Intelligent Database Agent

Developed By:

Muhammad Adeel Ather
33-CS/MS/01

Mubasher Feroze
49-CS/MS/01

Supervised By:

Dr. Malik Sikandar Hayat Khiyal

Department of Computer Science
Faculty of Applied Sciences
International Islamic University Islamabad
(2005)

10/07/10

MS
006.3
ATI

1- Agent-based computing.
2- Intelligent agents (computer software)
3- Deductive databases.

Acc. No. (FUIS) T-1244

Department Of Computer Science
International Islamic University Islamabad
Final Approval

August 10, 2005

It is certified that we have read the project report submitted by Mr. Muhammad Adeel Ather (Reg. No: 33-CS/MS/01) and Mr. Mubasher Feroze (Reg. No: 49-CS/MS/01) and it is our judgment that this report is of sufficient standard to warrant to acceptance by International Islamic University Islamabad for MS Degree in Computer Sciences.

Committee

1. External Examiner
   Mr. Shaftab Ahmed
   Senior Principal Engineer (R)
   Engineer Co-ordinator
   Computer Science & Engineering Department
   Bahria University, Islamabad.

   Shaftab Ahmed

2. Internal Examiner
   Dr. Syed Afaq ~~Ahmad~~ Husain
   Head,
   Department of Computer Engineering.
   International Islamic University, Islamabad

3. Supervisor
   Dr. Malik Sikandar Hayat Khiyal
   Head & Associate Professor,
   Department of Computer Science.
   International Islamic University, Islamabad

A dissertation submitted to

Department of Computer Science,

Faculty of Applied Sciences,

International Islamic University, Islamabad

as a partial fulfillment of the requirement

for the award of the degree of the MS in Computer Sciences.

Dedication

To our parents,

Our teachers

and

Our friends

Declaration

We here by solemnly declare that the developed software and this accompanied report neither as a whole nor as a part thereof has been copied from any source .It is further declared that we have developed this software as well as the accompanied report entirely on the basis of our personal efforts made under the kind guidance of our teachers .If any part of this report is proved to be copied out from any source or found to be reproduction of some other, we shall stand by the consequences thereof. We also declare that no part or whole of the work presented in this report has been submitted in support of any degree for other university or institution of learning.

Muhammad Adeel Ather
Reg # 33-CS/MS/01

Mubasher Feroze
Reg # 49-CS/MS/01

Acknowledgement

First and above all, gratitude is due to ALLAH who gives us health, strength and patience to complete this thesis .We are thankful to our teacher who guided us in this project in any way they could, also to our supervisor Dr. Malik Sikandar Hayat Khiyal for providing us help in this conducting the project .

We further wish to express our gratitude to people we met during project who have contributed to this thesis by offering guidance, sharing good advice, and providing tough critique when necessary.

Muhammad Adeel Ather
Reg # 33-CS/MS/01

Mubasher Feroze
Reg # 49-CS/MS/01

Project In Brief

Project Title:	Intelligent Database Agent
Undertaken By:	Muhammad Adeel Ather (Reg. No: 33-CS/MS/01) Mubasher Feroze (Reg. No: 49-CS/MS/01)
Supervised By:	Dr. Malik Sikandar Hayat Khiyal
Date:	8 March 2003
Start Date:	September, 2003
End Date:	March, 2005
Tools:	Visual Basic 6.0, ORACLE
Operating System:	Windows 2000/XP

Abstract

Intelligent Database Agent is a natural language based interface to databases. It is developed to facilitate the naïve user to use different types of databases without having any deep knowledge of database tables, relation between different tables and constraints of the database. With the existing systems, it is difficult for the user to easily understand the architecture and details of the database. This system provides the user with user friendly graphical interface and easy way to configure the different databases with the system.

Preface

This thesis is regarding the project work title "Intelligent Database Agent" submitted as partial fulfillment of requirement for the award of the MS Degree in Computer Sciences from International Islamic University, Islamabad.

Chapter one of the thesis presents an introduction of the project, giving the purpose, need and objectives of the project.

Chapter two of this thesis presents general background of Natural Language Processing techniques that are available and their practical applications in detail. Chapter three is system analysis, which covers requirement and domain analysis with the help of different software models. Chapter four of design analysis describes the design of the "Intelligent Database Agent". Chapter five give the implementation detail of the software along with some sample code. Chapter six defines Testing and Evaluation part showing the system stability.

