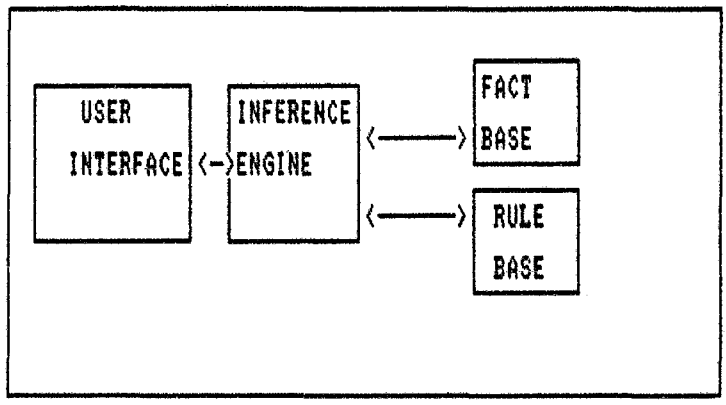

FIG 6.3 INTERNAL ARCHITECHTURE OF V.M.I.T.S

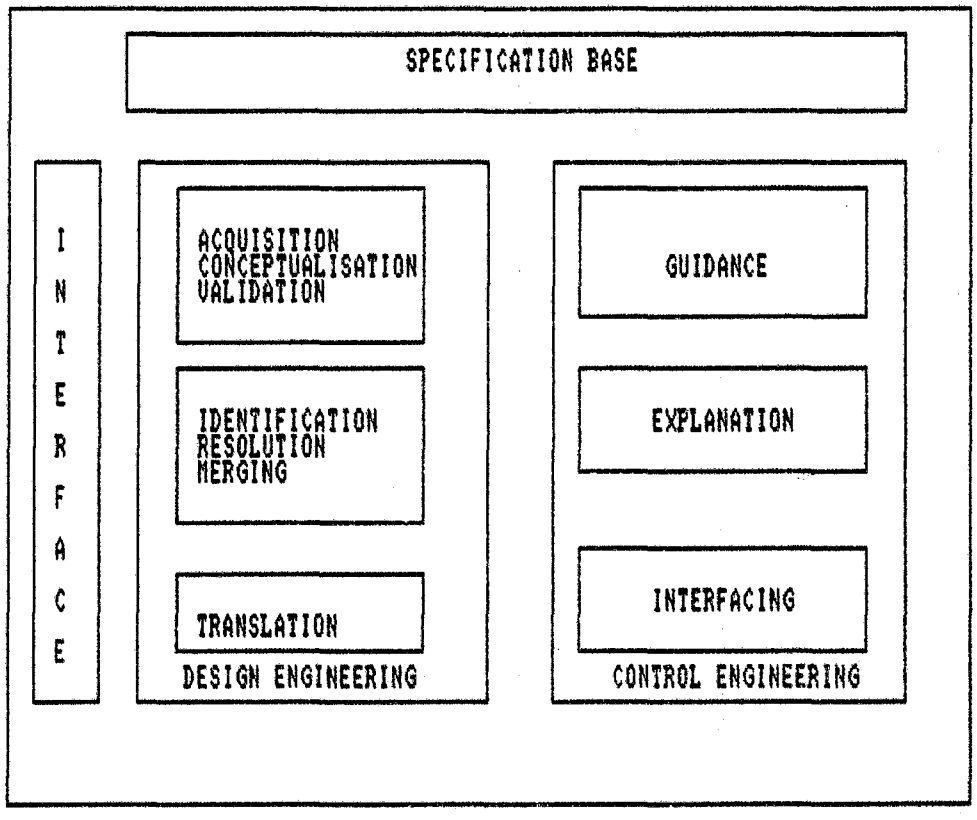

Fig. 6.2 Functional Architecture of VMITS.

formulated natural language queries, which require mostly "Yes/No" type answers. These queries are posed to the user through well structured screens aided with appropriate colors, appropriate examples, and definitions of the terms with which the user might not have been aware of, and also the whole query and interaction processes is appropriately guided by the control engineering's guiding process as show in Figure 6.4, that systematically assists the user to provide the required information.

