CHAPTER I

INTRODUCTION

1.1. Background and Problem Formulation

Expert systems are a relatively recent phenomenon, but their impact on today's world is impressive and is expected to grow in importance. These are computer programs based on Artificial Intelligence techniques, which embody some aspects of human knowledge and solve problems that ordinarily require human intelligence.

The database design process is recognized as a nonalgorithmic, complex and dynamic activity, i.e. it requires several interactions with the database designer, and is full of uncertainty. The experienced database designer manages to master the task of design because he uses on one hand the algorithmic part of the design, and on the other hand his experience which enables him to recognize the typical cases and to treat them by analogy, and to be attentive to certain delicate aspects.

Database design experts are relatively scarce. Hence there is a need to automate the process of database design.

For all these reasons an expert system approach for database design seems to be appropriate. This is the problem addressed in this research work.

There are three major elements of this research. First, a top-down methodology for logical relational database is presented. Second, the methodology alongwith some heuristics is formalized as a set of rules, which forms the knowledge base of the expert system for database design. Finally, the expert system is implemented in PROLOG.

There are several applications of the research. The systems can be used by all types of organizations (i.e. from small to large) to develop their own database internally. The system can also be used by consultants working in both the database and expert system areas.

1.2. Overview of Database Management System

Database defined

An organization's database may be thought of as the collection of data necessary to satisfy the information requirements of the users within that organization [81]. A database system is simply a computerized record-keeping system which is used to record and maintain data within the database [Date, 1981].

Development of database systems, as they are known today, began in late 1960's. Interest in these systems was generated as a result of the perceived inadequacy of conventional file processing systems. The two main shortcomings of conventional file processing systems were :

1) The inability to efficiently integrate numerous large files and; 2) The inability to support higher level data and file organizations, often called secondary file organization [24]. Database systems differ from file processing systems in that they provide (among other things) a means of overcoming these deficiencies.

Over the years, numerous text books and reference articles have been written about designing, implementing, and maintaining a database. Formal definition of database and database processing abound, but can be broken down into three general categories. The first category is typical of those authors whose interest lies in the technical aspect of database technology. It is essentially a designer's view of the database. Cardenas [24] states :

> A database is a collection of occurrences of a number of record types where the record types and their occurrences are interrelated by means of specific relationships.

A second approach is more concerned with the way the database is used from the standpoint of the organization :

> A database is a grouping of data elements structured to fit the information needs of all functions of an organization [89].

Finally, there is the definition used by Kroenke [81] which tries to span both the technical concern and the application concern of the database technology.

> A database allows an organization's data to be processed as an integrated whole. It reduces artificiality imposed by separate files for separate applications and permits users to access data in a manner which is more natural to them.

From the user's point of view, we can look on the database as a large collection of structured data stored in one or several computers in order to serve several data processing applications.

Database Management System

Between the user who will access the data and the physical storage of that data is a software interface known as database management system. A database management system

is a generalized tool for manipulating large databases. It is made available through special software for the purpose of interrogating, maintaining, and analyzing the data within the database [61].

A DBMS provides numerous facilities to manage the organization's database. To the programmer, a DBMS provides the ability to focus concern on data which is relevant to the problem at hand. To the user, on the other hand, a DBMS provides the ability to formulate queries or browse through a data set without the need for special programming.

Other generally accepted advantages of using a DBMS include the following :

1. Data independence : this includes both physical independence, i.e. the programmer is shielded from storage device concerns; and logical independence, i.e. the logical format of the data may be changed without requiring changes in the program which uses this data.
2. Non-redundancy of stored data : duplication caused by the same data items residing in more than one file is eliminated.
3. Data integrity : because data is stored only once, the probability of inconsistent updating or deleting

is eliminated.

