COMPUTER-AIDED RELATIONAL DATABASE DESIGN SYSTEM

Jessie Ching

Department of Computer Science

A thesis submitted for the degree of

Master of Philosophy

at The Chinese University of Hong Kong

May 1989

ABSTRACT

The past two decades have witnessed the consolidation of the basic concepts and theory of database design. The recent developments emphasize design methodologies and tools which have practical functions. Since logical database design is a complex and labor intensive process, an appropriate design aids can help making the complexity more manageable and the process less laborious. The work reported in this thesis addresses two aspects of database design; namely, mapping of conceptual schema to a relational schema and generating of optimal relational schema.

Entity-relationship (E-R) model was introduced in 1976 [Chen,1976] and since then has attracted considerable attention. Some work in the literatures has been devoted to translating from an E-R schema to a relational schema, and yet most of them have weaknesses. An algorithm which deals with "inter-entity" dependencies and transforms an E-R schema into relations is described. In addition, in order to carry on to eliminate "intra-entity" dependencies thoroughly, another algorithm which performs normalization is discussed.

This thesis presents an integrated technique which adopts the concepts of eliminating "inter-entity" and "intra-entity" dependencies and transforms an E-R schema to not only a set of relations, but a set of *optimal relations* which contains *minimum redundancies* and is *free of update anomalies*. This is the core of the CARDBD - a Computer-Aided Relational Database Design System.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my research supervisor, Dr. C. Yau for continuous friendly relationship, active involvement in my research, technical guidance and support of various kinds which I will never forget.

TABLE OF CONTENTS

PROGRESS OF THE RESEARCH

Two significant reports were written which indicate:

- how the ideas were gathered
- how the direction of research changes

The title of the mentioned reports and the date written are as follows:

(1) Computer-aided Relational Database Design (May,1988)

(2) Translating an E-R Schema into Optimal Relations (December,1988)

1 INTRODUCTION

The past decade has witnessed a widespread interest among database practitioners and researchers in Codd's relational model [Codd,1970]. One reason for this model's popularity is that, while it provides conceptual simplicity, it rests upon a rich theoretical foundation - the theory of sets and relations - and therefore makes database problems amenable to formal treatment.

At the heart of the relational model is the concept of data dependencies. Codd recognized very early the presence of certain anomalies in complex databases. To characterize the absence of these anomalies he introduced the concepts of functional dependency and normal forms. As stated, normalization theory begins with the observation that certain collections of relations have better properties in an updating environment than some other collections of relations containing the same data. The theory is based on a series of normal forms which provide successive improvements in the updating properties of a database.

Intuitively, the objective of relational database design is to decide how many relations should be in the database and what their underlying attributes should be. Due to the current difficulty of creating a schema with manual design procedures, when designing this schema, the necessity of keeping the conceptual schema stable has to be considered also. It is because any change to object which appears in more than one view requires that the mappings from the conceptual schema to each of these views be revised, usually manually at present.

Real world experience with database design indicates that no set of manual will suffice for this process. The size and complexity of this problem require computer support. On the other hand, a fully automatic design system in which a human is not actively involved may generate schemas which are not acceptable to users. Therefore, a computer assistance can be applied to the logical design process, and, when used with appropriate human interfaces, it can help the designer produce a logical design more quickly and with expectation of better quality.

CARDBD (Computer-Aided Relational Database Design) System includes detail guidelines and rules for the designer to follow. It treats database design as a step-by-step process and utilize such aids as forms for data collection, quantitative criteria for data grouping and key determination.

2. DATA MODELING

Data modeling involves shaping the facts collected during the data investigation process into data model concepts, deriving a conceptual model as a result. The conceptual model is required to support not only present applications and requirements, but also unknown future ones and its design should not be constrained by specific applications. It must be a global model reflecting the company-wide view of data, rather than reflecting a collection of local departmental views, some of which may have conflict. It should also reflect, to a certain extent, future company plans as well as present activities, to avoid the need for database restructuring as far as possible. The model derived will be a generalized model, independent of any specific DBMS.

The conceptual model must be as comprehensive as possible, and hence the DBA must ensure that his view of reality is sufficiently broad to meet this objective. Although his view may be the best model as far as reflecting the reality is concerned, it may not be the best model from the information system viewpoint. Therefore, the DBA has to abstract from reality, reducing his complex view to a view which will form the basis of the conceptual model. The modeled environment is simple and coordinated; everything involved is relevant, and most important of all, short-term and long-term goals are blended equally.

2.1 The Components of Data Modeling

To develop a database that satisfies today's as well as tomorrow's information needs, a conceptual model must be designed. When designing the conceptual model, efforts should be concentrated on structuring the data and relationships between the data elements of the organization. To achieve this goal, some basic components of the real world, which are also the data modeling components have to be considered. They are relationship, entity and attribute.

<u>2.1.1 Relationship</u>

There are a number of different *notations* currently in use for representing the relationships between modeled objects. In figure 2.1, all the examples represent a simple one-to-many relationship between object A and B.

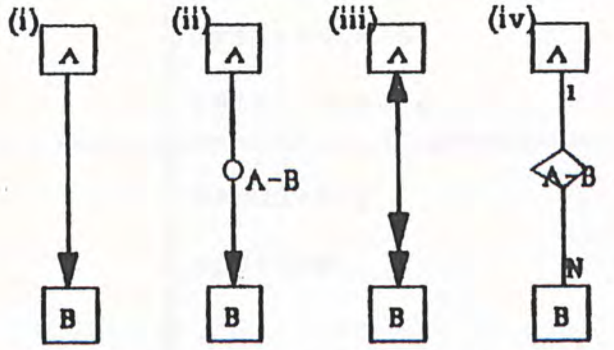


Figure 2.1 Different One-to-many Relationship Notations

Notations (i) and (ii) were those used in the early days of databases when the representation of one-to-many relationships was considered to be very important; especially for notation (ii), which was designed by Bachman for the diagrammatic representation of CODASYL database descriptions. but it was soon recognized that the real world has more to it than just one-to-many relationships between objects. It is necessary to bring the real world view and the restricted view together, and this led to the emergence of other database design methodologies, notations and techniques such as (iii) [Robinson,1981] and (iv). Notation (iv) was designed by P.P. Chen [Chen,1976] and has gained in popularity over the years [Chen,1980].

The *type* of a relationship consists of two main characteristics: the *degree of* the relationship and its *existence*.

Degree of relationship	Existence of relationship
one-to-one one-to-many many-to-many	mandatory optional

Table 2.1 Degree and Existence in relationship Types

2.1.2 Entity and Attribute

The next two data components to be considered are: the *entity* and the *attribute*. An entity may be defined as a person or thing which is capable of independent existence in the real world and which possesses characteristics in which we are interested. For example, a person, a television set, a car, a student, all of them are entities. All entities are distinguishable from each other in reality. As a matter of fact, an entity can be indivisible or composed out of other entity. For example, a watch is an entity; the hour hand in the watch can also be an entity, and so as the minute hand. In this case, the hour hand and the minute hand are considered as indivisible entities, whereas the watch is an entity composed of some distinct entities.

An attribute is a quality, feature or characteristics which a class of entities possesses and by a set of which all elements of this class are meaningfully described. An example for an attribute is as follows: a watch is an entity; but consider a mechanical watch and a digital watch, the manufacturing style may be the attribute of the entity in this case.

The distinction between an entity and an attribute is obvious. However, the distinction between them sometimes depends on context. Consider the case: color. To a car salesman, color may simply be an attribute of a car, while to a paint manufacturer, color is an entity in its own right which possesses characteristics such as chemical composition.

2.2 Two of the Most Common Approaches

The following sections present two of the most common modeling approaches. The first approach, entity-relationship modeling method, is representative of the class of methods that take entities and relationships as input. The second approach, data normalization and structuring, is representative of class of methods that take a list of fields and the associations among those fields as input.

2.2.1 Entity-relationship modeling approach

The technique to be described is called the *entity-relationship (E-R) approach* [Chen,1976]. The database designer first identifies the entities and relationships (discussed in section 2.1) which are of interest to the enterprise using the *entity-relationship diagrammatic technique*. It is a pure representation of the real world and should be indenpendent of storage and efficiency considerations. The database designer *first* designs the enterprise conceptual schema and *then* translates it to a user schema for his database system. The advantages of this two-phase approach are:

(i) The division of the design process into 2 phases make the process simpler and better organized.

(ii) Since this is not a DBMS specific model, changing from one database system to another is easy.

2.2.2 Data normalization and structuring

The data modeling approach mentioned in the previous section was a top-down approach. An alternative which has been widely used in the past is the normalization methodology, which can be considered as a bottom-up approach. It was originally devised as a database design tool for relational databases but has been found to be a useful tool for conceptual modeling and is an accepted technique. The advantages are:

1. It is a formal technique with each stage of the normalization process eliminating a particular type of undesirable dependency.
2. It highlights constraints and dependencies in the data, and therefore it is an aid to a better understanding of the nature of the data.
3. 3NF and higher normal forms produce well-designed databases which provide a high degree of data independence.

2.2.3 Comments on the two approaches

For the normalization approach, there is no distinction between entity set and relationship set, and no type and existence characteristics of relationships are shown.

A further difference between the normalization approach and the entity-relationship approach relates to the way in which relationships are represented. In the normalization approach entities are linked by means of attributes, while in the entity-relationship approach entities are linked by relationship sets.

In addition, the normalization approach is based on the notion that one-to-one relationships between objects are represented by means of entity sets (relations) while one-to-many relationships are represented by attributes in different relations which are drawn from a common domain. The main disadvantage of this is the existence of null values for some attributes.

The best approach to data modeling is probably to use entity-relationship aproach initially, and then apply the normalization rules to the attributes in the resulting entity and relationship sets, to ensure that the design is good. A skilled modeler will find that his model is in 3NF or a higher normal form before he applies any normalization rules. Normalization will however serve to verify that his model is a good one.

3 LOGICAL DATABASE DESIGN IN DATABASE DEVELOPMENT

Different database management systems will have their own specific constructs, rules and limitations which apply to the data model. The process of database design is that of mapping the generalized data model into the DBMS-specific data model. In order to transform a conceptual model to a relational one, some general concepts of relational data model is to be introduced.

3.1 General Concepts of Relational Data Model

The relational model was introduced by E.F. Codd of the IBM in 1970 [Codd,1970]. It is based on the mathematical concepts of relation and set.

The essential terminologies are defined as follows:

A *relation* is a mathematical term for a two-dimensional table which is characterized by rows and columns. Each column is called a *domain* containing all the values of an attribute . If the relation has *n* columns, then each row is referred to as an *n-tuple*. Also, a relation that has *n* columns or *n* attributes is said to be of *degree n*. In addition, a relation has several properties,

a) The entries in the table are single-valued (atomic).

b) The entries in any column are of the same kind.

c) Each column has a unique name and the order of the column is immaterial.

d) No two rows in the table are identical and the order of rows is insignificant.