At the end of Dissertation Appendices, Glossary and References are given. In the appendix – A, a general description of software is given.

Table of Contents

Chapter No.	Contents	Page No.
1. Introduction		2
1.1 Natural Language Processing (NLP)		2
1.2 Applications of NLP		2
1.2.1 Machine Translation		3
1.2.2 Database Access		3
1.2.3 Text Interpretation		3
1.3 Existing Problem		3
1.4 Objective		4
2. Literature Review		6
2.1 Practical Applications		7
2.1.1 Machine Translation		8
2.1.2 Database Access		9
2.1.3 Text Interpretation		10
2.1.3.1 Information Retrieval (IR)		10
2.1.3.2 Text Categorization		11
2.1.3.3 Data Extraction		11
2.2 Advantages of NLIDBs		11
2.2.1 No artificial language		11
2.2.2 Better for some questions		12
2.2.3 Discourse		12
2.3 Disadvantages of NLIDBs		12
2.3.1 Linguistic coverage not obvious		13
2.3.2 Linguistic vs. conceptual failures		13
2.3.3 Users assume intelligence		14
2.3.4 Inappropriate medium		14
2.3.5 Tedious configuration		14
2.4 Background History		15

Chapter No.	Contents	Page No.
3. Basic Concepts		25
3.1 Natural Language		25
3.2 Natural Language Processing		25
3.2.1 Computational Linguistics		26
3.2.2 User interfaces		26
3.2.3 Knowledge-Acquisition		26
3.2.4 Information Retrieval		26
3.2.5 Translation		26
3.3 Levels of Natural Language Processing		27
3.3.1 Phonetic or Phonological Level		27
3.3.2 Morphological Level		27
3.3.3 Syntactic Level		27
3.3.4 Semantic Level		28
3.3.5 Discourse Level		28
3.3.6 Pragmatic Level		29
3.4 Practical Applications of NLP		29
3.4.1 Machine Translation		29
3.4.2 Database Access		30
3.4.3 Information Retrieval (IR)		30
3.4.4 Text Categorization		30
3.4.5 Data Extraction		31
3.5 Software Agent		31
3.5.1 Types of Software Agents		31
3.5.1.1 Autonomous		31
3.5.1.2 Communicative		32
3.5.1.3 Perceptive		32
3.6 Efficient Parsing		32
3.6.1 Tokenization		32
3.6.2 Morphological Analysis		33
3.6.2.1 Inflectional Morphology		33

Chapter No.	Contents	Page No.
	3.6.2.2 Derivational Morphology	33
	3.6.2.3 Compounding	33
	3.6.3 Dictionary Lookup	33
	3.6.4 Scaling Up the Lexicon	34
	3.6.4.1 Morphological Analysis	34
	3.6.4.2 Capitalization	34
	3.6.4.3 Special Format	34
	3.6.4.4 Spelling Correction Routines	34
4.	**System Analysis**	37
	4.1 Analysis	37
	4.1.1 Requirement Analysis	37
	4.1.2 Domain Analysis	37
	4.2 Steps for Object Oriented Analysis	38
	4.3 Use Cases	38
	4.3.1 Use Case Analysis	38
	4.3.2 Actors	39
	4.3.3 Use Cases	39
	4.3.4 Use Case Expanded Format	42
	4.3.4.1 Use Case Login	42
	4.3.4.2 Use Case Client Database Details	43
	4.3.4.3 Use Case Fill Synonym Dictionary	44
	4.3.4.4 Use Case Extract Table Constraints	45
	4.3.4.5 Use Case Manipulate Table Synonyms	46
	4.3.4.6 Use Case Add Table Synonyms	47
	4.3.4.7 Use Case Edit Table Synonyms	48
	4.3.4.8 Use Case Delete Table Synonyms	49
	4.3.4.9 Use Case Manipulate Column Synonyms	50
	4.3.4.10 Use Case Add Column Synonyms	51
	4.3.4.11 Use Case Edit Column Synonyms	52