4. Data security : A DBMS generally provides multiple levels of access security.

Data models

In the context of database system, a data model is a term used to denote a set of concepts able to capture and describe information pertinent to the database application. A data model is a way of viewing data. It provides a basis for the construction of a DBMS. Traditionally, three main data models are identified : the hierarchical, the network, and the relational data model. By evaluating the relative advantages and disadvantages of different data models, one can evaluate and select a DBMS for a particular user application. Further more understanding of the differences in the data models assists in improving DBMSs in the future. Based on these three types of data models, three different categories of DBMS have been proposed. These are Hierarchical DBMS, Network DBMS and Relational DBMS. In general, the difference between the three types of DBMS lies in three specific areas :

1. The model defining the structure of the data ;
2. The way in which these systems place these data on the storage medium to be used and;
3. The method used to access this data once it is in place.

Figure-1.1 (adopted from Date [42]) shows how data is logically organized in each of the categories of DBMS.

In relational DBMS, the data is formatted into tables (Fig. 1.1). The S(supplier) table contains for each supplier, a supplier number, name, status code, and location. The P(part) table contains for each part, a part number, name, color, weight, and location. The SP(shipment) table contains for each shipment, a supplier number, a part number, and the quantity shipped. Each of these three tables is similar in form to a conventional sequential file, with rows corresponding to records in a file and columns corresponding to fields in a record.

In the hierarchical DBMS, the supplier and the part data is organized into four individual tree structures. Each tree consists of one part record occurrence together with a set of subordinate supplier record occurrences. We may compare this hierarchical view to a single file containing records arranged into four individual trees. Note that this file is more complex than the tables in the Fig. 1 since it contains two different types of records as well as links connecting occurrences of these records.

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P2	400
S2	P2	300
S3	P2	400

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London

Fig. 1.1a Sample data in Relational form.

P2	Bolt	Green	17	Paris
S2	Jones	10	Paris	300
S1	Smith	20	London	300

P2	Bolt	Green	17	Paris
S3	Blake	30	Paris	200
S2	Jones	10	Paris	400
S1	Smith	20	London	200

P3	Screw	Blue	17	Rome

P4	Screw	Red	14	London

S1	Smith	20	London	400

Fig. 1.1b Sample data in Hierarchicall form.

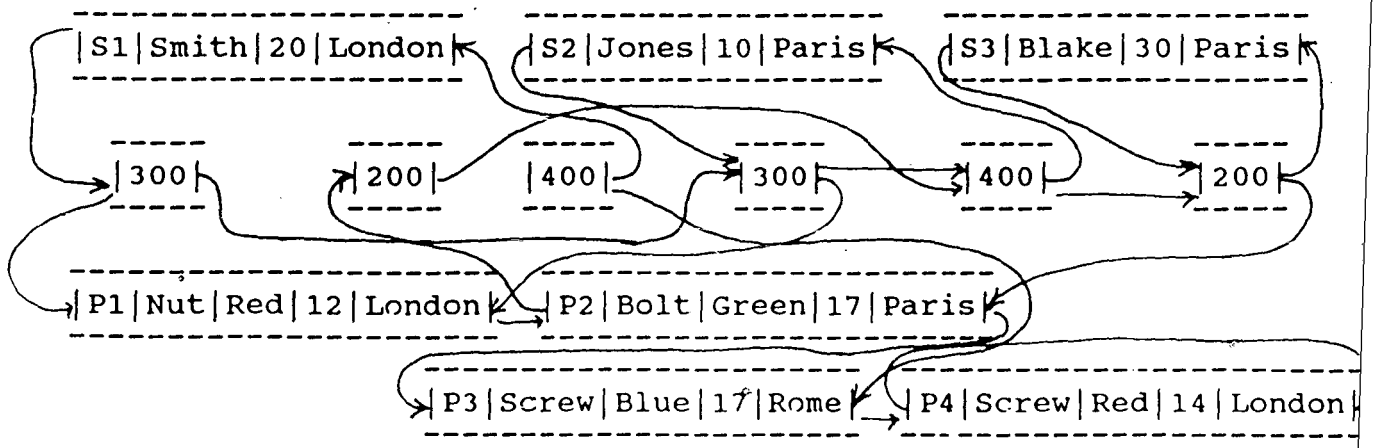

Fig. 1.1c Sample data in Network form.

