



UNIVERSITY
OF
JOHANNESBURG

COPYRIGHT AND CITATION CONSIDERATIONS FOR THIS THESIS/ DISSERTATION



- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- NonCommercial — You may not use the material for commercial purposes.
- ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

How to cite this thesis

Surname, Initial(s). (2012) Title of the thesis or dissertation. PhD. (Chemistry)/ M.Sc. (Physics)/ M.A. (Philosophy)/M.Com. (Finance) etc. [Unpublished]: [University of Johannesburg](https://ujdigispace.uj.ac.za). Retrieved from: <https://ujdigispace.uj.ac.za> (Accessed: Date).

WRIO
JONK

THE NORMALIZATION OF FRAMES AS A SUPERCLASS OF
RELATIONS

by

JACOB JONKER

DISSERTATION

submitted in fulfillment of the requirements for the degree of

MASTERS IN NATURAL SCIENCE

in

COMPUTER SCIENCE

in the

FACULTY OF SCIENCE

at

RAND AFRIKAANS UNIVERSITY

SUPERVISOR: PROF. E.M. EHLERS

NOVEMBER 1992

Opsomming	vi
Abstract	vi
CHAPTER I : Introduction.....	1
1.1 Frames and Relations	1
1.2 Differences vs. Similarities	1
1.3 Data, Information and Knowledge.....	1
1.4 Knowledge-Bases and Databases.....	3
1.5 Model Theoretic and Proof Theoretic Views of Databases.	3
1.6 Database Management Systems and Expert Systems	4
1.7 Knowledge-based Systems and Artificial Intelligence.....	5
1.8 Framework for study.....	6
CHAPTER II : Relations and Their Operations.....	8
2.1 Introduction	8
2.2 Relations	8
2.3 Example.....	10
2.4 Normalization	11
2.4.1 First Normal Form	12
2.4.2 Second Normal Form.....	12
2.4.3 Third Normal Form.....	15
2.4.4 Fourth Normal Form.....	17
2.4.5 Project-Join Normal Form.....	17
2.4.6 Domain-Key Normal Form.....	19
2.5 Integrity Constraints.....	19
2.5.1 Uniqueness property	19
2.5.2 Referential integrity.....	20
2.6 Conclusion	20
CHAPTER III : Frames	21
3.1 Introduction	21
3.2 Knowledge Representation.....	21
3.3 The development of Frames	22
3.4 Frames	22
3.4.1 Example Frames.....	23
3.5 Different Views of Frames.....	24

- 3.6 Application Areas of Frames 24
 - 3.6.1 Frames and Vision..... 25
 - 3.6.2 Frames and text..... 25
 - 3.6.3 Frames and Expert Systems..... 26
- 3.7 Frames as an Object-oriented Representation Scheme..... 26
- 3.8 Semantics and Basic Characteristics of Frames 26
 - 3.8.1 Frame Inference 27
 - 3.8.2 Seeing as..... 28
 - 3.8.3 Defaults 28
 - 3.8.4 Reflexive Reasoning..... 28
- 3.9 Conclusion 29
- CHAPTER IV : Formal Definitions of Frames 30
 - 4.1 Introduction 30
 - 4.2 Representation Languages..... 30
 - 4.2.1 Frame representation Language (FRL) 30
 - 4.2.2 Knowledge Representation Language (KRL) 31
 - 4.3 Mathematical Definitions..... 31
 - 4.3.1 Algebraic Structured Model of Frames 31
 - 4.3.2 Predicate Algebra Notation of Frames 32
 - 4.3.3 Functional mapping definition of Frames 33
 - 4.4 Formal Definition of a Frame..... 34
 - 4.4.1 Example..... 35
 - 4.5 Conclusion..... 36
- CHAPTER V : Problems With Frames 37
 - 5.1 Introduction 37
 - 5.2 Representing Knowledge..... 37
 - 5.2.1 Conflicting requirements 37
 - 5.2.2 Blindness 38
 - 5.2.3 Common sense knowledge 38
 - 5.2.4 Self-knowledge and meta-knowledge 38
 - 5.3 Problems with Frames as a Knowledge Representation Structure..... 38
 - 5.3.1 Lack of theoretical foundations and formalisms for using frames:..... 38
 - 5.3.2 Default Reasoning:..... 39
 - 5.3.3 Ineffective search 39
 - 5.3.4 Enforcing of Integrity constraints 39

5.4 Advantages of Using a Frame-based Representation Scheme	40
5.4.1 Natural Representation of Knowledge	40
5.4.2 Multiple relationships	40
5.4.3 Combining declarative and procedural information	40
5.5 Conclusion	40
Chapter VI : The Normalization of Frames	41
6.1 Introduction	41
6.2 Comparison of Frames and Relations	41
6.3 Normalization	44
6.3.1 Functional dependence and primary keys	46
6.3.2 Non-Derivable normal form	47
6.3.3 First Normal Form	49
6.3.4 Second Normal Form	50
6.3.5 Third Normal Form	52
6.3.6 Fourth Normal Form	53
6.4 The Relational Algebra	53
6.4.1 Select	53
6.4.2 Project	54
6.4.3 Join	55
6.4.4 Union	55
6.5 Motivation of normalization	56
6.6 Advantages of Seeing Frames as a Superclass of Relations	57
6.6.1 Lack of theoretical foundations and formalisms for using frames	58
6.6.2 Ineffective search	58
6.7 A Complete Example of Frame Normalization	58
6.7.1 The Example:	58
6.7.2 Normalization	61
6.7.2.1 Non-derivable normal form	61
6.7.2.2 First Normal Form	61
6.7.2.3 Second Normal Form	64
6.7.2.4 Third Normal Form	65
6.7.2.5 Fourth Normal Form	66
6.7.3 The Normalized Example	66
6.8 Conclusion	67
Chapter VII : An Overview of Expert Systems	68

7.1 Introduction.....	68
7.2 Expert Systems	68
7.3 Rule based Expert Systems.....	69
7.3.1 Domain specific knowledge.....	69
7.3.2 Domain-independent knowledge	69
7.3.3 Advantages of a rule-based expert system	70
7.3.4 Disadvantage of separation of knowledge and control	70
7.4 Frame based Expert Systems	71
7.4.1 Domain specific knowledge.....	71
7.4.2 Domain-independent knowledge	71
7.4.3 Advantages of a frame-based expert system.....	72
7.4.4 Disadvantages of frame-based expert systems.....	72
7.5 Hybrid Expert Systems.....	72
7.5.1 Domain Specific Knowledge	73
7.5.2 Domain-Independent Knowledge	73
7.5.3 Advantages of hybrid expert systems.....	73
7.6 Conclusion.....	74
Chapter VIII : The Integration of Knowledge-based Systems and Database Management Systems	75
8.1 Introduction.....	75
8.2 Motivation	75
8.3 Classification.....	76
8.4 Approaches to Integrate Expert Systems and Database Management Systems	78
8.4.1 Expert Database Systems	78
8.4.1.1 Architecture of an Expert Database System.....	78
8.4.1.2 Important issues about EDS	79
8.4.1.3 Inference versus Query Evaluation: A Comparison	80
8.4.1.4 Conclusion	80
8.4.2 The Extended Disjunctive Normal Form Approach.....	81
8.4.2.1 Advantages of the EDNF Approach	81
8.5 Using the Relational Model for Knowledge-bases.....	81
8.5.1 Knowledge Representation Using Views in Relational Deductive Databases.....	82
8.5.1.1 Knowledge-bases and Databases	83
8.5.1.2 Rules and Views.....	83

8.5.2 Extensions to the Relational Algebra	83
8.5.3 Deductive object bases	83
8.5.4 A Mapping from Frame-based knowledge to Nested Relations.....	84
8.5.5 Frames as a Superclass of Relations	84
8.5.5.1 An Alternative view of a Frame-based Expert System.....	84
8.6 Objections Against Using Relational Databases for Representing Knowledge	86
8.7 Database management systems and Knowledge-based systems as complementary technologies	88
8.8 Conclusion	88
Chapter IX : Normalization and the System Development Life Cycle of Expert Systems.....	90
9.1 Introduction	90
9.2 Steps in Developing an Expert System	90
9.2.1 Feasibility.....	90
9.2.2 Selecting a Development Tool.....	91
9.2.3 Knowledge Acquisition	91
9.2.4 Creating a Representation of the Knowledge	92
9.2.5 Verification and Evaluation	93
9.2.6 Maintaining and Expanding the System	93
9.3 Encoding the Knowledge	93
9.3.1 Represent the knowledge as collections of frame classes.....	94
9.3.2 Identify all the keys	94
9.3.3 Normalize the Frames	95
9.3.4 Determine defaults, domains and constraints	95
9.4 Conclusion	96
CHAPTER X : CONCLUSION	97
10.1 Introduction	97
10.2 Frames as a Superclass of Relations.....	97
10.3 Topics for Further Research	98
10.4 Conclusion	98
REFERENCES AND BIBLIOGRAPHY	99

Opsomming

Daar is sekere probleme met kennisvoorstelling, wat nie veroorsaak word deur kennisvoorstellingsstrukture wat onvoldoende is nie, maar deur die manier waarop die kennisvoorstellingsstrukture gebruik word. In die eerste deel van hierdie verhandeling kyk ons na die relasie model (vir databasisbeheerstelsels) asook na rame ('n kennisvoorstellingsstruktuur gebruik in ekspertstelsels), soos voorgestel deur M. Minsky [MIN75]. Ons verskaf dan ons eie definisie van rame, gebaseer op die definisie van Minsky. In die tweede gedeelte, vergelyk ons die twee modelle (die relasie model en ons model van rame), en ons toon aan dat rame gesien kan word as 'n superklas van relasies. As gevolg van die verwantskap tussen die twee modelle, definieer ons normalisering vir rame en ons ondersoek hoe die normalisering sekere van die probleme wat ons geïdentifiseer het, oplos. Daarna kyk ons na die integrasie van ekspertstelsels en databasisbeheerstelsels, en ons klassifiseer dan ook ons normalisering as so 'n poging. Ten slotte kyk ons na die plek van normalisering in die ekspertstelsel ontwikkelingslewensiklus.

Abstract

Knowledge representation suffers from certain problems, which is not a result of the inadequacies of knowledge representation schemes, but of the way in which they are used and implemented. In the first part of this dissertation we examine the relational model (as used in relational database management systems) and we examine frames (a knowledge representation scheme used in expert systems), as proposed by M. Minsky [MIN75]. We then provide our own definition of frames. In the second part, we examine similarities between the two models (the relational model and our frame model), establishing frames as a superclass of relations. We then define normalization for frames and examine how normalization might solve some of the problems we have identified. We then examine the integration of knowledge-based systems and database management systems and classify our normalization of frames as such an attempt. We conclude by examining the place of normalization within the expert system development life cycle.

CHAPTER I : Introduction

1.1 Frames and Relations

In this chapter we provide the rationale for wanting to establish frames as a superclass of relations. We look at the distinctions between data, information and knowledge, and at the differences in the state of technology in databases and expert systems. We also provide a framework for our study.

1.2 Differences vs. Similarities

Most textbooks on expert systems, devote some time to the differences between expert system tools and more traditional programs, like database management systems and management information systems (MIS) [BOW88, LUG89]. Specifying these differences serve to better define the concept of expert systems. The mistake that is made by many however is to stop as soon as they have identified these differences. This is very unfortunate, as, in general, just as much knowledge might be gained by looking at similarities between concepts, as can be by looking at differences. The main advantage of identifying similarities between two concepts, is that it is then possible to modify results proven or discovered for one concept and to re-use them or apply them to the second concept.

This dissertation is devoted, primarily to the similarities between expert systems, and database management systems, more specifically to the similarity between frames (a knowledge representation scheme, used in some expert systems) and relations (a model used by most commercial database management systems).

1.3 Data, Information and Knowledge

Data is said to refer to numeric or alphanumeric strings, and the processing of data is said to be limited to various sorting, filtering and collating operations performed on these elementary data fields.

When data is organized and meaningful to a person, it is called information. In other words, the right data, available at the right place and time is information. The fundamental

difference between data and information is therefore, not necessarily its structure, but its application.

Symbols are the representation of data or information as meaningful related objects, symbols in other words are a different representation of data and information that is meaningful to a person or group of persons.

Knowledge is said to be characterized in terms of three aspects: the *knower*, the *known* and the *process of knowing*. Knowledge in other words is the interaction between the knower and the known. [BOW88]

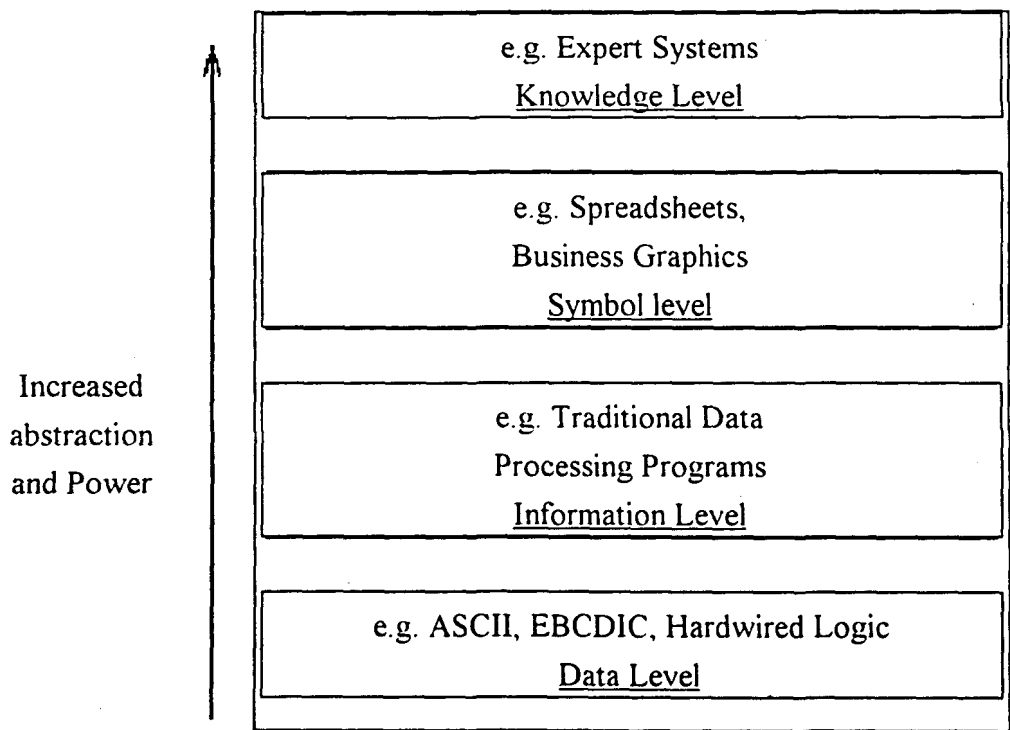


Fig. 1.1 Four levels of abstraction in computer systems. [BOW88]

The differences between knowledge, symbols, information and data are very subjective. If in some or other program the fact is stored that Peter's birthday is on the second of October, it is a piece of data, should this program inform me on the First of October, that tomorrow is Peter's birthday, this fact is called information, and if I know when Peter's birthday is it is called knowledge.

1.4 Knowledge-Bases and Databases

The reason for the previous discussion is simply to illustrate the following:

Although the internal structure of databases, and knowledge-bases generally differ, because different representation schemes are used in each (relations as opposed to rules), the 'items' contained within a database might be used as knowledge, and the 'items' contained within a knowledge-base might be used as data or information. The only real difference between a database and knowledge-base, is the use to which their contents are put. It is possible for two different applications to contain exactly the same 'databases' identical in both structure and content, but one might be classified as a database system and the other as an expert system.

Because of this similarity between knowledge-bases and databases, the following should be possible:

- 1) It should be possible to develop expert systems, that provide mechanisms to ensure integrity and optimize queries, using traditional database management techniques.
- 2) It should be possible to develop databases systems, using expert system development tools.

The second point is not intended to suggest developing database systems using expert system development tools, but is intended as a test for any expert system development tool. If an expert system development tool is not able to access its knowledge or data just as efficiently as any commercial database management system and does not provide the same level of integrity, you will be better off developing a serious expert system using database management software. An expert system tool that is worth anything should provide methods to manipulate its data that is at least equal to the methods used by database management systems [UCK91, REI89].

1.5 Model Theoretic and Proof Theoretic Views of Databases.

There are two distinct views of a relational database:

The first, the model theoretic view of a database, is the traditional perception of a relational database. In this view, a database is seen as a set of explicit base relations. In

this view, the processing of a query is seen as the equivalent to evaluating a formula over the tuples of the database.

The second, alternative view, proposed by Reiter is the proof theoretic view. In this view, a database is consistent of a set of facts, encoded as tuples in the relations and a set of deductive axioms that can be used to derive new facts. In this approach, processing a query is seen as the same as proving a theorem, that is, that some specified fact is the logical consequence of the axioms stored in the database. [UCK91, DAT90]

1.6 Database Management Systems and Expert Systems

The situation that currently exists is the following:

Database management systems that manage large volumes of data with a rather simple structure have been designed [DAT90, REI89, PRA87]. These database management systems have extremely effective data storage and retrieval functions that are able to efficiently manage the large volumes of data generated by a modern commercial database application. Expert systems that have been developed, use sophisticated representation schemes, but the aspect of handling large volumes of knowledge, has not been an issue, because most expert systems were designed for small experimental applications, that were able to keep all their knowledge in main memory, making even a sequential search through all its knowledge reasonably fast [REI89].

Using existing methods, to develop a real-world knowledge-based system raises new problems.

- It is difficult to assure integrity of knowledge during modification of any part of the knowledge-base.
- Efficiently searching through a large knowledge-base might also go beyond acceptable computational time. [REI89]

The same type of problems has already been solved for relational database systems [PRA87, DAT90, COD90, MAI83]. These problems were mainly solved by normalization of relations and by the definition of concepts such as primary and foreign keys of a relation. In this dissertation (chapter 6) we examine the similarities between frames and relations and we attempt to define normalization for frames. In figure 1.2 we see a summary of the current situation in database and expert systems.

Database Systems	Expert Systems
<ul style="list-style-type: none"> - Simple data structures - Large Volumes of data - Efficient mechanisms for: <ul style="list-style-type: none"> • Integrity • Storage and retrieval 	<ul style="list-style-type: none"> - Complex knowledge representation structures - Generally small amounts of knowledge - Ineffective mechanisms when processing large volumes of knowledge.

Fig. 1.2 Comparison of Database and Expert Systems.

1.7 Knowledge-based Systems and Artificial Intelligence

Artificial intelligence(AI) can be defined as the branch of computer science, that involves itself with the automation of intelligent behavior. The term Artificial intelligence is a misnomer, because it is impossible to create artificial intelligence without knowing the exact nature of intelligence. Current artificial intelligence applications, can more correctly be described as '*simulated*' intelligence, as these systems only seem to act intelligently. The term Artificial Intelligence is however used so frequently that it is doubtful that the more correct term will ever be adopted [TEL88, LUG89].

AI research can be divided into different research areas, which include:

1) Game playing:

This is one of the oldest areas in AI research, that focus primarily on strategy games, like chess and checkers [TEL88, LUG89]. The most important issues in game playing are: How to represent the different board configurations, and how to search the problem space.

2) Expert systems:

Expert systems or knowledge-based systems use domain-specific knowledge to solve problems in a specific problem domain [BOW88, LUG89].

3) Natural-language processing:

Natural language processing involves not only the parsing of sentences into individual parts of speech, and looking up the meaning of words in a dictionary. It also includes the application of extensive background knowledge about the domain

of discourse, to correctly interpret the meaning of human speech [LUG89]. An example of a system that processes natural language is BORIS [TEL88, BLA86]. BORIS was developed at Yale University and is able to understand large bodies of natural-language texts. BORIS, when given a story to read is able to answer questions about the characters in the story and why they took certain actions.