Chapter No.	Contents	Page No.
	4.3.4.12 Use Case Delete Column Synonyms	53
	4.3.4.13 Use Case Manipulate Keyword Synonyms	54
	4.3.4.14 Use Case Add Keyword Synonyms	55
	4.3.4.15 Use Case Edit Keyword Synonyms	56
	4.3.4.16 Use Case Delete Keyword Synonyms	57
	4.3.4.17 Use Case Analyze Query	58
	4.3.4.18 Use Case Log Off	59
	4.4 Domain Analysis	60
	4.4.1 Conceptual Diagram	60
5. System Design		63
	5.1 Activity Diagrams	63
	5.2 Class	69
	5.3 Class Diagrams	69
	5.4 Attribute	69
	5.5 Relationships	70
	5.6 Sequence Diagram	79
	5.6.1 Enter/Edit Table Synonyms	80
	5.6.2 Enter/Edit Column Synonyms	81
	5.6.3 Enter/Edit Keyword Synonyms	82
	5.6.4 Database Information	83
6. Implementation		85
	6.1 Structured Query Language	85
	6.2 Modules of Intelligent Database Agent	87
	6.2.1 Database Connection with NLP system	87
	6.2.1.1 Universal Data Link	88
	6.2.1.2 Client Database Details	89
	6.2.2 English Language Dictionary Interface	89
	6.2.3 Database Information Extraction	90

Chapter No.	Contents	Page No.
	6.2.3.1 Database Independent Information	90
	6.2.3.2 Database Related Information	90
	6.2.3.3 Database Semantics Information	91
	6.2.3.4 IDA Configuration	92
	6.2.4 IDA Internal Database	93
	6.2.5 Parsing and keywords extraction engine	96
	6.2.5.1 Tokenization	97
	6.2.6 SQL generation Engine	97
	6.2.7 User Interface	98
	6.3 Sample Code	98
7. Testing		106
	7.1 Objective of Testing	106
	7.2 Object Oriented Testing Strategies	106
	7.3 Types of Testing Done	106
	7.4 Evaluation	108
Appendix A: Intelligent Database Agent		111
A. Intelligent Database Agent		111
	A.1 Description	111
	A.2 Menu Items	112
	A.2.1 Configuration	112
	A.2.1.1 Client Database Details	113
	A.2.1.2 Fill synonym Dictionaries	114
	A.2.1.3 Extract Table Constraints	114
	A.2.1.4 Enter/Edit Table Synonyms	115
	A.2.1.5 Enter/Edit Column Synonyms	115
	A.2.1.6 Enter/Edit Keyword Synonyms	116
	A.2.1.7 Log Off	116
	A.2.1.8 Exit	116

Chapter No.	Contents	Page No.
	A.2.2 Query	117
	A.2.2.1 Analyzer	117
	A.2.3 Help	118
	A.2.3.1 About	118
Appendix B: Figures Use in Project Documentation		119
Appendix C: Glossary		122
References and Bibliography		125

Chapter # 1

Introduction

1. Introduction

Natural Language is a language which humans use when communicating with one another. A natural language exists as a result of evolution as opposed to invention. Consider French, English or German; these languages have evolved over centuries to become what they are today. On the other hand, languages such as COBOL, C++ and SQL were created in a relatively short period of time. Natural languages tend to have very large lexicons and highly complex grammars.

1.1 Natural Language Processing (NLP)

Natural Language is a natural mode of communication for human beings but it can not be understood by the computers. Different standard computer languages are used to interact with computers but those are very specific and need to be specialized. Its very difficult for every person to master these languages. Natural Language Processing is the way to convert the Natural Language into the Computer understandable language.

Natural language Processing can be used to perform a variety of useful tasks, ultimately it offers a more natural mode of communication between the system and user. Another application of natural language processing in conjunction with speech synthesis and speech recognition is to allow people with physical handicaps (such as paralysis) to use computers more effectively.

1.2 Applications of NLP

Natural language processing is being applied in more and more fields each day. The main applications of natural language processing are machine translation, database access, and text interpretation. The more successful applications of natural language processing have two common properties. First, they are focused on a particular domain instead of allowing discussion of any topic. Second, they focus on a particular aspect of comprehension instead of trying to understand the language completely.

