CHAPTER III

VIEW MODELING METHODOLOGY

3.1. INTRODUCTION

A view is a specification of the content of a database appropriate for a single task that some user or group of users perform. The collection of all user views can be taken as a specification of the required contents of the database. View modeling is the process of eliciting user views by analyzing the user requirements and the information needs of different applications and the user groups in an organization. According to Navathe and schkolnick [102] there are two major tasks in the view modeling :

- Extracting from user or from person incharge of application development the relevant parts of the real world.
- Abstracting this information in a form that completely represent the user views so that it can be subsequently used in the design.

View representation has been addressed mainly as a by product of data model development. According to Navathe and Schkolnick [102], the most prominent work done in this area has been the entity relationship (ER) data model

proposed by Chen [30] and the data abstraction methodology of Smith and Smith [119]. Navathe and Schkolnick have proposed their own data model, the Navathe_Schkolnick (NS) data model as a vehicle for modeling user views. In addition three methodologies have been developed exclusively for constructing user views. These are Bubble charting method by Martin [94], the Interactive Specification Methodology of Baldissera et al. [9], and the view creation system of Storey [120].

The objective of the present methodology is to expand the previous research on view modeling by providing the followings :

1. Rules and heuristics for distinguishing among entities, attributes and relationships.
2. Rules for identifying missing entities, relationships and data abstractions.
3. Rules for capturing some of the semantics of an application through Data Abstractions such as generalization and specialization hierarchies, and aggregations.
4. Rules and heuristics for identifying and resolving potential design problems such as synonyms, identifying redundant and conflicting hierarchies, and relationships.

5. Rules for determining primary keys of entities.

Finally, the methodology is described in a step-by-step manner that facilitates the development of a computer-aided system (such as the present VMITS) that can interactively obtain the information requirements of users for the view of the database and generate a user view by analyzing them.

This chapter is divided into four sections. In section 2 we give an overview of related works on view modeling. Section 3 describes the design methodology in detail, and section 4 gives a summary of this chapter.

3.2. REVIEW OF VIEW MODELING METHODOLOGIES

ER Model

The ER model proposed by P. Chen [30], consists of three classes of concepts- Entity, Relationship and Attributes, and a semantic link between entities - the IS_A hierarchy. The terms entity, relationship and attribute are here used in the sense of entity-set, relationship-set and attribute set. An entity set is a group of entities (abstract and concrete things which can be properly identified) which share the same property. A relationship set is a mathematical relation between entity

sets. An attribute is a function which maps an entity-set or a relationship set into a value-set. The IS_A hierarchy is differentiated into two types : subset-hierarchy and generalization-hierarchy. An entity E is a subset of another entity E_1 (E IS_A E_1) if every occurrence of E is also an occurrence of E_1. An entity E is a generalization of the entities E_1, E_2, ..., E_n, if, for each i=1,2,..,n, E_i IS_A E and each occurrence of E is also an occurrence of at most one of the entities E_1, E_2, ..., E_n. Semantics are captured by ER model through the concept of role and by connectivity of a relationship between occurrences of participating entities. When using ER model, clear identification and distinction of the main constructs is required, however, it may be difficult to determine whether something should be represented as an entity, as a relationship, or as an attribute of an entity or relationship. Since each view-level ER model is relatively small and focused upon a single application, these factors aid in reducing the complexity of view construction.

Data-Abstraction

In Smith and Smith's Data Abstraction methodology [14], both entities and relationships are represented as objects. Two different types of

relationships are represented : association of similar entities and association of related entities, known respectively as generalization and aggregation data abstraction types. Both are used to build a multidimensional hierarchy of objects that represent a user view. The Data-Abstraction approach has been described as conceptually elegant in terms of logical database design; however, it has been criticized when it is applied to the community view of data as opposed to the individual view when large-scale problems are involved. The Data-Abstraction approach demands a great deal of skill on the part of the designer.

NS Model

Navathe and Schkolnick drew upon the concept of Data-Abstraction methodology to develop their own model (NS-Model [102]). The objective of the NS model is to capture a great deal of semantic constraints and model a user view as explicitly as possible. The NS model has two types of constructs : objects and connectors, with object types being divided into entity and association types. User views are represented by a graph with entities and associations at its nodes and connectors between nodes. User views presented in a view diagram in the NS model and are represented in forms of entities, associations, attributes and connectors.

Associations are n-ary relationships among entities, entities and associations, or among associations. They represent facts, ideas or specific aspects of a relationship.

Bubble charting

Bubble charting is a method proposed by Martin [94], representing user views whereby the users are taught to draw representation of data structures. The fundamental piece of data represented by this technique is a data item - the smallest piece of data that is meaningful to a user. An example of a data item is student-name. This approach is in contrast to the development of a user view from higher-level concept such as entities and relationships. Bubble charts are drawn by first representing individual data items by ellipses, or bubbles. Associations between the data items are represented by drawing links in the form of single-headed and double-headed arrows between the data items. Single-headed arrows between two data-items imply that each value of the first item has one and only one value of the second item associated with it. Double-headed arrows indicate that one value of the first item may have zero, one or many values of the second item associated with it.

The Interactive Specification Methodology

The Interactive Specification Methodology (ISM), developed by Baldissera et al. [9], is a part of the project DATAID, dealing with the view modeling function in an interactive manner. The user provides the system with some elementary sentences of the form "A is_a B", "A has B" and "A verbal_construct b" that express his or her requirements. The system then recognizes the sentences to provide a canonical relational representation of the user's view. The underlying data model that represents views is an extension of the binary data model. After accepting input sentences describing the application requirements, the system takes the responsibility of:

- linking together different requirements of the user, so that the user need not be concerned with how he or she organizes his or her input sentences.
- checking that the representation is formally correct and detecting inconsistencies and redundancies.
- Presenting to the user different ways in which the collected requirements be interpreted and asking the user to resolve any ambiguities.
- Producing a schema for the design phase that follows.

The View Creation System

The view creation system developed by Storey [120], is an expert system that formalizes the view modeling methodologies as a set of rules and automates the process of generating user views. It engages the user in a continuous dialogue about the information requirements for the database application, develops an entity relationship model for the user's database view, and then converts the ER model to a set of Fourth Normal form relations. The view creation methodology is represented as a set of rules that forms the knowledge base of the VCS. The overall procedure that controls the creation of user views consists of the following steps :

Step-1. Identify entities, their attributes and candidate keys.