6.4 Illustration of VMITS session

The scenario presented in this session aims at illustrating the database design supports provided by VMITS. The presentation is based on an application of this system to design a database for a Transport company. The result of the database design session with VMITS are given in Appendix B. The screen numbers used in the following description are the screen numbers of the outcome of the different phases of this session given in Appendix B.

It is assumed that the requirements and various application areas of the case study for which the database is required to be designed , have already been analyzed and the set of user views(a user view defines a portion of a

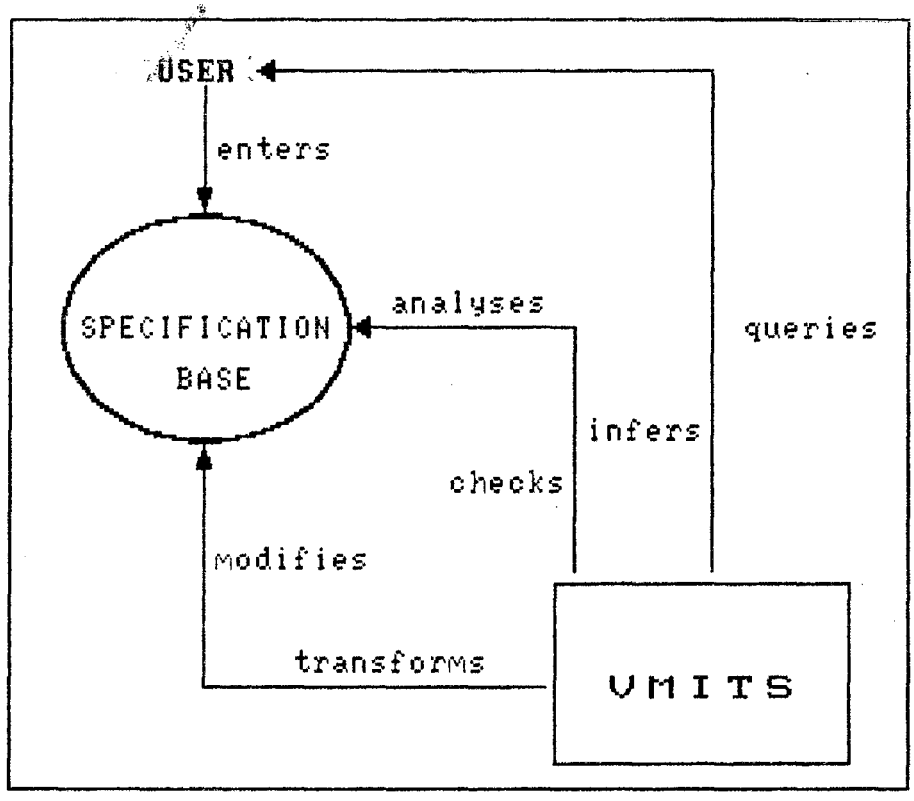

FIGURE 6.4 THE GUIDED PROCESS OF INTERACTION OF VMITS

database corresponding to major application(s)) have already been identified.

The design session proceeds as follows :

A. Preliminaries

First a few screen(1-6) introduces the user to the scope of the system, and then the main menu options are provided as follows:

(1) create/Add view.

(2) Integrate view.

(3) Translate view.

(4) Exit

Option-1 is used to create and add new views to a database system. When this option is selected by the user, the system asks the user to enter the name of the database and then to provide the list of views the user wants to create in this database system (Screen 7-8), and then the system proceeds to view modeling session. In the exit option, the system exits.

B. View modeling session. (Screen 3_23)

This stage is a bridge between the user's thinking and the database designers terminology. This session proceeds as follows :

B.1. Identify entities and their attributes

The user interested to design a view of database is likely to think in terms of objects about which he wants to

store information in or retrieve information from a database. To the database designer these objects are entities. The user is asked to provided a list of names of the objects identified by him/her. For each name provided by the user , the system checks if it is an entity name or an attribute name (the name A will be consider as attribute name if it is of the form X_Y or X# , where X is any string and Y is a key-indicating-attribute-name). If the name will be identified as an attribute, the user will be prompted to reconsider it and provided a different name of object under consideration. After obtaining an initial set of entities, the system then encourages the user to identify characteristics or properties of object which can be used to describe it. To the database designer these properties and characteristics are known as attributes. For each attribute name provided by the user, the system checks if it is an attribute of another entity, if it is then the system checks whether the entity already exists. If yes then the attribute will be added to the attributes of the entity, if not, then it will be added to the previous list of entities, and an assertion indicating that a relationship should exist between the original entity to which the attribute was belonging and the new entity identified now. Screens (11-15) show examples of this step.