1.2.1 Machine Translation

Machine translation uses natural language processing techniques to translate from one natural language to another. A typical translation system has a lexicon of 20,000 to 100,000 words and a grammar of 100 to 10,000 rules. It is very difficult to perform 100% accurate machine translation because it requires a detailed understanding of the text and of the situation to which the text refers.

1.2.2 Database Access

Database access via natural language allows the user to interact with the database without having to learn a formal language such as SQL, Access, C/C++, etc. There is a disadvantage to access the database through natural language because it can be confusing and frustrating for the user if their query fails, because it is outside of the system's competence. But these problems can be handled in a better way by developing the appropriate graphical interfaces or providing the guidelines about the limitations of the systems functionality.

1.2.3 Text Interpretation

Natural Language can also be used for the text interpretation. Information retrieval, Text Categorization and Data extraction fall under this category. Information retrieval is a collection of methods which can be used to retrieve documents relevant to a query from a group of documents. Text categorization is the sorting of natural language texts into fixed topic categories. Natural language processing has been successful in this area categorizing over 90% of news stories into their correct categories. Data extraction is the process of extracting useful data from a natural language text (which is often online), and placing this data in a structured database record or template.

1.3 Existing Problem

There are lots of Natural language interfaces to databases. But there are some problems with the existing interfaces. First of all, most of them are specific for specific domains, although they are more accurate but they can only be used for that specific domain.

For example, if a Natural Language Interface is developed for tourism industry then that will only cover the tourism domain. Another problem is that NLIDBs usually require tedious and lengthy configuration phases before they can be used. Also the interfaces given by such systems are not user friendly, Therefore, more and more intricate user interface mechanisms are being developed, from command line interfaces (CLI) to graphical user interfaces (GUI) and, more recently, even 3-dimensional representations. However, even though the on screen representations and ways of interactions become more sophisticated, they are still largely artificial, users have to learn the meaning of the various commands and icons, which parameters they take, and the varied forms how to interact with them. In short, computer systems have not yet been able to satisfactorily communicate with the user in the most human way: natural language.

1.4 Objective

To overcome the problems stated above, our objective is to develop an Intelligent Database Agent based on simple keyword extraction technique with front end based on artificial intelligence and natural language processing. The basic idea behind the creation of the Intelligent Database Agent is to communicate with the database without having in depth knowledge of the database like the database tables, database language and table relations etc. and without the help of specialized database query languages like SQL, instead using simple English language. The database Agent should convert the English query into structured query language understandable by the underlying database by applying knowledge of the database and the English language. Theme behind Intelligent Database Agent is the effective use of natural language based input given by the user through identifying and using the keywords present in the user's entered query. This information in the form of keywords will then be combined with database information and knowledge present in the system to make a language (Structured Query Language in this case) to talk to the underlying database.

Chapter # 2

Literature Review

2. Literature Review

A natural language is a language which humans use when communicating with one another. A natural language exists as a result of evolution as opposed to invention. Consider French, English or German; these languages have evolved over centuries to become what they are today. On the other hand, languages such as COBOL, C++ and SQL were created in a relatively short period of time. Natural languages tend to have very large lexicons and highly complex grammars [3].

Natural language processing can be used to perform a variety of useful tasks, ultimately it offers a more natural mode of communication between the system and user. Another application of natural language processing in conjunction with speech synthesis and speech recognition is to allow people with physical handicaps (such as paralysis) to use computers more effectively [3].

Natural language interfaces to databases are not in common use today for two main reasons: they are difficult to use and they are expensive to build and maintain. The "ease-of-use" problem is solved by wedding a menu-based interaction technique to a traditional semantic grammar-driven natural language system. Using this approach, all user queries are "understood" by the system. The "creation and maintenance problem" is solved by designing a core grammar with parameters supplied by the data dictionary and then automatically generating semantic grammars covering some selected subpart of the user's data. Automatically generated natural language interfaces offer the user an attractive way to group semantically related tables together, to model a user's access rights, and to model a user's view of supported joins paths in a database [1].