4) Machine learning:

Most expert systems are unable to learn. The objective of machine learning is to enable programs to learn, from experience, analogy and examples [LUG89].

5) Vision and sensory recognition:

This area of research attempts to provide computers with the ability to evaluate visual images and to take action on them. Examples of this are a system for recognizing a face from stored pictures of people, or a system that is able to drive a motor car, without driving into anything [TEL88].

The most promising area of AI research is expert and knowledge-based systems. It is however impossible to separate it from other AI research areas, as they are all interrelated. For example a system for machine learning might utilize an expert system, as well as a natural-language interface. Therefore, knowledge representation is not only important for expert systems, but for all areas of Artificial Intelligence.

1.8 Framework for study

We begin by giving the formal definition of relations, and looking at the normalization of relations and the motivation for the normalization process (chapter 2).

We then look at knowledge representation schemes, paying particular attention to frames (chapter 3).

In chapter 4 we look at the different formal definitions of frames and conclude by giving our own definition of frames.

We then look at problems with knowledge representation schemes in general, and at problems with frames in particular (chapter 5).

After formally defining frames and relations, we compare the two structures for differences and similarities, using the definition of relations, given in chapter 3 and our own definition of frames for this comparison. We then define normalization for frames, and examine the effect of 'normalization' on frames (chapter 6).

In the next chapter we examine expert systems and the distinction between frame- and rule-based expert systems (chapter 7).

In chapter 8 we take a look at the integration of database management and knowledge-based systems.

And in chapter 9, we provide a methodology for using normalization during the development of a frame-based expert system.

CHAPTER II : Relations and Their Operations

2.1 Introduction

A relation is a mathematical structure used as the foundation for the organization of data in a relational database. The relational model was originally presented by Dr. E.F. Codd in 1970. This, at that time, radically different model, led to a great deal of research and in the early 1980s relational databases started to appear, and soon replaced other database models, as the industry standard [PRA87].

In this chapter we will examine relations, and the normalization of relations. This is done to provide a basis for comparison, when a formal definition of frames is given later.

Basic Definitions:

Relation: In its simplest terms, a relation can be seen as a two-dimensional table, with each row in the table having the same format. Each row of a relation represents some object in the real world, and each column represents specific attributes or characteristics of these 'objects' [PRA87, DAT90, MAI83]. In the next section we will give a more detailed definition of the term relation.

Tuple: Each row of a relation is technically called a *tuple*. The number of tuples in a specific relation is called its *cardinality* [PRA87, DAT90].

Attribute: Each column of a relation is technically called an *attribute*. The number of attributes of a relation is called its *degree* [PRA87, DAT90].

Domain: The pool of values from which an attribute or group of attributes may draw their actual values is called a *domain* [PRA87, DAT90].

2.2 Relations

The following formal definition of a relation, is given by C.J. Date [DAT90], and is a slightly modified version of the original definition given by Codd.

A relation R on a collection of domains D_1, D_2, \dots, D_n
(not necessarily distinct) consists of two parts, a *body* and a *heading*.

The **heading** consists of a fixed set of attribute-domain pairs,

$$\{ (A_1:D_1), (A_2:D_2), \dots, (A_n:D_n) \}$$

such that each attribute A_j corresponds to exactly one of the underlying domains D_j ,
for $j = 1, 2, \dots, n$,
where n is the degree of the relation and,
 A_j is the name of attribute j .

The **body** consists of a time-varying set of n -tuples, where each tuple in turn consists of a set of attribute-value pairs,

$$\{ (A_1:v_{i1}), (A_2:v_{i2}), \dots, (A_n:v_{in}) \}$$

for $i = 1, 2, \dots, m$,
where m is the cardinality of the specific relation and,
 v_{ij} is the value of the i 'th tuple's j 'th attribute.

Properties of Relations

Relations possess certain properties, all of them direct, but not necessarily obvious, consequences of the above definition of a relation. The properties within any given relation are as follows:

- There are no duplicate tuples;
- Tuples are unordered (top to bottom);
- Attributes are unordered (left to right);
- All attribute values are atomic. [DAT90]

1. There are no duplicate tuples

This follows directly from the definition of the body of the relation as a mathematical set, as sets by definition do not contain any duplicate elements.

2. *Tuples are unordered (top to bottom)*

This property also follows from the fact that the body of the relation is a mathematical set. Sets in mathematics are not ordered.

3. *Attributes are unordered (left to right)*

This property follows from the fact that the heading of a relation is also defined as a set.

4. *All attribute values are atomic*

This last property is a consequence of the fact that all underlying domains, contain atomic values only. This property can also be stated as follows: Relations do not contain repeating groups. A relation satisfying this condition is said to be normalized.

2.3 Example

We will illustrate normalization of relations, using the table in figure 2.1, which in its current form is still unnormalized and strictly speaking, not a relation at all because some of its attribute values are not atomic. The relation we are using as an example has the following attributes:

- | | |
|---------------|---|
| Person number | - A unique number given to each student. |
| Person name | - The name of the student. |
| Course number | - The codes of courses already attended by the student. |
| Course name | - The name of each course. |
| Dept. number | - The Department the student belongs to. |
| Date | - The date on which each course was finished. |

Students

Person number	Person name	Course number	Course name	Dept. number	Dept. name	Date
003560	D. Adams	3016	Typing	L000	Languages	90/09/07
		3019	Orientation			91/01/03
		4057	Greek I			91/07/03
000834	L. Davies	3016	Typing	L000	Languages	89/01/31
		3017	Latin I			90/02/25
007876	P. Davies	4011	Physics I	E011	Science	90/01/07

Fig. 2.1 An Unnormalized relation containing information about students.

2.4 Normalization

Normalization is a process of transforming a relation, into equivalent relation(s), that eliminate certain problems (*update anomalies*), that might cause difficulty in updating the relation consistently, in its original form. Before it is possible to go into normalization in any detail, it will be necessary to first define some basic concepts:

Functional Dependence

Given a relation R, attribute Y of R is *functionally dependent* on attribute X of R -symbolically: $R.X \rightarrow R.Y$ (read "R.X functionally determines R.Y")

- if and only if each X-value in R has associated with it precisely one Y-value in R (at any one time). Attributes X and Y may be composite. [DAT90]

Functional dependence is a *semantic* notion. Recognizing the functional dependencies is part of the process of understanding what the data *means*. Functional dependence implies that given:

$R.X \rightarrow R.Y$, and given a value for R.X, it will be possible to determine a unique value for R.Y, by inspecting the relation R. It is important to realize that functional dependence can not be determined by only inspecting sample data, it is necessary to understand the meaning of the data to make any decision about functional dependencies [DAT90].

In our example the following functional dependencies are present:

Person number \rightarrow Person name, Dept. number

Course number \rightarrow Course name

Dept. number \rightarrow Dept. name

Person number, Course number \rightarrow Date.

Primary key

Given a relation R, attribute R.Y (R.Y may be composite) is the *primary key* of R if:

1. *All* attributes in R are functionally dependent on R.Y.
2. No subset of the attributes in R.Y also has property 1 [DAT90].

In other words the primary key of a relation R is the smallest subset of its attributes, that has a unique set of values for each of its tuples.

Given this definition of a primary key, and the formal definition of a relation, it is obvious that a relation always has a primary key even if the primary key consists of all of its attributes, since a relation may contain no duplicate tuples.

In the example the primary key is the combination of the *Person number* and the *Course number* attributes.

2.4.1 First Normal Form

A relation is in *first normal form* (1NF) only if it contains no repeating groups.

A relation containing repeating groups is, strictly speaking, not a relation at all and is called an *unnormalized relation*.

Students

Person number	Person name	Course number	Course name	Dept. number	Dept. name	Date
003560	D. Adams	3016	Typing	L000	Languages	90/09/07
003560	D. Adams	3019	Orientation	L000	Languages	91/01/03
003560	D. Adams	4057	Greek I	L000	Languages	91/07/03
000834	L. Davies	3016	Typing	L000	Languages	89/01/31
000834	L. Davies	3017	Latin I	L000	Languages	90/02/25
007876	P. Davies	4011	Physics I	E011	Science	90/01/07

Fig. 2.2 The students relation without repeating groups (1NF)

Removing the repeating groups in the example produces the relation in figure 2.2.

2.4.2 Second Normal Form

Even though a relation is in 1NF, problems may exist within the relation that will cause us to want to restructure it. Consider the relation in figure 2.2:

STUDENTS (Person number, Course number, Person name, Course name, Dept. number, Dept name., Date).

with functional dependencies:

Person number \rightarrow Person name, Dept. number

Course number \rightarrow Course name.

Dept. number \rightarrow Dept. name.

Person number, Course number \rightarrow Date.

As you can see in figure 2.2., the description of a specific course occurs several times in the relation. This redundancy causes several problems, of which the least is that it is a waste of space. The other problems caused by this are called *update anomalies* and they fall in the following categories:

- Update.
- Inconsistent Data.
- Additions.
- Deletions [PRA87].

In our example the following update anomalies, are evident in each of these categories:

1. Update

Changing the description of course 3016 requires not one change but several - each row where 3016 appears, has to be changed.

2. Inconsistent Data

There is nothing about the design that would prevent course 3016 from having two, or more, different descriptions in the database.

3. Additions

We have a problem when we wish to add a new course that has not been attended by anyone before.

4. Deletions

In the example, if we delete the record of P. Davies, we also lose the fact that course 4011 is Physics I.

These anomalies lead to the definition of second normal form, which eliminates the above mentioned update anomalies in these situations.

An attribute is a *nonkey attribute* if it is not part of the primary key [DAT90, PRA87].

A relation is in *second normal form* (2NF) if it is in first normal form and no nonkey attribute is dependent on only a portion of the primary key [PRA87, DAT90].

In our example (in 1NF), the relation:

Students(Person number, Person name, Course number, Course name, Dept. number, Dept. Name, Date)

with primary key: *Person number, Course number*, has the following nonkey attributes that are dependent on only a portion of the primary key:

Person number \rightarrow Person Name, Dept. number, Dept. name.

Course number \rightarrow Course name.

Person

Number	Name	Dept. number	Dept. name
003560	D. Adams	L000	Languages
003560	D. Adams	L000	Languages
003560	D. Adams	L000	Languages
000834	L. Davies	L000	Languages
000834	L. Davies	L000	Languages
007876	P. Davies	E011	Science

Course

Number	Name
3016	Typing
3019	Orientation
3017	Latin I
4011	Physics I
4057	Greek I

Training

Person number	Course number	Date
003560	3016	90/09/07
003560	3019	91/01/03
003560	4057	91/07/03
000834	3016	89/01/31
000834	3017	90/02/25
007876	4011	90/01/07

Fig. 2.3 Our example, normalized to 2NF

In 2NF our example will therefore consist of the following three relations:

Person (Number, Name, Dept number, Dept name)

Course (Number, Name)

Training (Course number, Person number, Date)

The contents of these relations are shown in figure 2.3

2.4.3 Third Normal Form

It is clear by looking at the example, that even in second normal form, update anomalies still exists because of redundant data. In the example descriptions of departments are duplicated. This redundancy, results in exactly the same set of problems that lead to the creation of the second normal form. For instance, it is possible for the department number L000 to have several different names in the present form of the Person relation. To eliminate this type of problems third normal form was defined.

A *candidate key* is any attribute, which could also function as the primary key

Any attribute (or collection of attributes) that determines another attribute is called a *determinant*.

A relation is in *third normal form* (3NF) if it is in second normal form and if the only determinants it contains are candidate keys.

(Note. This definition of third normal form is also called *BOYCE (CODD) NORMAL FORM* (BCNF), and was defined because the original definition of third normal form, was found to be inadequate for certain complex relations. [PRA87])

In our example (in 2NF) the relation:

Person (Number, Name, Dept. number, Dept. name),
has the following determinants that are not candidate keys:

Dept. number Dept. name,

Department number being a determinant for department name. In 3NF our example will therefore consist of the following relations:

Person (Number, Name, Dept number)

Department(Number, Name)

Course (Number, Name)

Training (Course number, Person number, Date)

The contents of these relations are shown in figure 2.4

Person

Number	Name	Dept. number
003560	D. Adams	L000
003560	D. Adams	L000
003560	D. Adams	L000
000834	L. Davies	L000
000834	L. Davies	L000
007876	P. Davies	E011

Department

Number	Name
L000	Languages
E011	Science

Course

Number	Name
3016	Typing
3019	Orientation
3017	Latin I
4011	Physics I
4057	Greek I

Training

Person number	Course number	Date
003560	3016	90/09/07
003560	3019	91/01/03
003560	4057	91/07/03
000834	3016	89/01/31
000834	3017	90/02/25
007876	4011	90/01/07

Fig. 2.4 Our example, normalized to 3NF

2.4.4 Fourth Normal Form

There exist a number of higher normal forms, of which the first is fourth normal form (4NF), which arises because of something called multivalued dependency.

In a relation with attributes A, B, and C, there is a *multivalued dependency* of attribute B on attribute A if a value for A is associated with a specific collection of values for B, independent of any values for C.

A relation is in *fourth normal form* (4NF) if it is in 3NF and there are no multivalued dependencies.

It is important to note, that multivalued dependency can be avoided by placing repeating groups present in an unnormalized relation, in separate relations, when converting to 1NF.

The other higher normal forms are fifth normal form (also called projection-join normal form) and the so called domain-key normal form.

2.4.5 Project-Join Normal Form

Project-Join Normal form (PJNF) attempts to find lossless decompositions to remove redundancy from relations. The project-join normal form was developed because it was discovered that relations exist that cannot be losslessly decomposed into two projections, but can be losslessly decomposed into three or more. [DAT90]. The example in figure 2.5 is given by Date [DAT90] to illustrate this point.

A join dependency is defined as follows:

Let $R = \{R_1, R_2, \dots, R_p\}$ be a set of relational schemes over U . A relation $r(U)$ satisfies the *join dependency* $*[R_1, R_2, \dots, R_p]$ if r decomposes losslessly onto R_1, R_2, \dots, R_p , in other words, if the original relation r can be obtained by the join of the relations containing attributes R_1, R_2, \dots, R_p . Where each R_{j_i} consists of multiple attributes of r , for $1 \leq i \leq p$.

Let R be a relational scheme and let F be a set of functional dependencies and join dependencies. R is in project-join normal form with respect to F if every

join dependency $* [R_1, R_2, \dots, R_p]$ implied by F that applies to R , $* [R_1, R_2, \dots, R_p]$ is implied by the key functional dependencies of R [MAI83]. Alternatively stated, a relation R is in project-join normal form if and only if every join dependency in R is implied by the candidate keys of R [DAT90].

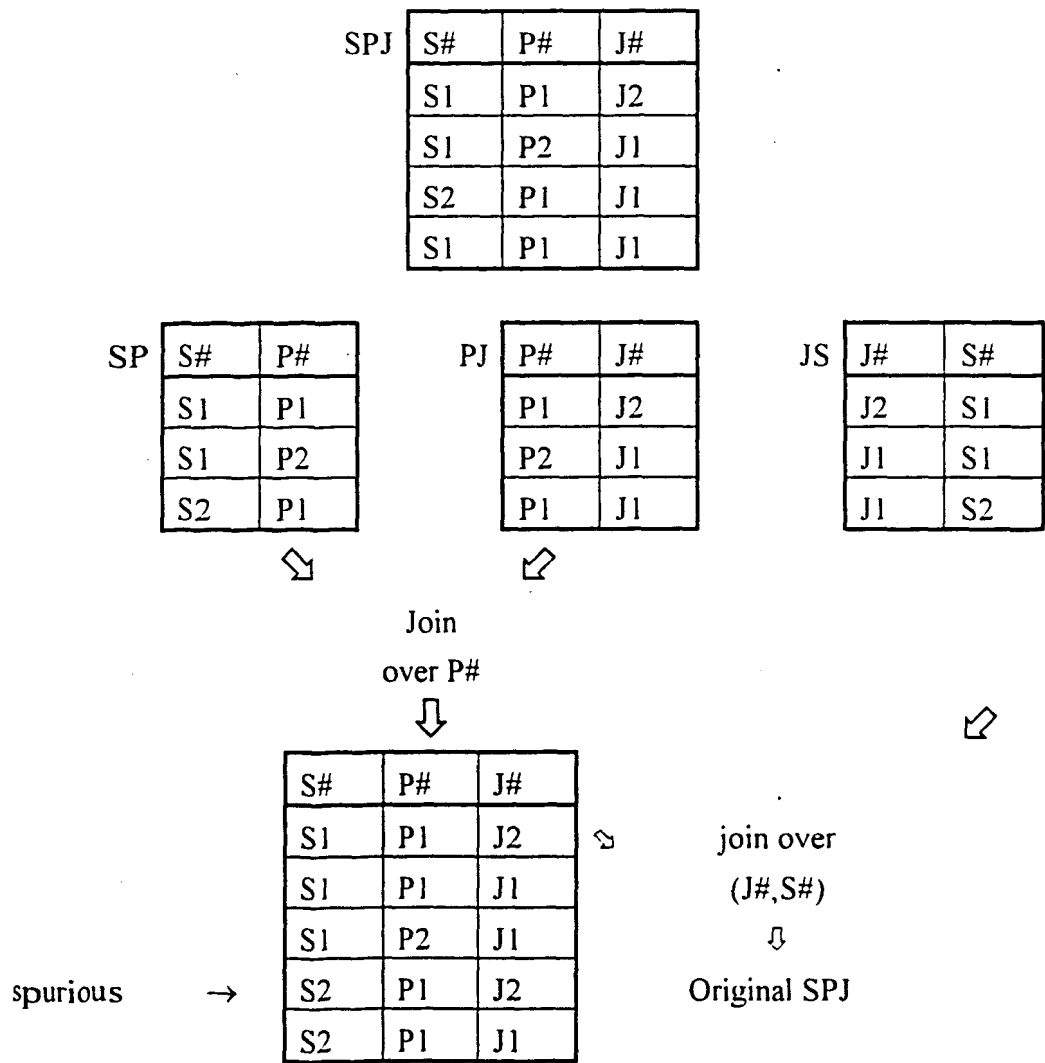


Fig. 2.5 SPJ is the join of all three of its binary projections but not of any two.

Given this definition of fifth normal form (project-join normal form) clearly to tell whether a relation R is in 4NF, but not in 5NF, we need to know the candidate keys and all the join dependencies in R . Discovering all the join dependencies, however is a nontrivial operation and therefore the process for determining whether a given relation is in 5NF, is still unclear. Fortunately relations needing to be decomposed to 5NF are rarely found in practice [DAT90].

2.4.6 Domain-Key Normal Form

A relation is in *domain-key normal form* (DK/NF) if and only if every constraint on the relation is a logical consequence of the key constraints and domain constraints [DAT90, PRA87].

A key constraint is a statement that a certain attribute or attribute combination is a candidate key.

A domain constraint is a statement that the values of a certain attribute lie within a certain set of values [DAT90].

A relation in DK/NF is already in 5NF [DAT90] and suffers from no update anomalies. Therefore no higher normal forms are needed [PRA87]. DK/NF is however not always achievable nor is any general means of converting a relation to DK/NF known.

2.5 Integrity Constraints

A relational database needs to take into account certain integrity constraints associated with relations. For the integrity of a relation to be maintained, the following should be enforced:

2.5.1 Uniqueness property

The value of the primary key in each row of a relation must be unique (since the primary key by definition identifies a tuple within a relation uniquely).

It is also valid to interpret this uniqueness property in terms of object identification: the value of the primary key uniquely identifies the object represented by a particular row within a relation, within the class of objects represented by that specific relation. In any other relation in the database where it is necessary to refer to that object, the reference is made by adding the primary key of the object to the relation referring to it. The fields in a relation, that contains the primary key of another relation, to refer to objects within that relation, are called a *foreign key*. Looking at our example in 3NF (figure 2.4), the *Course*

number field within the Training relation, is a foreign key, as it contains the primary key of a specific course within the course relation.