Step-2. Determine relationships, relationship attributes and cardinalities.

Step-3. Detect and resolve ambiguities, redundancies and inconsistencies.

Step-4. Select primary keys for entities.

Step-5. Represent entities and relationships as relations.

Step-6. Identify and resolve partial and transitive functional dependencies.

3.3. The View Modeling Methodology

The view modeling methodology implemented in the expert system VMITS is based on the Extended Entity-Relationship (EER) model, and employs the ideas from the Interactive Specification Methodology of Baldissera et al. [9], the logical relational design methodology of Teory et al. [125] and the view creation system of Storey [120]. The methodology can be divided into the following steps as illustrated in Fig. 3.1 :

1. Identify Entities and their attributes.
2. Identify relationships (including Unary, Binary, n-ary, and aggregations), their cardinalities, and attributes.
3. Define ISA hierarchies.
4. Identify missing relationships.
5. Identify and resolve potential design problems.
6. Define primary keys of entities.

The methodology can be used in the design of a computer-aided system (as shown in the expert system VMITS) that can systematically obtain the information requirements of users for a particular view of the database, assisting the designer to identify and resolve potential design problems, and finally producing an Extended-Entity-Relationship schema

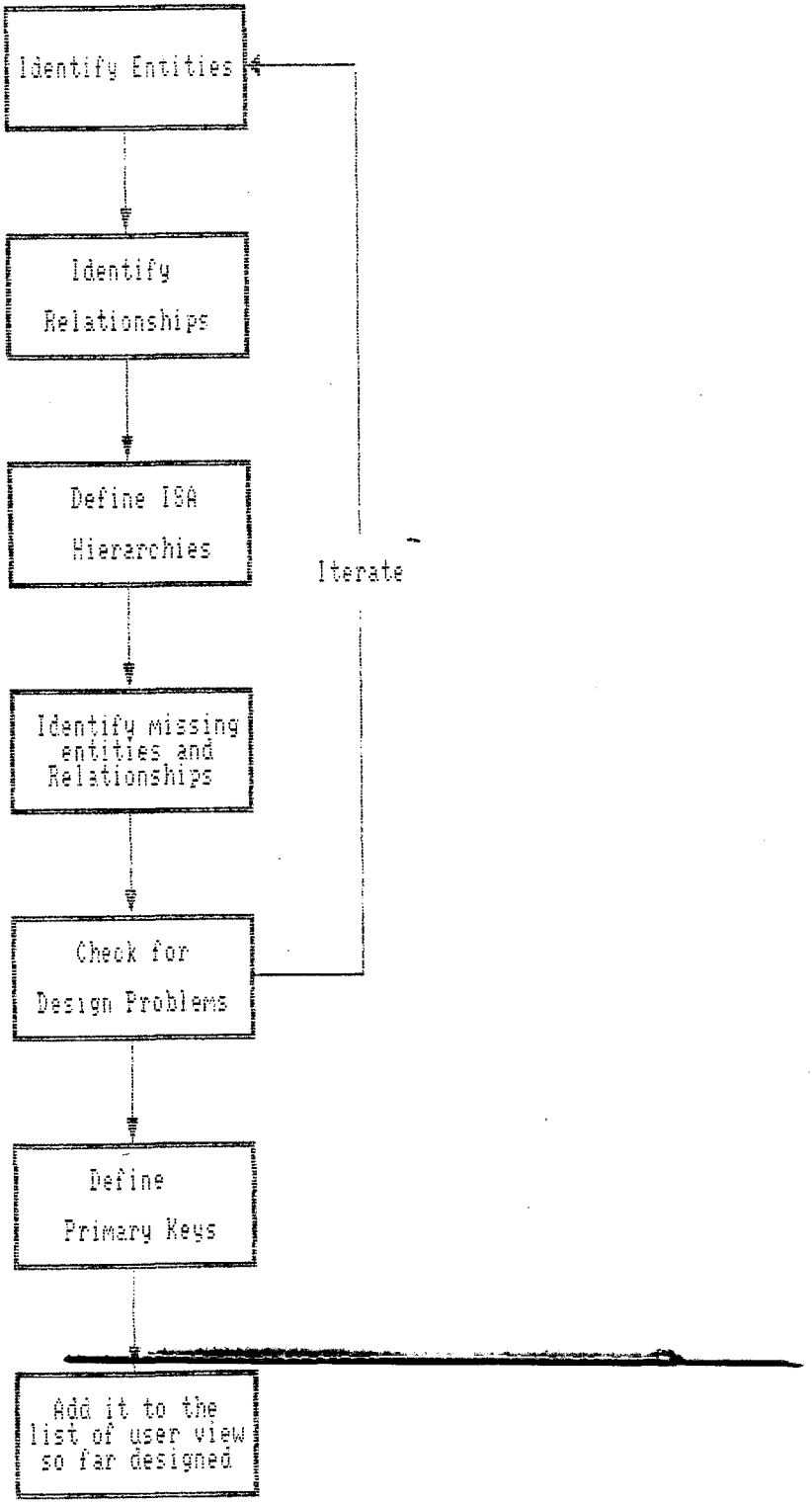

Figure 3.1 View Modelling Methodology.

corresponding to the view of the database. Each of the design steps is explained in detail in the following subsections.

3.3.1. Identify Entities

The first step in the view modeling process is the identification of entities. By analyzing the user requirements about the objects about which the user wants to store and retrieve information, the methodology tries to identify some entities. The other entities then can be identified in an iterative manner. For each entity identified, the attributes and the candidate keys must be defined. The design should then be checked for indications of missing entities.

3.3.2. Identify Attributes

For each entity, a set of attributes (properties or characteristics) is obtained. Since it is not easy to define the roles of entities, attributes and relationships in the database and the final objective of the methodology is to produce a good EER schema, the following steps need to be considered while identifying entity attributes :

Step-1. Attach attributes to entities that they describe most.