3.2 Relational Database Design and Normalization

The practising of normalization can help the analysts and database administrators to determine relations that are consistent with a conceptual model such that they do not have undesirable dependencies that would give troubles to updating, i.e. when creating, modifying or deleting data item values. These troubles are sometimes called *updating anomalies*.

Originally Codd defined three levels of normalization, first, second and third normal form (1NF, 2NF, 3NF) [Codd,1971]. Later an extended 3NF, Boyce/Codd Normal Form (BCNF) was introduced to eliminate certain rarer undesirable properties. Following the introduction of

BCNF, Fagin defined a fourth normal form (4NF) which overcomes some other storage anomalies. A clear and brief guide to the five normal forms is written by Kent may be found in [Kent,83].

3.2.1 Some terminologies on Normalization

The second and third normal forms require knowledge about *functional dependencies* [Codd,71], whereas the fourth and fifth normal forms are based on *multivalued* [Fagin,77] and *join dependencies*, respectively [Date,83].

In addition to the above dependencies, candidate key is also important in the practising of normalization. A *candidate key* is such that it could be chosen as a primary key of a relation. An attribute or set of attributes is referred to as a candidate key if it can uniquely identify record in a relation and, in the case of a set of attributes, no subset of that sets is itself a candidate key.

3.2.2 The importance of Normalization

It is important to note that the application of normalization to a conceptual schema (if properly done) does not result in any loss of information. The contribution of the step-by-step process lies in the elimination of certain undesirables properties in the representation of conceptual schemas.

Higher normal forms represent stronger rules from the 1NF which is proved to be much too weak. In fact, even if a relation is in the 1NF it still exhibits the insertion, deletion, and update anomalies. The hierarchy of normal forms is such that if a relation satisfies the requirements of some normal form, then it also satisfies the requirements of all lower-order normal form. The converse is not true, which means that the hierarchy of normal forms is proper, that is, every normal form is a stronger form than the preceding one in the hierarchy.

The reason one would use the normalization procedure is to ensure that the conceptual model of the database will work. This means, not that an unnormalized structure will not work, but only that it may cause some problems when application programmers attempt to modify the database. The database administrator must decide, after locating violations from normalization, whether the modifications will affect how the database will function.

4 CARDBD SYSTEM OVERVIEW

Logical database design is a complex and labor intensive process, as discussed in chapter 2 and chapter 3. Therefore, appropriate design aids can help make the complexity more manageable and the process less laborious. In this chapter, an overview of the Computer-aided Relational Database Design System (CARDBD) is presented.

The CARDBD System has been implemented on VAX/VMS environment. It is written in Pascal and consists of three main modules. The first module is E-R MODELING, which mainly makes it easier to gather all the appropriate data into a data dictionary and define them using notations of entity-relationship model. The second module is TRANSFORMATION, generating a set of relations based on the data gathered in the previous module. The last module of this system is NORMALIZATION, takes a set of data dependencies input by the designer to ensure the outcome is a set of relations in optimal normal form [Nijssen,1987]. Finally, a relational schema is created on the Rdb/VMS.

Of all the three modules of CARDBD, TRANSFORMATION and NORMALIZATION are considered to be the soul of the whole system. These modules reduce the human effort in database design and point the way toward further automation.

4.1 The Design Goals

Database design has been more than an art than a science, with no standardized set of rules or procedures. Therefore, experience and intuitive feeling have been the designer's main resource.

Using automated techniques, design information can be made available earlier in the design cycle and in a more complete manner than is usually obtained manually. CARDBD intends to:

- improve the design quality
- shorten the design cycle

4.2 Techniques Used in the System

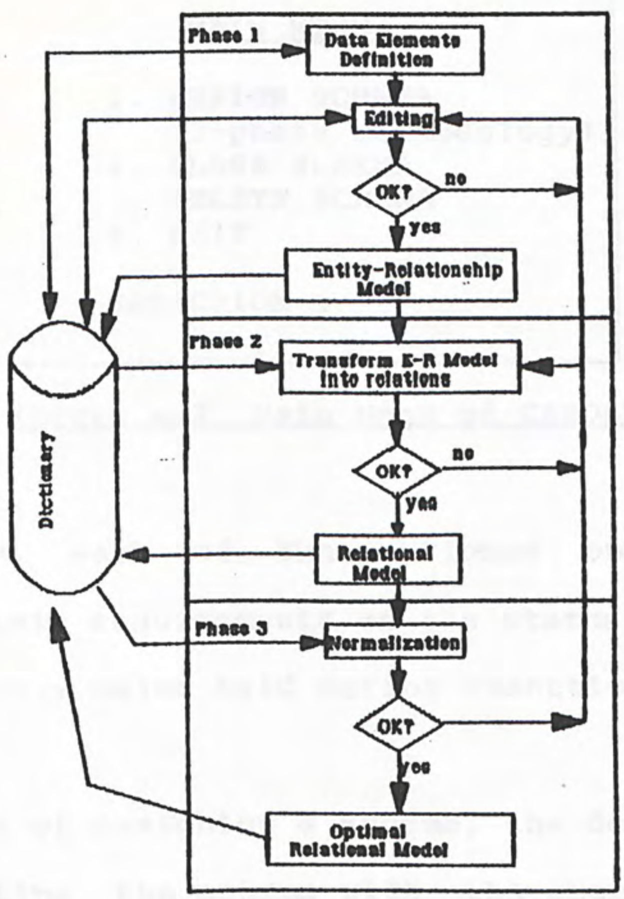
As mentioned in section 2.2, entity-relationship modeling approach can be considered as a two-phase approach. In the first phase E-R diagrammatic technique is used to identify entities and relationships. Converting E-R diagram to relations is regarded as the second phase.

The techniques used in CARDBD is the integration of the two modeling approaches discussed in chapter 2, i.e. to use E-R approach to do a top-down design initially, then apply the bottom-up normalization rules to transform the schema into an optimal form.

Therefore, when using CARDBD, the relational database design process can be divided into three phases, which are E-R MODELING, TRANSFORMATION, AND NORMALIZATION. The three phases are regarded as iterative procedures, the result of each phase will be a model. In addition, the overall process is also iterative. Results may be obtained in phase 2 or in the phase 3 that will require changes or re-interpretations in earlier phase to obtain required capabilities.

4.3 General Structure of the System

While most of the steps of database design process lend themselves to automation, considerable human intervention is required. The iterative procedures require frequent dialogues between the designer and the end users, and well-designed automation can provide insights and suggestions of specific problems to be discussed and resolved in these dialogues. Figure 4.1 represents the iterative procedures along with the major human decision points.



<u>Figure 4.1 The design of CARDBD system</u>

CARDBD is designed as a menu-driven package which starts with an opening menu (figure 4.2) that enables the designer to open, close, or delete a schema.

```
RELATIONAL DATABASE DESIGN SYSTEM

                 MAIN MENU

              1. DESIGN SCHEMA
                 (3-phase methodology)
              2. CLOSE SCHEMA
              3. DELETE SCHEMA
              4. EXIT

              SELECTION : _
```

Figure 4.2 Main Menu of CARDBD

In addition, each of the mentioned operation to a schema has certain requirements on the status of the system which is currently being held during execution.

In the case of designing a schema, the designer has the choice of creating the schema with the specified name or open another one if it does not exist. A schema can be opened only if there is no other schema opened currently.

In the case of closing a schema, the system will close the schema that is opened currently. If there is no opened schema, warning messages will occur.

In the case of deleting a specific schema, the schema must be exist. Trying to delete a non-existing schema will cause warning messages.

After the designer has chosen to work on a particular schema, another three options are available, which represent the three phases of the whole design methodology (figure 4.3).

```
              RELATIONAL DATABASE DESIGN SYSTEM

                     DESIGN SCHEMA MENU

    SCHEMA : ____________________

                   1. E-R MODELING
                   2. TRANSFORMATION
                   3. NORMALIZATION
                   4. EXIT TO MAIN MENU

                   SELECTION: _
```

Figure 4.3 Design Schema Menu

The following three chapters will discuss the components of CARDBD. Chapter 5 covers the first module of the system. Chapter 6 and 7 will discuss the second and the third module in order, and also how they both contribute to obtain optimal relations.

4.4 Data Dictionary in CARDBD

There are basically six indexed files in CARDBD to store the data needed for a single schema, and also five text files for reporting. Their usages are as the following table:

	E-R MODELING	TRANS.	NORMALI.
Entity File	I/U	I	
Relationship File	I/U	I	
Table File		U	I
Dependency File			I/U
Data Type File			I/U
Final File			U
Report 1	O		
Report 2	O		
Report 3		O	
Report 4			O
RDO			O
Legend :	I = Input	U = Update	O = Output

The detail retrieval and updating of the above files will be discussed clearly in the next three chapters.

5 E-R MODELING in CARDBD

E-R MODELING is the entry point of the whole design system. It provides database administrator with facilities to develop his conceptual schema in terms of an E-R model. Although an E-R model should be defined before entering the design process, it is also true that they frequently are changed during the course of the design process as the designers realize that what they initially specified (or thought they wanted) no longer corresponds exactly with what they now realize they need.

The objective of this module is to provide the designer with a tool for identifying and defining all the .relevant data items and the relationships among them for an enterprise. As a result, an Entity-relationship model is created. It is developed in order to facilitate database design by allowing the specification of an enterprise scheme. Such a scheme represents the overall logical structure of the database.

The E-R MODELING module is composed of two major functions which are namely the Entity Maintenance Function, and the Relationship Maintenance Function. Figure 5.1 shows the composition of E-R MODELING.

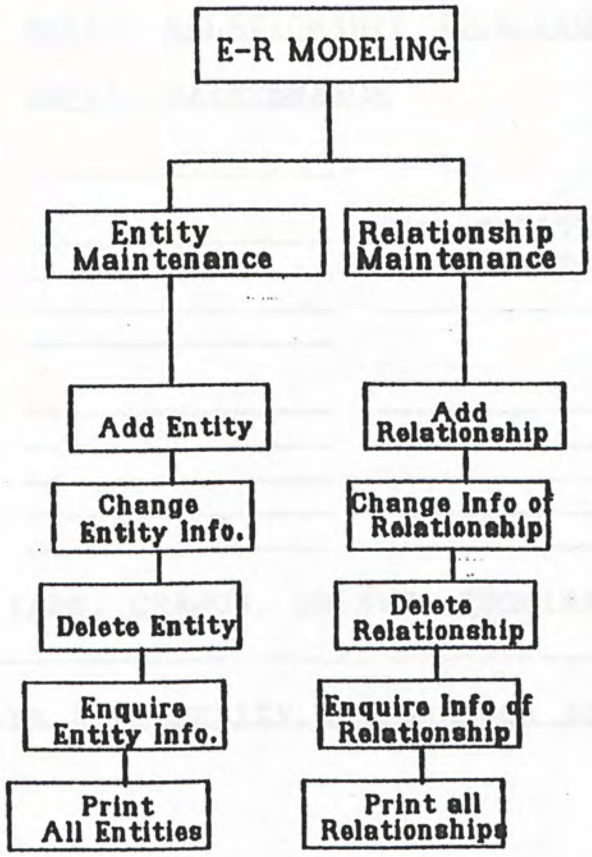


Figure 5.1 The Composition of E-R MODELING

In order to design a newly created schema or to modify an existing schema, the entity and relationship information must be taken in good care.