2.5.2 Referential integrity

Let D be a domain from which one or more primary keys draw their values. Let K be a foreign key, which draws its values from domain D . Every unmarked value that occurs in K must also exist in the database as the value of the primary key on domain D of some base relation. This also implies that a value is marked if and only if it is missing.

In our example (in 3NF), we can explain these terms as follows:

In the Course relation, the course number, must be *unique* for each course.

In the Person relation, we have the *foreign key*: Dept number.

For *referential integrity* to be maintained, in the Person relation, each reference to a Department in the Dept. number field, must be the number of a Department existing in the Department relation. [COD90]

2.6 Conclusion

In this chapter we examined the exact nature of relations, the problems encountered with data stored in relations and the manner in which these problems can be avoided by normalizing relations. In the following chapters, we will take a look at frames and examine the similarities between frames and relations, trying to establish frames as a superclass of relations, making the re-use of relational theory possible for frames.

CHAPTER III : Frames

3.1 Introduction

The two most important concerns in AI research are *knowledge representation* and *search*. Knowledge representation addresses the problem of capturing, in symbolic form, the chunks of knowledge required to simulate intelligence. The available knowledge representation structures limit and direct the way problems are solved in AI applications [MIN91]. Search is the systematic exploration of the problem space for valid solutions. In this chapter we will concern ourselves primarily with the representation of knowledge and specifically with frames.

3.2 Knowledge Representation

In any expert system, it is necessary to represent the essential chunks of knowledge in the problem domain. This is achieved by using one or other knowledge representation scheme. The different knowledge representation schemes available, are classified into the following categories by Mylopoulos and Levesque (1984) [LUG89]:

1. *Logic Representation Schemes*

These schemes use expressions in formal logic as their basis, for example, *first order predicate calculus*.

2. *Procedural Representation Schemes*

These schemes represent knowledge as the set of instructions necessary to solve a particular problem, for example *production systems*.

3. *Network representation Schemes*

These schemes capture knowledge as graphs, where each node represents an object and the arcs represent the relationships between different objects. For example, *semantic nets*.

4. *Structured representation Schemes*

These schemes extend networks, by allowing each node to be a complex data structure. Structured representation schemes include the following: *scripts*, *frames* and *objects*.

3.3 The development of Frames

The notation of a frame, goes back to the idea of a *schema*, developed in Gestalt psychology, and can be characterized as a complex holistic entity. These schemas were used by cognitive psychologists to describe situations by combining the descriptions of the entities involved in a single structure. Later they used schemas to model *human memory*. This finally led to Minsky's proposal, in 1975, to use frames as a representation scheme in AI applications [MIN75]. His informal definition of frames led to the development of a number of frame-based languages.

A year after Minsky's introduction of frames, Bobrow and Winograd introduced KRL (Knowledge Representation Language), a frame-based language that profited considerably from the frame concept, and from object-oriented languages such as Smalltalk [BOB77].

Another frame-based system came with the UNITS system, developed by Stanford University. UNITS, unlike KRL was designed to support expert system development, rather than natural-language processing [TEL88].

Later substantially modified versions of UNITS led to the meta-language RLL (Representation Language Language) and the commercial expert system tool known as KEE (Knowledge Engineering Environment) that was designed and developed by Intellicorp. [REI89, TEL88]

3.4 Frames

Minsky described frames as follows:

"Here is the essence of the theory: When one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a substantial structure called a frame. This is a remembered frame work to be adapted to fit reality by changing details as necessary" [MIN75].

We can think of a frame as a network of nodes and relations. The "top levels" of a frame are fixed, and represent things that are always true about the supposed situation. The lower levels have many terminals (slots) that must be filled by specific *instances* or data. For each terminal or 'slot', the conditions the values assigned to it, must meet can be specified. Simple conditions can be the domain from which the value assignment must be selected. More complex conditions can specify the necessary relationship between

different terminals. A Terminal assignment can be a value, or a pointer to a sub-frame. Collections of related frames, are linked together into *frame systems* [MIN75].

According to Luger and Stubblefield, a slot can have the following contents:

1. *Frame identification information.*

In other words, the unique name, or using database terminology, the primary key of a frame is contained within one of its slots.

2. *Relationships to other Frames.*

Information about inheritance, for instance, reference to the superclass of a frame might be contained within one of its slots.

3. *Descriptors for requirements for a frame to match a certain situation.*

This kind of slots contains constraints, specifying valid entries for other slots.

4. *Procedural information on use of the structure.*

This type of slot, contains attached procedures, that can describe how a slot might be used or which can test for valid entries of other slots.

5. *Frame default information.*

These slots contain values that are taken to be true when no information to the contrary is found.

6. *New instance information.*

The contents of some slots are left unspecified, until needed for solving a particular problem [LUG89].

This clearly is an extension of the original concept of a frame, as specified by Minsky.

3.4.1 Example Frames

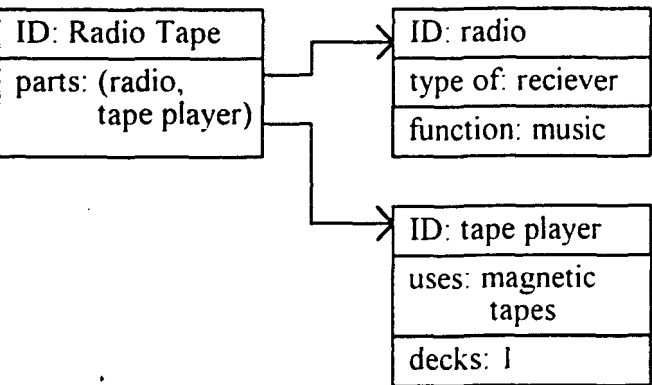


Fig. 3.1 A frame description of a radio tape player.

Figure 3.1 shows how a radio tape player may be represented using the frame approach. In this example we can see the following:

- i) Each frame has a slot containing a unique identifier for the specific frame.
- ii) Each slot has a slot name and a slot value.
- iii) The slot value of a frame can be a pointer to a sub-frame or,
- iv) a slot can be filled with an actual value.

3.5 Different Views of Frames

P. J. Hayes in his article the *Logic of Frames* [HAY80] identifies two different interpretations of what Minsky intended. He calls these two different views the metaphysical and the heuristic interpretations.

The "metaphysical" interpretation is, that to use frames is to make a certain kind of assumption about what entities shall be assumed to exist in the world being described. This view of frames is an assumption about what a program needs to know, rather than how exactly this knowledge will be presented.

The "heuristic" interpretation is, that frames are a computational device for organizing stored representations in computer memory and for retrieval and inference that manipulates these stored representations. [HAY80]

The important point made here is that a given representation language might be implemented in a number of different ways and that frames can be viewed as either, i.e. frames can be seen as a representation language (metaphysical interpretation), or as an implementation (heuristic interpretation). Therefore it is possible to implement frames in a number of different ways. This is illustrated by the number of different frame-based languages that exists (like FRL, KRL and UNITS) all being different implementations of the basic frame concept.

3.6 Application Areas of Frames

Frames were proposed by Minsky primarily for visual scene analysis. He proposed that different frames in a system be used to describe a scene from different view points. The area where frames have been used most often is however in natural-language processing.

Frames also form part of the knowledge representation structures provided by a number of expert system tools like KEE and Level 5 [MIN75, BOB77, TEL88].

'The real force of the frames idea was not at the representational level at all, but rather at the implementation level: a suggestion about how to organize large memories.' [HAY80]

This statement implies that the functionality of frames is more important than the exact structure of frames. Minsky designed frames as a representational structure to simplify problems, so that they are easier to solve.

3.6.1 Frames and Vision

Frames can be used to symbolically represent three dimensional images: rotation of the image is then seen as a transformation between different sub-frames of the image. Frames can also be used in visual scene analysis. If the current scene is a bedroom, the bedroom frame might be selected and can be used to identify the objects in the bedroom by comparing them to the objects known to occur most often in a bedroom. This is much easier than comparing the objects to all known objects. If on the other hand certain objects in a room are identified, assumptions can be made about the nature of the room. For instance a room containing a large number of chairs and a stage, is most probably a theater of some kind.

3.6.2 Frames and text

To understand linguistic activity involves larger structures than can be described with sentential grammar. The necessary background knowledge to understand language might be provided by frames. The following levels of frame structures are suggested to cope with language [MIN75]:

- a) *Surface Syntactic Frames* - Verb and noun structures, prepositional and word-order indicator conventions.
- b) *Surface Semantic Frames* - Action-centered meanings of words. Containing qualifiers and relations concerning participants, instruments, trajectories and strategies, goals, consequences and side-effects.
- c) *Thematic Frames* - Scenarios concerned with topics, activities, portraits, setting. Outstanding problems and strategies commonly connected with topics.

- d) *Narrative Frames* - Skeleton forms for typical stories, explanations, and arguments. Conventions about foci, protagonists, plot forms, development etc., designed to help a listener construct a new instantiated Thematic Frame in his own mind.

3.6.3 Frames and Expert Systems

Frames are used as a knowledge representation structure for developing expert systems. UNITS, for instance, is a frame-based expert system development tool [TEL88]. Most expert systems, however, that support frames, use them in conjunction with rules. KEE, a commercial expert system tool, uses frames to create structured hierarchies of rule sets to control search in rule-based expert systems.

3.7 Frames as an Object-oriented Representation Scheme

Frames and objects have the following in common:

1. They group information about single entities together.
2. Both frames and objects support inheritance, frames support superclasses by the addition of slots with labels like A-kind-of.
3. Both specify some kind of procedural attachments (called methods in the object oriented terminology.)

Frames and objects differ in the sense that frames do not require data encapsulation, or at least data encapsulation is not an issue when using frames. They also differ in the way inheritance is implemented. In the object oriented approach, a sub-class inherits all its superclass' properties (data structures), as well as all its methods. In the frame-based approach, inheritance is usually limited to the 'data' and doesn't include the procedural attachments. Frames however support more complex relationships between frame classes than the sub-class, superclass relationship, provided by objects [NEE91].

3.8 Semantics and Basic Characteristics of Frames

Looking only at the structure of frames is not enough. For complete understanding of frames, it is necessary to look at the semantics, or meaning of frames. It is important to understand how knowledge is represented in terms of frames and what this knowledge means as well as what the effect of different frame operations will be on this knowledge. A frame instance represents a specific object, i.e., an individual instance of a specific class of objects. Each slot of the frame represents an attribute of the object, or a *relationship*

between that frame and another frame [HAY80]. Minsky suggested that this relationship might be implemented as a pointer to a sub-frame. However, we would suggest that implementing this relationship as a foreign key might be more effective to make the knowledge more meaningful. Hayes says that frames are simply an alternative for expressing relationships between individuals, i.e., for predicate logic. Frames however have some important complexities not provided for by predicate logic, like default values. To understand frames more fully, we will take a look at the ways in which they are used.

3.8.1 Frame Inference

Frame inference is classified into three groups by Hayes:

a) Frame instantiation:

A frame is instantiated, in other words, a new frame instance is created by finding values for its slots.

b) "Criteriality" inference:

Criteriality inference states that if fillers for all the slots of a frame can be found, the assumption that the concept represented by the frame exists is true.

c) Matching:

Matching is defined by Hayes as a form of frame reasoning. He states that matching is the process of searching through a collection of frames to determine if a certain assumption is valid [HAY80]. Minsky's definition of matching is slightly different. He defines matching as a request for a new frame. The request can take one of the following forms:

- i) Find a frame with as many terminals as possible in common with a desired frame,
- ii) Find or build a frame with certain properties.
- iii) Find a frame that is like the old frame except for certain differences between them [MIN75].

Requests for frames should make provision for the following *excuses* for an inexact match:

OCCLUSION: In vision analysis for example, a part of an object might be obscured making it impossible to observe some critical aspect of its frame definition. A table for instance might be specified as having four legs, but at certain times some of its legs might not be visible, being obscured by other objects.

FUNCTIONAL VARIANT: In certain instances the *function* is more important than exact physical descriptions. For example, the legs of a chair's function is support, whatever their exact geometry, might be.

BROKEN: A visually inexact match might be explained by a physical defect, since reality is rarely perfect.

PARASITIC CONTEXTS: An object might look exactly like a chair except for its size, being too small, because it is a toy chair. Frames should make provision for recognizing models of reality as such. A related example in text understanding is the use of frames for understanding figurative speech [MIN75].

3.8.2 Seeing as

Frames might be interpreted as a way of looking at an entity that gives a correct view of this entity. This can create problems when seeing an object as being like another. The statement: "The little boy, Peter, is a pig," can cause all kinds of problems in a frame-based text understanding system. This sentence doesn't mean that Peter is a four legged animal, that might be turned into bacon. It is therefore necessary to be able to reason about Peter as a pig, knowing that he is in fact not one. The frame-based knowledge representation should make provision for distinguishing between likeness to an object and being an instance of that object [HAY80].

3.8.3 Defaults

An important aspect of frame reasoning is the idea of default values: A default is a value associated with a slot and is used in the absence of explicit information to the contrary. The slots using default values in a frame might be proven to be wrong with the addition of any new information and it is therefore necessary to validate the contents of these slots when new relevant information is found [HAY80]. According to Minsky, default assignments are weakly-bound to the slots of a frame [MIN75].

3.8.4 Reflexive Reasoning

Reflexive Reasoning is the use of existing knowledge to try to determine whether a new piece of knowledge is valid (true). Because of the structure of frames, it should be possible to validate a new piece of knowledge and determine if this knowledge is confirmed by existing knowledge. If the new knowledge is contradicted by default assumptions in existing frames, it is necessary to change the values of these slots to accommodate the new knowledge, if this new knowledge however is contradicted by the contents of slots known to be true, this new knowledge should be rejected.

3.9 Conclusion

In this chapter we looked at the original definition of a frame, as proposed by M. Minsky [MIN75]. This definition is still somewhat vague but provides the basis for a number of more 'formal' definitions of frames. In the next chapter, we examine some of these definitions of frames and conclude by proposing our own formal definition of a frame.

CHAPTER IV : Formal Definitions of Frames

4.1 Introduction

Minsky's proposal to use frames as a representation structure [MIN75] is rather informal. To formulate any meaningful theories about frames, a more rigid definition of the frame structure is necessary. A number of formal definitions of the frame structure are found in the literature. These definitions can be classified into two broad categories: *Representation Languages* (e.g. FRL and KRL) and *Mathematical Definitions* (not necessarily linked to a specific implementation of frames).

4.2 Representation Languages

In this section we will look at two frame-based languages and their relation to the original definition of frames

4.2.1 Frame representation Language (FRL)

FRL is one of the more well-known programming systems offering support for frames. In FRL each *frame* consists of a number of *slots*. There are two basic kinds of slots:

AKO slots (AKO = a kind of) and other slots. AKO slots define inheritance between different frames.

The other slots, or *localized slots*, are defined to consist of a number of *facets*. These facets can be classified as follows: *Value* facets (used for storing the value of a property), *Default* facets (providing background assumptions in cases where a specific value may be unknown. Each ordinary slot may also include a number of 'demon' procedures, which are activated, by accessing the values of a slot in a particular manner. The demon procedures are classified as **IF-NEEDED**, **IF-ADDED** and **IF-REMOVED** demons [KRE90]. Figure 4.1 shows the structure of a frame in FRL.

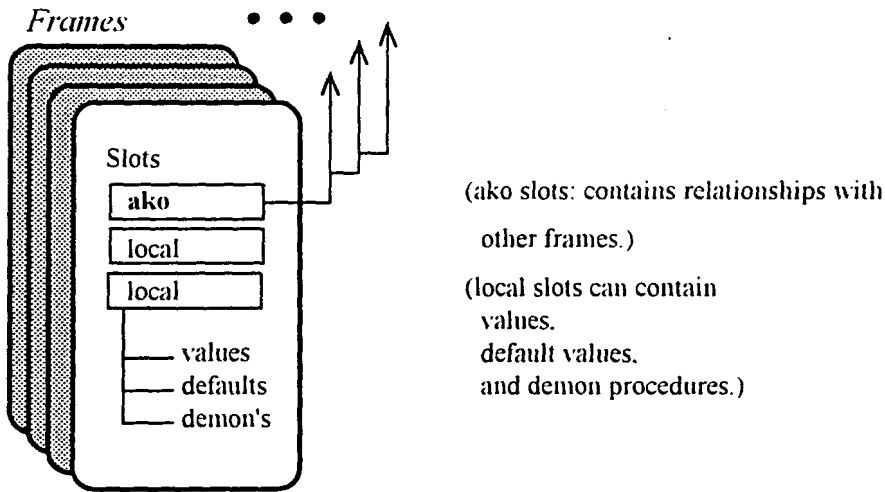


Fig. 4.1 Structure of a frame in FRL [KRE90].

4.2.2 Knowledge Representation Language (KRL)

D.G. Bobrow et al. (1977), describes a frame in the following manner: A frame is a data structure, potentially with an associated *name*, a reference to a *prototype* frame, and a set of *slots*. A frame that has another frame, as prototype, is called an *instance* of the prototype frame. A frame's important substructures and its relations to other frames are defined in its slots. A slot has a *slot-name*, a *value*, and possibly a set of attached procedures. The value of a slot might contain a specific value, a reference to a sub-frame, or in the case of a prototype, default and constraint information. All the slots of a frame need not be filled, only those necessary for the current reasoning process [BOB77].

4.3 Mathematical Definitions

There exist a number of different formal definitions of frames, that is not associated with a specific application. These formal definitions can be classified as mathematical definitions.

4.3.1 Algebraic Structured Model of Frames

This algebraic model of frames is defined in an article by J. B. Castellanos Peñuela [PEN89]. According to this article, a frame is a structure formed by different slots that can either have expressions or be empty.

For a frame M^i : M denotes the frame class and ' i ' denotes a specific instance of the frame class M .

Two frames M^i and M^j are related as follows, in each row ' k ' of M^i there is the same number of slots as in row ' k ' of M^j . A class of frames M has the following structure:

slot(1,1)	slot(1,2)	slot(1,t)
		.	
slot(2,1)	slot(2,2)	slot(2,p)
		.	
.....
.	.	.	.
slot(k,1)	slot(k,2)	slot(k,m)
		.	

All the frames in class M have k rows; and each of them has the same number of slots in a particular row. $SLOT^i_{(i,j)}$ is the slot in position (i,j) of frame M^i [PEN89].

The following operations are defined on frames, using this notation.

a) *Frames equality*

Two frames M^i and M^j is equal if:

- i) They belong to the same frame class, M , and
- ii) $SLOT^i_{(x,y)} = SLOT^j_{(x,y)}$ for valid values of x and y . [PEN89]

b) *Frames addition*

In the set of frames, the frame $(M^i + M^j)$ is defined as the frame containing in each $SLOT_{(x,y)}$ the contents of $SLOT^i_{(x,y)}$ plus the contents of $SLOT^j_{(x,y)}$ joined by the word "or", a non-exclusive disjunction.[PEN89]

This definition of frames does not include references to many of the essential characteristics of the frame structure in general, no reference are made to defaults, or procedural attachments.

4.3.2 Predicate Algebra Notation of Frames

The predicate algebra notation of frames is found in [THA88]. Frames or units are defined as several related predicate logic formulas aggregated into larger structures that are

identified with the characteristic objects of the domain of discourse. When information about one of these objects is needed, the appropriate frame is accessed and all the relevant facts about the object is retrieved at once. This representation of a frame takes the general form:

$$Frame_name(attribute_j, value_j) (j=1,...,n).$$

The definition of a frame, consist of a collection of related binary-predicates. Each pair (*attribute,value*) in the frame is called a slot; the pair, using this terminology, can also be called (*slot-name, slot-value*). This formal definition of frames, only makes provision for slot values. No provision is made for attached procedures or default values.