If an attribute A of an entity E_i is similar in name to that of another entity E_j or is of the form X_Y, or of the form X#, where X and E_j are similar (for example Department and Dept are similar) and Y is a generic key attribute name such as ID, KEY, NO, NUM, NUMBER, NAME, TYPE, CODE, etc., then A should be made as a separate entity E_j and a relationship between E_i and E_j should be added as per the following cases :

Case-1. A is needed for unique identification of entity E, then the ID_relationship "E_i ID E_j" should be added.

Example : Department (DIVN#, DEPT#, name)

Key is [DIVN#, DEPT#] Here DIVN# is an attribute of the entity DIVISION. Eventually, department will be made as a weak entity by deleting the attribute DIVN# from it and adding the ID_relationship "Department ID Division" as shown in the Figure 3.2.

Case_2. A is not needed for unique identification of E_i. In this case, the attribute suggest the existence of a relationship between E_i and E_j (see Fig. 3.3)

Step-2. Multivalued attributes and repeated attributes should be converted to entities.

If an attribute A of an entity E_i has multiple values for each occurrence of the entity, then it should be

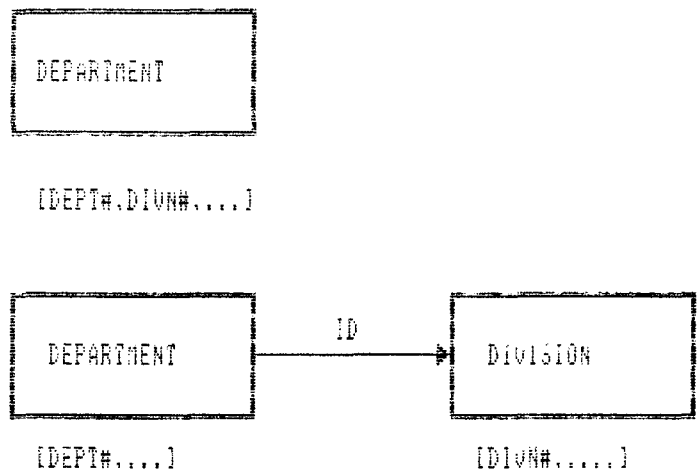

FIGURE 3.2 REPLACING ATTRIBUTE BY ENTITY

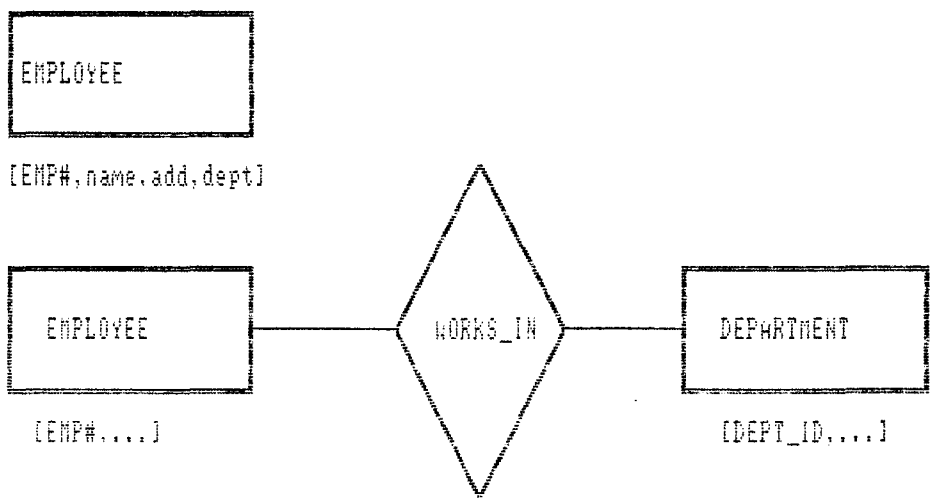

FIGURE 3.3 RELATIONSHIP REQUIRED INSTEAD OF ATTRIBUTES

converted to an entity E_j and a relationship should be added between E_i and E_j.

Example: EMPLOYEE (EMP#, name, qualification, job_title, salary)

An employee can have many qualifications, so qualification is a multivalued attribute.

This becomes:

EMPLOYEE (EMP#, name, job-title, salary)

QUALIFICATION (Degree, institute, year_of_Passing, grade)

A relationship is required between EMPLOYEE and QUALIFICATION as shown in Fig. 3.4.

Similarly, if some of the attributes of an entity E_i are of the form A_1, A_2, A_3,... A_n, where A's are repeating attributes then A should become a separate entity and a relationship should be added between E_i and A to reflect the original association.

Example: Employee (EMP#, name, qualific_1, qualific_2, qualific_3, job-title, salary)

Here qualific is a repeating attribute so it becomes a new entity.

Employee(Emp#, Name, Job-Title, Salary)

Qualification (Degree,...)

A relationship is added between Employee and Qualification as shown in Fig. 3.5.