5.1 Operations in Entity Information Maintenance

Entity information of an enterprise are manipulated by this maintenance function, which has four modes - add, change, delete, enquire.(Figure 5.2)

```
                ENTITY-RELATIONSHIP MODELING

                ENTITY MAINTENANCE
 ENTITY NAME  : ______________

 PRIMARY KEY  : ______________  WEAK ENTITY? (Y/N) _
                ______________  STRONG ENTITY NAME :
                ______________  ______________
                ______________

 ATTRIBUTES   : ______________  ______________
                ______________  ______________
                ______________  ______________
                ______________  ______________
                ______________  ______________

 ENTER ACTION (ADD, CHANGE, DELETE, ENQUIRE, EXIT) _
```

Figure 5.2 Entity Maintenance Screen

5.1.1 Adding entities

Entity names are unique, and no duplication of names are allowed. Whenever the designer adds an entity, the system will check the existence of the specified entity name. If there is no such entity exist, then the designer can proceed to enter the primary key and attributes of the entity. For each entity, a composite key composed of up to four fields is allowed, and must not be repeated as attributes. Weak entities are also catered. The designer simply answers whether or not the specified entity is a weak entity, and enters the strong entity as well.

The following are for the designer's considerations:

a) Strong entities should be defined before weak entities, since the existence of a weak entity is depended on the strong entity, and it needs to validate the existence of the strong entities to ensure integrity.

b) Prepare the following entity information:
   - Identify a unique name for each entity sets,
   - Identify all attributes which characterize each entity,
   - Of all the attributes identified, choose a primary key for it.

5.1.2 Changing entities

If modification has to be made to an updated entity, just enter the entity name, and the rest of the information will be shown. Then retype the proper fields.

5.1.3 Deleting entities

If an entity has to be deleted, two cases are considered:

a) If there are other entities depended on the 'entity to be deleted', then all the depending entities have to be deleted also.

b) All relationships, which contain the 'entity to be deleted' as the related entity, have to be deleted.

5.1.4 Enquiring entities

After the designer has chosen to run this mode, two options are available - enquiring entities individually on the screen, or print all the entities on hardcopy are available.

To view entity information individually on the screen, the designer should enter the entity name which he wants to enquire; if it exists, the rest of the information will be shown on the screen.

If printing all entities on hardcopy is chosen, a report will be generated and is named with the schema name plus an extension '.rpl'.

5.2 Operations in Relationship Information Maintenance

Same as the entity maintenance, the relationship maintenance function has four modes - add, change, delete, enquire. The data entry screen layout is as figure 5.3.

```
                ENTITY-RELATIONSHIP MODELING

                 RELATIONSHIP MAINTENANCE

 RELATIONSHIP NAME : ________________

    RELATED ENTITY          MAPPING
    ________________           _
    ________________           _
    ________________           _
    ________________           _

 ATTRIBUTES  :  ________________  ________________
                ________________  ________________
                ________________  ________________
                ________________  ________________
                ________________  ________________

 ENTER ACTION (ADD, CHANGE, DELETE, ENQUIRE, EXIT) _
```

Figure 5.3 Relationship Maintenance Screen

5.2.1 Adding relationships

Relationship is the linkage between two or more entities. To add a relationship, the designer must follow the rules:

a) All entities should be defined before relationships, since a relationship identifies associated entities, and it needs to validate the existence of the related entities to ensure integrity.

b) Prepare the following relationship information:
- Assign a unique name for each relationship set,
- Identify all the related entities, the degree of relationship and their membership classes. These will be discussed in section 5.3 and section 5.4.
- Determine attributes for a relationship.

5.2.2 Changing relationships

If modification has to be made on an updated relationship, just enter the relationship name, and the rest of the information will be shown. Then retype proper fields.

5.2.3 Deleting relationships

If deletion is necessary, enter the relationship name to be deleted, then the designer is prompted to confirm the deletion.

5.2.4 Enquiring relationships

After the designer has chosen to run this mode, two options are available - enquiring relationships individually on the screen, or print all the relationships on hardcopy are available.

To view relationship information individually on the screen, the designer should enter the relationship name which he wants to enquire; if it exists, the rest of the information will be shown on the screen.

If printing all relationships on hardcopy is chosen, a report will be generated and is named with the schema name plus an extension '.rp2'.

5.3 Importance of Mapping and Membership Class

Mapping has been regarded as the traditional vehicle for indicating the nature of the relationships between pairs of related data elements (or entities). A brief review of mappings is given as follows,

One-to-one mapping

'A lecturer teaches, at most, one course.'

'A course is taught by, at most, one lecturer'

One-to-many mapping

'A lecturer may teach many courses.'

'A course is taught by, at most, one lecturer.'

Many-to-one mapping

'A lecturer teaches, at most, one course.'

'A course is taught by many lecturers.'

Many-to-many mapping

'A lecturer may teach many courses.'

'A course may be taught by many lecturers.'

The 'at most' in the above mapping examples has the meaning that the quantity may be as specified, or zero. Therefore, there are two different ways in which an entity type can participate in a relationship. Some of the rules of an enterprise insist that every occurrence of an entity participates in the relationship, other rules of an enterprise allow occurrences of an entity to exist independently. The terms *mandatory* and *optional* will be used to distinguish between these situations.

For a relationship between two entity types, there are four possible combinations of membership classes, as illustrated in Fig.5.2. Knowledge of the membership classes of entities is important, as it may influence the design of data models and schemas.

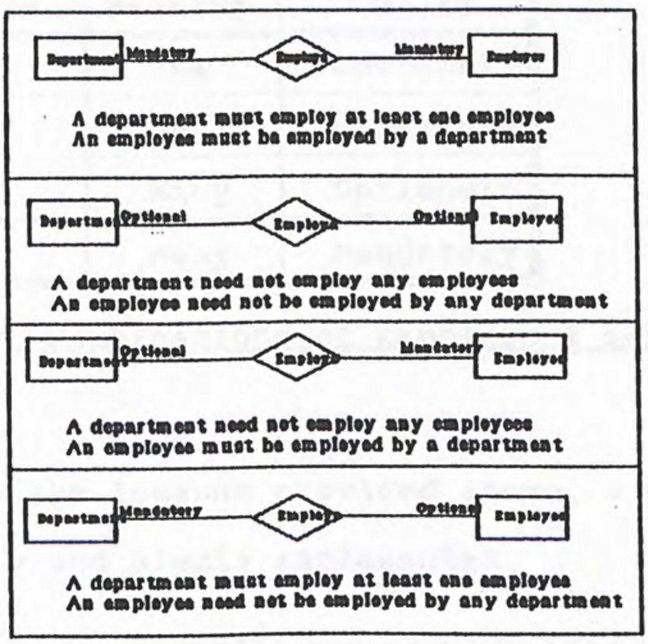


Figure 5.2 Possible combinations of membership class

5.4 The Representation of Mappings and Membership Classes

As discussed in last section that both the mappings and membership classes of all the related entities in a relationship set are two of the determining factors that will affect the schema; therefore, a set of symbols (illustrated in Figure 5.3) are used to represent different combination of mappings and membership classes.

Mapping Symbols	Mapping Meaning	Membership Class
N	One	Optional
U	One	Mandatory
M	Many	Optional
W	Many	Mandatory

Figure 5.3 Combinations of mappings & membership class

Based on the legends provided above, a relationship set can be easily and simply represented.

Example 1:

A department must employ at least one employee, and an employee must be employed by a department. Obviously, there is a one-to-many relationship between entities

DEPARTMENT and EMPLOYEE. Moreover, both of them are mandatory in this case. Therefore, a new representation of this particular relationship is U:W

Example 2:

A department need not employ any employees, and an employee must be employed by a department. Same as the previous example, it is a one-to-many relationship, but DEPARTMENT has different membership class, which is optional. Therefore, the representation will be N:M

Designer should not be confused the mapping symbols (such as N:M in example 2) with the conventional mappings. The point is that the mapping N:M will be interpreted as a many-to-many relationship in the conventional one.

5.5 Result of E-R MODELING

Upon entering the information of the entities and relationships, a conceptual schema using e-r modeling approach is created, in which entity and relationship information is store in two files named as the schema name plus an extension '.e' and '.r' respectively.

6 TRANSFORMATION IN CARDBD

After phase 1 (E-R MODELING) of the design system, in which all the preparation works is done, a database designer may then go to phase 2 of CARDBD, which is performed by the TRANSFORMATION module.

6.1 The Objective of TRANSFORMATION

TRANSFORMATION is built for the designer, using a set of transformation rules (discussed in the following sections), to transform a schema represented in E-R model to a set of relations. For each entity and for each relationship in the database, there is a unique relation which is assigned the name of the corresponding entity or relationship. Each relation has primary key, and a number of attributes.

6.2 What Information is Needed for TRANSFORMATION

The reader might want to have an idea of how the input of this module looks like. The input of this module are an

Entity Master File and a Relationship Master File.

Entity Master File:

Field Name	Description
e_name	Entity Name
e_pk	Primary key
e_at	Attribute(s)
e_dp	Supporting Entity

Relationship Master File:

Field Name	Description
r_name	Relationship Name
r_entity & mapping	Related Entity & its mapping in the relationship (max. 4 pairs)
r_at	Attribute(s)

6.3 Brief Explanations of the Transformation Algorithms

The first step of the algorithm is to get an entity one by one from the Entity Master File, then follow the rules described in section 6.5 and section 6.6. The second step is to get a relationship one by one from the Relationship Master File, and apply the rules given in section 6.7 to each relationship.

6.4 The Result Produced by This Module

Upon completion of this module, the result will be stored in a Table Master File as follows, with each record in the file corresponds to one relation.

Table Master File:

Field Name	Description
tab_name	Table Name
tab_pk	Primary key
tab_at	Attribute(s)

6.5 Rules for Transforming Strong Entities

Strong entity is a record in the entity master file whose field called "e-dp" contains nothing. To transform strong entity, we adopt the follow rule:

Let *S* be a strong entity in an Entity Master File *E* with primary key *SK* and attributes $SA_1, SA_2, \ldots, SA_n$ ∍ $0 \leq n \leq 10$. We represent this entity by a relation called *S* in Table Master File *T* with *(n+1)* distinct columns, each of which corresponds to the primary key *SK* and one of the attributes of *SA*.