B.2. Multivalued attributes.(Screen 16-17)

The system proceeds to :

(i) Give an example of multivalued attribute.

(ii) For each entity, ask the user

> is there any attribute that can take several values for each occurrence of the entity (Y/N) ?

(iii) if the answer to (ii) is 'no' then exit.

(iv) obtain the attribute from the user.

(v) Delete the attribute from the entity and add it as an entity to the list of entities.

(vi) Add the following assertion:

A relationship is needed between the newly formed entity and the original entity from which the attribute was deleted.

B.3 Input candidate keys (Screen 18-20)

The system will :

(i) Explain the concept of a key.

(ii) For each entity, Ask the user :

> Input the attributes that form the key.
Enter one attribute per line.
When done,press return.

(iii) Accept the attributes which the user gives.

(iV) If no key is provided for an entity (say E) then add (E_ID) as the key of E.

(iv) If any of the attribute belongs to another entity then ask the user :

> If the key of the second entity is required to determine the first entity (Y/N) ?

(v) If the answer to (iv) is 'NO' then ask the user to re-enter the attribute name.

(vi) Add the second entity name to the list of key attributes of the first entity instead of attributes given in (ii).

B.5 Determine ID-Relationships and ISA hierarchies

The system does the followings:

(i) If any candidate key of entity only contains another entity name, then do :

(a) delete the key of the entity.

(b) Add the ISA hierarchy " E1 ISA E2 ".

(ii) If any candidate key of an entity (say E1) contains another entity name (say E2) then do :

(a) delete E2 from the key of E1.

(b) Add the ID-Relationship " E1 ID E2".

B.5. Input relationships (screen 21-25)

B.5.1. Input Binary relationships (screen 22)

The system will :

(i) Give an example of a binary relationship.

(ii) Ask the user :
>Input Binary relationships in the form of "A VP B".

(iii) If the input does not confirm to the syntax of "A verb-phrase B" then ask the user to re_enter it.

(iv) accept the input.

(v) Check if any entity used in the relationship is not known to the system. If there is any then add it now as an entity.

(vi) If the input is of the form that "A verb_phrase A" then ask the user to provide two different roles for A. Accept the roles and convert this unary relationship to binary relationship.

(vi) for each entity E in this relationship (Screen 22)

> Give role of the entity E1 in the relationship :

(vii) Convert the input to a ISA hierarchy if appropriate.

B.5.2. Input N-ary relationships

The system does the followings :

(i) Give an Example of the concept of n-ary relationships.

(ii) Ask the user :

> Input N-ary relationship in the following ways.

> enter the name of the relationship :

> Give a list of entities over which the relationship is defined.

> Enter one entity per line. When done, press return.

(iii) Accept the relationship together with its entities.

B.5.3. Determine Cardinalities. (Screen 24)

The system does the followings:

(i) Explain the concept of cardinalities.

(ii) For each N-ary relationship R (E1, E2...., EN);

For each E_i (i=1,2,...n): Ask the user :

> Is there many values of E_i associated with any of the the remaining entities (Y/N) ?

(iii) If the Answer to (ii) is "Y" then define the cardinality of E_i as "MANY".

(iv) If the answer to (ii) is "NO" then define the cardinality of E_i as "ONE".

(v) If all the cardinalities of all the entities in a relationships are "MANY" , then the system ask the user to provide attribute for the relationship if there are any, and accepts them if the user provides (Screen 25).

B.6. Identify missing relationships

(i) Check if relationships are not defined between entities for which there are assertions in its internal database which indicates that a relationship is needed between them.

(ii) If the relationships are not defined then ask the user to provide one.

(iii) Identify solo entities (entities for which no relationships or ISA hierarchies exists, and ask the user to provide a relationship for such entities.

B.7. First modification (Screens 26-31)

At this state an initial design of the view of the database has been made. The system does the followings :

(i) Display the set of entities, relationships, ISA-hierarchies, and ID-relationships.

ii) Allow the user to modify this set.

iii) It displays the following menu :

1) Modify entities.

2) Modify relationships.

3) Exit.