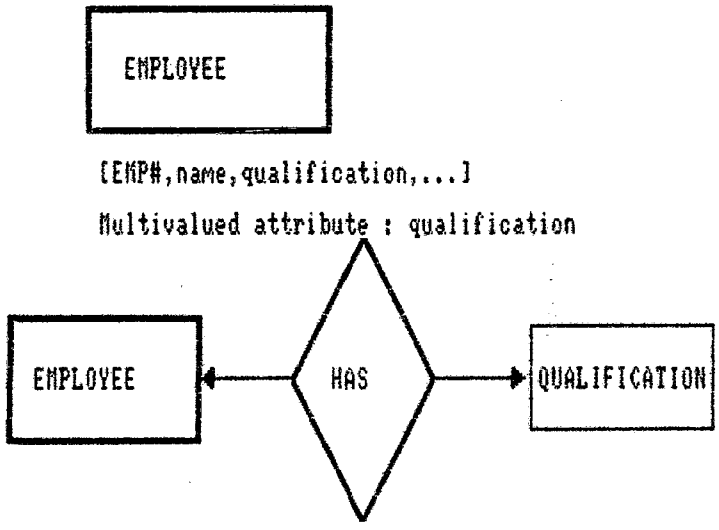

FIGURE 3.4 Multivalued attribute qualification becomes new entity

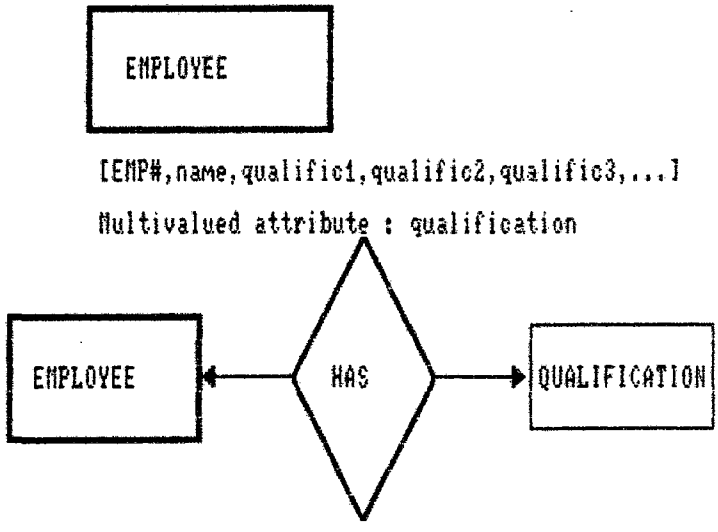

FIGURE 3.5 Repeated attributes become new entity

3.3.3. Identify Candidate Keys

A candidate key is a set of attributes that uniquely identifies each occurrence of an entity. For each entity identified, the candidate keys should be specified, and the primary key will be chosen from these candidate keys. The generic key attributes (such as ID, NO, NUM, NUMBER, TYPE, CODE , etc.) should be considered as possible candidate keys when they appear in the entities definition. In addition attributes of the form X#, Where X is similar to the entity to which it belongs should also be considered as possible candidate key when it occurs in the definition of the entity.

Example:

Employee (EMP#,name, add,...)

Keys are : [EMP#] and [NAME,ADD]

An entity is said to be a weak entity when it does not have its own attributes to uniquely identify it. Such entities should be identified by examining the candidate keys defined for it. If an attribute A appearing in the candidate key of an entity E_i is similar to that of another entity E_j or is of the form X_Y or X#, where X and E_j and similar and Y is a generic key name, then the attribute A Should be deleted (when the primary key of the entity is defined, it will be compensated by adding the key of E_i to the candidate key) and the entity E_j will be declared as a weak entity by adding

the ID_relationship "E_i ID E_j".

Example:

Employee (EMP#, name, position, salary,...)

Dependent (Name,Age,...)

Candidate key of Dependent is : [Emp_ID, Name]

Eventually the candidate key of Dependent will become [NAME] and the relationship "Dependent ID Employee" will be added as illustrated in Figure 3.6.

3.3.4. Identify Relationship

After identifying entities and their attributes the next step is to define relationships among the entities and other relationships. Although many database problems can be modeled by using Binary Relationships only, to capture more semantics and to make the system more flexible, the methodology also allows other types of relationships such as Unary and N_ary relationships among entities and Binary relationships between entities and relationships. The procedure for obtaining these three types of relationships consists of the following steps :

Step-1. Obtain Relationships.

Binary relationships between entities can be expressed in the form of "A verb-phrase B", where A and B are

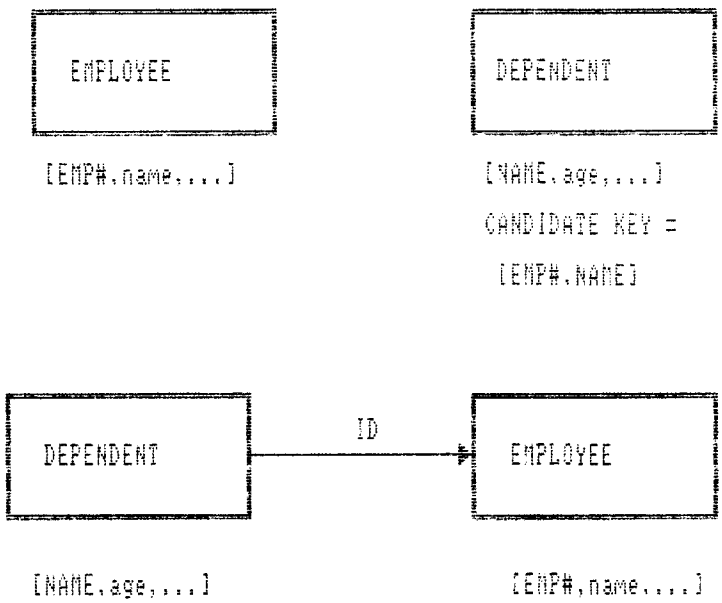

FIGURE 3.6 ID_relationship required instead of attributes

entities and verb-phrase is the relationship between the two. For example :

"Employee work-in company" is a binary relationship. Unary relationships can be specified in the form of "A verb-phrase A", where A is an entity and verb-phrase is the relationship. For example:

"Person Spouse of person" is a Unary relationship. N-ary relationships can be expressed as "relationship [A,B,C,D,...]", where A,B,C,D,... are entities. For Example:

"USE [ENGINEER,SKILL,PROJECT]" is a ternary relationship.

Step-2. Examine relationships for missing entities.

If any of the entity name used in the relationship is not defined previously as an entity then it should be defined at this stage.

Step-3. Identify redundant relationships and obtain roles for entities if appropriate.

If different relationships are defined over the same set of entities, i.e. if R1(Ent_set) and R2(Ent_set), then the relationship should be checked if they represent the same information in different ways. For each such pair of relationships identified, the designer may be asked to identify if both of them represent the same information or not. If yes, then one of the relationship should be deleted

from the design, otherwise the designer may be asked to provide different roles for each entities participating in the two relationships. Roles should also be obtained for each entity participating in unary relationships.

Example :

1. Redundant relationships (See Fig. 3.7)

Relationships : Department offers course

Course offered_by Department

Here both of the relationships represent the same information so one of them should be deleted from the design.

2. Unary Relationship (See Figure 3.8)

Person Spouse of person.

The entity person has roles male and female in this relationship.

3. Cycle (See Figure 3.9)

Faculty Teach course

Faculty Supervise Course

In the relationship "Teach" the entities Faculty and course have roles "Teaches" and "Taught_by" respectively, and in the relationship "Supervise" they have the roles "Supervise" and "Supervised_by" respectively.