Fig. 1.1 shows an organization of this same database in a network DBMS context. Just as in the hierarchical approach, the data is represented by records and links. But it is a more general structure than a tree. In a network model, a given record occurrence may have any number of superiors or subordinates, as opposed to a hierarchy where you are limited to only one immediate superior. The internal structure of a network file is even more complex than the hierarchical case.

Advantages of Relational Approach

A DBMS based on relational approach appears to possess a number of advantages over either of two types of DBMS. These advantages are noted by a number of authors throughout the database literature.

1. Simplicity and ease of use is one of the major advantages. Relational systems tend to be more user-friendly than either of the other two approaches.
2. Greater degree of logical and physical data independence. The relational model eliminates such major considerations as physical structure and position, and access paths and pointer mechanisms from the user interface. This is not possible in Network DBMS and only partially possible in Hierarchical DBMS.

3. Greater Access Flexibility. The powerful operators in the relational algebra and relational calculus offer almost unrestricted navigation through the database and permit a wider variety of applications.
4. Relational concepts are founded on the well developed theory of relations. The concept of relational completeness provides a distinct methodological approach for database design.
5. The use of normalization procedures in the relational approach allows better control on the data redundancy.
6. Relational DBMSs provide greater integrity and privacy.
7. Access philosophy is another advantage of relational approach. Both network and hierarchical systems require the user to navigate through the database via a DML. This requires the user to be familiar with the intricacies of the particular data model used. This is not necessary in the relational environment. In a relational system you simply specify what you want, while in the other two approaches you must also specify how to get it [129].
8. Another advantage of the relational approach is its relevance to the new field of database machines. The high degree of data independence afforded by the

relational model; and the regularity of the data model, makes the relational model very appropriate for hardware implementation using associative memories and parallel processors.

1.3. Database Design methodology

Database design can be defined as the process of capturing relevant information and processing requirements of an organization and mapping them into an underlying database management system [49].
Database design is typically divided into two levels - logical database design and physical database design. The logical database design is concerned with expressing the user's information requirements in a form that can be shown to be complete and correct, but which is independent of any particular implementation technology. The physical design transforms the logical design into a form that is suitable for the given hardware and the database management system. It maps the logical schema into an appropriate stored representation, and determines the physical parameters necessary to optimize the database performance against a set of required transactions [15].

Recent experience has shown that due to the growth in complexity and size of the databases, the initial

design is one of the most important factors affecting the success of a database application. For this reason a significant amount of research has been dedicated to developing methodologies for logical database design.

Relational Database Design Methodology

Relational database design has been accomplished with a variety of approaches, including the top-down, bottom-up, and combined methodologies. The traditional bottom-up approach is the universal Relational approach and assumes that all the semantics are captured by dependencies expressed over a universal set of attributes and the user does not need to be aware of how the attributes are grouped into relations. This approach leads to the well-known normalization methodology for database design. These methodologies, however, proved to be difficult to use and not entirely reliable, mainly because they required the expression of database semantics only by specifying the dependencies between attributes and because the underlying universal relational assumptions make the attribute naming critical and expressively complex. Furthermore it has been shown that inter-data-element dependencies resulting from the requirement analysis must be specified in the relational model in order to ensure proper integrity of the database and such

constraints are disregarded by normalization. At the theoretical level a top down approach has been investigated with regard to the UR assumption [86]. In practice, typically a few basic relations are defined by the requirement analysis process, and then a combination of the top-down and bottom-up approach is used. The combined approach becomes much more popular because of the introduction of a well-established conceptual design tool, the entity relationship model [30] into this process.

