

BUILDING AN EXPERT DATABASE SYSTEM  
IN C USING CLIPS AND PARADOX

BY


DIPTI R. BHARGAVA  
Bachelor of Science  
Nagpur University  
Nagpur, India


1986

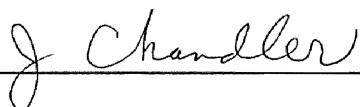
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
July, 1993

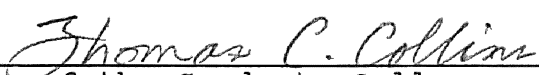
BUILDING AN EXPERT DATABASE SYSTEM  
IN C USING CLIPS AND PARADOX

Thesis Approved:

  
\_\_\_\_\_  
Thesis Advisor

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_  
Dean of the Graduate College

## ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my parents, to whom I dedicate this thesis, for having given me the opportunity to pursue higher studies in the US. Their love and support enabled me to accomplish this mission. Thanks also to my brothers, sisters and the rest of my family for their great love and moral support.

Special thanks to my dear friend Ravi without whose constant nagging this thesis would never have been completed. Gratitude is due to my friends Shashi, Anu, Ganesh Sundaram, Mr. Larry Watkins of UCC, Elaine Burges and Regina Henry of ISS.

My special thanks to my manager Maryanne Deaton at United Airlines for being so considerate and understanding during this effort. I also extend my thanks to my colleagues for their encouragement.

My thanks to Dr. M. Samadzadeh for his valuable suggestions during this rather difficult process.

Last but not least my thanks to my husband for being on my side.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION . . . . .	1
Statement of the Problem. . . . .	3
Proposed Thesis . . . . .	4
II. LITERATURE REVIEW. . . . .	5
Expert Systems. . . . .	5
Expert System Facilities. . . . .	7
Analysis of Knowledge . . . . .	9
Characteristics of an Expert System . . . . .	10
Database Management System. . . . .	10
Architecture of a DBMS. . . . .	12
Types of Databases. . . . .	13
Hierarchic DBMS. . . . .	13
Network DBMS . . . . .	13
Relational DBMS. . . . .	13
Intelligent Database Systems . . . . .	15
Main Memory Databases. . . . .	16
Expert Database System. . . . .	17
III. SYSTEM DESIGN AND IMPLEMENTATION . . . . .	19
Introduction to Paradox Engine. . . . .	19
CLIPS . . . . .	20
Organization of Data and Knowledge. . . . .	22
Interaction Between the Application, . . . . .	
CLIPS and Paradox . . . . .	26
Query Interface . . . . .	28
Guidelines in Constructing Query. . . . .	28
Parsing of Queries. . . . .	29
IV. SUMMARY AND CONCLUSIONS. . . . .	30
Adaptability to a New Knowledge Domain. . . . .	31
Limitations of the Current System . . . . .	32
Future Work . . . . .	33
BIBLIOGRAPHY. . . . .	35

Chapter	Page
APPENDIXES. . . . .	38
APPENDIX A - LIST OF TABLES. . . . .	39
APPENDIX B - LIST OF FIGURES. . . . .	45

## LIST OF TABLES

Table	Page
1. Fields of Flights.db . . . . .	40
2. Fields of Acbasic.db . . . . .	40
3. Fields of Acengine.db. . . . .	41
4. Fields of Acdimen.db . . . . .	41
5. Flight Data Stored in Flights.db . . . . .	42
6. List of Noise Words used by the Query Parser . .	42
7. Aircraft Engine Data Stored in Acengine.db . . .	43
8. Attribute Values Stored in Acdimen.db. . . . .	44
9. Basic Aircraft Data Stored in Acbasic.db . . . .	45

## LIST OF FIGURES

Figure	Page
1. Logical Components of a Knowledge-Based System . . . . .	47
2. Basic Concept of an Expert System. . . . .	48
3. Logical Architecture of a DBMS . . . . .	48
4. Flowchart Depicting Overall Algorithm. . . . .	49
5. Flowchart of Query Processing. . . . .	50

## CHAPTER I

### INTRODUCTION

The term 'expert database system' has been used to represent the confluence of concepts, tools and techniques from a number of areas including artificial intelligence, database management system, logic programming and information retrieval.

Computerized databases are essential and inseparable components of the vast majority of contemporary information systems. Many such systems utilize general-purpose tools, called database-management systems (DBMS), to provide an efficient and uniform access to, and control of, consistent information across single-user or multi-user environments[18]. A database management system is a generic tool in the sense that it is intended to support different kinds of databases, for a variety of application environments[18].

The functional capabilities of a database management system include:

1. support the independent existence of a database, apart from the application programs and systems that manipulate it[25];



2. provide a conceptual/logical level of data abstraction[34];
3. support the query and modification of databases[26];
4. accommodate the evolvability of both the conceptual structure and the internal organization of a database in response to changing usage and performance requirements[10];
5. control a database in terms of semantic integrity, security, concurrence, and recovery[34].

The evolution of the database management system is in a sense analogous to the development of abstract data types in high-level programming languages: the goal is to provide general-purpose mechanisms that support a higher level of abstraction for application designers, implementors, and users[33]. Given a traditional database as a platform to store, organize, control, and access a database, the database designer should focus on creating application software to act as an interface between the user and the database management system[28].

In the recent past, database-intensive application areas have emerged other than those for which traditional DBMS facilities were intended. Such areas include computer-aided design[30], information and document retrieval systems[25], legal and environmental systems[7] etc. In order to help end users express their data manipulation requirements, database management systems provide a

user-friendly interface such as SQL to end users[32]. These modern applications operate on a huge central database, shared by many "clients", and require a more "intelligent" way of data manipulation[18]. Apart from the standard database operations like insert, delete, update, and viewing records, these modern applications require the data management system to logically deduce new facts from existing data, and respond to a wider varieties of queries. In other words, these systems not only store values, but "chunks" of knowledge about the relationships between the various facets of data[8]. Adding logic capabilities to a database system provides an efficient way of dealing with facts and general rules. General rules used to represent knowledge is the application's domain at a high level of abstraction from a powerful modelling tool, as opposed to data representing specific facts that are handled by conventional database systems. Rules are easier to input, occupy less space, and are easier to change[22]. In an effort to meet modern database requirements, research is now focussing on these "intelligent" database systems termed "Expert Database Systems".

#### **Statement of the Problem**

As mentioned before, an expert database system consists of a combination of the storage/retrieval capabilities of a database management system and the inferring capabilities of an expert system. This could be achieved by:

1. incorporating inferring capabilities into a DBMS;
2. embedding the data into the knowledge base of an expert system;
3. developing an interface between the expert system and the database management system.

All the above methods have their advantages and disadvantages. The problem then can be stated as follows: Is there a way to develop an integrated expert database system based on the combination of the above three techniques, using existing tools and systems?

#### **Proposed Thesis**

This thesis presents a methodology to develop an integrated expert database system using CLIPS (a portable inference engine with C interface) and PARADOX (a portable networked database management system, again using C language interface) on a PC based environment. It demonstrates the methodology by applying it to a sample aircraft database.