6.6 Rules for Transforming Weak Entities

Weak entity is a record in the entity master file whose field called "e-dp" contains another entity name which is also exist in the entity master file.

Let W be a weak entity in the Entity Master File E with primary key WK and attributes $WA_1, WA_2, \ldots, WA_m \ni 0 \leq m \leq 10$. Let S be the strong entity on which W is depended. Let the primary of S be SK. We represent this entity W by a relation called W in the Table Master File with $WK \in T = (WK \in E \cup SK \in E)$; and $WA \in T = WA \in E$.

For example, let EMPLOYEE be a strong entity with EMPLOYEE NUMBER as the primary key, and EMPLOYEE NAME, ADDRESS, PHONE #, BIRTHDATE, SALARY, TITLE as the attributes. Let CHILD which has CHILD NAME as primary key, AGE and SEX as its attributes, be a weak entity which is depended on EMPLOYEE. By using this rules, CHILD can be transformed as a relation named CHILD, its primary key (composite key) is EMPLOYEE NUMBER & CHILD NAME, CHILD'S AGE and SEX are the attributes of the weak relation CHILD.

6.7 Rules for Transforming Relationships

First of all, the process of the transformation of entities is supposed to be finished before entering to transform relationships, since during the process of transforming relationship, there is a chance that the entity relations in the Table Master File have to be retrieved and re-processed again.

It is because a relationship must have at least two and at most four related entities; in addition, it also has different combinations of mappings and membership classes, therefore each record *R* in the Relationship Master File will be classified as one of the following four types: (for convenient, a sample relationship with two or more related entities are used for each case as examples to illustrated the transformation process in each case)

Case 1 : all mappings are 'one', and exist at least one 'mandatory' membership class.

e.g. Two entities has already been transformed into relations:

DEPT(<u>DEPT#</u>,BUDGET,LOCATION)

EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE)

There is relationship between DEPT and EMP named

DEPT-MGR. An employee may be the manager of at most one department, and a department must have exactly one employee as the manager. The mappings can be defined as DEPT:EMP (U:N)

Case 2 : all mappings are 'one', and all membership classes are 'optional'.

e.g. Two entities has already been transformed into relations:

EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE)

CAR (<u>CAR#</u>,YEAR,MAKE,COLOR)

There is relationship between EMP and CAR named EMP-CAR. An employee may be supplied at most one car from the company, and a company car can be assigned to at most one employee. The mappings can be defined as EMP:CAR (N:N)

Case 3 : there exist only one 'many' mapping, and not exist 'optional'.

e.g. Two entities has already been transformed into relations:

DEPT(<u>DEPT#</u>,BUDGET,LOCATION)

EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE)

There is relationship between DEPT and EMP named DEPT-EMP. An employee must work for one department, and a department must have at least one employee. The mappings can be defined as DEPT:EMP (U:W)

Case 4 : there are more than one 'many' mapping regardless of the type of membership classes or exists only one 'many' which is also 'optional'.

e.g. Three entities has already been transformed into relations:

DEPT (<u>DEPT#</u>,BUDGET,LOCATION)

SUPP (<u>SUPP#</u>,NAME,ADDR,PHONE)

PART (<u>PART#</u>,DESC,COST,WGT)

There is relationship between DEPT, SUPP, and PART named DEPT-SUPP-PART. For each part comsumed by a department, the particular part is supplied by one supplier only. The mappings can be defined as DEPT:SUPP:PART (M:N:M)

Based on each of the above catergories, different algorithms is used to handle different cases. In general, let a relationship $R \in$ Relationship Master File, and

$$\left\{e\_name, \left\{ \bigcup_{i=1}^{n} r\_entity, mapping, \ni 2 \le n \le 4 \right\}, r\_at \right\} \in R$$

A relationship falls into each of the following categories if and only if, Case 1:

$(\forall\ mapping \in R\ , \ni\ mapping = N \vee U)$

and

$(\exists\ mapping \in R\ , \ni\ mapping = U)$

step 1: prompt the designer to choose one of the $r\_entity \in R \ni mapping = U$

step 2: find tab_name $\ni$ $tab\_name =$ chosen r_entity

step 3: tab_at = $tab\_at \cup (\cup\ o\_pk$ of non-chosen $r\_entity)$

step 4: rewrite the record in Table Master File found in the step 2

Refer to the example of case 1, since there is only one membership class 'mandatory', therefore the entity will automatically be chosen entity. The resulting set of relations will become:

DEPT(<u>DEPT#</u>,BUDGET,LOCATION,*EMP#*)

EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE)

Note the the italic EMP# represents the modifications to the relation DEPT made in the transformation of relationships

Case 2:

($\forall$ *mapping* $\in R$, $\exists$ *mapping* = N)

step 1: create a record in Table Master File such that

tab_name = r_name

step 2: prompt the designer to choose one of the

r_entity $\in R$

step 3: find e_pk of E $\in$ Entity Master File, $\exists$ *E* = chosen *r_entity*

step 4: tab_pk = e_pk

step 5: tab_at = $\cup$ *e_pk* of non-chosen *r_entity*

Refer to the example of case 2, assume that the designer chooses CAR to be the control key. The resulting set of relations will become:

EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE)
CAR (<u>CAR#</u>,YEAR,MAKE,COLOR)
EMP-CAR (<u>CAR#</u>,EMP#)

The relation EMP-CAR is a newly added relation during the process of transforming relationships.

Case 3:

$(\forall\ mapping \in R, \ni |mapping = W| = 1)$

or

$(\neg\exists\ mapping \in R, \ni\ mapping = M \vee N)$

step 1: find tab_name $\ni$ *tab_name* = *r_entity* with *mapping* = W

step 2: tab_at = *tab_at* $\cup(\cup$ *a_pk* of *r_entity* with *mapping* = U)

step 3: rewrite the record in Table Master File found in the step 1

Refer to the example of case 3, since there is only one mapping 'many', therefore EMP will be retrieved again for further modifications. The resulting set of relations will become:

DEPT(<u>DEPT#</u>,BUDGET,LOCATION)
EMP (<u>EMP#</u>,ID,SEX,BDATE,NAME,ADDR,TEL,SAL,TLE,*DEPT#*)

Note the the italic DEPT# represents the modifications to the relation EMP made in the transformation of relationships.

Case 4:

$$(\forall\ mapping \in R, \exists\ mapping = M \vee W) \wedge (|\ mapping = M \vee W| > 1)$$

or

$$(\exists\ mapping = M) \wedge (|mapping = M| = 1)$$

step 1: create a record in Table Master File such that

tab_name = r_name

step 2: tab_pk = $\cup$ *a_pk* of *r_entity* with *mapping* = W $\vee$ M

step 3: tab_at = $\cup$ *a_pk* of *r_entity* with *mapping* = U $\vee$ N

Refer to the example of case 4, since DEPT and PART both have mappings 'many', therefore, both of their primary keys will be served as control key in a relation. The resulting set of relations will become:

DEPT (DEPT#,BUDGET,LOCATION)

SUPP (SUPP#,NAME,ADDR,PHONE)

PART (PART#,DESC,COST,WGT)

DEPT-SUPP-PART (DEPT#,PART#,SUPP#)

The relation EMP-CAR is a newly added relation during the process of transformating relationships.

6.8 Comments on the Results in Table Master File

The results of this phase is a set of relations with each relation corresponds to each record in the Table Master File named as the schema name plus an extension '.tab'. The set of relations is at least in the First Normal Form (1NF), and will act as part of the input to the next phase, NORMALIZATION, which normalizes them into a set of higher normal form relations such that some undesirable properties can be minimized.

7 NORMALIZATION IN CARDBD

This chapter discribes the detail set up of the last module of CARDBD, NORMALIZATION. The algorithms derived for this module is based on the concepts of functional dependencies (FDs) [Codd,1970] and multivalued dependencies (MVDs) [Fagin,1977]. A set of relations (not necessarily in the lowest normal form) will go through a process to ensure the outcome is optimized [Nijssen,1987] which means:

- there exists no repeated attributes and/or groups in all relations
- all redundancies and update anomalies are eliminated
- the number of relations is minimized.

7.1 The Overview of NORMALIZATION

This module is composed of several submodules, with each of them governs the main steps in the last phase of the design. The designer is supposed to complete the E-R MODELING and TRANSFORMATION before entering NORMALIZATION. Upon completion, a Table Master File is ready to serve as one of the most important data for this module. The system, in this stage, will display records in Table Master one by one.

The designer should examine each relation thoroughly and determine if there exists data dependency. GET DATA DEPENDENCY is designed for the designer to store all the existing data dependencies in the Dependency File for each relation in Table Master File. FILTER DEPENDENCIES will automatically be executed to eliminate redundancy exists in the Dependency File. Based on the revised Dependency File, the system will execute GENERATE OPTIMAL RELATIONS and write the finalized optimal relations to the Final Master File. As a matter of fact, up to this point, the system has already given us what we have asked for in the first place - to get an optimal relational schema.

Further works have been done to put the resulting optimal schema onto VAX Rdb/VMS, which is a relational database management system. GET DATA TYPE gets the appropriate information needed to generate a command file that has an RDO file type. Such a command file can contain all the definition statements required to create the database.

Figure 7.1 illustrates the program flow of NORMALIZATION and each of the submodules will be discussed in detail in the following sections.

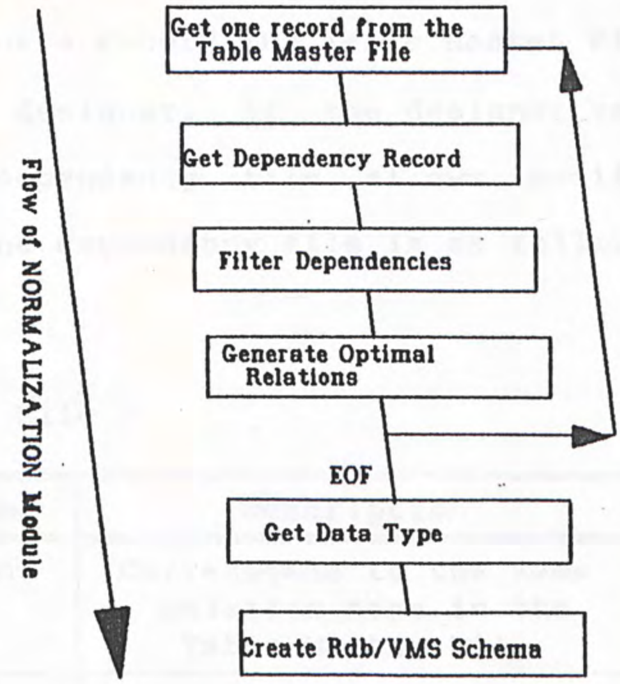


Figure 7.1 Program Flow of NORMALIZATION

7.2 Gathering Data Dependencies

For each record in the Table Master File, data dependencies are gathered and group as one record stored in the Dependency File named as the schema name plus extension '.dep'.