The Entity Relationship (ER) model has been most successful as a tool of communication between the designer and the end user during the requirements analysis and the conceptual design phases because of its ease of understanding and its convenience in its representation. One of the reasons for its effectiveness is that it is a top-down approach using the concept of abstraction. The number of entities in a database is typically an order of magnitude less than the number of data elements. Therefore, using entities as an abstraction for data elements and focusing on the inter-entity relationships greatly reduces the number of objects under consideration and simplifies the analysis. Although it is still necessary to represent data elements by attributes of entities at the conceptual level, their dependencies are normally confined to the other

attributes within the entities or in some cases, to those attributes associated with other entities that have a direct relationship to their entity. The major inter-attributes dependencies are between the entity keys of different entities that are captured in the ER modeling process. Special cases such as dependencies among unrelated entities can be analyzed upon identification in the data analysis.

This relational database design approach uses both the ER model and the relational model in successive stages. It benefits from the simplicity and ease of use of the ER model and the structure of the relational model. In this approach, the design process consists of four main phases :

1) View modeling
2) View integration
3) View translation
4) Physical design

View modeling

The first phase of the database design process is the requirements analysis phase during which the information requirements of various areas resulting in a preliminary specification of the information needs of various user groups

[15]. One of the most common approach to requirements specification is based on the concept of views. 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 set 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. A user view can be defined as the perception of users about what a database should contain [105]. According to Navathe and Schkolnick [102], there are two major tasks in the view modeling :

1. Extracting from user or from person incharge of application development the relevant parts of the real world.
2. Abstracting this information in a form that completely represents 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 [120]. Navathe and Schkolnick have also proposed their own data model, the Navathe and Schkolnick (NS) model as a vehicle for modeling user views. In addition there are three methodologies that have been developed exclusively for constructing user views. These are Bubble Charting method [94], the Interactive Specification methodology of Baldissera et al. [9], and the View Creation System of Storey and Goldstein [122].

View integration

The individual views are integrated into a common global view of the entire database. The final result of this activity is a conceptual schema that represents a global, high-level representation of all the requirements.

As shown in the Fig. 1.2, the activity of view integration can be performed at several points of database design process. It usually is performed during conceptual design. In that case, its goal is to produce an integrated schema starting form several application views that have been produced independently. According to Batini et al.[15], the reason for view integration is twofold :