## CHAPTER II

### LITERATURE REVIEW

#### Expert Systems

Expert systems or Knowledge-based systems is a branch of artificial intelligence that makes use of specialized knowledge to solve problems at the level of a human expert[23]. The British Computer Society's Committee of the Specialized Group on Expert Systems has produced the following definition for an expert system or knowledge-based system application:

'the embodiment within a computer of a knowledge-based component from an expert system shell in such a form that the machine can offer intelligent advice or take an intelligent decision about a processing function'[14].

The Butler Cox Foundation Report number 37 defined an expert system as:

'a computer system containing organized knowledge, both factual and heuristic, that concerns some specific area of human expertise; and that is able to produce inferences for the user'[14]. Professor Edward Feigenbaum of Stanford University has defined an expert system as '...an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult

enough to require significant human expertise for their solution'[2]. To do so, it emulates the decision-making ability of a human expert. The term emulates means that the expert system is intended to act in all respects like a human expert[23]. Figure 1, in Appendix B, depicts the logical components of an expert system.

The fundamental component of a expert systems is the knowledge-base. The knowledge-base contains the information required to emulate expertise, that will include some facts or data, and rules which express how the facts can be evaluated.

The inference engine operates on the knowledge-base and applies laws of logical inference or reasoning to control the flow of making deductions and drawing conclusions. That is, it causes the facts to be instantiated while attempting to reach goals and conclusions. All of the facts may not be instantiated; only those which are required as part of the inferencing will have values. A number of control strategies such as forward chaining and backward chaining may be associated with the inference engine[16].

The working store, or memory, is used by an expert system to store transient and dynamic information, such as data values, which facts have been instantiated, which rules have been fired, etc[20].

The user interface provides the link with the outside world. The knowledge acquisition subsystem is used by the knowledge-based system application builder to add facts and

process details to the knowledge-base. Some trace and debugging aids are usually provided to help validate and test the system.

The basic concept of an expert system is illustrated in Appendix B, figure 2.

The user supplies facts or other information to an expert system and receives expert advice in response. Internally, the expert system consists of two main components. The knowledge-base contains the knowledge with which the inference engine draws conclusions. These conclusions are the expert system's responses to the user's queries.

### **Expert System Facilities**

The following are the facilities provided by an expert system:

1. **Rules:** provide the most common format for specification of knowledge and knowledge processing in a knowledge-base of an expert system application. Rules are declarative, stating what is to be done and not how it is to be done.
2. **Inference Mechanisms:** The inference engine provided with each expert system tool is complex and employs a number of strategies for managing the use of knowledge-base components at run-time. The strategies determine the order in which the processing will be carried out. That is how the

rules will be invoked and if there is a conflict, which rule will be fired and when it will be fired.

Forward chaining, or data driven inferencing occurs when an attempt is made to reach a goal or conclusion using facts which have already been evaluated. All facts which can be derived, directly or indirectly, from some known facts are evaluated in an attempt to derive a goal value. This mechanism is commonly used when there is a very large number of possible solutions, given a large combination of possible fact values.

Backward chaining or goal driven inferencing occurs when an attempt is made to evaluate a conclusion or goal by resolving or evaluating a limited number of facts relevant to a line of reasoning. Backward chaining is commonly used when values of some key facts eliminate the need to evaluate a number of other facts, and when there are fewer possible goal values.

3. Demons, or event driven actions: are available to be executed at any time and can interrupt processing. A condition associated with each demon determine when it should be executed by the inference engine.
4. Hypothetical Reasoning: Some expert system tools allow a number of options to be explored

simultaneously in order to arrive quickly at a solution. Multiple hypothetical situations can be explored and the software maintains a number of states.

5. **Backtracking:** is a feature of Prolog or Prolog systems. Such systems instantiate facts once and use this value whenever the fact is referenced, unless instructed to find an alternative value for a fact, in which case backtracking may be used. Backtracking consists of reviewing what has been done (in terms of satisfying goals) and attempting to re-satisfy the goals.

### **Analysis of Knowledge**

There are many components of the knowledge that is the source of an expert's ability to perform[5]. They may be viewed generally as:

1. **Facts:** are statements that relate some element of truth regarding the subject domain, for example: A Boeing 747 has four engines.
2. **Procedural rules:** are well-defined rules that describe fundamental sequence of events and relations relative to the domain[31].
3. **Heuristic Rule:** are general rules that suggest procedures to be followed when procedural rules are not available[12]. The presence of heuristics contributes greatly to the power and flexibility



of expert systems and tends to distinguish expert systems from traditional software[9].

### **Characteristics of an Expert System**

An expert system is usually designed to have the following general characteristics:

1. **High Performance:** The system must be capable of responding at a level of competency equal to or better than an expert in the field[21];
2. **Adequate Response Time:** The system must also produce solutions in a reasonable time, comparable to or better than the time required by an expert to reach a decision[14];
3. **Good Reliability:** The expert system must be reliable and not prone to crashes[23];
4. **Flexibility:** Because of the large amount of knowledge that an expert system may have, it is important to have an efficient mechanism for adding, changing, and deleting knowledge[21].

### **Database Management System**

There are numerous definitions of a database, including the following given by James Martin;

"A database is a collection of inter-related data stored together with controlled redundancy to serve one or more applications in an optimal fashion; the data is stored so

that it is independent of programs which use that data"[1,14].

The data is controlled by a database management system which is a proprietary software for handling the storage and retrieval of data. One can define a database management system as a software tool for developing applications requiring access to shared information[14]. A database management system provides the following:

1. data independence. The DBMS separates specification of processing from the data. Processing works on the logical view of the data and the DBMS maps the logical view on to the physical representation of the data[6];
2. data integrity. Checks on the consistency of the data[14];
3. data concurrency and consistency. Since the database is a shared resource the DBMS allows a number of users to access the database concurrently. The DBMS can inhibit concurrent access to the same data instances by setting locks on the data[29];
4. recovery. The DBMS logs changes made to the database by all users; if a user aborts the current transaction then all changes made within it are automatically undone by the DBMS. A DBMS also provides facilities to backup the database[14];

5. access control. The DBMS controls who can access what parts of the database and in what mode. Access control is important when all data is held as one logical unit[14];
6. data maintenance. Facilities to unload and reload, reorganize and re-structure data are generally supplied with a DBMS[14].

### **Architecture of a DBMS**

An overview of the architecture of a DBMS is shown in Appendix B, figure 3. There are three logical views of the data in a database.

1. Internal View: defines the physical structure of the data, specifying where records should be placed, clustered, etc., across physical files and how they should be physically accessed, such as by using an index or a serial search[34].
2. Global View: provides a logical view of the database, specifying such things as record layouts, data item definitions. There is a mapping between the internal view and the global view[34].
3. External View: provides an access control mechanism for processing, since only those data items which are essential for an application to function are included in the definition to be used. All the other data items and record types,

which are not included in the view, are hidden and unavailable to the application[34].

Users and applications access data through an external view using a DBMS specific language, generally termed a data manipulation language. Some databases have a language which has unique syntax, but most relational databases have standardized on the database language SQL.