Each record in Table Master File will be processed one by one until the end of the file. The first time the designer goes through this process for a particular schema, the Dependency File is empty. Therefore, a series of data dependencies for a record in Table Master File have to be entered by the designer. If the designer reprocesses the records, the dependency file allows modifications. The structure of the dependency file is as follows:

Dependency File :

Field Name	Description
Relation	Corresponds to the same relation name in the Table Master File
dependencies	Combinations of FDs and MVDs (Total: Maximum 20)

7.2.1 Considerations on identifying data dependencies

The format and the restrictions of the set of FDs and MVDs input interactively by the designer are as follows:

Format : $a_1[,a_2[,a_3[...]]]$ FD|MVD $b_1[,b_2[,b_3[...]]]$

Restrictions: (i) A FD|MVD B, A and B are two disjoint sets such that $|A| \leq 5$ and $|B| \leq 5$

(ii) Of all the FDs entered by the user, if $|B|>1$, then each of $b_1[,b_2[,b_3[...]]]$ is assumed to exist mandatory in accordance with the determinant.

7.2.2 An example of defining data dependencies

This section mainly illustrates the process of identifying data dependencies and how to represent them when entering into the system. Figure 7.2 is a sample layout of the screen when a relation called ACADEMIC [Fagin,1977] from the Table Master File is retrieved and waiting the designer to determine if there exists data dependencies.

```
 ACADEMIC                  3 _ _ _ _ _ fd 4  6  _ _ _
 1: Class                  7 _ _ _ _ _ fd 8  9  _ _ _
 2: Section                1 2 _ _ _ _ mv 11 12 _ _ _
 3: Student                1 2 3 _ _ _ mv 5  _  _ _ _
 4: Major                  1 2 11 _ _ _ fd 12 _  _ _ _
 5: Exam                   1 2 _ _ _ _ mv 3  4  5 6 _
 6: Year                   1 _ _ _ _ _ mv 10 _  _ _ _
 7: Instructor             1 2 _ _ _ _ mv 7  8  9 _ _
 8: Rank                   _ _ _ _ _ _ _ _ _ _ _
 9: Salary                 _ _ _ _ _ _ _ _ _ _ _
10: Text                   _ _ _ _ _ _ _ _ _ _ _
11: Day                    _ _ _ _ _ _ _ _ _ _ _
12: Room                   _ _ _ _ _ _ _ _ _ _ _
13: Max_Student_allowed    _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _
                           _ _ _ _ _ _ _ _ _ _ _

                           Data Correct ? (y/n) _
```

Figure 7.2 Screen Layout for Entering Data Dependencies

On the left hand side of the screen, each attribute in the relation is assigned a number, and the primary key is underscored. The reason of numbering each attribute is for the sake of convenience for the designer to fill the dependencies in the spaces provided on the right hand side of the screen. Detail explanations on each of the dependencies will be provided in the following.

provided in the following.

<u>Dependency #1 : 3 fd 4 6</u>

It can be translated as "student functionally determines major, year". Each student's major and his year of study can be found out if we know that particular student.

<u>Dependency #2 : 7 fd 8 9</u>

It can be translated as "instructor functionally determines rank, salary. By knowing a particular instructor, we can also find out his rank and salary.

<u>Dependency #3 : 1 2 mv 11 12</u>

It can be translated as "<u>class, section</u> multivalued determines day, room". By knowing a particular class and section, we can find out more than one meeting places meeting time.

<u>Dependency #4 : 1 2 3 mv 5</u>

It can be translated as "<u>class, section, student</u> multivalued determines exam". By knowing the class and section that a particular student enrolled in, we can find out more than one exam scores.

Dependency #5 :   1 2 11 fd 12

It can be translated as "class, section, day functionally determines room". By knowing the meeting time of a class and section, we can find out only one meeting place.

Dependency #6 :   1 2 mv 3 4 5 6

It can be translated as "class, section multivalued determines student, major, exam, year. There are more than one student with major and year enrolling in a particular class and section with more than one exam scores.

Dependency #7 :   1 mv 10

It can be translated as "class multivalued determines text". For each class, several number of text books are used.

Dependency #8 :   1 2 mv 7 8 9

It can be translated as "class, section multivalued determines instructor, rank, salary". There are more than one instructor a rank and a salary teaching in a particular class and section.

7.3 The Filtering of Dependencies

After the all the possible data dependencies have been entered for particular relation of the Table Master File, another process will be carried out - Filtering Dependencies. The identification of the dependencies by the designer is one of the major human interaction points in this system, besides, it also acts as a tool to help, or rather, to force them to understand the relationship within a relation more thoroughly. The more detail the designer understands, the greater the possibility of containing redundancies in the group of dependencies. Therefore, this is an intermediate stage which apply certain rules to eliminate all the redundancies of the data dependencies input by the designer.

7.3.1 Rules for filtering dependencies

Every two dependencies will be put together twice and tested by a couple of rules. We identify each dependency in the pair as source and *target*. Each dependency in the pair will also change their status from source to target and vice versa, depends on what the status is at the first round. Refer to section 7.2.1, the right hand side of a dependency is the side that contains those attributes *b*, and the left hand side of a dependencies is naturally the determinant. Redundancy occurs when two pairs of data dependencies of a single relation have the following properties :

Property (i) : $(sourceRHS) \cap target = target$

Action Taken : source RHS = (target LHS) Union (source RHS - target)

Property (ii) : $(targetRHS) \cap source = source$

Action Taken : target RHS = (source LHS) Union (target RHS - source)

Property (iii) : $(|source| = |target|)$

and $(source = target)$

and $|sourceLHS| > |targetLHS|$

Action Taken : delete target

Property (iv) : $(|source| = |target|)$

and $(source = target)$

and $|sourceLHS| < |targetLHS|$

Action Taken : delete source

Property (v) : $(|source| = |target|)$

and $(source = target)$

and $|sourceLHS| = |targetLHS|$

and source = fd

Action Taken : delete source

Property (vi) : $(|source| = |target|)$

and $(source = target)$

and $|sourceLHS| = |targetLHS|$

and source = mv

Action Taken : delete target

7.3.2 Examples of filtering dependencies

The purpose of this section is to explain the properties listed in the previous section, and to illustrate what action has to be done if such properties are encountered. This is the continuation of the example used in section 7.2.2

Again, after the process of defining dependencies for a relation of the Table Master File, record will be created for storing the relation name and the group of dependencies in the Dependency File.

The eight data dependencies of relation ACADEMIC are as follows:

3	fd 4,6	(1)
7	fd 8,9	(2)
1,2	mv 11,12	(3)
1,2,3	mv 5	(4)
1,2,11	fd 12	(5)
1,2	mv 3,4,5,6	(6)
1	mv 10	(7)
1,2	mv 7,8,9	(8)

Example 1:

Dependency (1) and (6) form a pair:

    Source = 3 fd 4,6   (1)
    Target = 1,2 mv 3,4,5,6   (6)

This particular pair of dependencies will contain property (ii) in section 7.3.1 By applying the action taken of property (ii), the filtering process will be:

replace target RHS by 3 union (3,4,5,6 - 3,4,6)

3 union 5

3,5

Therefore, target will become 1,2 mv 3,5

Example 2:

Dependency (2) and (8) form a pair:

    Source = 7 fd 8,9   (2)
    Target = 1,2 mv 7,8,9   (8)

This particular pair of dependencies will contain property (ii) in section 7.3.1 By applying the action taken of property (ii), the filtering process will be:

replace target RHS by 7 union (7,8,9 - 8,9)

7 union 7

7

Therefore, target will become 1,2 mv 7

Example 3:

Dependency (3) and (5) form a pair:

Source = 1,2 mv 11,12 (3)

Target = 1,2,11 fd 12 (5)

This particular pair of dependencies will contain property (iv) in section 7.3.1 By applying the action taken of property (iv), the filtering process will be: delete source, which is dependency (3)

Therefore, dependency (3) no longer exist.

Example 4:

Dependency (4) and newly formed (6):

Source = 1,2,3 mv 5 (4)

Target = 1,2 mv 3,5 (6)

This particular pair of dependencies will contain property (iii) in section 7.3.1 By applying the action taken of property (iii), the filtering process will be: delete target, which is dependency (6)

Therefore, dependency (6) no longer exist.

7.4 Generating Optimal Relations

This process also needs the Table Master File and the Dependency File as information supplier, because all the dependencies in the Dependency File are identified based on the Table Master File. In a sense, the Dependency File is a tool to make the relations in the Table Master File simpler, and this leads to the introduction of the Final Master File, which contains a simpler version of the Table Master File.

Although there are several kinds of data dependencies, there are only two kinds - Functional Dependency (FD) and Multivalued Dependency (MVD), which are considered as the most important in respect to reality.

7.4.1 Algorithm of generating optimal relations

The idea of generating a set of optimal relations comes from the notions and definitions of the Boyce/Codd Normal Form (BCNF) and the Fourth Normal Form (4NF). Since MVDs is a subset of FDs, the way that the algorithm treats MVDs is more or less the same with the FDs.

For each FD in a dependency record, generate a new relation in the Final Master File. FD's LHS will be the primary key, FD's RHS will be the attributes.

For each MVD in a dependency record, generate a new relation in the Final Master File. Both the MVD's LHS and the MVD's RHS will be the primary key. There will be no attribute in this case.

Generate an extra relation for the original relation from the Table Master File. Such a relation will not contain all the MVD's RHS and FD's RHS of the dependencies.

7.4.2 An example

The information provided is as the following. The left hand side contains the original relation from the Table Master File. The right hand side contains the set of data dependencies after the filtering process.