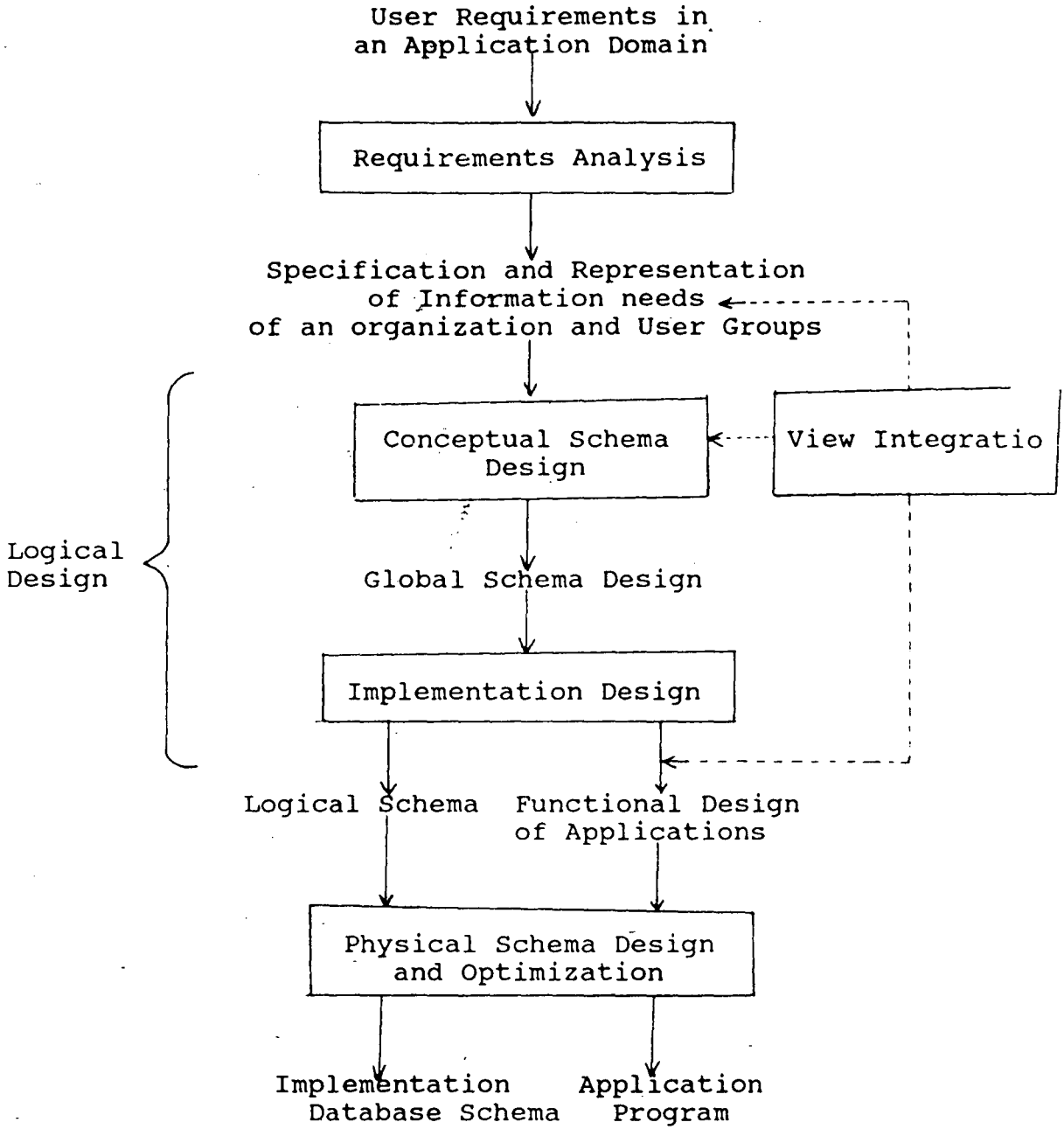

Figure 1.2. Phases of Database Design.

1. The structure of the database for large applications (organizations) is too complex to be modeled by a single designer in a single view.
2. User groups typically operate independently in organizations and have their own requirements and expectations of data, which may conflict with other user groups.

Another possibility is to perform integration even before the conceptual design step is undertaken. In this case, view integration still occurs, however views are less formal and are mostly in the form of narrative description of requirements. The last possibility shown in the Fig. 1.2 is to perform integration after the implementation design step, that is, start from schemes expressed as implementable logical schemes. This is the approach followed in the methodologies based on the relational model that do not advocate a conceptual step, and model requirements directly in forms of the relational model. Batini et al. [15] surveys a number of methodologies for both purposes.

View translation

The logical data model design produces a logical schema (a set of relations) which must be flexible enough to

support user requirements and achieve operational efficiency. On the basis of a categorization of ER constructs and a set of mapping rules, each relationship and the associated entities are transformed into a set of candidate relations. Redundant relations are eliminated. The candidate relations are normalized to the highest degree using normalization algorithms. This step is also known as implementation design because it represents the transformation of the conceptual schema into a logical schema of the database design. There is a large body of work in the literature leveled to the transformation of ER model to the Relational model. Most of the earlier works focused on the original ER model (Chen [30], Jajodia [74], Briand et al. [22], Storey and Goldstein [122], Chuang [34]). Later view translation methodologies included abstraction of the extended ER model (Teory et al. [125], Markowitz and Shoshani [92]).

Physical Design

The physical design deals with the choice of an optimal set of access structures to be built upon the relations in order to improve performance of the database.