Step_4. Examine Unary relationships for missing subtype entities.

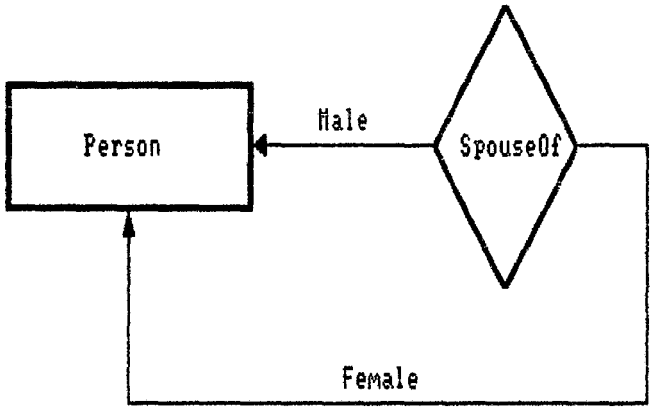

Figure 3.7 Unary Relationship.

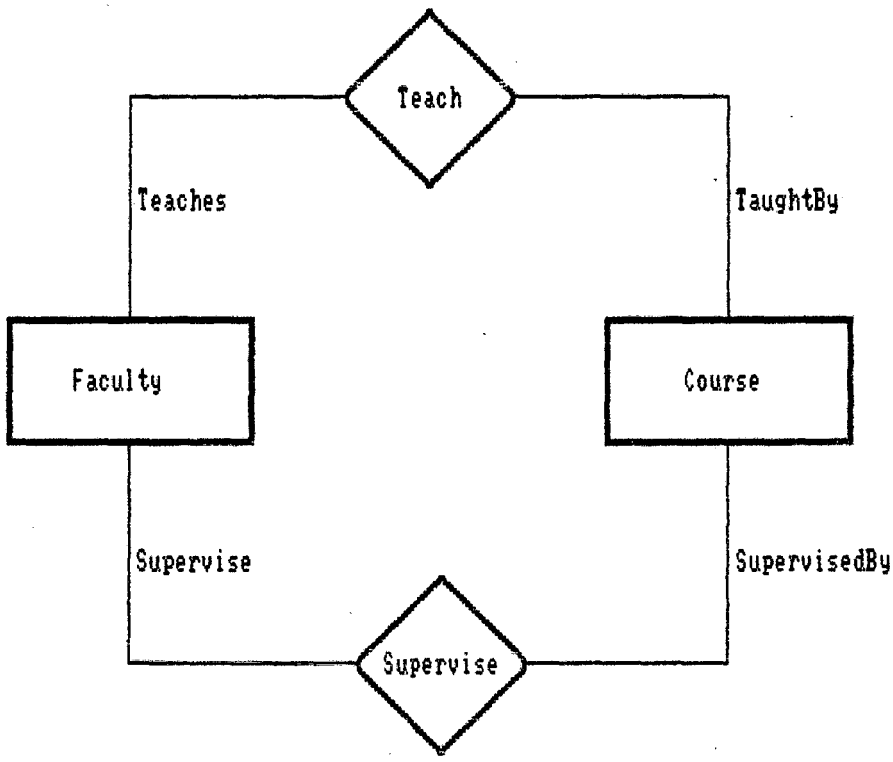

Figure 3.8 Entities with roles in the cycle.

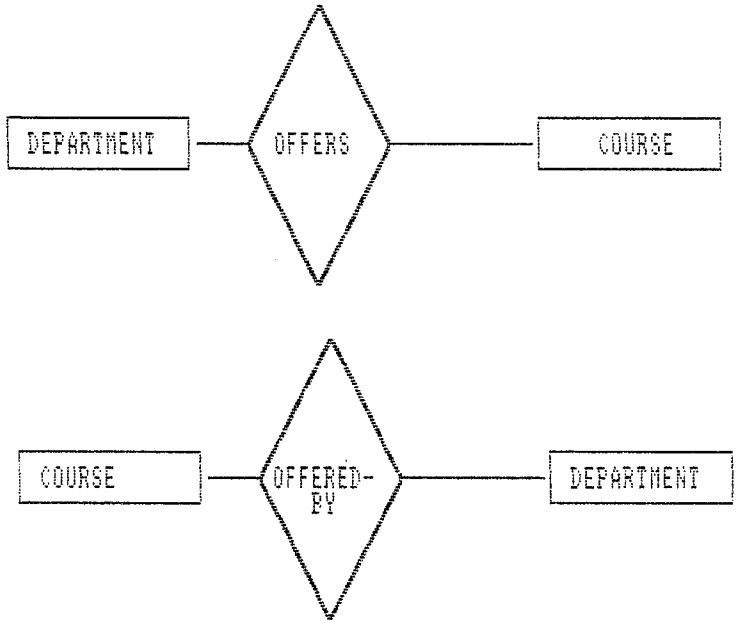

FIGURE 3.9 Redundant Relationships

For each Unary relationship "E verb-phrase E" it is required to obtain a further refinement in the kind of E entity types that can exist. This is determined by the role that each entity plays in the relationship and could result in the following actions :

1. The addition of two new entities E_1 and E_2.
2. The addition of two generalization hierarchies:
   a) E_1 ISA E
   b) E_2 ISA E
3. Modification of the original Relationship:

The unary relationship "E verb_phrase E" is modified by the binary relationship:

E_1 verb_phrase E_2

Example : (See Figure 3.10 and 3.11)

Relationship : person spouse-of person

Person has roles male and female in this relationship.

Becomes :

Male_person Spouse_of female_person

Male_person ISA person

Female_person ISA person

Step-5. Assign Cardinalities

For a binary relationship "E_1 Verb-Phrase E_2", the cardinality describe the maximum number of E_1 values that can

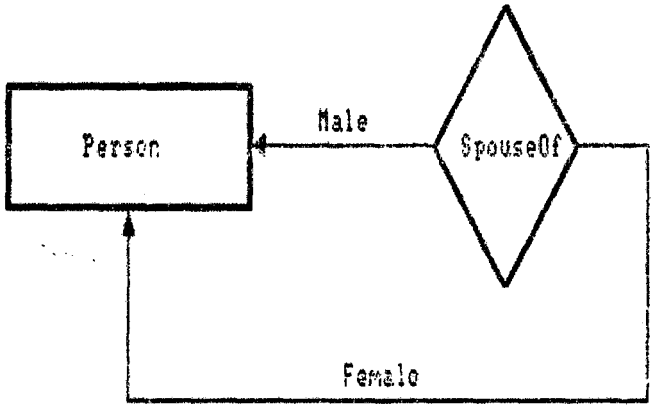

Figure 3.10 Entity with roles in unary Relationship.