4.3.3 Functional mapping definition of Frames

This definition of frames is given by Reimer and Scheck in their article: *A frame-based knowledge representation model and its mapping to nested relations* [REI89]. They define a frame knowledge-base as a sequence of three mappings. The first assigns a *frame name* to a *frame structure*. The frame structure is given by a second mapping which is defined upon the set of slot names for that frame. The third mapping specifies the slot structure, i.e. the actual permitted entries of the slot.

$$FRAMES = \{f | f: Fnames \rightarrow SLOTS\}$$

Fnames denotes the set of frame names and *SLOTS* denotes the set of mappings each of which represents a frame by specifying its slots:

$$SLOTS = \{f | f: Snames \rightarrow SENTRY\}$$

Snames denotes the set of slot names and *SENTRY* denotes the set of mappings each of which yields the actual and permitted entries of a slot:

$$SENTRY = \{f | f: \{act, perm\} \rightarrow 2^{Entries}\}$$

The set *Entries* comprise frame names as well as terminal values which are just plain character strings:

$$Entries = Fnames \cup Strings.$$

It is important to note that frames are defined in this manner by Reiter and Schek, specifically to provide a mapping from frames to nested relations. It is for the same reason that we will provide our own formalization of frames in the next section - to facilitate the comparison of frames and relations.

4.4 Formal Definition of a Frame

Taking into account the preceding definitions of a frame and the way in which relations were defined in chapter 2, we define a frame as follows:

A frame class consists of two parts, the *prototype* of the frame class, and specific *instances* of the frame class.

A **prototype** of a frame $F_p = (S, P)$

where:

S is the set of slots, $S = (s_1, s_2, \dots, s_k)$, $k \geq 1$.

$s_i = (n, c, d)$.

n = slot-name.

c = constraint/ slot type, i.e. domain for slot values.

d = default value.

P is the set of procedures, associated, with each frame,

$P = \{p_1, p_2, \dots, p_m\}$, $m \geq 0$. If $m=0$ then P is defined as the empty set.

Note: We only define a set of attached procedures. This does not mean that our model does not support demons. The distinction between procedures and demons is in the way they are activated. Therefore whenever we speak of attached procedures, we might as well have used the term demons, as the way in which a procedure or demon is activated has no effect on our model.

An **instance** of a frame $F_F = (v_1, v_2, \dots, v_k)$, $k \geq 1$.

where:

v_i = value of slot i of Frame instance F_F .

4.4.1 Example

Frames, describing cats and cat owners, using our definition of frames look like this:

Prototypes:

CATS

n	c	d
Name	Character String	
Lives	Integer	9
Legs	Integer	4
Fur Color	Colors	Ginger

P={ Try_to_ Kill, Find_owner }

CAT OWNER

n	c	d
Name	Character String	
Cat Name	Character String	
Address	Character String	

P={ }

Instances:

CATS

Name	Lives	Legs	Fur Color
Winston			
Max	8		Brown
Watson	1	3	Black

CAT OWNER

Name	Cat Name	Address
Peter	Winston	Hyde Park
Peter	Max	

From this example we can gather that Winston is a ginger cat with 9 lives and 4 legs and he, as well as a cat named Max, is owned by Peter. On the other hand, Watson is a very unlucky cat, having only 1 life left, 3 legs and no owner.

4.5 Conclusion

In this chapter we looked at a number of different definitions of frame-like structures. Taking into account these definitions, and the original definition of frames as given by Minsky [MIN75], we presented a formal definition of frames, using set notation. Our definition of frames provides for the notion of different frame classes, instances of frame classes, default slot values, constraints on slot values, and attached procedures.

In chapter 5, we will try to motivate the normalization of frames by looking at problems associated with frames, and in chapter 6, we will use our definition of frames, to provide a basis for comparison between frames and relations, and to define the normalization of frames.

CHAPTER V : Problems With Frames

5.1 Introduction

In this chapter, we look at the problems associated with representing knowledge in symbolic form. We then examine specific problems associated with frames, as an example of a knowledge representation structure. After identifying these problems, we take a look at the advantages of using frames as the primary knowledge representation scheme for a knowledge-based system.

5.2 Representing Knowledge

As we have stated before, knowledge representation is one of the primary concerns of AI research. Although a vast amount of research has already been done into knowledge representation, there are still many problems experienced when knowledge is encoded using the available representation schemes. These problems are generally caused by:

- (i) The way a particular set of knowledge is encoded using the available knowledge representation structures.
- (ii) Restrictions imposed by the use of a specific knowledge representation scheme.
- (iii) By a combination of (i) and (ii).

In this section we look at specific problems encountered with knowledge representation, that is not associated with any particular representation scheme.

5.2.1 Conflicting requirements

A problem solving process needs knowledge in a form that is both *expressive* and *efficient*. In other words, the knowledge representation should be able to express all the necessary knowledge for solving a problem in such a way that the problem can be solved within an acceptable time limit. Unfortunately it is generally so that the more expressive a representation scheme is the less its computational efficiency is and it is therefore often necessary to sacrifice expressiveness for increased efficiency. [LUG89]

5.2.2 Blindness

The term blindness refers to the inability to solve a problem even though all the necessary knowledge to solve the problem is available. Blindness is a result of the way knowledge is represented: It is possible, using any knowledge representation scheme, to represent the same piece of knowledge in several different ways, each of which will facilitate certain inferences on the knowledge and obscure others. It may therefore happen that all the knowledge necessary for solving a particular problem is available, but because the knowledge was structured with certain kinds of problems in mind, an AI system using this knowledge might still be unable to solve the problem. Blindness illustrates the fact that the problems of representing real world knowledge are not necessarily caused by defects in current knowledge representation schemes, but in the way that they are used.

5.2.3 Common sense knowledge

Even the most elementary of decisions made by humans on a day to day basis requires a vast amount of so called common sense knowledge. Again the problem is not that current representation schemes are inadequate for representing common sense knowledge, but rather how to encode this knowledge into a specific structure in such a way that it would be meaningful [LUG89, TEL88].

5.2.4 Self-knowledge and meta-knowledge

Most knowledge representation schemes lack the ability to include detailed knowledge about the structure of the knowledge. This is largely a result of the separation of the knowledge representation structure and the part of an AI system (inference engine) that searches through the knowledge [KRE90, TEL88].

5.3 Problems with Frames as a Knowledge Representation Structure

Frames as a knowledge representation scheme has certain unique problems of its own in addition to the problems mentioned in section 5.2. In this section we examine these problems.

5.3.1 Lack of theoretical foundations and formalisms for using frames:

The original definition of frames by Minsky [MIN75] was rather vague and it led to a number of different 'formal' definitions of frames [BOB77, THA88, PEN89,

5.2.2 Blindness

The term blindness refers to the inability to solve a problem even though all the necessary knowledge to solve the problem is available. Blindness is a result of the way knowledge is represented: It is possible, using any knowledge representation scheme, to represent the same piece of knowledge in several different ways, each of which will facilitate certain inferences on the knowledge and obscure others. It may therefore happen that all the knowledge necessary for solving a particular problem is available, but because the knowledge was structured with certain kinds of problems in mind, an AI system using this knowledge might still be unable to solve the problem. Blindness illustrates the fact that the problems of representing real world knowledge are not necessarily caused by defects in current knowledge representation schemes, but in the way that they are used.

5.2.3 Common sense knowledge

Even the most elementary of decisions made by humans on a day to day basis requires a vast amount of so called common sense knowledge. Again the problem is not that current representation schemes are inadequate for representing common sense knowledge, but rather how to encode this knowledge into a specific structure in such a way that it would be meaningful [LUG89, TEL88].

5.2.4 Self-knowledge and meta-knowledge

Most knowledge representation schemes lack the ability to include detailed knowledge about the structure of the knowledge. This is largely a result of the separation of the knowledge representation structure and the part of an AI system (inference engine) that searches through the knowledge [KRE90, TEL88].

5.3 Problems with Frames as a Knowledge Representation Structure

Frames as a knowledge representation scheme has certain unique problems of its own in addition to the problems mentioned in section 5.2. In this section we examine these problems.

5.3.1 Lack of theoretical foundations and formalisms for using frames:

The original definition of frames by Minsky [MIN75] was rather vague and it led to a number of different 'formal' definitions of frames [BOB77, THA88, PEN89,

REI89, KRE90]. This in itself is not a problem, when one realizes that Minsky's definition of a frame should be seen as a general description of a structured representation scheme and that all these 'formal' definitions can be seen as implementations of this general concept. However even though the structural characteristics of frames are quite clearly defined, the following problems still exist:

- i) There is no formal specification of how to encode knowledge as frames: It is possible to encode a collection of knowledge in a number of different ways using frames, some of which will work better than others. At this time it is still not clear how to select the 'best' frame representation for a given set of knowledge, or how to transform from one frame-based representation to a 'better' representation of the same set of knowledge.
- ii) There is no formal definition of valid operations for frames.
- ii) There are no definite guide-lines of how to search a collection of frames to solve a particular problem.

In short, the structure of frames is clearly defined and well understood, but the functional components (operations and search) of frames are still vague.

5.3.2 Default Reasoning:

Default reasoning is a very powerful characteristic of frames, but it can cause problems. These problems are usually a result of the manner in which default reasoning is implemented. The system should always keep in mind that default values might be wrong and should be able to recover once a default value has been proven wrong. Default values should only be used if no other way to determine a value can be found.

5.3.3 Ineffective search

Mechanisms to optimize search through a large number of frames do not exist in many frame-based systems. It is necessary to structure knowledge in such a way within frames to improve the effectiveness of search. This might be done by defining key slots for frames and indexing these key slots.

5.3.4 Enforcing of Integrity constraints

It is possible to enforce integrity constraints within frames by defining valid domains for slot values and checking slot values with attached procedures (demons), but the responsibility to do this lies with the user. [PAT91]

5.4 Advantages of Using a Frame-based Representation Scheme

In spite of the problems associated with frames, frames are still very widely used as a knowledge representation scheme in many expert system tools because there are a number of advantages to using frames. In this section we will take a look at some of these advantages.

5.4.1 Natural Representation of Knowledge

One of the requirements of a knowledge representation scheme is that it should provide a natural way of representing knowledge [LUG89]. Frames make use of the natural tendency of humans to group knowledge into classes and can therefore be said to fulfill this requirement.

5.4.2 Multiple relationships

Frames not only provide for inheritance (using AKO slots or superclass slots), but makes it possible to define much more subtle relationships between different objects, and different classes of objects.

5.4.3 Combining declarative and procedural information

The motivation behind combining procedural information with the structural information makes it possible to program in terms of generic operations and moves control to the object itself. It is possible to design event driven programs using frames with attached procedures or demons because of this integration.

5.5 Conclusion

In this chapter we saw that although frames are a very popular representation scheme for use in knowledge-based systems, there are still a number of problems associated with them. In the next chapter, we will attempt to solve some of these problems by establishing frames as a superclass of relations.

Chapter VI : The Normalization of Frames

6.1 Introduction

In previous chapters, we defined both relations and frames. In the first part of this chapter we compare these two models and identify their structural similarities. We also take a look at the main differences between the two models. Taking into account the similarities and differences between these two structures, we then redefine normalization, for frames. In chapter 8 we will look at other attempts to establish a relationship between knowledge representation and databases, when we look at the integration of data and knowledge-bases.

6.2 Comparison of Frames and Relations

The following table contains the definitions of Frames and Relations, with the definition of similar parts of the two models in the same rows.

Definition of Frames	Definition of Relations [DAT90]
A frame class consists of two parts, the <i>prototype</i> of the frame class, and specific <i>instances</i> of the frame class.	A relation R on a collection of domains D_1, D_2, \dots, D_n (not necessarily distinct) consists of two parts, the <i>heading</i> and the <i>body</i> .
<p>A prototype of a frame $F_p = (S, P)$ where:</p> <p>S is the set of slots, $S = \{s_1, s_2, \dots, s_k\}$, $k \geq 1$.</p> <p>$s_i = (n, c, d)$, $1 \leq i \leq k$</p> <p>n = slot-name.</p> <p>c = constraint/ slot type, i.e. domain for slot values.</p> <p>d = default value.</p>	<p>The heading consists of a fixed set of attribute-domain pairs,</p> $\{ (A_1:D_1), (A_2:D_2), \dots, (A_n:D_n) \}$ <p>such that each attribute A_j corresponds to exactly one of the underlying domains D_j (for $j = 1, 2, \dots, n$, where n is the degree of the relation).</p>

<p>P is the set of Procedures, associated, with each frame,</p> <p>$P = \{p_1, p_2, \dots, p_m\}$, $m \geq 0$. If $m=0$ then P is defined as the empty set.</p>	<p>No equivalent in the definition of relations.</p>
<p>An instance of a frame</p> <p>$F_F = \{(n_1:v_1), (n_2:v_2), \dots, (n_k:v_k)\}$, $k \geq 1$. where: n_i = name of slot i, v_i = value of slot i of Frame instance F_F.</p>	<p>The body consists of a time-varying set of n-tuples, where each tuple in turn consists of a set of attribute-value pairs,</p> <p>$\{ (A_1:v_{i1}), (A_2:v_{i2}), \dots, (A_n:v_{in}) \}$</p> <p>(for $i = 1, 2, \dots, m$, where m is the cardinality of the specific relation).</p>

From the table the following similarities between the two structures can be seen:

- a) The *Prototype* of a frame class is similar to the *Heading* of a Relation.
- b) The *Instances* of a frame class is similar to the *Body* of a Relation.
- c) A *slot* of a frame is similar to an *attribute* of a relation.
- d) Each slot has certain *constraints* or values from which the slot can take its values, and each attribute comes from a specific *domain*.
- e) A collection of instances of frames in the same frame class is similar to the body of a relation.

Also using the table we can identify the following differences between the two structures:

- a) In the definition of a frame prototype, the slot-name, n , in the triple $s_i=(n,c,d)$ is equivalent to the Attribute, A_j , of a relation and the constraint c , is equivalent to the domain D_j . The default values defined for frames however doesn't have an equivalent definition in terms of relations.
- b) The set of procedures associated with a frame class also doesn't have a related definition in terms of relations.

When looking at these similarities and differences, it is evident that the differences between frames and relations are isolated to the prototype/ heading parts of the two models; this means that both structures might be represented as two-dimensional tables.

Relations and Frames can be seen as exactly the same structure if the following holds:

- (i) The set of procedures associated with each frame class is defined as the empty set, and
- (ii) All default values are defined as being empty.

This means that Frames might be seen as a superclass of relations, having all the properties of relations, plus associated procedures and default values.

It is possible to include default values in a relational structure by defining within the body of the relation, one tuple containing default values for all the attributes and using these default values whenever an attribute is empty (figure 6.1). It is however impossible to implement associated procedures for the relational model without any substantial change to the model. This means that when redefining normalization for frames, special attention will have to be paid to the effect normalization will have on the associated procedures.

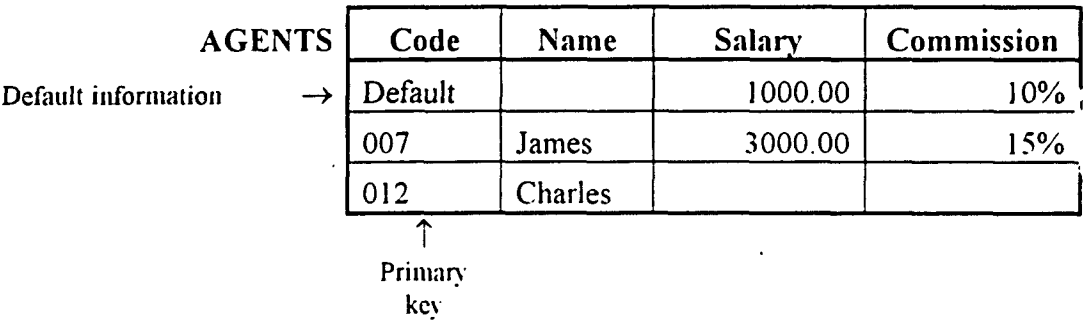


Fig. 6.1 A relation containing default information.

Procedures associated with frames can be classified in the following ways: [BAR81, BOB77, KRE90]

- i) They can be classified by the way they are triggered. In general procedures will be triggered when a given slot is empty or, when a given slot is needed. A procedure can also be activated when the value of the slot it is attached to is changed.
- ii) They can be classified by their function. An attached procedure might do any or a combination of the following:
 - Prompt the user for the value of a slot.
 - Perform integrity checks on the value of a slot.
 - Derive the slot value from other slots in the frame system.
 - Propagating information when a slot value is obtained.

- iii) They can be classified by the way they are used into two general classes: *servants* and *demons*. Demons are procedures that are activated automatically, and servants are procedures that are activated on demand. In general a servant procedure will only be activated, when it is invoked by the user or by a demon procedure.

According to Codd [COD90] the main difference between a database and a knowledge-base is the following: A database is in general large but not very rich, in other words a database in general contains relations with a small number of attributes but with a large number of tuples. A knowledge-base is in general rich but not very large. Real world knowledge-bases will however tend to be both large and rich. Normalization will not be able to do anything to the size of these knowledge-bases, but it will minimize the richness of each frame class, making operations on the knowledge-base easier and more effective.

6.3 Normalization

We will try to illustrate the basic issues in frame normalization with appropriate examples. The knowledge-base used to illustrate normalization contains the following frames:

Person (Name, Sex, Date_of_birth, Age, Brothers, Sisters)

Marriage (Husband, Wife, Children, Start_date)

Note: We use the term *frames* when talking about the prototypes of frame classes and when we are talking about frame instances, it should however be clear from the context in which the term frames is used, whether we are referring to frame prototypes or frame instances.

We will assume the following for this example:

- i) Each person has a unique name. In the absence of unique names, it will be necessary to use an artificial identifier to uniquely identify each person. This assumption is only made for the sake of simplicity.
- ii) The Husband slot will contain the name of a male person.
- iii) The Wife slot will contain the name of a female person.
- iv) Each person can only be involved in one marriage.

We will have the following attached procedures:

Person:

IF-NEEDED-age [compute age from current date and date of birth]

CHECKS:

Name unique.

Marriage:

CHECKS:

Husband male,

Wife female,

Husband, Wife pair unique,

Husband unique, Wife unique (enforcing that each person can have only one marriage)

These frames might contain the following instances:

Person:

Name	Sex	Date_of birth	Age	Brothers	Sisters
Abe	Male	45/08/03			
Sandra	Female	45/05/31			
Peter	Male	68/09/16			Mary Ann
Mary	Female	70/08/07		Peter	Ann
Ann	Female	73/07/04		Peter	Ann
Suzanne	Female	67/04/02			
Pat	Male	68/07/07		John	
John	Male	75/01/05		Pat	
Pete	Male	90/03/12		Sam	
Sam	Male	92/03/06		Pete	

(Age slot only filled if needed by attached procedure: IF-NEEDED-age)

Marriage:

Husband	Wife	Children	Marriage date
Abe	Sandra	Peter Mary Ann	65/05/30
Peter	Suzanne	Pete Sam	89/02/14
Pat	Mary		91/12/10

Fig. 6.2 Example frame instances (unnormalized)

6.3.1 Functional dependence and primary keys

We can rephrase the definition of functional dependence and of a primary key as follows for frames:

Functional Dependence:

Given a frame F, slot Y of F is *functionally dependent* on slot X of F - symbolically: $F.X \rightarrow F.Y$ (read "F.X functionally determines F.Y")
- if and only if each X-value in F has associated with it precisely one Y-value in F (at any one time). Slots X and Y may be composite.