### **Types Of Databases**

This section describes some of the kinds of databases.

Hierarchic Database. This DBMS has a number of data units organized in tree structures, each data unit has one and only one owner, but it may have one or more member units. IBM's Information Management System/Virtual Storage (IMS/VS) is an example of such a DBMS[14].

Network DBMS. Network DBMS implements a network view of the data. Records can be inter-connected in general networks as well as hierarchies, thereby providing a more flexible structure[14].

Relational Database. Relational DBMS is perceived to hold data in a series of two dimensional rows (record occurrences) and columns (attributes)[3]. Relationships between rows in different tables are represented by the storage of attributes from other tables within a table. Now tables can be formed by selecting rows or columns from

existing tables or by joining tables[3]. As a canonical example of a relational database, consider the database, named COMPANY, of an enterprise where information is kept about employees and the departments in which they work. Such a database may consist of the following relations:

```
EMP(eno, ename, age, salary, edno);
```

```
DEPT(dno, dname, floor, mgrno)
```

EMP and DEPT are two relations, with five and four attributes respectively, whose intended meaning is rather straight forward from their names. In this database, the EMP relation will contain one tuple for each employee and the DEPT relation will contain one tuple for each department of the enterprise. Queries on this database may involve one relation, where operations belong to {=, <>, <=, >=} for example salary > 30K or may combine two relations for example edno = dno.

Thus the relational database theory incorporates

1. a model of data;
2. an algebra for manipulating that data;
3. a calculus for expressing requirements of 1 and 2.

Each relation can be supported by at most one primary index and an arbitrary number of secondary indices, which are built based on the values of the relation tuples for some attribute(s)[36]. Such a variety of indices give the system the ability to accelerate the processing of queries. Oracle and Ingres are two of the best known relational database products.

Intelligent Database Systems. There are essentially three ways to provide intelligent database system capabilities. These are:

1. add logic/inference capability to a DBMS;
2. couple together a database system with a logic-programming system;
3. add DBMS facilities to a logic-programming system.

Research work has tackled all three possibilities.

Option 1 involves the addition of an inference engine and additional data constructs to a DBMS.

For second option consider the Prolog system. A Prolog system uses an existing DBMS a backend server. The methodology is to modify the processing of a Prolog program so that it collects together, unevaluated subqueries that access data stored in the relational DBMS. The subqueries are then translated into the query language of the DBMS and sent to it for processing.

Option 3 would require significant amounts of development effort since the natural data constructs associated with logic programming do not lend themselves to efficient handling of simultaneous multi-user access.

Main Memory Databases. In a main memory database system data resides permanently in main physical memory in a conventional database system it is disk resident[11]. In a conventional database system, disk data may be cached into memory for access; in a main memory database system the

memory resident data may have a backup copy on disk. So in both cases, a given object can have copies both in memory and on disk. The key difference is that in main memory database system the primary copy lives permanently in memory[11].

As semiconductor memory becomes cheaper and chip densities increase, it becomes feasible to store larger and larger databases in memory, making main memory database systems a reality. Because data can be accessed directly in memory, main memory database systems can provide better response times and transaction throughputs, as compared to conventional database systems[11].

A concept used in rapid handling of large volumes of textual data was proposed by Marguerite F. called KWIC and KWOC by the author. This method is based on indexing of essential keywords in the data instead of concepts[17]. A KWIC (keywords in context) index uses essential keywords found in the current context. A KWOC (keywords out of context) index lists essential keywords used out of the current context and is designed to complement the KWIC index[15].

### **Expert Database System**

An expert database system(EDBS) can be defined as a system for developing applications requiring knowledge-directed processing of shared information[13]. In Freundlich's view an expert database system is a database

that stores not only values but "chunks of knowledge" [8]. Zari describes traditional database management systems as passive, that is queries on transaction are executed only when explicitly requested. On the other hand, he describes a database augmented by transaction-triggered processing capabilities as an "active database, and he classifies expert database system into the category of active databases[34].

An expert database system architecture comprises of two major components. It involves a data-management system, and an expert system[9,27]. The data concerning the application resides in a database, the storage and retrieval of which is handled by the database management system. The expert system is used to perform intelligent processing of this data.

An expert database system supports applications that require knowledge directed processing of shared information. This definition allows us to envision a wide spectrum of architectures for such systems. They may be loosely coupled as for e.g, an expert system which retrieves data from a database. They may be tightly coupled in that either the expert system or the database system or both "understand" how the other functions and can take advantage of their knowledge to improve the performance of the expert database system[13]. In its most advanced form a tightly coupled system may embody, in an integrated fashion, characteristics found in both the expert systems and the database management



systems. Loose coupling means that both the expert system and the database management system will maintain their own functionality and will communicate through a well-defined interface. The database may be viewed as a data server for the expert system. For example, an expert system may send SQL queries to the database system to obtain needed data. Conversely, the database system might send messages to the expert system, placing data onto its blackboard or working memory. In addition, the database management system could pose questions to the expert system in much the same way a user might consult an expert system[13].

Generis is an integrated system encompassing database, expert system technology. This is a commercially available product. Generis has evolved from the FACT System which is claimed to be the world's first Intelligent Knowledgebase Management System, that uses the Generic Associative Technique to 'combine the power of relational database technology with user-defined inference and action rules which are held in the same database'. Data can be viewed in tables or in frames based on a subject. The system links all tables in an application into a network.

## CHAPTER III

### SYSTEM DESIGN AND IMPLEMENTATION

In this chapter, the design and implementation of a composite Expert Database System is presented. As mentioned before, the basic tools used are as follows:

1. Paradox relational database management system: used as a repository for context independent data;
2. Paradox ENGINE: used as a C language interface to the Paradox database management system;
3. CLIPS (ver. 5.1): used as the forward chaining inference engine to deduce context-based facts from rules defined in the knowledge base;
4. C programming language.

The methodology is demonstrated by applying it to a sample aircraft database. In the present work, twelve flights and fourteen aircraft are considered. This is solely with the aim of demonstrating the methodology and is not to be considered a limitation on the size of databases the methodology can handle.

#### Introduction to Paradox Engine

Powerful database applications have historically been difficult and slow to learn and use[19]. Paradox provides a

contradiction to this by proving to be powerful and complex yet simple and easy to learn. Paradox(ver. 3.5) is a full featured relational database management system, made commercially available by Borland International. It operates both in a single-user stand alone system and in a multi-user networked environment.

Paradox 3.5 also has the capability to provide access to data stored on Structured Query Language(SQL) database servers through the SQL link. This could be an extremely useful feature for the development of Expert Database Systems.

Paradox Engine is a comprehensive library of C functions and Pascal procedures and functions that can be invoked by application programs written in C and Pascal. The Paradox Engine also allows the manipulation of data residing in Paradox tables in both single-user and multi-user environments.

For more information on this refer to the Paradox manuals[19].