The relation in the Table Master File:	The set of dependencies after the filtering process:
ACADEMIC 1: <u>Class</u> 2: <u>Section</u> 3: Student 4: Major 5: Exam 6: Year 7: Instructor 8: Rank 9: Salary 10: Text 11: Day 12: Room 13: Max_Student_allowed	 3 fd 4 6 7 fd 8 9 1 2 11 fd 12 1 2 3 mv 5 1 mv 10 1 2 mv 7

Generating FD : 3 fd 4 6

The new relation will be: ACADEMIC1

Primary Key : Student

Attributes : Major, Exam

Generating FD : 7 fd 8 9

The new relation will be: ACADEMIC2

Primary Key : Instructor

Attributes : Rank, Salary

Generating FD : 1 2 11 fd 12

The new relation will be: ACADEMIC3

Primary Key : Class, Section, Day

Attributes : Room

Generating MVD: 1 2 3 mv 5

The new relation will be: ACADEMIC4

Primary Key : Class, Section, Student, Exam

Generating MVD: 1 mv 10

The new relation will be: ACADEMIC5

Primary Key : Class, Text

Generating MVd: 1 2 mv 7

The new relation will be: ACADEMIC6

Primary Key : Class, Section, Instructor

Adding the Original Relation:

The relation remains as : ACADEMIC

Primary Key : Class, Section

Attributes : max_student_allowed

7.4.3 The results obtained

The output of this module, also is the final result of the whole design system, is as follows:

Final Master File:

Field Name	Description
fnl_name	Final Table Name
fnl_pk	Primary key
fnl_at	Attribute(s)

7.5 Linkage to VAX Rdb/VMS

The previous sections have clearly presented the major function of NORMALIZATION. Since the whole system is implemented on VAX 780 (VAX/VMS), it is worthwhile to provide an extension of CARDBD, i.e. to utilize the available VAX Rdb/VMS, in order to create a database schema on such an environment. VAX Rdb/VMS is a general purpose database management system based on the relational data model. Other DIGITAL software products, such as DATATRIEVE, can take advantage of the features of Rdb/VMS.

7.5.1 Defining database on Rdb/VMS

This section discusses the use of RDO statements to define the database. There are three ways you can enter the RDO statements, but the one that the system adopted is:

Use an editor to create a command file that has an RDO file type (for example, personnel.RDO). Such a command file can contain all the definition statements required to create the database. Then the RDO command procedure can be executed at the RDO> prompt. Simply type an at sign (@) followed by the name of the command procedure file: RDO> @personnel.

<u>7.5.2 Get data types</u>

As mentioned in the previous section, we have to use RDO command file to define a database. Inside the RDO command file, the definitions of fields and relations are very important. The basic syntactical requirements for defining a field are: field name and data type. The requiremens for defining a relation are: relation name, associated field names.

Therefore, the purpose of this process GET DATA TYPES is to group all key attributes and non-key attributes from all the relations of the Final Master File into a set with unique attribute names. Then, the designer will be prompted to enter the data type for each element from the set, and the data type entered are stored in a file named as the schema name plus the extension '.tpe'.

The following table lists the characteristics for each data type which will be catered in CARDBD.

VAX Rdb/VMS Data Type	Size (Bits)	Range/ Precision	Other Parameters
SIGNED WORD	16	-32768 to 32767	n = scale factors
SIGNED LONGWORD	32	-2**31 to (2**31)-1	n = scale factors
F_FLOATING	32	approx. 7 decimal digits	none
DATE	64	n/a	none
TEXT	n bytes	0 to 16383 char.	n = # of characters (unsigned integer)
VARYING STRING	Varies	0 to 16383 char.	n = max # of characters (unsigned integer)

Six bytes are used to store the data type of each field. The following table shows how the data type are stored.

First Byte	Second to Sixth Byte
'W' = Signed Word	scale factor
'L' = Signed Longword	scale factor
'F' = F_Floating	N/A
'D' = Date	N/A
'T' = Text	number of characters
'V' =Varying String	max number of characters

7.5.3 Generating RDO command file

After the designer has already entered (or checked) all the data types of each attribute of relations in the Final Master File, the RDO command file can then be generated. Note that the RDO command file has the file extension of '.RDO'. Using the example ACADEMIC, the process from getting data types and the generation of RDO commmand file are as follows:

Fields	Data Type	Explanations
Class	T6	TEXT of 6 char.
Section	T2	TEXT of 2 char.
Student	T8	TEXT of 8 char.
Major	T3	TEXT of 3 char.
Exam	W2	SIGNED WORD SCALE FACTOR 2
Year	T1	TEXT of 1 char.
Instructor	T8	TEXT of 8 char.
Rank	T4	TEXT of 4 char.
Salary	L2	SIGNED LONGWORD SCALE FACTOR 2
Text	V100	VARYING STRING of max 100 char.
Day	T1	TEXT of 1 char.
Room	V10	VARYING STRING of max 10 char.
Max Student allowed	W	SIGNED WORD

Based on the above data types entered for each field, the field definitions are then syntacticaly generated.

e.g.

```
DEFINE FIELD CLASS
        DATATYPE IS TEXT
        SIZE IS 6.
  ...
```

```
DEFINE FIELD EXAM
     DATATYPE IS SIGNED WORD SCALE 2.
...
DEFINE FIELD SALARY
     DATATYPE IS SIGNED LONGWORD SCALE 2.
...
```

After all the fields are defined, each relation in the Final Master File is retrieved and the relation definitions are generated as follows,

e.g.

```
DEFINE RELATION ACADEMIC.
     CLASS.
     SECTION.
     MAX_student_allowed.
END ACADEMIC RELATION.
```

<u>7.5.4 Comments on the schema created</u>

Since the field definitions are defined on the three basic required elements: field name, field data type and field size, the schema built on the VAX Rdb/VMS is a database which contains the minimal structures requirement.

7.6 Complete Algorithm for NORMALIZATION

In this chapter, the NORMALIZATION is partitioned and presented in separate sections, the following is the whole idea, or rather, the algorithm for NORMALIZATION.

```
while Table Master File not EOF do
begin
   i := 0
   Final Master File := ø;
-----------------------------------------------------------
   display the relation;
   ask the user to input all the FDs and MVDs of the
relation
   into two sets FD and MVD respectively;
-----------------------------------------------------------
   D:=FD∪MVD ;
   for each d∈D do
   begin
      source := d
      for each d∈D do
         target := d
         if (sourceRHS)∩target=Ø
           then next target
         if (sourceRHS)∩target=target
           then sourceRHS:=targetLHS∪(sourceRHS-target)
                next target
         if (targetRHS)∩source=source
           then targetRHS:=sourceLHS∪(targetRHS-source)
                next target
         if (|source|=|target|) and (source=target)
           then if |sourceLHS|>|targetLHS|
                  then D := D - target
                  else D := D - source
                if |sourceLHS|=|targetLHS|
                  then if source is FD
                          then D := D - source
                          else D := D - target
   end
```

to be continued on next page

continued

```
    R := tab_relation
    for each f∈FD do
         i := i + 1
         fnl_name  :=  Rᵢ
         fnl_pk    := LHS of f
         fnl_at    := RHS of f
         R := R - RHS of f
         add Rᵢ to Final Master File
    for each m∈MVD do
         i := i + 1
         fnl_name  :=  Rᵢ
         fnl_pk    := (LHSofm)∪(RHSofm)
         fnl_at    := blanks
         R := R - RHS of m
         add Rᵢ to Final Master File
       add R to Final Master File

end while;
```

8 CONCEPTUAL AND LOGICAL DESIGN CASE STUDY

A case study embodying many of the concepts of computer-aided relational database design will be presented. Emphasis is given to the iterative nature of the process and to control of the process and to the control of the process by the human designer acting in response to the information presented in the edit, diagnostic, and design reports. The iterations will be carried through the conceptual and logical design phases. The application of physical design will not be considered.

8.1 Business Nature Description

This case study involves the design of a Rdb/VMS database for inventory control and accounts receivable of a cold storage company. The nature of business is to provide storage space for customers in order to receive payments for storage charges as its major income.

This company owns a 17-storey building, starting from the 3rd floor to the 17th floor are for rental purposes.

There are three different storage environment for different types of stock or for different requests by customers. The three storage environments are as follows:

Freezer -- temperature below -8c

Chiller -- temperature between -4c - 0c

General -- room temperature

Most of the customers are provision companies, and the stock that they store are mostly frozen meat. The company charges different customers with different types of stock at different rates. For goods stored in *freezer* or *chiller*, the company charges customer: a certain rate times the gross weight of the goods. For goods stored in *general*, the company charges the customer: a certain rate times the volume of goods.

8.2 Different Types Of Operation

8.2.1 Incoming Operations

If a customer wants to store goods in the warehouse, he should hire his own lorry and workers to transport the goods to the warehouse and to unload all the goods to the loading/unloading platform. The staff of the company are only responsible to allocate the goods to a location in a proper storage environment. Upon receiving the goods, the company will issue a temporary receipt (T/R) to the customer to acknowledge the quantity and the gross weight of the received goods. On this particular temporary receipt, the received goods will also be assigned a *lot number*. The floor on which the LOT is allocated, and all the particulars which associate with that particular LOT will also be recorded on the T/R.

8.2.2 In Case of Banker Goods

Sometimes another company lends money to that customer to buy that LOT of goods, then the lender is the banker of that particular LOT. Therefore the banker's name will be recorded on the T/R.

Customer Name	Date: dd/mm/yy T/R No. 99999
Banker Name	Transportation Company

Lot No.	Desc.	Marks	Qty.	@Net Wgt	@ Gross Wgt

Remarks	Prepared by
	______________ Signature

Sample T/R

8.2.3 Godown Warrant Issuing

One godown warrant (G/W) is issued for one LOT, and at the same time, a transaction will be added to account receivable. Once the G/W is issued, the amount that is charged on the G/W is valid on a one-month period.

8.2.4 Storage Renewal

If there is any goods of a particular LOT remains in the warehouse on the first day of the next month, a procedure called *renewal* has to be done.

The calculation of renewal charges is simply: the volume or gross weight of quantity remained times rate.

8.2.5 Minimum Charges

The minimum storage charges is HK$20 for chiller and freezer, HK$10 for general.

8.2.6 Outgoing Operations

The staff will deliver the specified quantity of goods to the loading/unloading platform with the instructions of a Delivery Order (D/O) of a customer.

Customer:	D/O# 999999
Transportation Co.	Date: dd/mm/yy

Lot Number	Description	Quantity

Received By :	Customer Signature : (with Company Chop)

Sample D/O

8.3 Design Logical Schema For This Cold Storage Company

8.3.1 E-R MODELING

In the first design phase, a number of entities are identified as follows:

(Note that there is restriction on the length of the name of an entity, its primary key and attributes. Each of them cannot exceed 20 characters in order to be represented in CARDBD)

Account Receivable

This entity will be named as ACCT-RECEIV. It represents various kinds of invoices such as the invoices that go along with the issuing of godown warrants, the invoices for the storage renewal charges, and some other charges. It has an issue date, total amount of invoice, due date, outstanding amount of the invoice and the payment voucher number if payment has been made.

Banker

This entity will be named as BANKER with banker name, address, telephone number, name of contact person.

Customer

This entity will be named as CUSTOMER with customer name, address, telephone number, name of contact person, and credit limit.

Delivery Order

This entity will be named as DO, with an issue date, customer who issues the delivery order.

Employee

This entity will be named as EMPLOYEE with all the personal particulars.

Floor

This entity will be named as FLOOR with its storage type and capacity.

Payment

This entity will be named as PAYMENT with a payment date, payment amount, and invoice(s) that paid.