Primary key:

Given a frame F, slot F.Y (F.Y may be composite) is the *primary key* of F if:

1. All the slots in F are functionally dependent on F.Y.
2. No subset of the slots in F.Y also has property 1.

The frames in the example have the following primary keys:

Person: Name.
Marriage: Husband, Wife.

Note: Because of the uniqueness requirement of a primary key it is not possible to specify default values for slots forming part of the primary key.

Defining a primary key for a frame class has the following advantages:

- i) Clarifying the semantics (meaning) of a frame
- ii) It makes referring to a specific frame easy and natural, using foreign keys (the inclusion of the primary key of one frame, as a reference to that frame, in another).
- iii) It can make search through the knowledge-base more effective, if indexes are maintained on the primary keys and foreign keys of each frame class. Using indexes makes it possible to go directly to a frame whose primary key is known instead of searching through all the frames in the knowledge-base.

6.3.2 Non-Derivable normal form

A slot value is *derivable*, if it is possible to determine a value for that slot by using information already stored within the knowledge-base. In the example, the age slot are left empty and only computed whenever it is needed, by the IF-NEEDED-age procedure. The age slot is an example of a derivable slot.

Other examples of derivable slots, are the *Brothers*, and *Sisters* slots. Explicitly storing this information in the frame causes the following problems during updates: When a new baby is born, it is necessary to update the frames of all his brothers and sisters and it is also necessary to search for all the brothers and sisters and add them to this new instance of the person frame.

A frame is in *non-derivable normal* form if all the derivable slots are left empty and only filled if their values are needed.

The value of a derivable slot might only be used without activating the associated procedure, if it is known that the knowledge-base has not changed since computing it the last time, but in general the value of a derivable slot has to be computed every time it is used.

Figure 6.3 illustrates non-derivable normal form, for the example we introduced earlier in this section.

Person:

Name	Sex	Date_of birth	Age	Brothers	Sisters
Abe	Male	45/08/03			
Sandra	Female	45/05/31			
Peter	Male	68/09/16			
Mary	Female	70/08/07			
Ann	Female	73/07/04			
Suzanne	Female	67/04/02			
Pat	Male	68/07/07			
John	Male	75/01/05			
Pete	Male	90/03/12			
Sam	Male	92/03/06			

(Age slot only filled if needed by attached procedure: IF-NEEDED-age)
(Brother slot only filled if needed by attached procedure: IF-NEEDED-brothers)
(Sisters slot only filled if needed by attached procedure: IF-NEEDED-sisters)

IF-NEEDED-brothers (name of target person)
Searches for all male members with same parents as target person.
Returns all brothers of target person

IF-NEEDED-sisters (name of target person)
Searches for all female members with same parents as target person.
Returns all sisters of target person

Marriage:

Husband	Wife	Children	Marriage date
Abe	Sandra	Peter Mary Ann	65/05/30
Peter	Suzanne	Pete Sam	89/02/14
Pat	Mary		91/12/10

Fig. 6.3 Example frame instances in non-derivable normal form.

6.3.3 First Normal Form

A relation is in first normal form, if it contains no repeating groups, or more simply stated if all its attributes are atomic. For frames, we will have to take into account the non-derivable normal form and modify the definition to the following:

A frame is in *first normal form* (1NF) if: it is already in non-derivable normal form, it contains no repeating groups, and all its IF-NEEDED procedures, computing derivable slot values, returns only one value at a time.

The IF-NEEDED procedure's returning only one value is required to maintain the atomic value characteristic, associated with the definition of relations, and our definition of frames.

In First normal form for frames, our example will look like this:

Person:

Name	Sex	Date_of birth	Age	Brother	Sister
Abe	Male	45/08/03			
Sandra	Female	45/05/31			
Peter	Male	68/09/16			
Mary	Female	70/08/07			
Ann	Female	73/07/04			
Suzanne	Female	67/04/02			
Pat	Male	68/07/07			
John	Male	75/01/05			
Pete	Male	90/03/12			
Sam	Male	92/03/06			

(Age slot only filled if needed by attached procedure: IF-NEEDED-age)
(Brother slot only filled if needed by attached procedure: IF-NEEDED-brother)
(Sister slot only filled if needed by attached procedure: IF-NEEDED-sister)

IF-NEEDED-brother (name of target person, n)	
	Searches for all male members with same parents as target person.
Returns the n'th brother found.	

IF-NEEDED-sister (name of target person, n)	
	Searches for all female members with same parents as target person.
Returns the n'th sister found.	

Marriage:

Husband	Wife	Children	Marriage Date
Abe	Sandra	Peter	65/05/30
Abe	Sandra	Mary	65/05/30
Abe	Sandra	Ann	65/05/30
Peter	Suzanne	Pete	89/02/14
Peter	Suzanne	Sam	89/02/14
Pat	Mary		91/12/10

Fig. 6.4 Example frame instances in first normal form.

6.3.4 Second Normal Form

We know that a relation is in *second normal form* (2NF) if it is in first normal form and no nonkey attribute is dependent on only a portion of the primary key.

We can rephrase these definitions as follows for frames:

A *nonkey slot* is a slot that does not form part of the primary key

A frame is in second normal form (2NF) if it is in first normal form and no nonkey, slot, that is not a derivable slot, is dependent on only a portion of the primary key.

When looking at functional dependence, it is important to realize that an attached procedure's slot is functionally dependent on the slots used by the procedure as parameters.

In our example in 1NF, the primary key of marriage is the collection of slots husband, wife, children. The date of the marriage is however only dependent on the husband and wife slots. This can cause inconsistent data, because it is possible for two different dates of marriage to exist within the knowledge-base, for the same marriage. To avoid this it is necessary for the frames to be normalized to second normal form.

In Second normal form for frames, our example looks like this:

Person:

Name	Sex	Date_of birth	Age	Brother	Sister
Abe	Male	45/08/03			
Sandra	Female	45/05/31			
Peter	Male	68/09/16			
Mary	Female	70/08/07			
Ann	Female	73/07/04			
Suzanne	Female	67/04/02			
Pat	Male	68/07/07			
John	Male	75/01/05			
Pete	Male	90/03/12			
Sam	Male	92/03/06			

- (Age slot only filled if needed by attached procedure: IF-NEEDED-age)
- (Brother slot only filled if needed by attached procedure: IF-NEEDED-brother)
- (Sister slot only filled if needed by attached procedure: IF-NEEDED-sister)

IF-NEEDED-brother (name of target person, n)

Searches for all male members with same parents as target person.

Returns the n'th brother found.

IF-NEEDED-sister (name of target person, n)

Searches for all female members with same parents as target person.

Returns the n'th sister found.

Marriage:

Husband	Wife	Marriage Date
Abe	Sandra	65/05/30
Peter	Suzanne	89/02/14
Pat	Mary	91/12/10

Children:

Husband	Wife	Child
Abe	Sandra	Peter
Abe	Sandra	Mary
Abe	Sandra	Ann
Peter	Suzanne	Pete
Peter	Suzanne	Sam

Fig. 6.5 Example frame instances in second normal form.

6.3.5 Third Normal Form

A relation is in *third normal form* (3NF) if it is second normal form and if the only determinants it contains are candidate keys. Rephrasing this definition, we get the following definition of third normal form for frames:

A *candidate key* is any slot, which could also function as the primary key.

Any slot (or collection of slots) that determines another slot is called a *determinant*.

A frame is in *third normal form* (3NF) if it is in second normal form and if the only determinants it contains are candidate keys or the determinants it does contain, only determines derivable slots, by being parameters in attached procedures.

In our example, the date of birth slot is a determinant of the age slot, but this doesn't cause any problems, because values for the age slot are only computed when needed and because the same procedure is used every time, two different frames with the same date of birth, will always have the same age. Because of this our example is already in third normal form, as all its determinants only determines derivable slots.

6.3.6 Fourth Normal Form

Rephrasing the definitions for fourth normal form for relations we propose the following definitions for frames:

A frame is in fourth normal form if it is in third normal form and there are no multivalued dependencies.

In a frame with slots A, B, and C, there is a multivalued dependency of slot B on slot A if a value for A is associated with a specific collection of values for B, independent of any values for C.

Given this definition, it is important to note that it is possible to avoid the problem of multivalued dependency, by placing each repeating group, when transforming to first normal form, in its own frame class. Because of this fourth normal form is hardly ever used because multivalued dependency can be avoided by correctly transforming to First normal form. This is illustrated in our example in section 6.7 at the end of the chapter.

6.4 The Relational Algebra

The relational algebra is a theoretical way of manipulating a relational database. Because of the similarities between relations and frames, the relational algebra might also be used to manipulate a frame-based knowledge-base. The relational algebra can be especially useful in specifying attached procedures in a frame-based knowledge representation. We propose that relational algebra can be used for frames, notwithstanding the differences between frames and relations and have extended the relational algebra operations as found in [PRA87] as follows:

6.4.1 Select

The select command takes a horizontal subset of a frame, in other words it causes a certain frame instance or group of frame instances to be selected. eg. To find all the information associated with the person Pat the relational algebra statement would be:

```
SELECT FROM person : Name="Pat"
```

The result of this select, can be seen in figure 6.6

6.4.2 Project

The Project command takes a vertical subset of a frame, in other words the project command causes only certain columns to be included in a new frame.

eg. To create a frame containing only the names and date of birth of each person the relational algebra statement would be:

PROJECT Person OVER (Name, Date of birth)
GIVING ANSWER

Person:

Name	Sex	Date_of_birth	Age	Brother	Sister
Abe	Male	45/08/03			
Sandra	Female	45/05/31			
Peter	Male	68/09/16			
Mary	Female	70/08/07			
Ann	Female	73/07/04			
Suzanne	Female	67/04/02			
Pat	Male	68/07/07			
John	Male	75/01/05			
Pete	Male	90/03/12			
Sam	Male	92/03/06			

PROJECT Person OVER (Name, Date_of_birth)

Name	Date_of_birth
Abe	45/08/03
Sandra	45/05/31
Peter	68/09/16
Mary	70/08/07
Ann	73/07/04
Suzanne	67/04/02
Pat	68/07/07
John	75/01/05
Pete	90/03/12
Sam	92/03/06

SELECT person: Name = "Pat"

Name	Sex	Date_of_birth	Age	Brother	Sister
Pat	Male	68/07/07			

Fig. 6.6 An example of the Select and Project operations

6.4.3 Join

The Join command is used to join two relations /frame classes together, based on a common slot. If we join the *Children* and *Person* frames, found in figure 6.6, using the *Child* and *Name* slots, we get the frame in figure 6.7

Join [Person, Children] (Name,Child)

Name	Sex	Date_of_birth	Age	Brother	Sister	Husband	Wife	Child
Peter	Male	68/09/16				Abe	Sandra	Peter
Mary	Female	70/08/07				Abe	Sandra	Mary
Ann	Female	73/07/04				Abe	Sandra	Ann
Pete	Male	90/03/12				Peter	Suzanne	Pete
Sam	Male	92/03/06				Peter	Suzanne	Sam

Fig. 6.7 The result of a Join operation

6.4.4 Union

The union operation appends a group of frame instances to an existing group of frame instance, with the same number and type of slots. The union operation is similar to the union operation defined for sets and just like the union operation for sets removes duplicate entries.

<p>IF-NEEDED-brothers (name_of_person)</p> <p>T_frame = SELECT FROM children: Child=name_of_person <i>{Now T_frame contains only the person this query concerns}</i></p> <p>T_frame=JOIN[T_frame,children](husband;wife, husband;wife)</p> <p>T_frame=PROJECT T_frame OVER (child) <i>{Now T_frame contains the name of the person and all his or her}</i> <i>{brothers and sisters}</i></p> <p>T_frame = SELECT FROM T_frame: child<>name_of_person <i>{Remove the name of this person from T_frame}</i></p> <p>T_frame = JOIN[T_frame,person](child, name)</p> <p>T_frame = SELECT FROM T_frame: sex = 'Male'</p> <p>brothers = PROJECT T_frame OVER (name)</p>

Fig. 6.8 An Attached Procedure for a frame, using Relational Algebra.

We can define attached procedures for our example using these relational algebra operations. In figure 6.8 we implemented the IF-NEEDED-brothers procedure, as found in our example, using relational algebra statements. These examples of relational algebra operations, applied to frames, clearly illustrates that relational algebra operations can be applied to frames, as we have defined them, in the same way they are applied to relations.

6.5 Motivation of normalization

One might at this stage ask, why go to all the trouble of normalizing frames:

It should be obvious, normalizing frames has the following advantages:

- 1) Simplifying updates to the knowledge-base, for example adding a child is simplified to adding a entry to the Person frame and the children frame, instead of having to make changes to all the frames containing brothers and sisters of this child.
- 2) Preventing inconsistent data, in the unnormalized form, it is possible for the same marriage to have different starting dates, this is eliminated in the normalized form.
- 3) In its normalized form it is possible to add a marriage even when they do not have children. Where as in the unnormalized version it was necessary to leave that slot empty.
- 4) In the normalized form, it is possible to delete a person from the knowledge-base, without losing information about a marriage, if that person is an only child.

- 5) Normalization causes related information to be grouped together, it also forces one to take a close look at the relationships between different frames, therefore, the process of normalization can be seen as a valuable tool to discover the semantic structure of the knowledge being represented.

One objection that might be raised against normalization of frames and even against normalization of relations is that, for anybody who is any good at developing knowledge bases (or databases), it is natural to automatically place all frames (or relations) in third normal form. This might be true in a sense, but the following are also true:

- i) Everybody makes mistakes at one time or another therefore it is better to have a method for checking your own representation against a representation known to be free of certain update anomalies (third normal form or higher).
- ii) Especially in knowledge-bases, the information stored in the knowledge-base and the relationships between different objects will be generally more complex than in the example, and the only way of assuring the absence of update anomalies in the knowledge-base will be by carefully examining the semantics of the knowledge it contains, using normalization, rather than one or other ad hoc method.

6.6 Advantages of Seeing Frames as a Superclass of Relations

Seeing frames as a superclass of relations has certain advantages for frame-based knowledge representation:

- i) Normalization with all the advantages mentioned in the previous section.
- ii) The re-use of methods developed for relational databases in frame based systems. For example methods to access data concurrently, methods for ensuring privacy and security of data etc. In other words this relationship between frames and relations, makes it possible to integrate knowledge-bases and databases, making these facilities of database management systems available for use in knowledge-bases.
- iii) The re-use of operations defined for relations (relational algebra), like for instance the JOIN operation for Frames, rather than having to develop similar operations that operates on frames. We saw an example of this in figure 6.7.

In chapter 5, we looked at a number of problems with frames:

6.6.1 Lack of theoretical foundations and formalisms for using frames

Seeing frames as a superclass of relations makes it possible to use the theoretical foundations of relations, in a slightly modified form, for frames. It is now possible to use a wide range of operations, defined for relations on frames.

6.6.2 Ineffective search

We can now use the same techniques used in relational databases to optimize queries on the database, to improve the effectiveness of search through a frame-based knowledge-base.

6.7 A Complete Example of Frame Normalization

6.7.1 The Example:

An expert system that identifies plant diseases, should contain the following frames.

Diseases Frame with slots:

Disease Name	
Plant name	The name of plants affected by this disease
Affected Part	Primary part of plant affected by this disease eg. stem or roots.
Symptoms	The symptoms of this disease
A_or_S	For each symptom, is this ALWAYS a symptom or only SOMETIMES, in other words is it essential for a plant to show this symptom to have this disease.
Nr of Symptoms	How many symptoms is associated with this disease (used determine likelihood of a plant having this disease).
Cure	How can this disease be cured

Associated Procedures:

DISPLAY CURE

Given any of the known diseases, this procedure will display the treatment given in the Cure slot of the specific disease instance.

Goal frame with slots:

Plant name	The name of a sick plant
Affected Part	Which part of the plant is affected by the disease
Likely diseases	Diseases that might cause the observed symptoms
Nr of symptoms	The number of symptoms, known for each disease, that was observed.
Likelihood	Percentage likelihood of each disease being the cause of the illness

Associated Procedures:**START**

This procedure will start a query by getting from the user, the name of the affected plant, and the part of the plant that is affected, filling the **Plant name**, and **Affected part** slots of the goal frame.

It will then fill the **likely diseases** slot of the goal frame with all the diseases found in the Diseases frame, that is known to affect the specific part of the plant.

This procedure will then call another procedure to compute the likelihood of each disease being the cause of the observed symptoms.

COMPUTE LIKELIHOOD

This procedure does the following:

- 1) For each disease identified as a possibility in the goal frame, determine, by querying the user, if the symptoms which is always present in a plant affected by this disease is present in the plant being tested. If the symptoms, known to always be present for a specific disease, is not found in the plant being tested, delete this disease instance from the goal frame.
- 2) For each of the diseases remaining after step 1, determine which of the symptoms associated with the disease is found in the plant being tested, by querying the user.
- 3) During both step 1 and step 2, keep track for each disease, of the number of symptoms observed.
- 4) Compute the likelihood of each disease being the cause of the observed symptoms, by expressing the number of symptoms observed for each disease as a percentage of the number of known symptoms of the disease.

Diseases:

Disease name	Plant name	Affected Part	Symptoms	A_or_S	Nr of Symptoms	Cure
Bacterial Cancer	Apricot tree Cherry tree Plum tree	Bark	Exudes Gum Bark dies Leaves show small brown spots.	Always Always Sometimes	3	Spray with copper oxychloride during leaf-fall.
Borers	Eucalyptus Pine Tree	Bark	General debilitation of bark Ringbarking	Always Sometimes	2	Tree surgery. followed by fertilizer treatment
Shot hole	Apricot tree Almond tree	Leaves	Brown spots with reddish margins on leaves Spots fall out Sunken black holes in fruit	Always Always Always	3	Prune infected twigs; apply copper oxychloride at leaf-fall
Leaf blister sawfly	Eucalyptus	Leaves	Raised brown bisters on leaves Leaves fall from tree prematurely	Always Sometimes	2	Spray with dimethoate
Leaf Curl	Peach tree Almond tree	Leaves	Infested portion of leave are at first pink and thickened Completely infected leaves are pale green to yellow	Always Always	2	Spray with copper oxychloride before early bud swell.
Brown rot	Peach tree Apricot tree Cherry tree	Fruit	Brown spots first occur on blossom Brown area spread on fruit until fruit are rotten.	Always Always	2	Spray with benomyl at bud swell.

Fig. 6.9 Sample contents for Diseases Frame.

6.7.2 Normalization

6.7.2.1 Non-derivable normal form

Looking at the different frame classes of the example, we see the following slots that can be computed:

Diseases Frame:

The *nr of symptoms* slot can be computed by counting the number of symptoms associated with a specific disease. Thus in non-derivable normal form, the Diseases frame should have an IF-NEEDED procedure attached to the *nr of symptoms* slot, that computes a value for this slot only when one is required.

Goal Frame:

The likelihood slot can be computed for each disease. This frame however is already in non-derivable normal form, as this slot is computed by the COMPUTE LIKELIHOOD procedure.

6.7.2.2 First Normal Form

A frame is in first normal form, if all its attached procedures, associated with a specific slot returns single values and it contains no repeating groups.

In the example we find the following repeating groups:

Diseases Frame:

The Plant name slot is a repeating group, as well as the symptoms slot.

Removing these two repeating groups can be done in two ways,

- 1) Simply place each plant name and symptom in its own frame instance, and fill the empty Disease Name slots with the appropriate disease name. Removing the repeating groups in this way however will cause multivalued dependency, making it necessary to normalize up to fourth normal form.
- 2) Split the Diseases frame class into two frame classes, one containing symptoms and the other plant names, avoiding multivalued dependencies and assuring that when these frames are in third normal form, they will already be in fourth normal form.

Removing the repeating groups using method number 2, we get the following frame classes:

Plants (Plant name, Disease Name)
Without any associated procedures

Diseases(Disease Name, Affected Part, Symptoms, A_or_S, Nr of Symptoms, Cure)
With the associated procedures:
IF-NEEDED number of symptoms
DISPLAY CURE

Goal Frame:
The Likely diseases slot is a repeating group.