### CLIPS

CLIPS is an acronym for C Language Integrated Production System. CLIPS is a forward chaining rule based language that has inferencing and representation capabilities. CLIPS was designed at NASA/Johnson Space Center. It was designed to be highly portable and easily integrable with external systems at a relatively low cost.

Because of its high portability CLIPS has been installed on a wide variety of computer systems ranging from PC's to CRAY supercomputers[23].

CLIPS comprises of three basic elements:

1. fact-list: global memory for asserted facts;
2. knowledge-base: rules governing the facts;
3. inference engine: controls how knowledge is manipulated and facts are inferred.

CLIPS is based on a very fast pattern-matching algorithm known as the RETE algorithm[24]. The basic premise of this algorithm is outlined below.

The inference engine typically makes inferences by deciding which rules are satisfied by facts in the current context, prioritizes the satisfied rules, and executes the rule with the highest priority. A rule is said to be satisfied when the pattern on the LHS of a rule match with existing facts. The other satisfied rules are placed in a list called the agenda.

As each of the rules on the agenda are executed, new facts may be created and/or old facts deleted. This in turn might lead to some rules which are not on the agenda to be satisfied and some rules on the agenda to be unsatisfied. The list of satisfied rules must therefore be dynamically maintained and updated in an efficient manner.

If a rule is satisfied the entire process of matching facts and rules is repeated. This method in which the rules

dictate the search process for facts, becomes very inefficient as the number of rules increases.

The Rete's algorithm is based on the paradigm of facts directing the search process for rules that are satisfied. Typically very few facts in an expert system change in one execution cycle, which in turn implies that only a small subset of rules would be affected by the changes in facts. This property is known as Temporal Redundancy[24]. The Rete's algorithm exploits this property by saving the state of the matching process from cycle to cycle and recomputing the changes in this state only for the changes that occur in the list of facts.

### Organization of Data and Knowledge

Context-independent data about aircraft configuration and flights are stored in Paradox database files in the form of tables. The rules which govern this data and help in inferring context-dependent facts are stored in a knowledge base file.

Two different classes of data are stored in the Paradox files as four tables. The first class stored in "Flights.db", consists of the fields shown in table 1, Appendix A. It contains information about different Flight Numbers and the class of flight. Allowable classes are 'A', 'B', and 'C'. If a flight's class is 'A', then aircraft which can carry more than 350 passengers and cargo weighing more than 100,000 lbs can execute these flights. Aircraft

which accommodate more than 150 and less than or equal to 350 passengers, while carrying cargo loads between 40,000 lbs and 100,000 lbs can execute flights of Class 'B'. All other aircraft can execute flights of Class 'C'. It is worth noting that Flight numbers are unique and they form the primary search key for this table. Also, this table of records does not store any references to aircraft.

The other class of data pertains to different types of Aircraft and their configuration parameters. These parameters ( totally 22 in number ) are distributed into three tables, namely " Acbasic.db", " AcEngine.db " and " Acdimen.db ". As the names of these tables suggest, basic parameters like Passenger and Crew capacities, maximum flying altitudes etc., are stored in " Acbasic.db ", engine type, number of engines, etc., are stored in " Acdimen.db ". The field names and types for each of these tables are presented in Appendix A, tables 2 through 4. Note that in all the three tables, the field " Aircraft-type " is present as the Primary key. This is to facilitate the relational linkage of the three tables.

The actual data stored in each of these tables is depicted in Appendix A, tables 5, 7, 8 and 9.

Functions that interact with the Paradox engine are implemented in a systematic way as outlined below:

1. The Paradox engine is started and initialized;
2. The desired table is opened;
3. A record buffer is dynamically allocated;

4. Fields of the record needing access are defined;
5. Necessary operation(s) are carried out;
6. The record buffer is freed;
7. The opened table is closed;
8. The Paradox Engine is shut down.

The abstract rules which govern the inferencing capabilities of the Expert Database system are stored in a knowledge base file. CLIPS requires that these rules (and certain "facts") be stored in file with a ".CLP" extension. Facts in the context of knowledge base of a regular expert system are "chunks" of information. However, those facts that are context-free are stored in the database of an EDBS as far as possible. Facts that are dynamic in nature and are added or removed in a specific context (agenda) are used in a knowledge Base of an EDBS. A fact in CLIPS consists of one or more fields enclosed in matching left and right parentheses. Good programming style dictates that facts be represented as (<relation name> <value1> .....<valueN>) which explicitly declare the relationships between various values.

An example follows:

```
(is-class-of-aircraft B)
```

Rules in any EDBS are necessary to infer new facts from the data in the database. CLIPS requires that these rules follow the following syntax :

```
(defrule <rule name>    [<optional comment>]
<<patterns>>          ; Left - Hand Side (LHS) of the rule
```

```

=>                ; THEN arrow
<<action>>        ; Right - Hand Side (RHS) of the rule

```

The group of all facts known to CLIPS is stored as a list known as a fact-list. New facts can be added or removed from the fact-list using the CLIPS function `assert` or `retract`, on the RHS of the rule, respectively.

Variables within CLIPS rules are always written in the syntax of a question mark followed by a symbolic field name. The 'bind' function can be used to bind the value of a variable to that of an expression.

Comments in a rule begin with a semicolon and can continue until the end of the line.

An example of a simple rule follows:

```

(defrule calc_cargo_volumes "Calculate Cargo Volumes"
; IF forward and rear cargo volumes of an aircraft is
; known
(vol_of_cargo_compts ?aircraft ?forward vol ?rear vol)
=> ;THEN
; Assert total cargo volume as sum of forward and rear
; cargo volumes
(bind ?total_vol (+ ?forward_vol ?rear_vol))
(assert (total_cargo_volume ?aircraft ?total_vol)))

```

In the present work, CLIPS is embedded within the application. Calls to CLIPS are made like any other function call. Embedding CLIPS involves the following steps.



1. The application provides a main program, and includes "stdio.h" and "clips.h" in the main program file;
2. CLIPS is initialized by the main function prior to loading rules;
3. Application defined functions which are to be called by CLIPS, are also made known to CLIPS in the main program file;
4. The header files are customized to the applications operating environment(including memory usage, specialized functions, etc.);
5. The applications code is then compiled and linked with all CLIPS files except the object version of the CLIPS main program file.

#### **Interaction Between the Application, CLIPS and Paradox**

The flowchart depicting the overall algorithm of the application is presented in Appendix B, figure 4. Since CLIPS is used in an embedded form, CLIPS is initialized as the first step in the main function. The rules in the knowledge base file are then loaded. The main menu is displayed. The main menu entries are classified into two categories:

1. Entries to view, insert or delete records in the conventional database;

2. Entry to allow users to pose queries to the inference engine and get solution, if any exist.

As already mentioned, entries in category 1 above bypass CLIPS and interact directly with the Paradox Engine functions and the Paradox data files. Deletion of a record in the database for example, would involve determining the primary key identifying the record and calling the appropriate function to delete the record.

The query entries in category 2 interact with CLIPS which, if required, interact with the application functions that deal with the Paradox engine and the Paradox database files. The following example outlines the interactions involved in a typical query solving process.