Lot

This entity will be named as LOT with unique Lot number, and all the lot's information in a particular shipment. Each lot of goods has an incoming date, incoming quantity, actual quantity including those on the loading/unloading platform which are pending, and actual quantity available for another delivery order.

Stock

This entity will be named as STOCK with a description of a particular stock.

Entity	Primary Key	Attributes
ACCT-RECEIV	AR-no	AR-issue-date AR-total AR-paid-vouchno AR-outstanding AR-due-date
BANKER	BK-code	BK-name BK-address BK-tel BK-cont-person
CUSTOMER	CUS-code	CUS-name CUS-address CUS-tel CUS-con-person CUS-Crlimit
DO	DO-no	DO-date DO-cust DO-tran-detail
EMPLOYEE	Emp-no	Emp-name Emp-address Emp-Tel Emp-birthdate Emp-HKID-no Emp-date-joined Emp-start-sal Emp-present-sal Emp-title

Entity	Primary Key	Attributes
FLOOR	FL-no	FL-strg-type FL-capacity
PAYMENT	PAY-vouc-no	PAY-date PAY-amount PAY-ar
LOT	LOT-no	SHIPMT-no SHIPMT-label SHIPMT-origin LOT-marks LOT-indate LOT-ini-qty LOT-avl-qty LOT-upd-qty LOT-unit-wgt LOT-unit-vol
STOCK	STK-code	STK-desc

A number of relationships among the above entities are studied and gathered in the following:

between Banker, Lot

This relationship will be named as BK-LOT.

- A Banker can own more than one Lot.
- A Lot can at most be owned by a Banker.
- A Banker need not own a Lot.
- A Lot need not be owned by Banker.

between Floor, Lot

This relationship will be named as FL-LOT.

- A Floor can have more than one Lot.

- A Lot can be allocated in more than one Floor.
- A Floor can have no Lot, i.e. empty.
- A Lot must be allocated on at least one Floor.

between Floor, Employee

This relationship will be named as FL-IC.

- A Floor must have exactly one Employee as in-charge.
- An Employee might not be a Floor in-charge.
- A Floor in-charge is in charge of one Floor only.

between Stock, Lot

This relationship will be named as ST-LOT.

- Each kind of Stock can have many Lots.
- A Stock can have no Lot at a certain moment.
- A Lot must have exactly one type of stock.

between Stock, Customer

This relationship will be named as ST-CU.

- A Customer stores more than one Stock.
- A Customer can have no Stock at a certain moment.
- A type of Stock can be stored by many customers.
- A Stock might not exist at a certain moment.

between Payment, Customer

This relationship will be named as PY-CU.

- A Payment must be paid by exactly one Customer.
- A Customer can have no Payment.
- A Customer can hove more than one Payment.

between Lot, Customer, Account receivable

This relationship will be named as LOT-C-A.

- A Customer can have more than one Lot.
- A Lot can be owned by exactly one Customer.
- For each combination of Lot & Customer, there might be at least one invoice or more than one invoice.

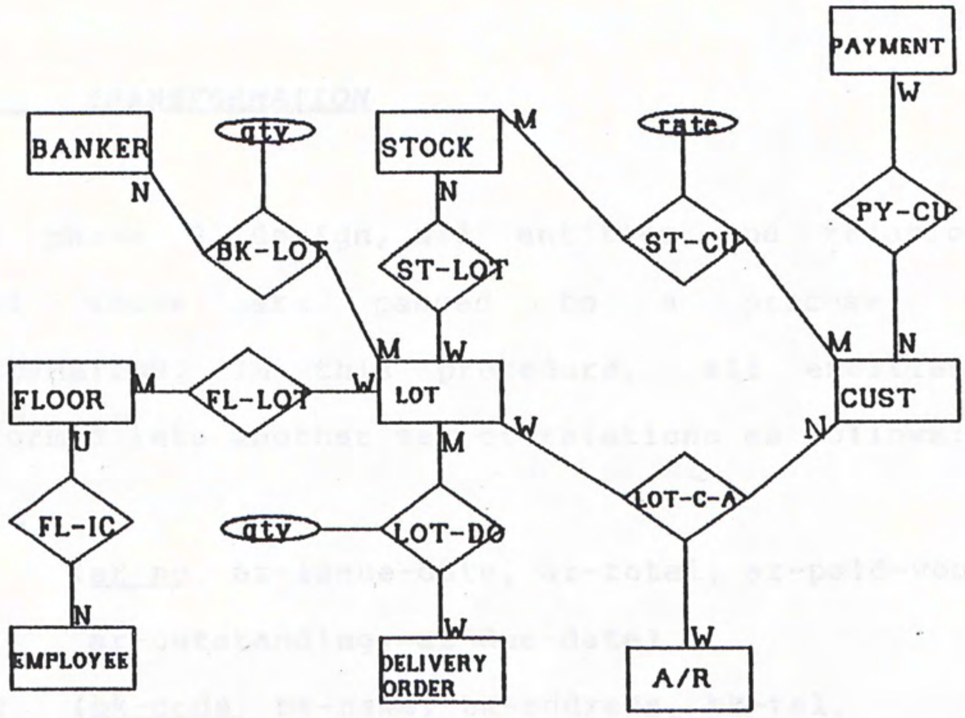
between Lot, Delivery order

This relationship will be named as LOT-DO.

- A Lot can have more than one Delivery order.
- A Lot can have no Delivery order.
- A Delivery order contains at least one Lot.

Relationship	Related Entities	Mapping	Attribute
BK-LOT	Banker Lot	N M	BK-LOT-qty
FL-LOT	Floor Lot	M W	
FL-IC	Floor Employee	U N	
ST-LOT	Stock Lot	N W	
ST-CU	Stock Cust	M M	rate
PY-CU	Payment Cust	W N	
LOT-C-A	Lot Cust AR	W N W	
LOT-DO	Lot DO	M W	LOT-DO-qty

The Entity-relationship diagram of the above environment is as follows:



<u>Entity-relationship Diagram of the Cold Storage Company</u>

8.3.2 TRANSFORMATION

In the phase 2 design, all entities and relationships defined above are passed to a process called TRANSFORMATION. In this procedure, all entities are transformed into another set of relations as follows:

AR (ar-no, ar-issue-date, ar-total, ar-paid-vouchno, ar-outstanding, ar-due-date)

BANKER (bk-code, bk-name, bk-address, bk-tel, bk-cont-person)

CUSTOMER (cus-code, cus-name, cus-address, cus-tel, cus-cont-person, cus-crlimit)

DO (do-no, do-date, do-cust)

EMPLOYEE (emp-no, emp-hkid-no, emp-name, emp-address, emp-tel, emp-birthdate, emp-date-joined, emp-start-sal, emp-present-sal, emp-title)

FLOOR (fl-no, fl-strg-type, fl-capacity)

PAYMENT (pay-vouc-no, pay-amount, pay-ar)

LOT (lot-no, shipmt-no, shipmt-label, shipmt-origin, lot-no, lot-marks, lot-indate, lot-ini-qty, lot-avl-qty, lot-upd-qty, lot-unit-wgt, lot-unit-vol)

STOCK (stk-code, stk-desc)

The transformation of the relationships are treated in a different way.

BK-LOT (<u>lot-no,</u> bk-code, bk-lot-qty)

FL-LOT (<u>fl-no</u>, <u>lot-no</u>)

LOT-C-A (<u>lot-no</u>, <u>ar-no,</u> cus-code)

LOT-DO (<u>lot-no</u>, <u>do-no</u>, lot-do-qty)

ST-CU (<u>stk-code</u>, <u>cus-code</u>, rate)

The relationship *FL-IC* cannot be transformed like the others because it has mappings U:N. In this case, the system gets the relation FLOOR which has the mapping U in FL-IC, then it would be modified automatically as follows:

FLOOR (<u>fl-no</u>, fl-strg-type, fl-capacity, emp-no)

Also, the relationship *ST-LOT* cannot be transformed like the others because it has mappings N:W. In this case, the system will search for the relation which has 'many' and 'mandatory' mapping , and then merges the primary key of the other entity into the relation.

Therefore, the relation *LOT* would be modified automatically as follows:

LOT (<u>lot-no</u>, shipmt-no, shipmt-label, shipmt-origin, lot-marks, lot-indate, lot-ini-qty, lot-avl-qty, lot-upd-qty, lot-unit-wgt, lot-unit-vol, stk-code)

Finally, the relationship *PY-CU* cannot be transformed like the others because it has mappings N:W. In this case, the system will search for the relation which has 'many' and 'mandatory' mapping , and then merges the primary key of the other entity into the relation.

Therefore, the relation *PAYMENT* would be modified automatically as follows:

PAYMENT (pay-vouc-no, pay-amount, pay-ar, cus-code)

8.3.3 NORMALIZATION

Each relation of the result produced by the transformation process is to be displayed on the screen and prompt the designer to enter up to 20 data dependencies associated to that relation. After the dependencies are keyed, this process will filter the set of dependencies to make sure that no dependency representation redundancy is exist. Then, followed by the execution of the normalization algorithm, and finally the particular relation is normalized. If the designer finds that there is no FDs or MVDs among the relation, he may opt not to enter any for this relation.

In the case of this example, all relations have no data dependencies except ar, banker, customer, payment and lot.

Explanation 1:

Banker (bk-code, bk-name, bk-address, bk-tel, bk-cont-person)

A banker can have more than one branch, therefore bk-address, bk-tel, bk-cont-person are multivalued depend on the key.

Explanation 2:

Customer (cus-code, cus-name, cus-address, cus-tel, cus-cont-person, cus-crlimit)

Same reason as explanation 1.

Explanation 3:

Payment (pay-vouc-no, pay-date, pay-amount, pay-ar, cus-code)

Since partial payment is allowed, therefore, one payment voucher multivalued determines transaction in account receivable.

Explanation 4:

Lot (lot-no, shipmt-no, shipmt-label, shipmt-origin, lot-marks, lot-indate, lot-ini-qty, lot-avl-qty, lot-upd-qty, lot-unit-wgt, lot-unit-vol)

It is possible to have several lots of goods in one shipment, therefore, one shipment multivalued determines lot, and lot-no functionally determines all the attributes concerning the particulars of a lot, and shipmt-no functionally determines shipmt-label and shipmt-origin.

Explanation 5:

AR (ar-no, ar-issue-date, ar-total, ar-paid-vouchno, ar-outstanding, ar-due-date)

It is possible to have partial payment, i.e. several payment vouchers for one invoice.