1.4. Database Design Example

The following example illustrates the different phases of database design. The database to be

designed is for a project-oriented company. View modeling :

In this phase, first the application domain is analyzed to identify the different views for each group of users or for applications. Let us assume that there are two views identified - Personnel view and the Project view. For each view, the user's data need must be identified. In the personnel view, the user wants to be able to keep track of employees, their manager, departments worked in and the different divisions of the departments. In the project view, the user wants to identify the employees, and the projects the employees are working on. Employees can work on several projects but only in one department. The entities and the relationships that reflect the two user views are given below :

Personnel view :

Entities :

EMPLOYEE : [ID, NAME, SALARY]

MANAGER : [ID, TITLE]

DEPARTMENT : [DEPT#, DIVN#]

DIVISION : [DIVN#]

Relationships :

EMPLOYEES WORKS_IN DEPARTMENT [LOCATION]
(n) (1)

MANAGER IS_A EMPLOYEE

DEPARTMENT ID DIVISION

Project view :

Entities :

EMPLOYEE : [ID, NAME, SALARY]

PROJECT : [ID]

Relationships :

EMPLOYEES WORKS_ON PROJECT [ST_DATE]
(m) (n)

The numbers below the entity types in the relationships are the cardinalities.

View integration :

The ER diagram of the global integrated view is shown in Fig. 1.3 and the entities and relationships of the global view are given below :

Entities :

EMPLOYEE : [EMP_ID, NAME, SALARY]

MANAGER : [EMP_ID, TITLE]

DEPARTMENT : [DEPT#, DIVN#]

DIVISION : [DIVN#]

PROJECT : [PROJ-ID]

Relationships :

EMPLOYEE WORKS_IN DEPARTMENT [LOCATION]
(n) (1)
EMPLOYEE WORKS_ON PROJECT [ST_DATE]
(m) (n)
MANAGER IS_A EMPLOYEE
DEPARTMENT ID DIVISION