Assume that the user poses the query "Compare the engine weights of Boeing 747 and DC10". Processing proceeds as follows:

1. Based on the query, facts are asserted in CLIPS. In this example, the fact "comp-attr engine-weight DC10 BOEING-747" is asserted in the CLIPS knowledge base;
2. CLIPS inference engine is then invoked;
3. Depending on the rules in the knowledge base and currently available facts, rule(s) are fired. In this example, one of the rules fired would have the following pseudo-code:

```
(if(comp-attr ?attr ?a1 ?a2) is defined then
  get-value of attr of ?a1 as ?val1
  get-value of attr of ?a2 as ?val2)
```

```
    if ?val1 > ?val2 then
        print ?a1 is more than ?a2
    else
        print ?a2 is more than ?a1
);
```

4. If Paradox functions need to be called by CLIPS during the inferencing process, they are invoked. In this example, the firing of the above rule involves the invocation of Paradox function to determine the engine weights of DC10 and Boeing 747, this is done through the use of application's function 'get-value';
5. The solutions, if any, are printed. In our example the reply, "BOEING 747 has more engine weight than DC10" would be printed to the screen.

### Query Interface

The ease with which a user can query an Expert Database System helps determine its widespread usability. Ideally, users would prefer to pose queries in a natural language like English without having to phrase their queries in a specified manner. Flowchart for query parsing is as shown in Appendix B, figure 5.

The query parsing mechanism implemented for the present system allows users to pose queries in a restricted form of the English language.

### **Guidelines in Constructing Query**

The user poses a query to the EDBS in plain English in the context of aircraft based on the following guidelines:

1. Queries can be in upper or lower case;
2. Queries should not contain any characters other than alpha-numeric characters;

### **Parsing Of Queries**

When a query is specified by the user, the keyword in the query string is determined and removed. The noise words in the query string are then eliminated. The noise words were chosen such that parsing of the query is simplified. These words can be typically categorized as words not adding any information to the query understanding. Table 6, in Appendix A, lists the noise words in the present system.

This query is then parsed by a Finite State Recognizer. As the query is parsed, a query structure identifying the relevant query construct is built.

## CHAPTER IV

### SUMMARY AND CONCLUSIONS

An Expert Database System(EDBS) for a sample aircraft database was successfully developed and tested, using the methodology described in chapter III. The system was developed using the C programming language in a single-user PC environment. Context independent data about aircraft was stored in Paradox database files. The Paradox engine was used to develop the interface between the Paradox database files and the system. CLIPS (ver 5.0) was used for its efficient inference engine. Rules constituting the EDBS's knowledge base were stored in a CLIPS knowledge base file.

The main features of the methodology adopted in developing the above system are:

1. the use of a widely-used conventional programming language like C;
2. the use of a highly versatile relational database management system like Paradox, with conventional programming language interface to the database files (like Paradox engine);
3. use of an efficient inference engine like CLIPS;
4. use of an efficient query-parsing mechanism (designed to interpret queries posed in a natural

language such as English) independent of the relational database system or the inference engine;

5. enmeshing the inference engine with the database management system.

The EDBS has a menu-driven user-interface. It allows the user to insert, delete, or view records in the relational database files. It also allows users to pose queries in English. An ambiguous query causes the system to prompt the user for clarification as far as possible.

#### **Adaptability to a New Knowledge Domain**

The methodology used in developing the present system could easily be used in developing an Expert Database System for any knowledge domain. However to make the system more efficient, some code segments are domain-specific. Major modification of this system to operate in a different knowledge domain would involve the following changes:

1. Creating the Paradox database files and storing context-independent data about the domain in these files;
2. Modifying the functions, written using Paradox engine's routines which assist in interfacing with the Paradox database files. This is necessary since the functions take into account the type of data stored in the database files, like int, char strings, etc;

3. Developing the CLIPS rule base for the desired knowledge domain;
4. Changing the CLIPS-Paradox interface functions to be useful with the new interface;
5. Modifying the query parsing structure and functions to incorporate the parsing of all possible queries in the new domain. This includes modification of the list of noise words and the list of synonyms.

#### **Limitations of the Current System**

Following are the limitations of the current system:

1. The current system does not deal with a very large database. In very large databases, efficient access of secondary storage becomes a crucial factor. This issue has not been dealt within the current system;
2. The current system also uses a small representative knowledge base. As the number and complexity of rules in the CLIPS knowledge base increases the memory requirements for rule processing would also increase. While the current hardware technology has made memory economically cheap, memory availability would still be a factor in very large systems. Proper ordering of the patterns of rules could help restrict the memory crunch in such systems. These techniques have not

been consciously used in developing the current system.

3. This system is implemented in a single-user PC environment. Typically, large databases are accessed across a network of machines by multiple user. Issues such as data integrity, updates, etc. would then have to be considered. While Paradox and CLIPS are designed for a single-user and a multi-user environment this system is limited to operate in a single user environment;
4. The current system is limited by the 640KB memory barrier encountered in personal computer.

#### **Future Work**

The following are some of the areas where future work on the current system is suggested:

1. Linking the SQL database server to the current system would greatly enhance the usability of the system;
2. Expanding the current system to work in a networked PC environment and/or Unix workstations;
3. Expanding the knowledge and database of the current system to aid in solving a variety of airline industry related problems;
4. Overcoming the limitation of 640KB real memory availability in PC's. This could be done by using



tools to map the executable to the protected memory area.

## BIBLIOGRAPHY

- [1] Amit B., Rafiul A. ; Using a Relational Database to Support Explanation in a Knowledge-Based system, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 6, December 1992, pp 572-581.
- [2] Andriole S., Hopple G. ; Applied Artificial Intelligence: A Source Book, 1991.
- [3] Gautam B., Gadia S. K. ; Relational Database systems, IEEE Transactions on Knowledge and Data Engineering, Vol 5., No. 5, February 1993, pp 76-87.
- [4] Brodie M., Jarke M. ; On Integrating Logic Programming and Databases, 1989.
- [5] Charles R., Richard C.W. ; Knowledge Intensive Software Engineering Tools, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 5, October 1992, pp 424-430.
- [6] Cheong Y., Hyoung-Joo K. ; Lawrence J. H., Classification and Compilation of Linear Recursive Queries in Deductive Databases, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 1, February 1992, pp 52-67.
- [7] Di Giorgi R.M., E.Fameli, R.Nannucci ; Expert System and Database Interaction in the legal Domain, 1990.
- [8] Freundlich Y. ; Knowledge Bases and Databases Converging Technologies, Diverging Interests, IEEE Computer Science Press, Washington D.C., 1990, pp. 51-57.
- [9] Gunther O. ; Implementation of Heuristic Search in an Expert Database System, in Decision Support Systems 7, 1991, pp. 233-240.
- [10] Jiawei H., Yandong C., Nick C. ; Data-Driven Discovery of Quantative Rules in Relational Databases, IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, February 1993, pp 29-40.