The result of normalization is as follows:

ar	(ar-no, ar-issue-date, ar-total, ar-outstanding, ar-due-date)
ar1	(ar-no, ar-paid-vouchno)
banker	(bk-code, bk-name)
banker1	(bk-code, bk-address, bk-tel, bk-cont-person)
bk-lot	(lot-no, bk-code, bk-lot-qty)
customer	(cus-code, cus-name, cus-crlimit)
customer1	(cus-code, cus-address, cus-tel, cus-cont-person)
do	(do-no, do-date, do-cust)
employee	(emp-no, emp-hkid-no, emp-name, emp-address, emp-tel, emp-birthdate, emp-date-joined,

emp-start-sal, emp-present-sal, emp-title)

fl-lot (fl-no, lot-no)

floor (fl-no, fl-strg-type, fl-capacity, emp-no)

payment (pay-vouc-no, pay-date, pay-amount, cus-code)

payment1 (pay-vouc-no, pay-ar)

lot (lot-no, lot-marks, lot-indate, lot-avl-qty, lot-upd-qty, lot-ini-qty, lot-unit-wgt, lot-unit-vol, stk-code)

lot1 (shipmt-no, shipmt-label, shipmt-origin)

lot-c-a (lot-no, ar-no, cus-code)

lot-do (lot-no, do-no, lot-do-qty)

st-cu (stk-code, cus-code, rate)

stock (stk-code, stk-desc)

The above mentioned three phases construct the main theme of CARDBD. Based on the above resulting relations, and the appropriate data type definitions for each field, a schema can be created on any relational database management system. However, in this research, a schema on VAX Rdb/VMS will be created.

8.4 Remarks To This Case Study

In this case study we have tried to illustrate the concepts of computer-aided relational database design. In doing so, we have emphasized the iterative nature of the process and the human control that must be exercised. One of the major contributions of these procedures is in helping to identify and understand the characteristics of the data gathered and relationships among them. Besides, editing reports and resulting reports are also available. By providing such reports early in the design process, and enabling repetive iterations, these automated procedures should materially assisted the designer in obtaining an efficient design for current requirements and future requirements as he is able to forsee them and in reducing the time required to complete the design study.

9 CONCLUSION

This chapter concludes the research reported in this thesis. Besides presenting the major contributions, it also outlines the areas of further research, and suggestions are made about how to address the issues of these areas.

9.1 The Contributions of the Research

In the following, the attempts made in each of these areas to meet the requirements of practical applications are summarized.

9.1.1 E-R MODELING: conceptual modeling

It should be stressed again that CARDBD is a tool to assist the designer. The real logical design is still done by the human designer. Although generating a set of higher normal form relations seems to be the duty of TRANSFORMATION and NORMALIZATION, the importance of E-R MODELING should not be neglected. The existence of E-R MODELING enables the designer to have a deeper understanding of "inter-entity" dependencies, and the structures of the enterprise scheme.

9.1.2 TRANSFORMATION: conceptual to relational model

Ever since Entity-relationship model was introduced, a lot of research works have been carried on to investigate an ideal methodology to translate an E-R schema into relational data model. For example, [Dumpala,1983] proposed an algorithm for E-R model to relational model; however, this algorithm does not consider null cases in relations. Moreover, [Forth,1986] presents an algorithm for the same purpose, but the transformation rules on the weak entities is really a little bit too "weak".

The transformation algorithm presented in this thesis solves the problems mentioned above. It utilizes the mappings in relationship sets defined in E-R MODELING and therefore can eliminate all the unnecessary relations at this stage.

9.1.3 NORMALIZATION: generating optimal relations

By integrating techniques concerning "inter-entity" and "intra-entity" dependencies, and other important ingredients into CARDBD, the generated relations :

contain minimum redundancies,

are free of update anomalies, and lastly,

the number of relations is minimized.

As defined earlier, a set of relations which fulfills the above is considered to be a set of *optimal relations*.

9.2 The Directions for Further Research

It has been and remains the goal of this reseach to develop software that automates the process of logical database design. To this end all modules of CARDBD are implemented on VAX/VMS using Pascal as the programming tool. As mentioned before, with the assistance of such a design system, the database administrators when designing databases, will not have as many problems and difficulties encountered as designing manually. However, there is no limitations on building a design system; the more extensive the design system, the shorter the design cycle and the better the design quality.

The research reported in this thesis leaves open a number of areas which deserves further attention. In the following sections, these areas will be outlined and suggestions about how to deal with them will be given.

9.2.1 Auto-generating E-R Diagram

The E-R MODELING is built based on the concepts of Chen's Entity-Relationship Model. In CARDBD, all the information related to the E-R Model are stored in the Entity Master File and the Relationship Master File, in which the designer cannot visualize the actual relationships between the entities instantly. Instead, the designer can at most get both reports from Entity File and Relationship File and then relates them together manually. This matter brings attention to an area that uses the notions of Entity-relationship Diagram.

The proposed solution is to build another submodule that will take the information in the Entity and

Relationship Master File as input, and then an entity-relationship diagram will be auto-generated. This sub-module may be treated as a graphical interface for CARDBD.

It is an hopeless matter to define formally what is a pleasant ERD and what is not. In any case, several aesthetic criteria may be identified. Research has to be done in order to choose effective criteria and test their relevance in a real environment. Two well-admitted aesthetics [Carpano,1980] [Warfield,1977] [Batini,1985] valid independently of the graphic standard, are:

- Minimization of crossings between connections;
- Even Distribution of symbols and connections.

9.2.2 Optimized the linkage to VAX Rdb/VMS

The schema defined in VAX Rdb/VMS by CARDBD contains the minimal requirement for a workable database. In order to build a schema which is tailor-made that fits the real-world environment, more research has to be done to smoothen the data definition process, such that various features and clauses in defining fields and relations that are available in VAX Rdb/VMS schema definition can be fully utilized [Digital,1985]. For example, a typical field definition for an Rdb/VMS relation is:

```
DEFINE FIELD  ROOM_TYPE
   DESCRIPTION IS /* Hotel room type code */
       DATATYPE IS TEXT SIZE IS 2
       VALID IF
           ROOM_TYPE EQ "S"
         OR ROOM_TYPE EQ "D"
         OR ROOM_TYPE EQ "SS"
         OR ROOM_TYPE MISSING
         MISSING VALUE IS "??"
         EDIT_STRING FOR DATATRIEVE IS "XX".
```

From another point of view, the linkage can be treated as an interface between CARDBD and any specific relational database management system which this thesis uses VAX Rdb/VMS as an example.

REFERENCES

Batini, C. and Santucci, G. (1980), *Top-Down Design in the Entity-Relationship Model*, Entity-Relationship Approach to System Analysis and Design, Chen, P.P. (ed), North-Holland.

Batini, C. et al. (1985), *A Graphical Tool for Conceptual Design of Database Applications*, Computer-aided Database Design - The DATAID Project, North-Holland.

Bernstein, P.A. (1976), *Synthesizing Third Normal Form Relations from Functional Dependencies*, ACM Transactions on Database Systems, Vol.1, No.4, December 1976, pp. 277-298.

Brathwaite, K.S. (1985), *Data Administration: Selected Topics of Data Control*, John Wiley & Sons, New York.

Carpano, M. (1980), *Automatic Display of Hierarchized Graphs for Computer Aided Decision Analysis*, IEEE Trans. on Systems Man and Cybernetics, SMC-10N11

Chamberlin, D.D. (1976), *Relational Data-Base Management System, Computing Surveys*, Vol.8, No.1, March 1976, pp.43-67.

Chan, E.P.F. and Lochovsky, F.H. (1980), *A Graphical Database Design Aid using the Entity-Relationship Model*, Entity-Relationship Approach to Systems Analysis and Design, Chen P.P. (ed.), North-Holland Publishing Company.

Chen, P.P. (1976), *The Entity-Relationship Model -- Toward a Unified View of Data*, *ACM Transactions on Database Systems*, Vol.1, No.1, March 1976, Pages 9- 36.

Chen, P.P. (1977), *The Entity-Relationship Approach to Logical Database Design*, Q.E.D. Monogragh Series Data Base Management.

Chen, P.P. ed.(1980), *Entity-Relationship Approach to Systems Analysis Design*, North Holland.

Chen, P.P. (1985), *Database Design Based on Entity and Relationship*, Principles of Database Design, Yao, B.S. (ed), Prentice-Hall, New Jersey.

Codd, E.F. (1970), *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, Vol.13, No.6, June 1970, pp.377-387.

Codd, E.F. (1971), *Further Normalization of the Data Base Relational Model*, Courant Computer Science Symposium 6, Database Systems, Prentice-Hall, Englewood Cliffs, N.J., pp.65-98.

Date, C.J. (1983), *Introduction to Database Systems (Volume II)*, Addison Wesley.

Deen, S.M. (1977), *Fundamentals of Data Base Systems*, Macmillan, Australia.

Digital, (1985), *VAX Rdb/VMS Guide to Database Design and Definition*.

Dumpala, S.R. and Arora, S.K. (1983) Schema Translation Using the Entity-relationship Approach, Entity-relationship Approach to Information Modeling and Analysis, P.P. Chen (ed.), Elserier Science Publishers B.V. (North-Holland), pp.337-355.

Fagin, R. (1977), *Multivalued Dependencies and a New Normal Form for Relational Databases*, ACM Transactions on Database Systems, Vol.2, No.3, September 1977.

Forth, H.F. and Silberschatz, A. (1986), *Database System Concepts*, McGraw-Hill, Singapore.

Hubbard, G.U. (1981), *Computer-Assisted Data Base Design*, Van Nostrand Reihold Company, New York.

Hubbard, G.U. (1985), *Computer-Assisted Hierarchical Database Design*, Principles of Database Design Volume 1: Logical Organizations, Yao,S.B. (ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Kent, W. (1983), *A Simple Guide to Five Normal Forms in Relational Database Theory*, Comm. ACM 26, 2, February 1983.

Melkanoff, M.A. and Zaniolo, C. (1980), *Decomposition of Relations and Synthesis of Entity-Relationship Diagrams*, Entity-Relationship Approach to Systems Analysis and Design, Chen P.P. (ed.), North-Holland Publishing Company.

Ng, P.A. and Paul, J.F. (1980), *A Formal Definition of Entity-Relationship Models*, Entity-Relationship Approach to Systems Analysis and Design, Chen, P.P. (ed), North-Holland.

Nijssen, G.M. and Leung, C.M.R. (1987), *From a NIAM Conceptual Schema into the Optimal SQL Relational Database Schema*, The Australian Computer Journal, Vol19, 2, May 1987.

Prakash, N. (1984), *Understanding Data Base Management*, Tata McGraw-Hill Publishing Company Limited, New Delhi.

Robinson, H. (1981), *Database Analysis and Design*, Chartwell-Bratt.

Ullman, J.D. (1980), *Principles of Database Systems*, Computer Science Press, Rockville, Maryland.

Warfield, J. (1977), *Crossing Theory and Hierarchy Mapping*, IEEE Trans. on Systems Man and Cybernetics, SMC-7N7

Yau, C. (1987), *Methodology for Normalization*, Lecture Notes CSC431, (The Chinese University of Hong Kong).