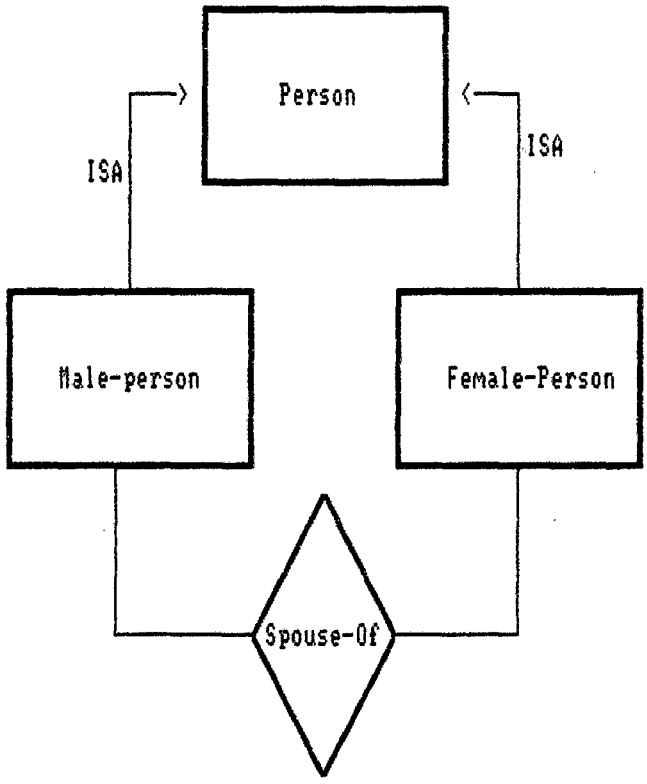

Fig. 3.11 Addition of Subtype Entities

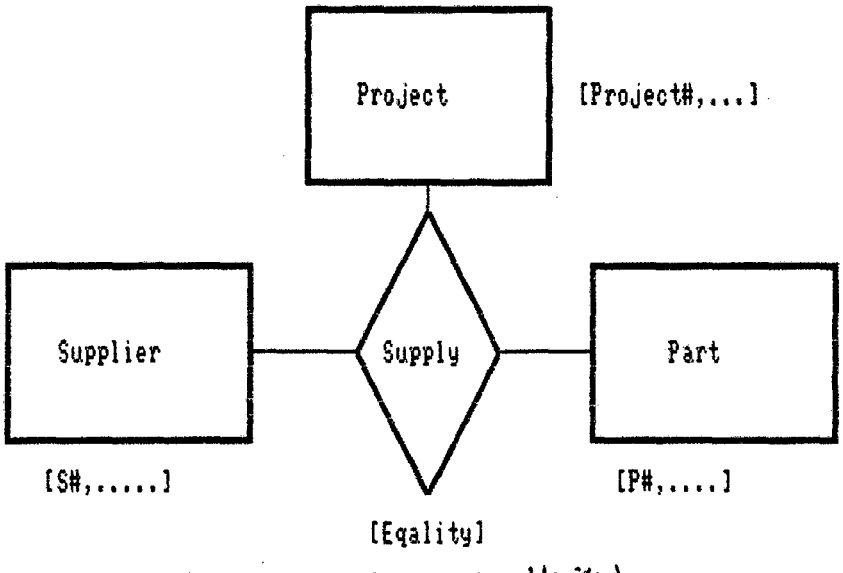

Figure 3.12 Relationship Attribute

occur for each value of E_2 and vice versa. For n_ary relationship "R [E_1,$E_2$2 ,....., E_n)" the cardinality of E_i, for i=1,...,n, describe the maximum number of E_i value that can be associated with each occurrence of the sequence E_1, E_2,.., E_{i-1}, E_{i+1},...,E_n.

Example :Employee Work_in Company

(M) (1)

In this relationship many employees can work in one company, and each company can have many employees.

The cardinality of E_i in the n-ary relationship "R [E_1,.....,E_n)" can be obtained by posing the questions :

- Is there many values of E_i for any of the remaining entities?
  (If yes then the cardinality of E_i 'one', otherwise it is 'many')

Step-6. Identify Relationship Attributes if appropriate

As with entities, relationship can also have attributes (properties or characteristics of the relationships as a whole). However, unlike entities which usually have corresponding attributes, only certain types of relationships can have attributes. For n-ary (n >= 2) relationship defined over entities E_1, E_2,....,E_n attributes exist only when none of the entities E_i for i=1...n, has cardinality equal to 1.

Suppose that the n-ary relationship $R(E_1, E_2,\ldots,E_n)$ has a relationship attribute (called R_a) and the cardinality of E_i=1, Then R_a is a function of the entities E_1, $E_2,\ldots\ldots,E_n$, that is, $R_a = f(E_1,\ldots\ldots,E_n)$. However, since there is one and only one value of E_i for every value of the sequence $(E_1,\ldots, E_{i-1},,E_{i+1},\cdots E_n)$, $E_i = g(E_1,\ldots,E_{i-1}, E_{i+1},\ldots,E_n)$, i.e. E_i is a function of the remaining entities. So we have, $R_a = h(E_i)$, which implies that R_A is probably an attribute of E_i instead of the relationship as a whole.

Example (See Figure 3.12)

Relationship : SUPPLY [Supplier,Part,Project]: [Quantity]
Here Quantity is an attribute of this relationship.

3.3.5. Define ISA Hierarchies

A number of researchers have proposed various higher-level data modeling concepts to capture more information stored in the database. These concepts are commonly known as Data Abstractions. The most common form of abstractions are generalization and subset hierarchies, Aggregation, and Associations. In the present EER model we allow only the Generalizations and subset hierarchies to which we will commonly refer to as "ISA hierarchies" and aggregations. The following steps are used to define ISA hierarchies:

Step_1: Convert Relationships to ISA Hierarchies if appropriate.