- [11] Hector G., Kenneth S. ; Main Memory Database Systems: An Overview, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 6, December 1992, pp 509-516.
- [12] Heidelberg and Kaiserslautern ; Current developments in knowledge Acquisition :EKAW'92 :6th European Knowledge Acquisition Workshop, May 1992.
- [13] Kershberg L. ; Expert Database Systems: Proceedings from the Second International Conference, April 1988.
- [14] Kerry R. ; Integrating Knowledge-Based and Database Management Systems, 1990.
- [15] Larry A. ; KWOC Indexes and Vocabulary Comparisons of Summaries of LC and DC Classification Schedules, Journal of American Society for Information Science, September-October 1971, pp 322-325.
- [16] Luger George F., William A. S. ; Artificial Intelligence: structures and strategies for complex problem solving, 1993.
- [17] Marguerite F. ; The KWIC Index Concept: A Retrospective View, American Documentation, April 1966, pp 57-70.
- [18] Mcleod D. and Yanover P. ; Expert Database Systems: Proceedings from the Second International Conference, January 1990, pp. 245-253.
- [19] Paradox Refernce Manual, Borland.
- [20] Patterson D.W. ; Introduction to Artificial Intelligence and Expert systems, 1990.
- [21] Pilkington, Rachel M. ; Intelligent help: Communicating with knowledge-based system, May 1992.
- [22] Ramirez R., Dattero R., Choobineh J ; Representing Generalizations and Exceptions in Expert Database Systems, in Decision Support Systems 6, 1990, pp. 29-44.
- [23] Riley G. ; Expert Systems: Principles and Programming, 1989.
- [24] Riley G. ; CLIPS Reference Manual, July 1989.
- [25] Scheugraf E. J., M.F. van Bommel ; Automatic Indexing of Document Databases by Cooperating Expert Systems, 1990.

- [26] Silberchatz Av, et al ; Database Systems: Achievements and Opportunities, October 1991, Vol 34, No. 10.
- [27] Smith J.M. ; Expert Database Systems: A Database Perspective, Expert Database Systems: Larry Kershberg, Editor, 1986, pp. 4-15.
- [28] Michael S. ; The Integration of Rule and Database Systems, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 5, October 1992, pp 415-423.
- [29] Subrata D. K. ; Deductive databases and logic programming, 1992.
- [30] Sycheyder E.C. ; Relational Database Applications in Manufacturing System Design, 1990.
- [31] Tjoa A.M., Wagner ; Database and Expert Systems Applications, 1990.
- [32] Tzy-Hey C., Edward S. ; A Universal Relation Data Model with Semantic Abstractions, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 1, February 1992, pp 23-33.
- [33] Xu Wu, Tadao I. ; KDA: A Knowledge-Based Database Assistant with a Query Guiding Facility, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 5, October 1992, pp 443-453.
- [34] Yannis E.I., Saulys T., Whitsitt A.J. ; Conceptual Learning in Database Design, ACM Transactions on Information Systems, Vol 10, No. 3, July 1992, pp. 265-293.
- [35] Zarri G.P. ; A Proposal of Integrating AI and DB technique in Database and expert System Applications, August 1990, pp. 307-314.
- [36] Zhu J., Maier D., Abstract Objects in an Object-Oriented Data Model, in Expert Database Systems: Proceedings of the Second International Conference, April 1988, pp. 73-106.

## APPENDIXES

APPENDIX A

LIST OF TABLES

TABLE I

## FIELDS OF FLIGHTS.DB

FIELD NUMBER	FIELD NAME	FIELD TYPE	LENGTH	PRIMARY KEY
1	Flight_No	Numeric	N/A	
2	Class	Alphabet	1	

TABLE II

## FIELDS OF ACBASIC.DB

FIELD NUMBER	FIELD NAME	FIELD TYPE	LENGTH	PRIMARY KEY
1	Aircraft_type	Alphabet	28	*
2	No_crew	Numeric	N/A	
3	No_of_bag_compt	Numeric	N/A	
4	No_passengers	Numeric	N/A	
5	Max_takeoff_ld	Numeric	N/A	
6	Max_cargo_ld	Numeric	N/A	
7	Max_altd_fly	Numeric	N/A	
8	Max_cspeed	Numeric	N/A	

TABLE III

## FIELDS OF ACENGINE.DB

FIELD NUMBER	FIELD NAME	FIELD TYPE	LENGTH	PRIMARY KEY
1	Aircraft_type	Alphabet	28	*
2	Engine_type	Alphabet	31	
3	No_of_engines	Numeric	N/A	
4	Wt_engine	Numeric	N/A	

TABLE IV

## FIELDS OF ACDIMEN.DB

FIELD NUMBER	FIELD NAME	FIELD TYPE	LENGTH	PRIMARY KEY
1	Aircraft type	Alphabet	28	*
2	Cabin_length	Numeric	N\A	
3	Cabin_width	Numeric	N\A	
4	Cabin height	Numeric	N\A	
5	Acraft_length	Numeric	N\A	
6	Wing_Span	Numeric	N\A	
7	Cargo_Vol	Numeric	N\A	



TABLE V

FLIGHT DATA STORED IN  
FLIGHTS.DB

FLIGHTS	FLIGHT NUMBER	CLASS
1	112	C
2	173	A
3	179	A
4	456	A
5	790	B
6	889	A
7	1132	B
8	1190	C
9	1234	A
10	8812	B

TABLE VI

LIST OF NOISE WORDS USED BY  
THE QUERY PARSER

A AN AIRCRAFT AIRCRAFTS ALL  
AND AIRPLANES APPLICABLE  
CRUISING COMPARTMENTS DURING FOR  
FLYING HAS HAVE HAVING EACH IS  
KNOW OF OFF NUMBER OR TO THAN  
SIZE SPAN PLANES WITH THE  
RESPECT VALUE

TABLE VII  
 AIRCRAFT ENGINE DATA STORED  
 IN ACENGINE.DB

AIRCRAFT TYPE	ENGINE TYPE	# OF ENGINES	WT. OF ENGINE
AIRBUS	GENERAL ELECTRIC CF6-50C	2	51000
BOEING 707	PRATT & WHITNEY JT3D-3	4	18000
BOEING 717	PRATT & WHITNEY JT3D-7	4	19000
BOEING 727	PRATT & WHITNEY JT8D-7	3	14000
BOEING 737	PRATT & WHITNEY JT8D-7	2	14000
BOEING 747	PRATT & WHITNEY JT9D-3	4	43500
BOEING 757	PRATT & WHITNEY JT8D-9	3	14500
BOEING 767	PRATT & WHITNEY JT8D-9	2	14500
DC 10	GENERAL ELECTRIC CF6-50A	3	49000
DC 7	PRATT & WHITNEY JT3D-1	4	17000
DC 8	PRATT & WHITNEY JT3D-7	4	19000
DC 9	PRATT & WHITNEY JT8D-9	2	14500
LOCKHEED JETSTAR	PRATT & WHITNEY JT12A-8	4	33000
LOCKHEED TRISTAR	ROLLS ROYCE RB211-228	3	38000