Diseases:

Disease name	Affected Part	Symptoms	A_or_S	Nr of Symptoms	Cure
Bacterial Cancer	Bark	Exudes Gum	Always		Spray with copper oxychloride during leaf-fall.
Bacterial Cancer	Bark	Bark dies	Always		Spray with copper oxychloride during leaf-fall.
Bacterial Cancer	Bark	Leaves show small brown spots.	Sometimes		Spray with copper oxychloride during leaf-fall.
Borers	Bark	General debilitation of bark	Always		Tree surgery. followed by fertilizer treatement
Borers	Bark	Ringbarking	Sometimes		Tree surgery. followed by fertilizer treatement

Shot hole	Leaves	Brown spots with reddish margins on leaves	Always		Prune infected twigs: apply copper oxychloride at leaf-fall
Shot hole	Leaves	Spots fall out	Always		Prune infected twigs: apply copper oxychloride at leaf-fall
Shot hole	Leaves	Sunken black holes in fruit	Always		Prune infected twigs: apply copper oxychloride at leaf-fall
Leaf blister sawfly	Leaves	Raised brown bisters on leaves	Always		Spray with dimethoate
Leaf blister sawfly	Leaves	Leaves fall from tree prematurely	Sometimes		Spray with dimethoate
Leaf Curl	Leaves	Infested portion of leave are at first pink and thickened	Always		Spray with copper oxychloride before early bud swell.
Leaf Curl	Leaves	Completely infected leaves are pale green to yellow	Always		Spray with copper oxychloride before early bud swell.
Brown rot	Fruit	Brown spots first occur on blossom	Always		Spray with benomyl at bud swell.
Brown rot	Fruit	Brown area spread on fruit until fruit are rotten.	Always		Spray with benomyl at bud swell.

Attached Procedures:

IF-NEEDED number of symptoms
DISPLAY CURE

Plants:

Disease name	Plant name
Bacterial Cancer	Cherry tree
Bacterial Cancer	Plum tree
Bacterial Cancer	Apricot tree
Borers	Eucalyptus
Borers	Pine Tree
Shot hole	Apricot tree
Shot hole	Almond tree
Leaf blister sawfly	Eucalyptus
Leaf Curl	Peach tree
Leaf Curl	Almond tree
Brown rot	Peach tree
Brown rot	Apricot tree
Brown rot	Cherry tree

Fig. 6.10 Frames in First Normal Form**6.7.2.3 Second Normal Form**

A frame is in second normal form (2NF) if it is in first normal form and no nonkey slot, that is not a derivable slot, is dependent on only a portion of the primary key.

For each of the frame classes, we have the following primary keys and functional dependency's:

Plants Frame class:

Primary key: Plant name, Disease

Diseases Frame class:

Primary key: Disease Name, Symptoms

Functional Dependency's:

Disease Name → Cure, Affected Part

Disease Name, Symptoms → A_or_S, Nr of Symptoms

Goal Frame class:

Primary key: Plant Name, Affected Part, Likely diseases

Functional Dependency's:

Plant Name, Affected Part, Likely diseases → Nr of Symptoms, Likelihood.

In second normal form we will have the following frame classes:

Plants (Plant name, Disease name)

Diseases(Disease Name, Affected Part, Cure)

With the associated procedure:

DISPLAY CURE

Symptoms (Disease Name, Symptoms, A_or_S, Nr of Symptoms)

With the associated procedure:

IF-NEEDED number of symptoms

Goal (Plant name, Affected Part, Likely disease, Nr of symptoms, Likelihood)

With the associated procedures:

START

COMPUTE LIKELIHOOD

6.7.2.4 Third Normal Form

A frame is in *third normal form* if it is in second normal form and if the only determinants it contains are candidate keys or the determinants it does contain, only determines derivable slots, by being parameters in attached procedures.

In our example, we find the following determinants that is not part of the primary key:

Goal Frame class:

Nr of Symptoms → Likelihood

The likelihood slot however is a derivable slot, its value being determined by the COMPUTE LIKELIHOOD procedure.

Because of this, our example is already in third normal form.

6.7.2.5 Fourth Normal Form

We have no multivalued dependencies in our example, because of the way in which we have transformed to first normal form. Therefore our example is already in fourth normal form.

6.7.3 The Normalized Example

Figure 6.11 shows the contents of the normalized example:

Diseases:

Disease name	Affected Part	Cure
Bacterial Cancer	Bark	Spray with copper oxychloride during leaf-fall.
Borers	Bark	Tree surgery, followed by fertilizer treatement
Shot hole	Leaves	Prune infected twigs; apply copper oxychloride at leaf-fall
Leaf blister sawfly	Leaves	Spray with dimethoate
Leaf Curl	Leaves	Spray with copper oxychloride before early bud swell.
Brown rot	Fruit	Spray with benomyl at bud swell.

Plants:

Disease name	Plant name
Bacterial Cancer	Cherry tree
Bacterial Cancer	Plum tree
Bacterial Cancer	Apricot tree
Borers	Eucalyptus
Borers	Pine Tree
Shot hole	Apricot tree
Shot hole	Almond tree
Leaf blister sawfly	Eucalyptus
Leaf Curl	Peach tree
Leaf Curl	Almond tree
Brown rot	Peach tree
Brown rot	Apricot tree
Brown rot	Cherry tree

Symptoms:

Disease name	Symptoms	A_or_S	Nr of Symptoms
Bacterial Cancer	Exudes Gum	Always	
Bacterial Cancer	Bark dies	Always	
Bacterial Cancer	Leaves show small brown spots.	Sometimes	
Borers	General debilitation of bark	Always	
Borers	Ringbarking	Sometimes	
Shot hole	Brown spots with reddish margins on leaves	Always	
Shot hole	Spots fall out	Always	
Shot hole	Sunken black holes in fruit	Always	
Leaf blister sawfly	Raised brown bisters on leaves	Always	
Leaf blister sawfly	Leaves fall from tree prematurely	Sometimes	
Leaf Curl	Infested portion of leave are at first pink and thickened	Always	
Leaf Curl	Completely infected leaves are pale green to yellow.	Always	
Brown rot	Brown spots first occur on blossom	Always	
Brown rot	Brown area spread on fruit until fruit are rotten.	Always	

Fig. 6.11 The normalized example

6.8 Conclusion

In this chapter we established frames as a superclass of relations, frames having all the properties of relations plus in addition, having default values specified for slots, as well as attached procedures. This relationship between frames and relations has a number of advantages, the primary advantage being that we were able to redefine normalization for frames.

In the next chapter, we will take a closer look at expert systems, and the role of frames as a representation scheme for expert systems, and in chapter 8 we will take a closer at the similarities between databases and knowledge-bases, as we examine the integration of the two.

Chapter VII : An Overview of Expert Systems

7.1 Introduction

This chapter contains a short overview of expert systems. We examine the structure of rule-based, as well as frame-based expert systems, and the advantages of each. Finally we examine the advantages of combining these representation schemes.

7.2 Expert Systems

An *expert system* is defined as a computer based system, that is software and hardware, that is capable of performing a task, usually performed by a human expert. Expert systems simulate the part of human intelligence that can be formulated as a set of facts and heuristic rules [LUG89]. Expert systems are also known as *knowledge-based systems*. In general, an expert system represents knowledge as small discrete chunks, using logic rules, production rules or frames [SMI86]. The knowledge contained within an expert system is usually confined to a single domain. Therefore its problem-solving capacity is usually also limited to a single field of technical expertise. It should be noted that an expert system is classified as such, because of its functional characteristics, rather than because of its structure or specific design aspects.

According to Bowerman and Glover, there are three different options for building an expert system:

- 1) Use an expert system building shell, with all the capabilities required by the specific project.
- 2) Use an expert system building shell, with most of the capabilities required by the specific project, that also provides a high-level programming interface, that can be used to develop the remaining facilities required.
- 3) Use a programming language to develop the expert system in question from the ground up. This programming language can be anything, from a traditional programming language like Pascal to languages like Prolog and LISP [BOW88].

This means that a software system need not be developed using specific structures, tools, languages or design methodologies to be classified as an expert system.

7.3 Rule based Expert Systems

In this section we examine the structure of a rule based expert system, as well as the advantages and disadvantages of this structure.

A rule based expert system can be partitioned into two parts, one containing the domain specific knowledge and the other containing the domain-independent knowledge.

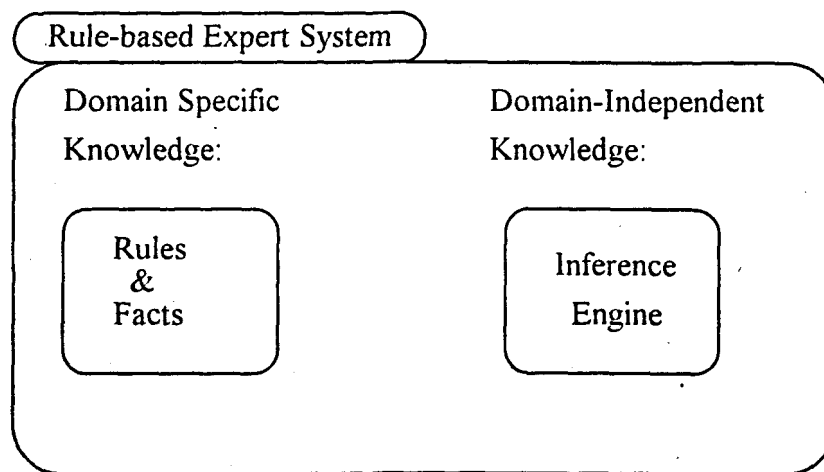


Fig. 7.1 The Primary parts of a rule-based expert system [BOW88].

7.3.1 Domain specific knowledge

The domain specific knowledge, also known as the knowledge-base in a rule-based expert system, consists of rules in the form IF...THEN... or IF....THEN....ELSE... These rules are used by the inference engine, to solve problems in conjunction with some case-specific knowledge or in other words values of variable for the specific problem being solved. [LUG89, BOW88]

7.3.2 Domain-independent knowledge

The domain independent knowledge, also known as the inference engine, has two parts: the rule interpreter and the scheduler. The interpreter evaluates rules in the system to find those whose conditions are satisfied. The scheduler decides the order in which to fire the

rules identified by the scheduler. The inference mechanisms used by the inference engine is either forward-chaining or backward-chaining. Forward chaining infers conclusions from facts in the database, backward chaining on the other hand attempts to find probable causes for conclusions or goals [BOW88]. Two examples of famous rule based expert systems are MYCIN and PROSPECTOR [TEL88].

7.3.3 Advantages of a rule-based expert system

- 1) Making the representation of knowledge more natural: Representing knowledge as "IF..THEN" rules are closer to the way humans describe their knowledge than knowledge imbedded within lower level computer code.
 - 2) The separation of knowledge and control makes it possible for expert system builders to focus on capturing and organizing knowledge, rather than on the details of the control of inference.
 - 3) The separation of knowledge and control makes it possible to change part of the knowledge-base without having to make any changes to the inference engine or the rest of the knowledge-base.
 - 4) Because of the separation of knowledge and control it is possible to use the same inference engine with different sets of knowledge (rules).
 - 5) It is possible to experiment, using a set of rules with different inference engines.
- [LUG89, SMI86].

7.3.4 Disadvantage of separation of knowledge and control

The main disadvantage of the separation of knowledge and control that we identify is the following: Because an inference engine has to be very general in the way it evaluates and fires rules, it is not always possible to take advantage of patterns within a specific set of rules to make the firing of rules more effective. In other words, knowledge representation can be kept simple, using rules, because performance-oriented details are omitted. To make up for these missing details, it is necessary to use powerful, but inefficient control strategies to control search in the rule base [SMI86]. This is an illustration of the general tendency within all problem solving methods: The more general a method is, the weaker it will be for specific cases.

7.4 Frame based Expert Systems

In this section we examine frame based expert systems and see how they compare to rule based expert systems.

Looking at the structure of a frame based expert system (figure 7.2), it seems that it is very similar to a rule based expert system. It can also be divided into a part containing domain specific knowledge and a part containing domain independent knowledge, but as we shall see in the rest of this section, this is about as far as the similarity goes.

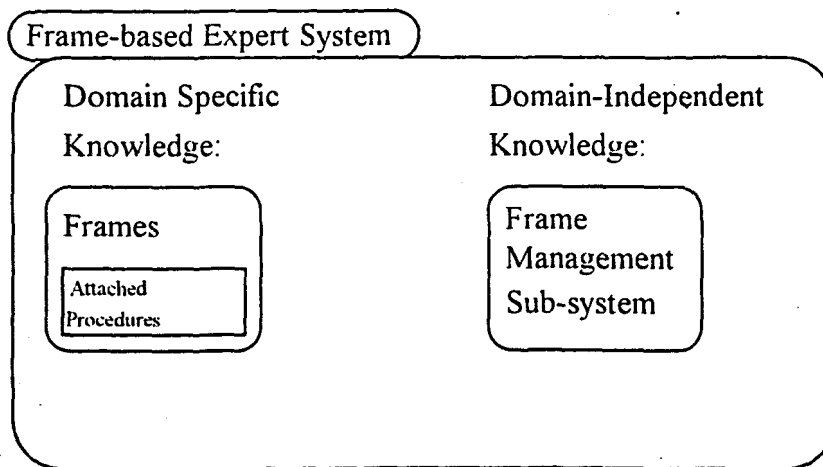


Fig. 7.2 The structure of a frame based expert system

7.4.1 Domain specific knowledge

The domain specific knowledge or knowledge-base in a frame-based expert system consists of a number of frame classes, each with possibly a number of attached procedures and a number of instances. Frames contain facts, rather than rules. The rules in a frame based system for deducting new knowledge is embedded within its procedural attachments[BAR81, BOB77, MIN75]. Because of this difference in its knowledge-base, the inference engine for a frame-based system has a completely different function than that of a rule based system.

7.4.2 Domain-independent knowledge

The inference engine for a frame-based system is not completely in charge of inference as such. Therefore we will rather call it the frame-management sub-system. The frame-management sub-systems' function is to select an initial frame to activate and to transfer control from one frame class to another. Inference is implemented by the procedures

attached to frames. In a typical frame-based expert system, the frame-management subsystem will transfer control to a goal frame, which will attempt to fill all its empty slots, by asking the user for information, deducting information using its own attached procedures, and activating other frames to find information. The only role of the frame-management sub-system being the transfer of control between different frames [MIN75].

7.4.3 Advantages of a frame-based expert system

- 1) Frames are a natural way to represent knowledge because of human's tendency, to group knowledge into classes, making it easy to represent knowledge as frames [MIN75].
- 2) Using Frames makes it possible to create generic systems for, say natural-language understanding, using frames, and to use these in a number of different applications.
- 3) Using Frames makes it possible to control inference much more closely than using rules. This makes it possible to optimize inference whenever definite patterns in knowledge are known.

7.4.4 Disadvantages of frame-based expert systems

Because knowledge and control are not separated in a frame-based expert system, a frame-based expert system has none of the advantages offered by this separation. It is further necessary to specify all inference yourself when developing a frame-based expert system, making it a much more complex task than developing a rule-based expert system.

7.5 Hybrid Expert Systems

Because of the above mentioned disadvantages of a frame based system, expert system development tools that support only frames as a representation structure is rarely found. Most expert system tools that support frames are hybrid systems, using both rules and frames to represent their knowledge. A few examples of such 'hybrid' expert systems are KES, Acquaint, The Intelligence Compiler and Goldworks [TEL88]. In figure 7.3, we can see the general structure of a hybrid system, using both frames and rules.

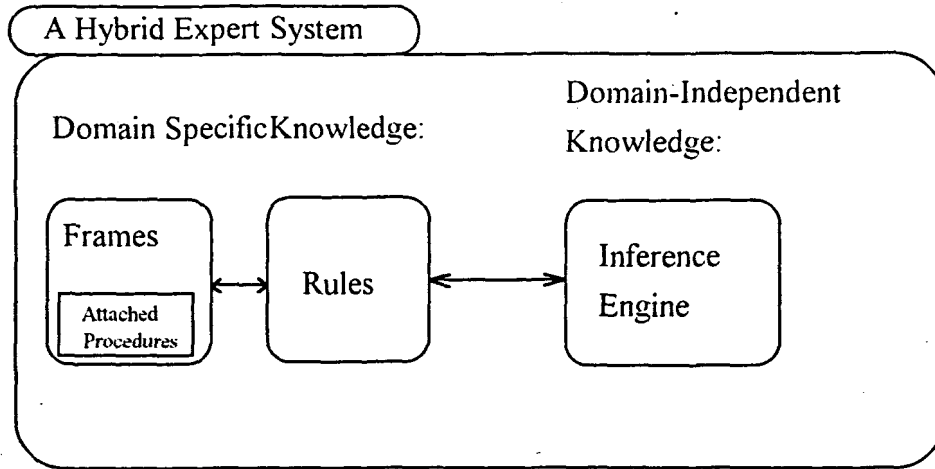


Fig. 7.3 The Structure of a Hybrid expert system.

A hybrid expert system can again be divided into two parts, Domain Specific and Domain-Independent Knowledge.

7.5.1 Domain Specific Knowledge

The domain specific knowledge of a hybrid expert system consists of both frames and rules. The interaction between frames and rules are the following: The rules can access facts stored within individual frames, activating attached procedures when necessary [TEL88]. The reverse is also true, frames in a hybrid system can deduct new knowledge by using their attached procedures or by using the combination of the rules and inference engine.

7.5.2 Domain-Independent Knowledge

The inference engine of a hybrid expert system evaluates rules and selects which rule to fire next. It must also be able to transfer control between different frames. Thus its domain independent knowledge combines the requirements of both a rule-based and a frame-based expert system.

7.5.3 Advantages of hybrid expert systems

Hybrid expert systems provide the best of both worlds: Using a hybrid expert system it is possible to separate knowledge and control, using the combination of rules and inference engine. It is possible to group knowledge into classes, using frames, and to make use of patterns within the knowledge to optimize the inference process by using attached procedures.

7.6 Conclusion

In this chapter we saw how the rule and frame-based representation schemes can complement each other when used in combination, each providing some desirable characteristics while eliminating some of the other's weaknesses.

It should be noted at this stage, that normalization of frames as we defined it in the previous chapter should also be used to normalize the frames used in a hybrid system. Normalization of frames in a hybrid system will eliminate redundancy within frames, prevent update anomalies, and facilitate, making the frame-based part of the system more effective. In a hybrid system, added care should however be taken when normalizing frames, because it will be necessary in some cases to modify the structure of rules that access these frames.

In the next chapter we will examine the integration of expert systems and database management systems, and how the use of database technology might eliminate some of the inherent weaknesses within expert system tools.

Chapter VIII : The Integration of Knowledge-based Systems and Database Management Systems

8.1 Introduction

In this chapter, we examine the integration of knowledge-based systems or expert systems and database management systems. We look at the motivation for this integration and at the different approaches taken to integrate expert systems and database management systems. Finally we examine some examples of attempts to integrate expert systems and knowledge-based systems, found in the literature.

8.2 Motivation

Integrating expert systems and database management systems is an on going area of research. The following are some of the reasons for wanting to integrate expert systems and database management systems:

- In expert systems, the general nature of inference causes a performance bottleneck. In some cases, inference might be replaced by a query evaluation in a Database management system, improving performance [SMI86].
- In most business applications of expert systems, some sharing of information between the expert system and databases might be necessary [SMI86]. Even if the knowledge-base and database remain separate, rules often need to be tied to specific fields in the database [BOW88], e.g., an Expert system identifying market trends, might need to access trade figures for the past year, which might be available in a traditional database system. A Database management system might also have to obtain information from an expert system running in the same environment. To simplify this kind of integration it is desirable to filter these connections through a shared database and to keep the variables passed each way to the absolute minimum [BOW88].
- Database management systems is especially useful for the management of shared data between multiple users. As expert systems develop, the requirement of multiple users will become more of an issue. Integrating database management and expert systems will provide mechanisms for managing multiple users of a single knowledge-base.