The generalization and subset hierarchies have been captured in the form of the following binary relationships :

1. is_a
2. contains
3. possess
4. instance-of
5. component_of
6. Possess_by
7. Part_of
8. Contained_in

Such relationships are replaced by the ISA hierarchy as illustrated in Figure 3.13.

Step-2. Identify and resolve Attribute Synonyms between the entities Participating in the ISA Hierarchies.

Attribute synonyms arise if different names are associated with the same attributes of the entities participating in the ISA hierarchies. For Example:

Employee (EMP#, name, add, dob,..)

Manager (EmM#, name, address, birthdate, type..)

Here the attributes DOB and BIRTHDATE both refers to the same property that is the date of birth, and similarly both ADD and address refers to the same address of the employees.

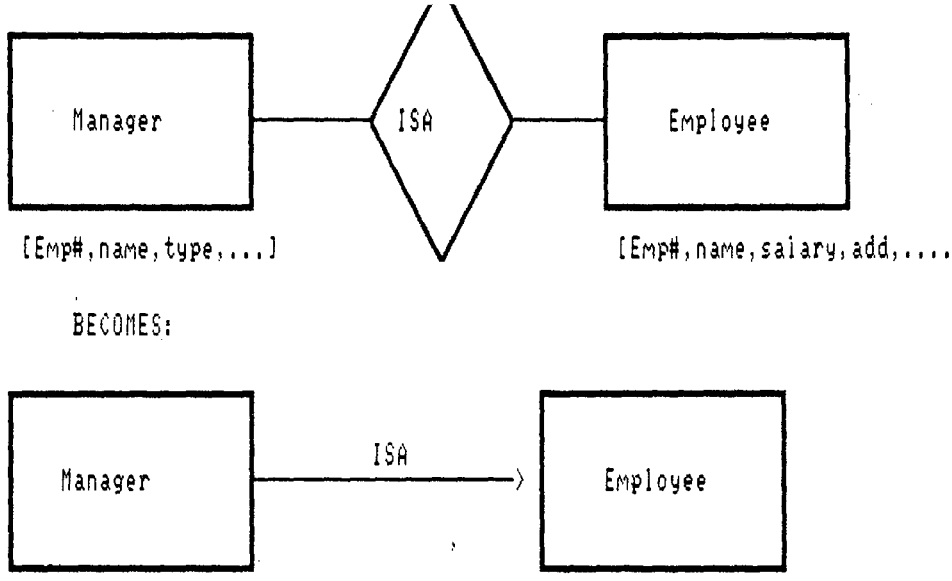

Figure 3.13 ISA hierarchy is needed instead of relationship

They should be identified and resolved by assigning same name (for example DOB and ADD in this case) to the attribute in both the entities.

Step-3. Adjust the Attributes and Keys for Generic and Specialized entities in the ISA Hierarchies.

The attributes that are common to a set of entities in an ISA hierarchy should be stored at the highest level entity and deleted from the specific entities below it. The specific entities can simply inherit these attributes and also the key of the generic entity in order to minimize the redundancies that appear in the design.

Example :

ISA hierarchies:

Manager ISA Employee

Sales_Manager ISA MAnager

Initial set of attributes for entities:

Employee (EMP#, name, add, phone_no, salary)

Manager (type, name, add, phone_no)

Sales-Manager (name, divn#, quantity, phone_no, add)

Final set of attributes after adjustment :

Employee (EMP#, name, add, phone_no, salary)

Manager (type)

Sales_Manager (div#, quantity)

3.3.6. Identify Missing Relationship

To ensure that the EER model designed is a complete representation of the real world, it is necessary to check for the missing and redundant relationships. The following heuristics may be used for identifying missing relationships:

- An entity that does not appear in any relationship or is not a weak entity might signal a missing relationship.
- The appearance of some attributes in one entity that refers to some other entity suggests that a relationship should exist between the two entities. This situation arises in a numbers of ways (see section 3.3.2).

(a) If an entity E_i originally has a multivalued attribute that was converted to another entity E_j, then a relationship should exist between E_i and E_j.

(b) If an entity E_i originally had repeated attributes of the form $X_1, X_2, \ldots, X_n$ and X becomes a new entity, then a relationship should list between E_i and X.

(c) If the name of an entity E_j, was originally similar to an attribute of another entity E_i, (for example the entity Department and the attribute Dept are similar) then a relationship should exist between E_1 and E_j.

(d) If the name of an entity E_i was a part of the name of an attribute of another entity E_j (for example the entity Department is a part of the attribute Department_ID) or

if the attribute of the entity E_j is of the form X_keyname (where key name is a generic key attribute name) or of the form X#, and X and E_i are similar, then a relationship should exist between E_i and E_j.

Example:

Employee (Emp#, name, Add, Salary, Dept#)

Becomes :

Employee (Emp#, name, add, position, salary)

Department (Dep#,.....)

A relationship should exist between employee and department (for example "Employee work_in Department".

- An entity E_i, that has two one-to-many relationship with two other entities E_j and E_k which are not directly related, suggests that a relationship might exist between E_j and E_k.

Example:

The two relationships:

Department Offers Courses

(1) (M)

Department Has Faculties

(1) (M)

suggests that there could be a relationship between faculty and course entities (e.g. faculty teach course).

- Two entities E_j and E_k which are not directly related but are linked via a third entity by two many-to-many relationships suggests that a relationship between E_j and E_k might be missing.

Example: The Two relationships:

Student (M) Registered_for (M) Course

Faculty (M) Teach (M) Student

Suggests that a relationship between faculty and course (e.g. course assigned_to faculty) might be missing.

3.3.7. Identify and Resolve Potential Design Problems

At this point of design an initial EER model has been designed. It should now be examined for potential problems such as entity and relationship synonyms, redundant and missing hierarchical links and relationships, and hierarchical conflicts.