TABLE VIII

ATTRIBUTE VALUES STORED IN ACDIMEN.DB

AIRCRAFT TYPE	CABIN LENGTH	CABIN WIDTH	CABIN HEIGHT	AIRCRAFT LENGTH	WING SPAN	CARGO VOLUME
AIRBUS	175.09	28.06	17.07	147.12	147.12	4869
BOEING 707	104.10	11.08	7.07	145.01	130.10	1665
BOEING 717	111.06	11.08	7.07	152.11	145.09	1775
BOEING 727	72.08	11.08	7.02	133.02	108.01	900
BOEING 737	62.02	11.06	7.02	94.00	93.00	650
BOEING 747	185.00	20.00	11.04	231.04	195.08	5190
BOEING 757	92.08	11.08	6.11	153.00	108.00	1485
BOEING 767	68.06	11.06	7.02	100.00	93.00	875
DC 10	150.00	18.00	10.90	181.04	161.04	6000
DC 7	102.01	11.06	7.03	150.06	142.05	1390
DC 8	100.00	10.00	6.09	187.05	148.05	2500
DC 9	100.00	10.00	6.09	125.07	93.05	1019
LOCKHEED JETSTAR	128.02	6.02	6.00	60.05	54.05	2000
LOCKHEED TRISTAR	135.11	18.11	7.11	178.08	155.04	2528

TABLE IX

BASIC AIRCRAFT DATA STORED  
IN ACBASIC.DB

AIRCRAFT TYPE	CREW SIZE	# OF BAGGAGE COMPARTMENTS	# OF PASSENGERS	MAX. TAKEOFF LOAD	MAX. CARGO LOAD	ALTITUDE	CRUISING SPEED
AIRBUS	5	3	331	302000	70020	35000	582
BOEING 707	4	2	181	257000	46849	42000	618
BOEING 717	5	2	219	333600	53900	39000	605
BOEING 727	3	2	131	160000	34500	36500	607
BOEING 737	2	2	115	105000	32100	36000	553
BOEING 747	3	2	500	775000	164745	45000	600
BOEING 757	3	2	189	190500	41000	33500	599
BOEING 767	2	2	130	115500	35700	34000	576
DC 10	5	3	380	555000	104913	32700	570
DC 7	5	1	179	325000	34360	36500	580
DC 8	5	1	259	350000	67735	35100	600
DC 9	2	1	125	114000	34195	34000	561
LOCKHEED JETSTAR	2	1	10	42000	2926	37400	570
LOCKHEED TRISTAR	5	1	200	430000	86183	42000	550