> Enter your Choice :

iv) If the user selects choice (1), then the system displays the following menu :

1) Add entities.

2) Delete entities.

3) Rename entities.

4) Add attributes to entities.

5) Delete attributes from entities.

6) Rename attributes of entities.

7) Exit.

> Enter choice :

Depending upon the choice of the user the system takes the user to the required step.

Similar options are there for modifying relationships.

B.8. Detect Design errors

The system does the followings :

1. Identifies and resolves synonyms by comparing pairs of entities, relationships, and attributes.
2. Identifies redundant relationships and redundant ISA hierarchies, and resolves them.

B.9. Final Modification

Allow the user to make a final modification of the view.

B.10 Determine primary key for strong entities

The system then selects the simplest candidate keys(keys with least number of attributes) as the primary key of the entity from its set of candidate keys. An arbitrary choice is made when there are more than one simple key having the same number of attributes.

B.11. End modeling session (Screen 32)

The system then display the view designed in this session and then repeats the session for the next view in the view list. When all the views are designed the system then exits the session and goes back to the main menu of the

system. The user then can select the second option to activate the view integration session.

C. View Integration Session

C.1 Preliminaries of view integration(Screen 34)

The system first introduces the concepts and the steps of view integration process to the designer. Here the systems expects the designer to be familiar with both database concepts and application area for which the database is designed. The steps for this session are :

1. Select two views from integration.
2. Do conflict resolution.
3. Do merging and restructuring.
4. Do integration completion.

C.2. Select Views(Screen 35)

The system first displays the list of views for integration, and allows the user to select any two views. The two views selected in this stage are loaded into the systems database, and the system proceeds to the new stage.

C.3 Conflict Resolution

The conflict resolution stage includes the following steps :

1. Homonym Conflicts (HC1, HC2, HC3)
2. Synonym Conflicts (SC1, SC2, SC3, SC4)
3. Type Conflicts (TC1, TC2)
4. Degree Conflicts (DC1, DC2, DC3)
5. Cardinality Conflict (CC)
6. Role Conflict (RC)
7. Hierarchical Conflicts (HYC1, HYC2)
8. Key Conflicts (KC)

Homonym conflicts are identified by comparing pairs of Entity-Entity, Relationship-Relationship, and Entity-Relationship names. For each pair of homonyms determined, the user is asked to determine whether or not they have the same meaning, and the system resolves the conflicts by renaming them. Screen-39 and 40 shows examples of Entity-Entity homonym conflicts identification and resolution.

Synonym conflicts, unlike homonyms, cannot be completely identified by the system. It only identifies Entity-Entity synonyms and some Relationship-Relationship synonyms by using the heuristic described in chapter 4. To determine the other Relationship-Relationship, and the Attribute-Attribute synonyms, it presents to the user a blank matrix, where the rows list all the elements of one view, and column lists all the elements of the other view. Screens 41 and 42 show an example of matrices for the cases of Attribute-Attribute synonym conflict (SC4).

The type conflicts, the cardinality conflicts, the role conflicts, the degree conflicts, the hierarchical conflicts, and the key conflicts are all determined by the system, and are presented to the user for interactive resolution. Screen-43 shows an example of Hierarchical conflict analysis step.

C.4. Merging

After resolving all types of conflicts between two views the system proceeds to the next step, that is, the merging and restructuring step. This is done in the following three stages using the merging and restricting algorithms given in Chapter 4.

1) Entity-Relationship merging : Entity and relationship pairs which are identical in meaning are merged first.
2) Entity-Entity merging : Pair of entities which are identical or related are merged and restructured using the inter-schema properties generated during conflict analysis phase.
3) Relationship-Relationship merging : Pairs of similar relationships are merged into a single relationship.

Once the two views are integrated into a single view, the system displays the integrated view , deletes the

two views from the viewlist and adds the integrated view into this list views for integration. This integration phase is repeated till all the views are integrated into a single view, and then the system proceeds to the third stage of database design, that is, the view Translation stage.

D. View Translation Stage

The conceptual schema generated in the last stage is now automatically translated into a set of 5NF relations and a list of inclusion dependencies using the translation algorithm described in chapter-V. The conversion of the EER model to a well formed EER model is not used in the present system. The output of this stage are shown in screen-47, which represents the relational schema for the Transport Information System.