Entity Synonyms

Entity synonyms occur when different names are assigned to the same entities in the design. The following situations suggests that a pairs of entities E_i, and E_j might be either synonyms or related in some way such as one is a subset of other or both are subsets of a common entity. They may also be used to identify missing hierarchical links and

entity types. The following heuristics are used to identify entity synonyms :

1. If the entities E_i and E_j have a common candidate key, then they are candidates for synonyms. For example :

   Employee: (Emp#, name, add, salary...)

   Manager: (Emp#, name, type...)

Here the entities have identical candidate key emp# and since they are not synonyms, the hierarchical link "Manager ISA Employee" may be identified and added to the design.

2. If the entities E_i and E_j have similar names, then they are candidates for synonyms.

This occurs in a number of ways :

(a) E_i is a part of E_j (e.g., Student and PG-student use similar names)

(b) E_j is a part of E_i.

(c) Both E_i and E_j have a common part (e.g. Pg-student and Ug_student have a common part).

(3) Both the entities are subset of a generic entity. For example :

   Research_Scholar ISA Student

   Phd-Student ISA Student

Suggest that Research-scholar and Phd-student may be synonyms.

(4) The entities participate with other entities in identical relationships. For example:

Relationships: E_1 Verb_Phrase E_2

and E_2 Verb_phrase E_3

Suggest that E_2 and E_3 might be either synonyms or related in some way that has not yet identified. For example:

Manager work_in department

Secretary work_in department

Suggests that the entities Manager and Secretary might be synonyms or related through the hierarchies "Manager ISA Employee" and "Secretary ISA Employee". Such hierarchical links should be obtained from the designer and included in the system.

Relationship Synonyms

Relationship synonyms occur when same information is represented by different relationship names. By examining relationships that are defined over same set of entities, the relationship synonyms can be identified. For Example:

E_1 Verb_Phase_1 E_2

E_1 Verb_phase_2 E_2 (or E_2 verb_phase_2 E_1)

Suggest that verb-phase-I and verb-phase - 2 may by synonyms.

Identify Redundant Hierarchies

An ISA hierarchy is redundant when it can be obtained from the transitive closure of the remaining ISA hierarchies. For Example :

Hierarchies : E1 ISA E2

E2 ISA E3

E1 ISA E3

Suggest that "E1 ISA E3" is redundant as E1 is transitively a subset of E3. Such hierarchies should be identified and deleted from the design.

Identify Hierarchy-Hierarchy Conflicts

Hierarchy-Hierarchy conflicts arise when contradictory hierarchical links exist between the same entities in the design.

For example:

Hierarchies : E1 ISA E2

E2 ISA E1 (or E2 is transitively subset of E1)

Suggests that there exist contradictory hierarchical links between E1 and E2. Such conflicts should be identified and are resolved by asking the designer to identify the correct ISA relationships.

Identify Hierarchy-Relationship Conflicts

A Hierarchy_Relationship conflict arise when there exist a hierarchical link and also a relationship between the same entities in the design. Form Example:

Hierarchy : E_1 ISA E_2

Relationship : E_1 Verb_Phrase E_2

Suggest that the same information is represented in two different ways. Such conflicts are resolved by rechecking the pair of entities and deciding whether to keep the relationship information or the hierarchical link information in the design.

Example:

Employee manage_by Manger

Manager ISA Employee

convey that the relationship and the hierarchical link are redundant and one of them should be deleted from the design.

3.3.8. Iterations :

The initial steps in the design process can be iterative. New entities and attributes might come to the mind of the designers as the relationship are identified and the design is examined for design errors. Thus these steps might need to be reiterated until the EER model appears to be an

accurate representation of the user view of the required database.

3.3.9. Define Primary Keys

The final step in the design is to define the primary keys for the entities. The entities can be categorized into three types : (i) entities which do not depend on other entities for their unique identification and which are not subset of other entities, known as strong entities,(ii) entities which depend on the other entities for their unique identification known as weak entities; and (iii) entities which are subset of other entities, known as specialized entities. We describe below the methods for defining the primary key for these three types of entities.

3.3.9.1. Primary Keys of Strong Entities

Each strong entity has a set of candidate keys that uniquely identifies each occurrence of it. The primary key for these entities are selected from the set of its candidate keys by choosing the simplest candidate key (that is the candidate key with least number of attributes) from the set. If there is a tie for the simplest candidate key, then choose the one that is the most application specific. For Example:

Entity: Employee (Emp#, Name, Add, Dob,...)

Candidate keys: [EMP#], [NAME,DOB], and [NAME,ADD]

Here the candidate key [emp#] will be selected as the primary key as it is the simplest one, and the remaining candidate keys will be made as alternative keys.

3.3.9.2 Primary Keys of Weak Entities :

Weak entities require in addition to their candidate keys the Primary key of other entities of their unique identification. The primary key of such weak entity, can be identified by first selecting the simplest candidate key from set of candidate keys and then, adding the primary key of the entity on which it depends for it's unique identification to its simplest candidate key. For Example:

ID-Relationship: Dependent ID Employee

Entities: Employee (Emp#, Name...)

Dependent (Name, Age,...)

Primary key of employee: [Emp#]

Candidate key of dependent : [Name]

The primary key of dependent is defined as: [EMP#,NAME]

3.3.9.3. Primary Keys of Entities in ISA Hierarchies

In an ISA hierarchy "A ISA B" anything that is true for the entity B is also true for the entity A, and hence the primary key of A is also the primary key of B. For entities participating in the ISA hierarchies the primary key of the

highest level entity (that is the highest generic entity) is chosen as their primary key.

Example:

ISA hierarchies: Sales_Manager ISA Manager

Manager ISA Employee

Here Employee is the highest-generic entity.

Primary key of employee : [EMP#]

So the primary keys of both Manager and Sales-Manager will also be same as [EMP#].

3.4. Summary :

This section has presented a step-by-step methodology for designing user views of a database application. The methodology is based on an extended entity relationship model. It has shown how the EER model corresponding to the user view of the database can be constructed interactively. Motivation for this work is an increasing need for effective and easy to use design support tools in any system development environment and in particular in database applications which becomes more and more large and semantically complex and where the development cost should be limited. The methodology is provided with several rules and heuristics to facilitate the design of an interactive tool for designing user views of a database, and it is implanted in our expert system VMITS.