TH-4512

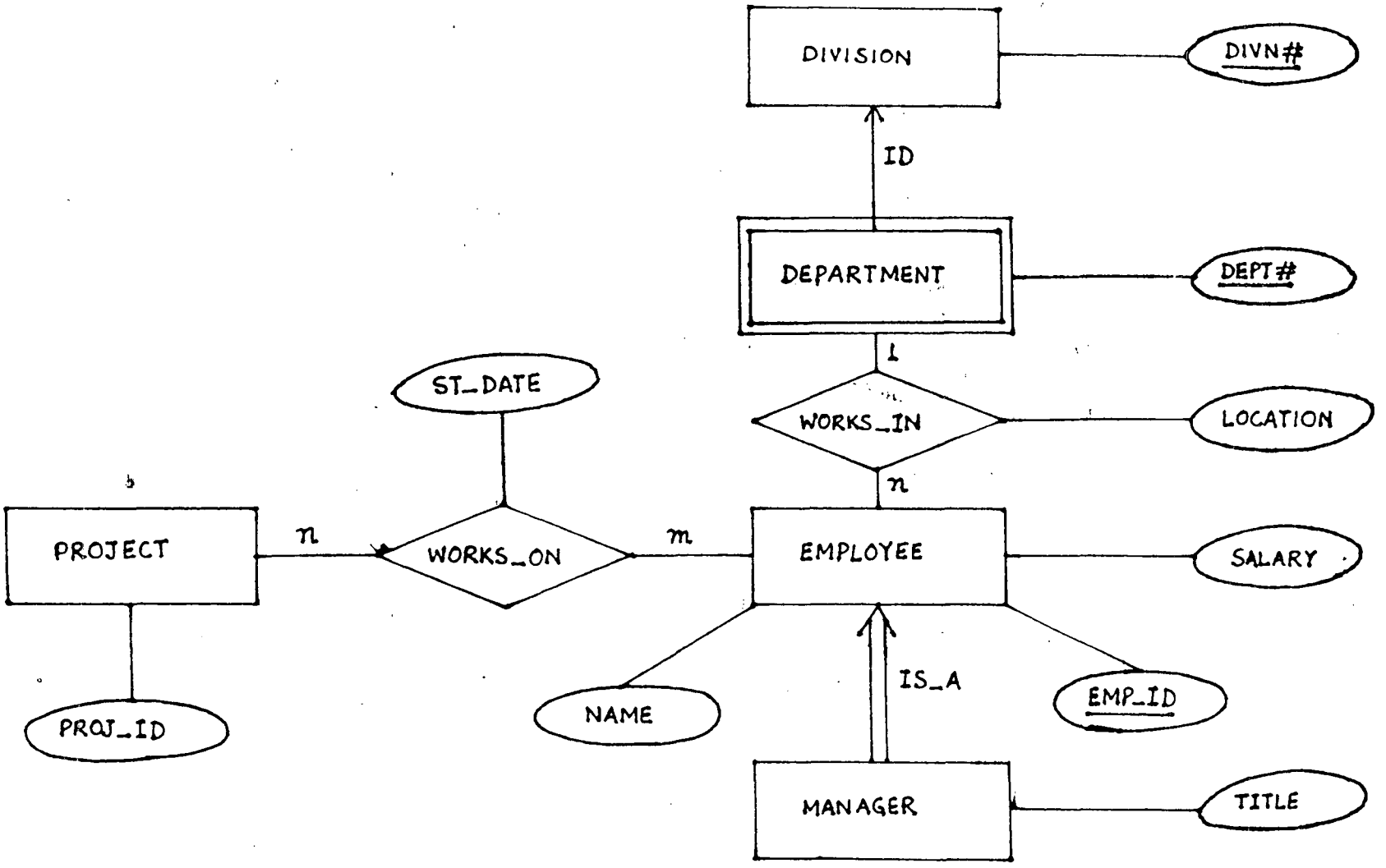

Fig. 1.3. Global Integrated View of Project_Personnel Database

View translation :

In the relational data model, this would be represented as :

EMPLOYEE (EMP_ID, NAME, SALARY)

MANAGER (EMP_ID, TITLE)

WORKS_IN (EMP_ID, LOCATION, DEPT#, DIVN#)

DEPARTMENT (DEPT#, DIVN#)

WORKS_ON (EMP_ID, PROJ_ID, ST_DATE)

1.5. The Traditional Approach To Database Design

Traditionally, database design has been carried out on an adhoc basis [20]. It has usually been performed by a database design expert who obtains information about a user's data need through interviewing, examining existing documents and systems, and other such means. The designer then integrates information requirements of various users and applications. The user and the designer then interact to validate this conceptual design, which requires several iterations of requirements analysis and conceptual design phases. When this has been completed, the designer translates the conceptual design into the logical data model required for implementation. The last step is to express the logical data model in a form compatible with the database management system software to be used.

The traditional approach to database design suffers from the following weaknesses :

1. They require the use of scarce resources - the expert database designer.
2. The designer's knowledge of an application is necessarily second-hand.
3. The quality of design produced is highly dependent upon the experience and insight of the database designer.

Database designers have observed that database design is still an art practiced by a few with tools other than the designer's experience and intuition and that in current database modeling, a designer's ability to perceive the environment and identify entities and their relationships is crucial for successful database design. It has been described that database design is a difficult task because there are so many options available while deciding both logical and physical structures. Usually, the design of a database is done by someone who is unfamiliar with the application domain. It is, therefore, the responsibility of the designer to ensure that he or she understands the application domain. Designers typically do so through interviewing users to determine their information needs for a database and then constructing the actual design.

However, users have difficulties in describing their data needs accurately and completely, and, as a result, frequently the results are misinterpreted. On the other hand, end users typically have a good understanding of their application environment, but little familiarity with database technology [78]. So the conventional approach to the design of databases has the drawback that to specify a database schema, it requires the user to have knowledge about both the domain and the data model. This is a very tedious and error-prone task both for the expert and the end users.

If the methodology of the expert database designer could be codified in a computer program, it could be replicated and applied wherever needed, alleviating the problem caused by the scarcity of human experts, and it would free the users from mastering the intricacies of the data base concepts and data models and to become the designer of their own databases. The main objective of the knowledge based system is to develop expert systems, to automate the process of database design.