- Database management and expert systems is complimentary technology and the current trend is to require aspects of both to develop database and expert system applications.

From the previous discussion, we can see the need for integrating the two different technologies. Doing this however is not as simple as integrating two different software systems. To integrate expert systems and database management systems, it is necessary to accommodate differences in the theoretical foundations of each.

According to Whang and Navathe [WHA92], the motivation for integrating expert systems and database systems, is to provide processing of logic queries (search) with the following characteristics:

- Feasibility and correctness: Each search should be guaranteed to terminate and should produce correct results.
- Coupling efficiency: The database should provide efficient access to data on secondary storage.
- Search efficiency: The search should focus on relevant data only, in other words, the search process should not evaluate irrelevant tuples (tuples not necessary to formulate the results) and it should evaluate each relevant tuple only once.

Through integration of expert systems and database management the above mentioned characteristics should be achievable.

How this integration is done will be examined in the next sections.

8.3 Classification

There is primarily two ways of integrating expert systems with database management systems: The first, called *tight coupling*, is when the definition of a database management system is extended to include things like rule management and inferencing, thus adding the deductive capabilities of an expert system to the capabilities of a traditional database management system. Tight coupling is primarily achieved by extending the definition of the relational structure [UCK91, WHA92]. The second method, called *loose coupling*, is when the expert system and database management system exist as two independent systems and communication between the two occur at the level of a database query language [WHA92].

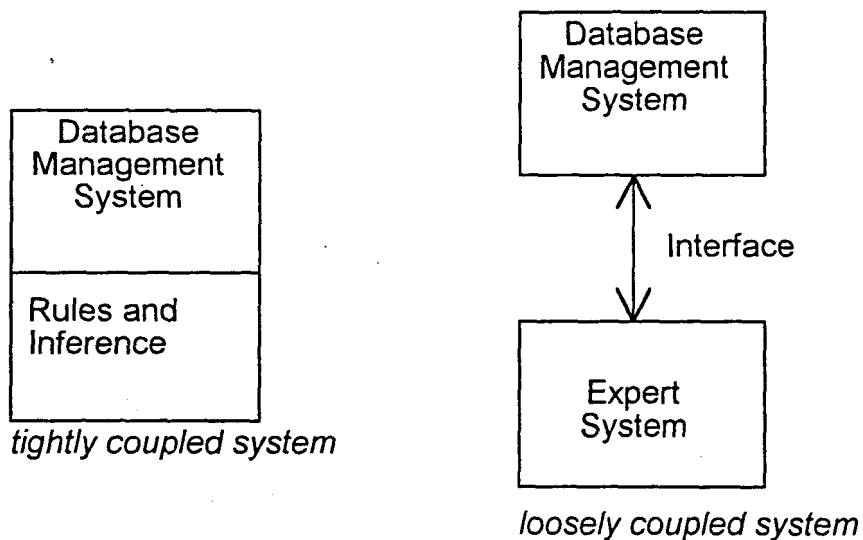


Fig. 8.1 Different methods of integrating expert and database management systems.

Systems integrating expert systems and database management systems can also be classified as homogeneous or heterogeneous systems. In the heterogeneous approach, two distinct systems exist, one for conventional database management and the other for theorem proving, and these two systems are coupled through an interface [UCK91]. In homogeneous systems, data management and theorem proving function are integrated to form a single system [UCK91]. The difference between tight coupled and homogeneous systems is that tight coupling refers to the logical view and homogeneous refers to the physical view of the system. When using tight coupling there still might be two distinct systems, but the users view is that there is only one system, because he uses one representation to formulate queries and deductive axioms [UCK91].

The approaches to integration can also be divided into the following categories:

- Logic based approaches
 - e.g. Prolog interfaces to a Database management system.
- AI frame interfaces to databases
 - e.g. KEE interface to relational databases.
- Object oriented approaches
- Extended relational approaches [BAN89]
 - e.g. The nested relations used by Reiter and Schek [REI89]

The system resulting from integration of expert systems and database management system is in essence an expert system, the only difference it has to conventional expert systems, is that it is much more effective in answering certain kinds of queries.

8.4 Approaches to Integrate Expert Systems and Database Management Systems

Although all attempts at integrating expert and database management systems can be classified using the terms defined in the previous section, there still remains a number of different ways of integrating the two technologies. In this section we will examine some of the attempts found in the literature. We will also take a look at the advantages of the different approaches that are used.

8.4.1 Expert Database Systems

An expert database system (EDS) is defined as "a system for developing applications requiring knowledge-directed processing of shared information." The motivation for expert database systems is the fact that search is the common underlying function of expert systems and database management systems. [SMI86]

8.4.1.1 Architecture of an Expert Database System

An EDS consists of three kinds of components:

- Expert systems, for knowledge directed processing.
- A database management system for managing shared information.
- Specialized processors for handling specialized data formats. [SMI86]

The third component, the ability to handle specialized data, like visual data, is required in a growing number of so called multi-media applications. A general propose EDS should therefore be capable of managing this type of data. The reason for keeping each ES in a separate module is to encapsulate each problem solving activity within a single module. The basic architecture of an EDS is shown in figure 8.2.

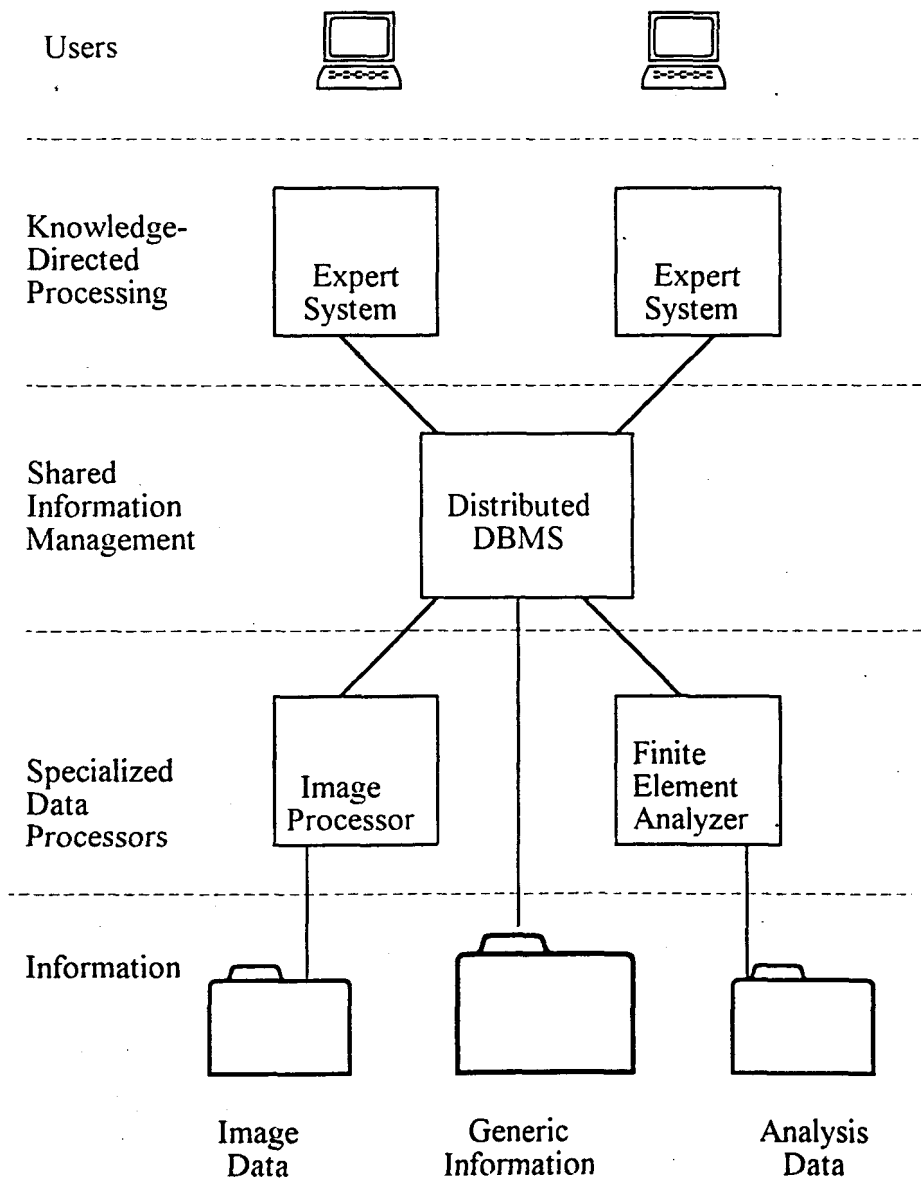


Fig. 8.2 Architecture of an EDS [SMI86]

8.4.1.2 Important issues about EDS

The two main issues are the following:

- How should functionality be distributed across the expert system and the database management system? Should the database functionality be stripped to a bare minimum and all other services be provided by the expert system, or should the stripping be done the other way around?
- How should the system appear to the user? Should it appear as a tightly coupled system, with no visible boundary between database management system and

expert system, or should there be a definite boundary between the two, with different representation and query languages for each?

In the rest of this section, we will focus on the first of these issues. The general answer to the question of distributing functionality is this: The main reason for integrating expert systems and database management systems is to improve performance, a function should therefore be allocated to the system (database management or expert) where it can be executed in the most efficient manner.

8.4.1.3 Inference versus Query Evaluation: A Comparison

- Query evaluation is a much simpler process than deductive inference and in general query evaluation can be expected to be much faster than deductive inference.
- The knowledge representation for deductive inference is more powerful than the knowledge representation for query evaluation.
- Deductive inference is more flexible than query evaluation.
- The flexibility of deductive inferencing is not needed in many applications where some of the knowledge will never change. Integrity constraints (knowledge which never change) in deductive inferencing is not distinguished from other knowledge and therefore the search process can not take advantage of them. This knowledge should rather be built into the search process as integrity constraints, giving improved performance.
- The ability to provide explanations for deductions is not limited to deductive inferencing, as often believed. Query evaluation has a comparable query explanation capability, which involves backtracking the query evaluation tree. [SMI86]

8.4.1.4 Conclusion

Taking into account the previous discussion the following conclusions about the decision for distributing functionality between expert system and database management system are reached:

- An expert system and a database management system containing the equivalent sets of data or knowledge can both support similar query and explanation functions, the only difference being that the database management system can answer queries much more efficiently than the expert system.
- Expert systems have a more flexible way of representing knowledge, but in most cases this added flexibility is unnecessary.

- The search mechanism employed in expert systems is very powerful, but it is also computationally expensive, because it uses the same method of search for all its knowledge. On the other hand, database management systems have a much less powerful, but also much more efficient search mechanism. [SMI86]

The key design criterion for deciding which functions to incorporate in which system is therefore the following: Where a powerful search mechanism is really needed the inference mechanisms of the expert system should be used and in all other cases the database management system should be used for search, to achieve maximum efficiency.

8.4.2 The Extended Disjunctive Normal Form Approach

The Extended Disjunctive Normal Form (EDNF) approach is suggested by Whang and Navathe [WHA92] and is used to achieve performance in a loosely coupled integrated expert, database management system. The EDNF is an extension of the disjunctive normal form of relational algebra so as to include recursion. The EDNF is used to construct optimized queries for loosely coupled systems and is therefore a mechanism to optimize the interface (see figure 8.1) between the expert system and database management system in a loosely coupled system.

8.4.2.1 Advantages of the EDNF Approach

- This approach is used for loosely coupled systems, it is therefore possible to use existing database management systems without any modification.
- It improves performance by optimizing queries directed to the database management system.
- One of the main drawbacks of relational algebra is that it does not provide any recursive functions. This approach extends the relational algebra to provide for recursive logic queries. [WHA92]

8.5 Using the Relational Model for Knowledge-bases

Much of the work that can be categorized as attempts to integrate knowledge-bases and databases management systems, do not use expert systems at all but are attempts to use the relational model, in a particularly creative manner or to modify it slightly, to add expert system capability to it. In other words, much of the work on integrating expert systems and database management systems are at the level of extending the models used by

database management systems (primarily the relational model) to include logic-based capabilities. In this and in the following sections, we will look at this kind of attempts at integration. It is possible to use the relational model as it is as a basis for building knowledge-bases. There is however a few limitations to using this model:

- Knowledge-bases need to be able to access generic-level data about objects with the same ease as accessing specific instances of objects. The generic data also need to be easily changed in a knowledge-base. Relational database management systems usually do not provide mechanisms to do this [LOC91].
- Due to normalization usually associated with relations, related information is distributed across several relations, making the representation more complex to understand. [LOC91]
- Relational algebra do not provide for recursion. This is adequate for database queries, but inference in expert systems, usually requires recursion. [BAN89, WHA92, SMI86, DAT90]
- The relational algebra does not provide any list processing facilities. [BAN89]
- Relations requires that attributes be single valued, where most knowledge-bases requires non-atomic valued attributes. [BAN89]
- The Relational model do not provide any methods for attaching procedures to relations. [BAN89, DAT90]
- Relational database management systems usually do not provide any mechanisms to support making deductions (inferencing). [BAN89]

Because of these limitations a number of additions is proposed to the relational model to make it a more viable method for implementing knowledge-bases. In this section we will examine some of the ways in which the relational model is augmented to be used as a basis for knowledge-base management systems.

8.5.1 Knowledge Representation Using Views in Relational Deductive Databases

A deductive database is defined as a homogeneous logic-based database system in which only one programming system is used for both database access operations and inferential functions [UCK91].

8.5.1.1 *Knowledge-bases and Databases*

According to Uckan [UCK91], knowledge is information that can be derived from data stored within a relational database. This knowledge can be expressed using a variety of knowledge representation schemes, including frames and production rules. Rules can be used to derive new information from the database and therefore, the knowledge-base might consist of a number of rules on how data stored in the knowledge-base might be used to derive information.

8.5.1.2 *Rules and Views*

Production rules can be substituted by virtual relations or views in a relational database. Many relational data base management systems support views. This makes it possible to implement production rules using views in a relational system. According to Uckan, views can be used with the same functionality as rule definitions, using languages like SQL. The only weakness being the processing of recursive rules [UCK91].

8.5.2 **Extensions to the Relational Algebra**

As we mentioned earlier, relational algebra does not support any form of recursion or deduction. Banks et al. [BAN89] defines what they call a *Deductive Algebra* (DEAL) for relations. DEAL is based on SQL (Structured Query Language: an equivalent of relational algebra) and aspects of Prolog. DEAL strengthens the basic operations of SQL to support deductions, functions, molecular objects (as opposed to atomic objects) and virtual attributes. DEAL can also be used for list processing. DEAL is thus an extension of the relational algebra, that makes it more feasible to implement a knowledge-base using the relational model.

8.5.3 **Deductive object bases**

A deductive database extends the relational framework, using mathematical logic. In a deductive database, it is possible to represent facts in a database in terms of normal relations, deductive rules and integrity constraints [SHE90]. In an object-oriented database, the user can define high level objects and their associated set of customized operations. A deductive object base is a combination of the previous two and can be implemented as a system consisting of a rule base, a relational database and a set of primitive operations that can be directly evaluated [SHE90]. A deductive object base integrates database management systems, expert systems, and object-oriented programming, to provide a more effective model for developing expert systems.

8.5.4 A Mapping from Frame-based knowledge to Nested Relations

Reimer and Schek [REI89] proposes a mapping from Frames to Nested Relations. They introduce the frame representation model with a strong type system. They allow for polymorphic type hierarchies and recursive type definition. They map this to a one NF² relation per frame prototype and one meta NF² relation where each tuple describes the schema of a prototype-frame relation. Inheritance is recorded within the meta-relation by explicitly naming sub- and superframes. Nesting is used only to capture the part-of relationship. This mapping makes it possible to implement a Frame-based knowledge-base, using a nested relation model.

8.5.5 Frames as a Superclass of Relations

Our attempt to establish frames as a superclass of relations can also be seen as an attempt to integrate expert systems and database management systems. We define frames in such a way, that it is possible to see frames as an extension of the relational model. Frames as a superclass of relations, again gives rise to the idea that relations, can be used to form the basis of knowledge-based system, even if they are used in a modified form.

8.5.5.1 An Alternative view of a Frame-based Expert System

Because of our definition of frames as an extension of the relational model, we can look at a frame based-expert system in terms of a relational database system.

In chapter 6 we showed that the following relationship exists between frames and relations: A frame is equivalent to a relation, with certain features in addition to those of a relation namely attached procedures and default values. We illustrate this relationship between frames and relations in figure 8.3.

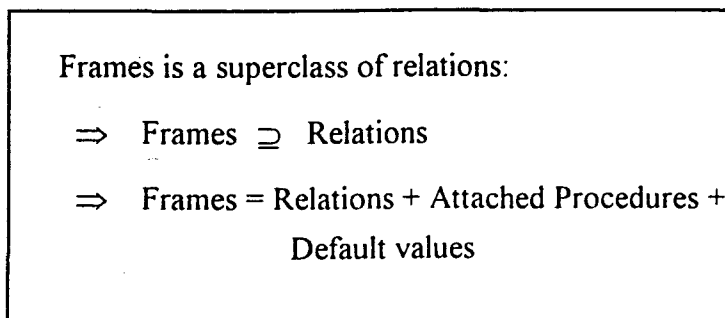


Fig. 8.3 The relationship between Frames and Relations

It is this relationship between frames and relations that prompts us to attempt to define a frame-based expert system in terms of a relational database. Because frames is a superclass of relations, it is clear that a frame-based expert system can not be represented by a relational database on its own.

A data dictionary is defined as a part of a database management system that contains the following information:

- Information about the structure of data.
- Characteristics of the data.
- Usage of the data contained within the database.

In other words, the data dictionary contains meta-data, that can include procedures (usage of data) and default values (characteristics of the data).

A frame-based expert system, using our definition of frames, can be seen as a relational database management system with an integrated data dictionary. The database contains the frame instances, and the data dictionary contains the following: Information about how the database is to be used (attached procedures). Default values for empty slots, and information of the constraints or domain of each slot. We illustrate this alternative view of a frame-based expert system, in terms of a database management system and data dictionary in figure 8.4.

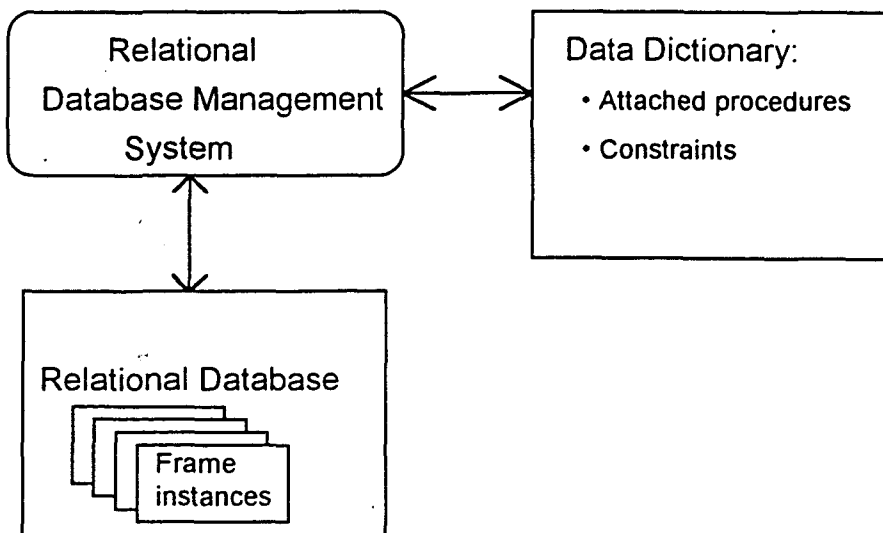


Fig. 8.4 A frame-based expert system in terms of a relational database and a data dictionary.