APPENDIX B

LIST OF FIGURES

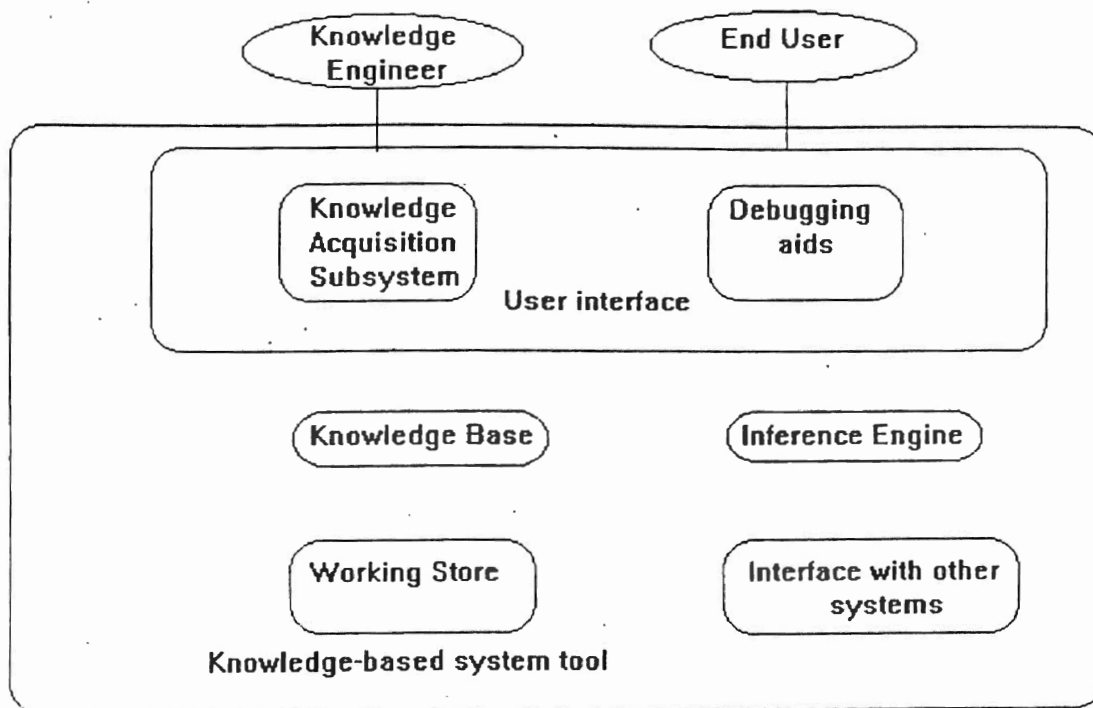


Figure 1. Logical Components of a Knowledge-Based System

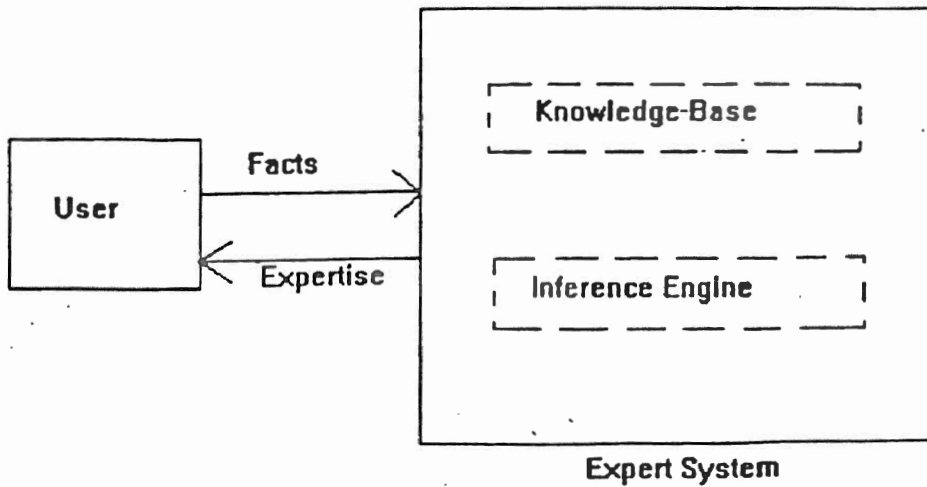


Figure 2. Basic Concept of an Expert System

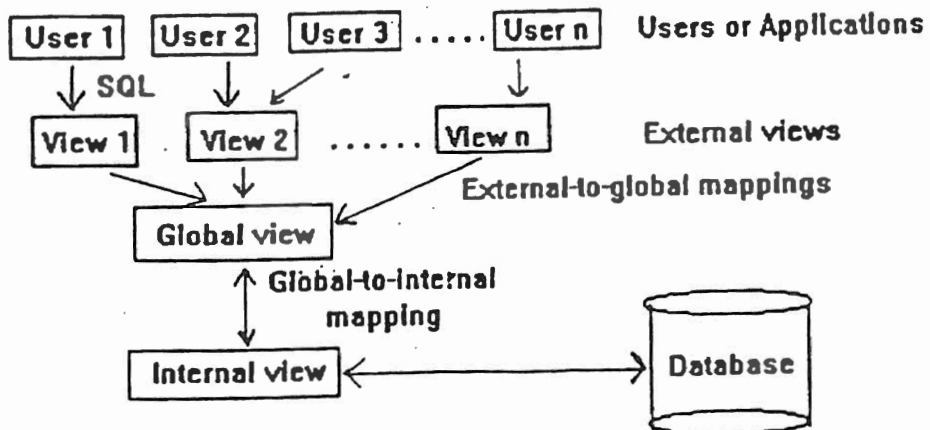


Figure 3. Logical Architecture of a DBMS

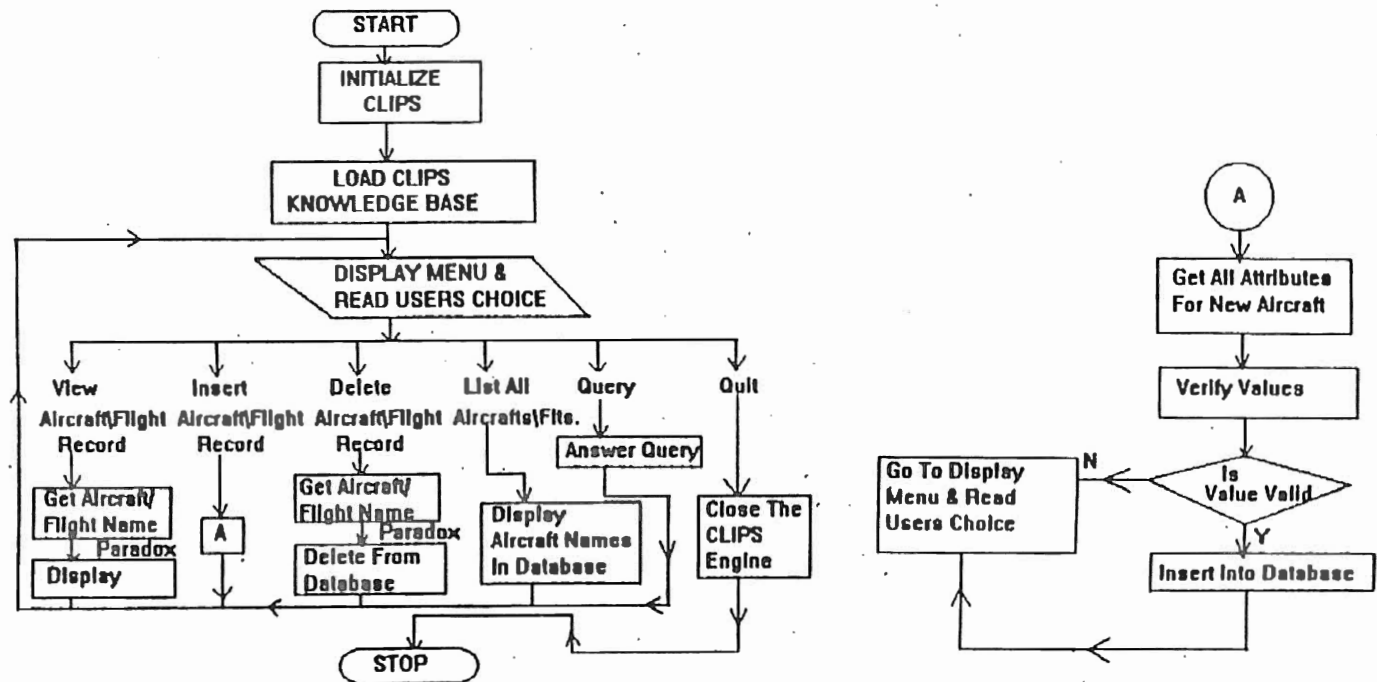


Figure 4. Flowchart Depicting Overall Algorithm



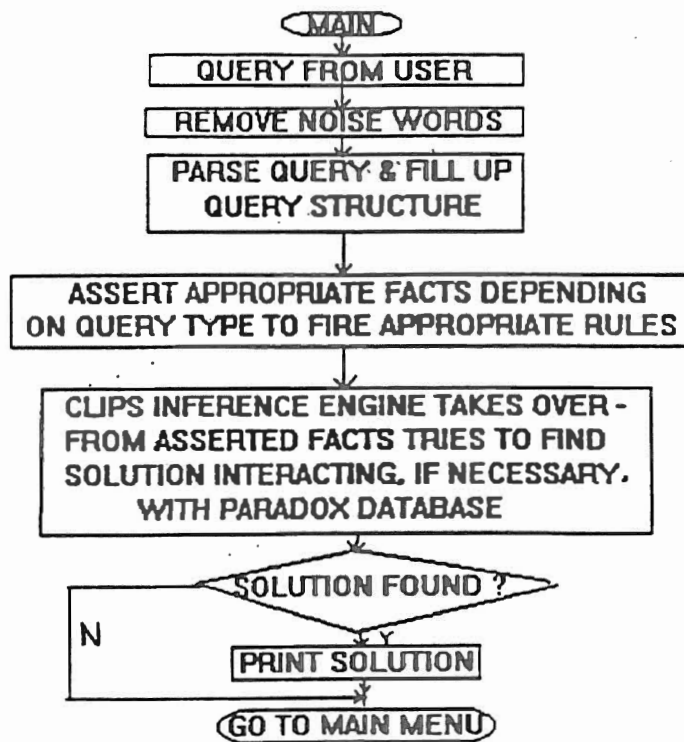


Figure 5. Flowchart of Query Processing

VITA 2

DIPTI R. BHARGAVA

Candidate for the Degree of

Master of Science

**Thesis:** BUILDING AN EXPERT DATABASE SYSTEM IN C USING CLIPS  
AND PARADOX

**Major Field:** Computer Science

**Biographical:**

**Personal Data:** Born in Ahmedabad, India, May 12, 1966,  
the daughter of Rameshwarnath Bhargava and  
Chandrakanta Bhargava.

**Education:** Received Bachelor of Science Degree from  
Nagpur University, Nagpur, India in May, 1986;  
received Master of Science in Mathematics from  
Bombay university, Bombay, India in May, 1988;  
completed requirements for the Master of Science  
degree at Oklahoma State University in July, 1993.

**Professional Experience:**

Operations Research Analyst, United Airlines, 1200  
East Algonquin Road, Elk Grove Village, Chicago,  
Illinois, February, 1992, to present.

Research Assistant, University Computer Center,  
Oklahoma State University, August, 1991, to  
January, 1992.

Research Assistant, Department of Business  
Administration, Oklahoma State University, May,  
1990, to May, 1991.