Knowledge based approach to database design

The knowledge based systems are computer programs, based on Artificial Intelligence techniques, which embody

some aspects of human knowledge and expertise and perform tasks ordinarily done by human beings.

The knowledge-based approach to designing a system for any kind of task starts by asking what knowledge (i.e. what facts and reasoning abilities) is used by a human expert in solving this task. This knowledge is then encoded in data structures and procedures that represent the knowledge explicitly, and that are separate from the inference procedures that apply this knowledge to solve problems.

Developing a knowledge-based system for database design involves constructing a set of rules that summarizes various design phases, as well as a general rule interpreter that applies combination of these rules to solve database design problems.

This approach has been found to have several interrelated advantages over traditional techniques for organizing software as follows :

- It is easier to make incremental improvements. Since the knowledge is represented explicitly, separate from the code that applies the knowledge to solve problems, the system's capability can be incrementally extended by adding to the knowledge base.

- It is easier for the system to explain what it is doing and why.
- It is easier for the human expert to determine what is incorrect or incomplete about the system's knowledge, and explain how to fix it.
- It is easier to interactively use a human expert.
- It accepts incomplete specifications.
- It interacts with the human user with an easy-to-use interface.

Some of the important expert systems that have been developed for database design problems are : The Interactive Specification methodology which attempts to automate the generation of user views; the View Creation System, which automates the process of generating and specifying user views; and the GESDD - a Generalized Expert System for Database Design [49], which is an intelligent database design tool used both for developing a design methodology and in applying this methodology to design a database schema for an enterprise, from the requirements specification phase to the logical data model design phase.

1.6. RESEARCH OBJECTIVE

The basic objective of this research is to

develop a rigorous methodology for robust relational database design, and to implement the methodology by developing an Expert System - View modeling, Integration and Translation System (VMITS), to completely automate the process of database design.

The research investigates the fundamental components of what is commonly referred to as Logical Database Database Design in the context of relational system. It analyzes the reasoning process carried out by the expert database designer during the design process in order to reproduce them in the expert system. Thus, it formalizes, on the one hand, the algorithmic part of the design (for example, algorithm of normalization) and on the other hand, the heuristics (experimental rules of the designer) to produce a rule set which will form the knowledge base of the expert system VMITS.

1.7. Importance of the study

The VMITS formalizes a methodology, as a set of rules, which forms the knowledge base of the expert system. The methodology produces database design that are not only accurate representations of reality, but flexible enough to accommodate future processing requirements. It also reduces the number of data dependencies that must be analyzed, using extended ER model

conceptualization, and maintains data integrity through normalization.

The Interactive Specification methodology [Baldissera et al., 1979] is an automatic tool for generating user views, but it has not addressed the view-specification process. The VCS [Storey and Goldstein, 1988] is an expert system which addresses the view modeling and view translation phases of the relational database design. The generalized expert system for database design [Dogac et al., 1989] is an expert system for generating the methodology and using the methodology for database design, however, the system is intended to aid, as opposed to replace, a database designer, so it again requires the use of scarce database design expert. Hence, the need remains for automation of the database design process to deal with all the three phases of logical database design, that is starting with the requirements analysis and view modeling, the view integration and conceptual schema design, the view translation and the logical schema design to produce a set of fifth normal form relations.

1.8. ORGANIZATION OF THE THESIS

The presentation of the research is organized in the following manner :

Chapter II gives an overview of the extended entity_relationship model which forms the basis of the

methodologies for logical database design.

In chapter III, the view modeling methodology based on the EER model is described.

Chapter IV discusses the view integration methodology.

In chapter V, a well_formed EER model based on the graph theory and the normalization principles of relational model is described and an algorithm for translating a well_formed EER schema into fifth normal form relational schema is given.

Chapter VI describes the architecture and the implementation of the Expert system VMITS.

In chapter VII, the conclusion, importance and the directions for future development of the research, are presented.

The Turbo Prolog program for the expert system is given in the Appendix-A, and, an illustration of the design session using the system is described in Appendix-B.