This alternative view of a frame-based expert system suggest the following:

- It might indeed be possible to develop some expert systems, using a relational database management system that includes an integrated data dictionary.
- Frame-based expert systems might be made more efficient, if they are developed, using relational database management systems, rather than frame-based development tools.

There are however some objections to the use of relational databases for representing knowledge. In the next section, we will look at some of these objections and we will try to put them in the correct perspective.

8.6 Objections Against Using Relational Databases for Representing Knowledge

P.C. Lockemann et al. [LOC91] raises the following objections against using databases as knowledge bases:

- DBMS provide exactly one level of abstraction. Inheritance, on the other hand, requires an unlimited number of levels of abstraction. It is however possible to simulate inheritance on the instance level, but this leads to unwieldy relational schemas.

This is true, but inheritance can be implemented by extending the relational model, an example of such an extension is found in deductive object bases, and object oriented databases.

- When mapping a knowledge-base to a database, it is not clear how to determine the optimal relational structures for representing the knowledge.

We believe that the optimal mapping from knowledge to relations might be achieved by normalization.

- Normalization causes information about a single object to be divided across a number of different relational schemes, making the representation of knowledge, less natural and making expensive join operations necessary to retrieve this information.

It is true that normalized knowledge is more difficult to understand, but normalization is essential to eliminate update anomalies as we have seen in chapter 6. Join operations, however need not be expensive: Join operations can be optimized by maintaining indexes on the fields used for the joins. Inefficient join operations will only be a factor if they are implemented inefficiently.

From these objections it is clear that the use of the relational model for knowledge representation is not without its sacrifices. It is indeed possible to represent any knowledge, using the relational model in a slightly modified form. There are however certain types of knowledge that can be represented much better using expert systems techniques. This seems to point towards using the expert database system approach that represents knowledge using relations whenever possible and uses expert systems techniques in the few cases where they are better suited to representing a specific fact.

It also seems that much of the disadvantages of using relations for representing knowledge, rather than knowledge representation schemes is that they are more difficult to use, but not necessarily more effective. In other words, there seems to be a trade-off between the simplicity of using a specific knowledge representation scheme and how effective it will be. This opinion is confirmed by the current trend to separate the development and delivery systems. A knowledge-based system, developed using an expert system development tool, might be translated to compiled C code [HEL91].

People rarely think of it this way, but the only reason for using complex knowledge representation schemes for designing knowledge-based systems, is to simplify their implementation, but it is even possible to implement an expert system directly in assembler, it will only be much more difficult and take much longer to finish than using higher level techniques.

8.7 Database management systems and Knowledge-based systems as complementary technologies

Database technology is a mature technology [REI89, SMI86]. There are a vast number of commercial database management systems available. These systems are able to meet all the demands in terms of speed, security, management of shared information etc., required by commercial database applications. There is however a class of applications, namely knowledge-based applications, which they cannot easily support. These problems are solved by expert systems development tools. Expert systems development tools, however have their own set of problems, like inefficient search and too slow response times with a large amount of rules, to name a few.

No matter how these two technologies are integrated, they can both benefit from each other: The extensions to database models, made to support knowledge-based applications, makes it possible to develop more complex database systems than before, that combines both traditional database aspects, and knowledge. Expert system technology on the other hand can borrow a great deal from databases. Many of the problems currently experienced with expert systems was also teething problems with database technology and rather than developing new solutions for expert systems, we can learn from the way similar problems were solved for databases, and in many cases we can re-use these solutions with only small changes.

8.8 Conclusion

Although there are still some significant differences between data and knowledge-bases, it seems as if the distinctions between them are becoming increasingly vague. Considerable work has been done to establish common ground between the two technologies and in the long run, both technologies can only benefit from each other. The previous discussion of the different attempts to integrate expert systems and database management systems however highlight a very important problem with this integration: There is a general agreement that this integration can have some advantages, but the manner in which this is to be achieved is still not clear. A great deal of confusion is also starting to develop concerning terminology. Terms like expert system, knowledge-based system, deductive database system, object-oriented database and logic-based systems, are being used in the

same context. This also illustrates the fading of the borders between database and expert systems.

Another problem with the integration of expert systems and database system is that research is done from two directions. From the database community, there is attempts to extend the capabilities of databases, to include functions, traditionally only found in expert systems. And from the expert system community there is an attempt to make expert systems more efficient by using database technology. We believe that meaningful results will be achieved much sooner if there is closer cooperation between these two approaches.

Chapter IX : Normalization and the System Development Life Cycle of Expert Systems

9.1 Introduction

In this chapter, we look at how normalization fits into the development life cycle of an expert system. In the first part of this chapter, we examine the different steps in developing an expert system, and in the second part, we provide a methodology for the use of normalization when developing an expert system, using a frame-based expert system development tool.

9.2 Steps in Developing an Expert System

The development of expert systems is a very wide topic and it is possible to write several books about it. Therefore this section serves only as a summary of the steps in the development life cycle.

9.2.1 Feasibility

Before developing an expert system, it is necessary to determine whether the problem being solved should indeed be solved by the development of an expert system and if indeed this is a problem requiring an expert system solution. It is also necessary to determine whether it will be feasible to develop such an expert system.

A system requiring expert system development generally has some of the following features [LUG89]:

- A requirement for symbolic reasoning.
- A focus on problems that do not respond to algorithmic solutions, but that relies on heuristic search as a problem-solving technique.
- It has to solve problems, using inexact, missing or poorly defined information.
- It has to capture and manipulate qualitative features, rather than numeric data about a problem area.
- It has to deal with semantic meaning, as well as syntactic form.

- It has to give answers that is not necessarily optimal but that is sufficient. This has to do with solving problems in situations where an exact or optimal solution is impossible or too expensive to achieve.
- It has to solve problems using large amounts of domain-specific knowledge.

According to R. Tello [TEL88] the following are a few important rules of thumb that can be used to determine whether the development of a specific expert system will be feasible:

- The problem domain should be specialized and common sense knowledge should not be essential in solving these problems.
- The problem domain should neither be too easy nor too difficult for the current human experts.
- The expert system should include the capability to keep track of the problem space.
- The developer of the expert system must be able to count on a commitment from one or more experts to help in developing the system, for a considerable amount of time.
- The experts should be able to communicate the contents of their expertise, and they should agree on the way in which problems are solved.

9.2.2 Selecting a Development Tool

The selection of an expert system development tool will be influenced, by [TEL88, LUG89, BOW88]:

- The knowledge representation structures provided by the expert system shell.
- The tools provided for developing a user interface.
- The cost of the different expert system shells.
- The integration provided with other systems, i.e., graphics, database and programming interfaces.
- The machine requirements of the development tools.

9.2.3 Knowledge Acquisition

Knowledge acquisition is the process by which knowledge is obtained for transferal to the expert system.

Knowledge acquisition is an iterative process that consists of three phases [BOW88, TEL88]:

- Extracting knowledge from knowledge sources to establish the initial knowledge required by the system.
- The ongoing formulation of knowledge, to add to the knowledge-base.
- Testing and verifying that the knowledge-base is correct and complete.

This knowledge acquisition process is illustrated in figure 9.1.

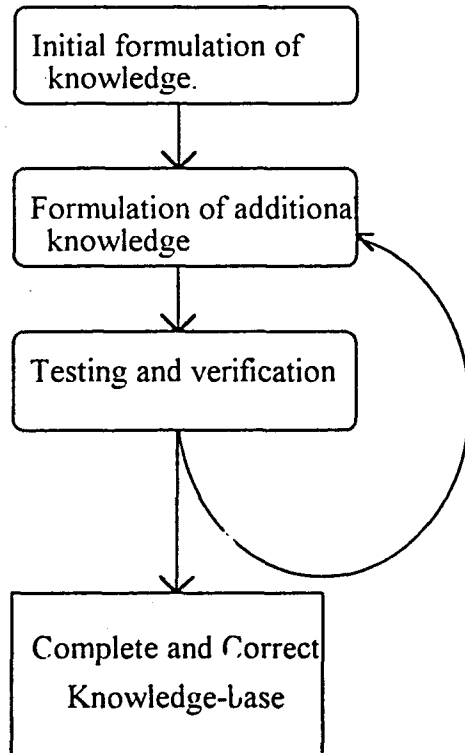


Fig. 9.1 The three phases of knowledge acquisition.

Knowledge can be acquired from the following sources [SEL85]:

- One or more human experts.
- Text books and manuals.
- Examples.

9.2.4 Creating a Representation of the Knowledge

Creating a representation of the knowledge or encoding the knowledge, using the available representation structures, is a central issue to the development of expert systems. In section 9.3 we will examine this encoding of knowledge using frames more closely and provide a methodology for the normalization of frames.

9.2.5 Verification and Evaluation

The verification of an expert system usually involves "hands-on" testing by domain experts and the intended users of the system.

9.2.6 Maintaining and Expanding the System

If an expert system advances to the stage where it is being used then, just like any other piece of software, it will have to be maintained and it might at some stage, be necessary to add a number of extensions to the original system.

9.3 Encoding the Knowledge

Before encoding the knowledge it is necessary to decide which representation scheme will be used. In this section, we examine, how knowledge can be encoded, using frames. We also suggest a methodology for normalizing the frames. The use of frames as a knowledge representation structure is indicated by the following:

- Frames are ideal for systems that require reasoning about stereotyped situations [MIN75, JAC86].
- Frames offer mechanisms for dealing with exceptions and defaults, that can not be easily handled by any other representation scheme [JAC86].
- Frames are recommended when the knowledge structure of the domain is hierarchical and dependent; and the degree of knowledge heuristics is well-defined [MOC92].
- Frames are appropriate for object-oriented systems, in other words frames are appropriate to represent knowledge that focuses on related objects [OLS92].
- Frames can be used for knowledge-based applications that focus on gathering or sorting data-activated procedures. In other words, any change in data will activate procedures. This especially appropriate for monitoring computations [OLS92].
- Frames are useful for representing descriptive information about objects.

To encode the knowledge, gathered in the knowledge acquisition phase, it is necessary to complete the following steps:

- Represent the knowledge as collections of frame classes.
- Identify all the keys.
- Normalize the frames.
- Determining defaults, constraints and domains.

9.3.1 Represent the knowledge as collections of frame classes

Representing the knowledge as collections of frame classes can be achieved in the following manner:

- Try to create a separate frame class for each type of object encountered.
- Determine the primary key for each of these frame classes.
- Determine the properties of each of these frame classes, in other words, determine which slots are associated with each frame class.
- Determine the procedural information to be associated with each frame class, i.e., define all the attached procedures associated with each frame class.

9.3.2 Identify all the keys

Primary keys:

The primary key of each frame class should already be determined in the previous step.

Candidate keys:

It is necessary to identify all the candidate keys so that their uniqueness property might be enforced in subsequent steps, by defining attached procedures that will check this uniqueness property, thus maintaining the integrity of the knowledge-base.

Foreign keys:

Foreign keys, are by far the most important category, as they are used to establish relationships between different frame classes. It is also necessary to add attached procedures to ensure the referential integrity of foreign keys.

9.3.3 Normalize the Frames

To normalize the frames, it is necessary to do the following:

- Identify the derivable slots of each frame class.
- Taking into account these derivable slots, transform the frames to non-derivable normal form.
- Normalize the frames to 1NF.
- Identify the functional dependency between the slots within each frame class.
- Use the functional dependencies, along with the primary key to transform all the frames to 2NF.
- Use this information, along with the keys determined in the previous steps, to normalize the frames, to 3NF, and if necessary, 4NF, using the definitions of normalization as defined in chapter 6.

9.3.4 Determine defaults, domains and constraints

For each slot of each frame class, the following should be determined:

- Determine a default value for each slot. A default value should be a value that is true in general for the slot, otherwise, the default value should be left empty. It is important to remember that a slot that is part of the primary key or of a candidate key, can not have a default value, because of the uniqueness property. Also it is necessary to define attached procedures that prevent slots without default values or IF-NEEDED procedures attached to them from being empty.
- Establish the pool of valid values that may be used to fill each slot. In other words determine the domain of each slot and make sure that the domain restrictions are enforced, if they are not automatically enforced by the frame-management system.
- Determine any other constraints that limit the valid values for each slot, and make sure that they will be enforced.

Determining the defaults, domains and constraints can be done at any stage, we however suggest doing it only after the frames have been normalized, because after normalizing all the functional dependencies and relationships between different frame classes have been determined. If for instance, the default values for all the slots, are determined before normalization, new information about the structure of the knowledge might influence the defaults, invalidating previous assumptions made about their values.

9.4 Conclusion

In this chapter, we provided an overview of the development life cycle of an expert system and we examined how normalization can form part of the knowledge encoding phase. We suggest that normalization should form part of any attempt to map a knowledge-base to frames. This normalization process can also be used when developing an expert system, using a hybrid expert system development tool. In such a case the normalization will only be applied to that part of the knowledge, represented by frames.

CHAPTER X : CONCLUSION

10.1 Introduction

In this chapter we summarize what we have achieved in this dissertation. We also examine possible topics for further research that arises from our study.

10.2 Frames as a Superclass of Relations

The main objective of this dissertation was to establish frames as a superclass of relations. We have seen that frames can indeed be seen as an extension of the relational model, with all the characteristics of the relational model and in addition a few characteristics of its own namely default values and attached procedures. This relationship between frames and relations enabled us to do the following:

- We defined normal forms for frames:
 - non-derivable normal form,
 - first normal form,
 - second normal form,
 - third normal form and
 - fourth normal form (chapter 6).
- We applied relational algebra operations to frames, providing a mechanism by which to manipulate the contents of a frame. We also discovered that it was possible to define the attached procedures of frames, using the relational algebra operations (chapter 6).
- We were able to define an alternative view of a frame-based expert system, in terms of a relational database management system, with an integrated data dictionary (chapter 8).

The normalization of frames, as a superclass of relations, enabled us to transform an initial representation of a collection of knowledge as frames, to a representation, using frames that we know to be free of any update anomalies.

10.3 Topics for Further Research

The following is some topics, arising from this research, that require further looking into:

- It might be interesting, to implement an expert system, using a frame-based expert system development tool, as well as implementing the same expert system using a commercial database management system with an integrated data dictionary and to then compare the systems in terms of the following:
 - Development time.
 - Response time.
 - How easy or difficult it is to change each system to incorporate new knowledge or change existing knowledge.
- The integration of expert systems and database management system also deserves some further attention especially in terms of combining the different approaches of integrating expert systems and database management systems.
- The normalization of frames, if we classify frames as an object-oriented representation raises the question whether normalization might also benefit other object-oriented representations of data.

10.4 Conclusion

We end this dissertation on the same note we started, by reiterating the point, that a great deal can be gained by comparing different technologies for similarities, rather than differences.

We used *similarities* between frames and relations to define normalization for frames. This clearly shows the advantage of examining related technologies for similarities, even though there are also a number of differences between them.

REFERENCES AND BIBLIOGRAPHY

- [BAN89] B. J. Banks, S.M. Deen , L.A. Garcia, S. M. Harding, A. C. Herath. "Design and Implementation of DEAL". *Data and Knowledge Base Integration- Proceedings of the Working Conference held at the University of Keele*. S.M. Deen and G.P. Thomas (ed.), pp. 29-58, 1989.
- [BAR81] A. Barr & E. A. Feigenbaum (ed.). *The Handbook of Artificial Intelligence Volume I*, Pitman, 1981
- [BLA86] W.J. Black. *Intelligent Knowledge Based Systems*, Van Nostrand Reinhold, U.K. 1986.
- [BOB77] D.G. Bobrow, T. Winograd. "GUS, A Frame-Driven Dialog System." *Artificial Intelligence*. Volume 8, pp.155-173. North-Holland Publishing Company, 1977.
- [BOW88] R.G. Bowerman, D. E. Glover. *Putting Expert Systems into Practice*. Von Nostrand Reinhold, 1988.
- [COD90] E. F. Codd, *The relational model for database management : version 2*. Addison-Wesley, 1990.
- [DAT90] C.J. Date. *An Introduction to Database Systems Volume I*, 5th ed. Addison-Wesley, 1990.
- [HAY80] P. J. Hayes. "The Logic of Frames." *Frame Conceptions and Text Understanding*. Metzing D. (ed.). Walter de Gruyter, Berlin, 1980.
- [HEL91] M. Heller. "Once it works, does it still qualify as AI?" *Byte*. January 1991 pp. 267-278.
- [JAC86] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley Wokingham, England, 1986.

[KRE90] W. Kreutzer. *Programming for Artificial Intelligence : Methods, Tools and Applications*. Addison-Wesley Singapore, 1990.

[LOC91] P. Lockemann, H. Nagel & I. Walter. "Databases for knowledge bases: empirical study of a knowledge base management system for a semantic network." *Data & Knowledge Engineering* Vol 7 pp. 115-154. North-Holland Publishing Company, 1991.

[LUG89] G. F. Luger & W.A. Stubblefield. *Artificial Intelligence and the Design of Expert Systems*. The Benjamin/Cummings Publishing Company California, 1989.

[MAI83] D. Maier. *The theory of relational databases*. Pitman Publishing Limited London, 1983.

[MIN91] M. Minasi. "More complex knowledge representation." *AI Expert*, Volume 6 nr. 1 pp. 15-20, January 1991.

[MIN75] M. Minsky. "A framework for representing knowledge." *The Psychology of Computer Vision*. Winston P. (ed.). McGraw-Hill New York, 1975.

[MOC92] R. J. Mockler and D.G. Dologite. *Knowledge-Based Systems: An Introduction to Expert Systems*. Macmillan Publishin Company, New York, 1992.

[NEE91] Neena B. AI tools and Object Oriented Programming Techniques. *New Science Report on Strategic Computing*, Volume 1 nr. 3 , pp. 2(1), 1991.

[OLS92] D. L. Olson, J. F. Courtney, jr. *Decision Support Models and Expert Systems*. Maxwell Macmillan International, 1992.

[PAT91] N. W. Paton and O. Diaz. "Object-oriented databases and frame-based systems: comparison." *Information and Software Technology*, Volume 33 nr. 5, pp. 357-365, 1991.

[PEN89] Juan B. Castellanos Peñuela et al. "An Algebraic Structured Model of Formal Knowledge Representations (Frames and Rules)." *International Journal*

of *Computer Mathematics*, Volume. 2, pp. 1-9. Gordon and Breach, Science Publishers, Inc, 1989.

[PRA87] Phillip J. Pratt & J.J. Adamski. *DATABASE SYSTEM: Management and Design*. Boyd & Fraser Publishing Company Boston, 1987.

[REI89] U. Reimer & H.J. Schek. "A frame-based knowledge representation model and its mapping to nested relations", *Data & Knowledge Engineering*, Volume 4 pp. 321-352. North-Holland Publishing Company, 1989.

[SEL85] P. S. Sell. *Expert Systems- A Practical Introduction*. Camelot Press, Southampton, 1986.

[SMI86] J. M. Smith. "Expert Database Systems: A Database Perspective", *Expert Database Systems Proceedings From the First International Workshop*. pp. 3-15. Kerschberg, L. (ed), Benjamin/ Cummings Publishing Company, Inc, 1986.

[TEL88] E. R. Tello. *Mastering AI Tools and Techniques*. Howard W. Sams & Company, 1988

[THA88] I. A. Thayse. *From standard logic to logic programming*. Anchor Press Britain, 1988.

[UCK91] Y. Uckan. "Knowledge Representation Using Views in Relational Deductive Data Bases." *Journal of Systems Software*, Volume 15, pp. 217-232, Elsevier Science Publishing Co., Inc, 1991.