Although some of the numerous Natural Language Interfaces to Databases (NLIDBs) developed in the mid-eighties demonstrated impressive characteristics in certain application areas, NLIDBs did not gain the expected rapid and wide commercial acceptance. For example, in 1985 Ovum Ltd. [63] (p.14) was foreseeing that "By 1987 a natural language interface should be a standard option for users of Database Management System (DBMS)

and 'Information Centre' type software, and there will be a reasonable choice of alternatives." Since then, several commercially available NLIDBs have appeared, and some of them are claimed to be commercially successful. However, NLIDBs are still treated as research or exotic systems, rather than a standard option for interfacing to databases, and their use is certainly not wide-spread. The development of successful alternatives to NLIDBS, like graphical and form-based interfaces, and the intrinsic problems of NLIDBs are probably the main reasons for the lack of acceptance of NLIDBs.

In recent years there has been a significant decrease in the number of papers on NLIDBs published per year. Still, NLIDBs continue to evolve, adopting advances in the general natural language processing field (For example, discourse theories), exploring architectures that transform NLIDBs into reasoning agents, and integrating language and graphics to exploit the advantages of both modalities, to name some of the lines of current research. Generic linguistic front-ends have also appeared. These are general-purpose systems that map natural language input to expressions of a logical language (For example, the CLE system [8]). These generic front-ends can be turned into NLIDBs, by attaching additional modules that evaluate the logic expressions against a database

2.1 Practical Applications

Natural language processing is being applied in more and more fields each day. The main applications of natural language processing are machine translation, database access, and text interpretation. The more successful applications of natural language processing have two common properties. First, they are focused on a **particular domain** instead of allowing discussion of any topic. Second, they focus on a **particular aspect of comprehension** instead of trying to understand the language completely [5].

- Successful applications focus on a particular domain and a particular aspect of comprehension.

2.1.1 Machine Translation

Machine translation uses natural language processing techniques to translate from one natural language to another. Taum-Meteo is one successful application of machine translation. It translates weather reports from English to French. This system is successful because the language used in weather reports is quite regular and consistent **(a restricted domain)**. SPANAM is another natural language translation system which translates between Spanish and English. Although SPANAM is **less accurate**, it operates in a broader domain (the text can be on any subject). Spanish-English machine translation (SPANAM) has been operational at the Pan American Health Organization (PAHO) since 1980. As of May 1984, the system's services had been requested by 87 users under 572 job orders, and the project's total output corresponded to 7,040 pages (1.76 million words) that had actually been used in the service of PAHO's activities. The translation program runs on an IBM mainframe computer (4341 DOS/VSE), which is used for many other purposes as well. Texts are submitted and retrieved using the ordinary word-processing workstation (Wang OIS/140) as a remote job-entry terminal. Production is in batch mode only. The input texts come from the regular flow of documentation in the Organization, and there are no restrictions as to field of discourse or type of syntax. A trained full-time post-editor, working at the screen, produces polished output of standard professional quality at a rate between two and three times as fast as traditional translation (4,000-10,000 words a day versus 1,500-3,000 for human translation). The post-edited output is ready for delivery to the user with no further preparation required.

The SPANAM program is written in PL/I. It is executed on the mainframe at speeds as high as 700 words per minute in clock time (172,800 words an hour in CPU time), and it runs with a size parameter of 215 K. Its source and target dictionaries (60,150 and 57,315 entries, respectively, as of May 1984) are on permanently mounted disks and occupy about 9 MB each.

- A typical translation system has a lexicon of 20,000 to 100,000 words and a grammar of 100 to 10,000 rules.

- It is very difficult to perform 100% accurate machine translation because it requires a detailed understanding of the text and of the situation to which the text refers.

2.1.2 Database Access

Database access via natural language allows the user to interact with the database without having to learn a formal language such as SQL, Access, C/C++, etc. There is a disadvantage to this. It can be confusing and frustrating for the user if their query fails, because it is outside of the system's competence. For example a natural language interface for a database may understand "south of the equator", but possibly not "in the southern hemisphere" even though both phrases have the same meaning. [Russell & Norvig, 1995, p. 693]

As a natural language (NL) interface, question answering on relational databases allows users to access information stored in databases by requests in natural language, and generates as output natural language sentences, tables, and graphical representation. The NL interface can be combined with other interfaces to databases: a formal query language interface directly using SQL, a form-based interface with fields to input query patterns, and a graphical interface using a keyboard and a mouse to access tables. The NL interface does not require the learning of formal query languages, and it easily represents negation and quantification, and provides discourse processing. The use of natural language has both advantages and disadvantages. Including general NLP problems such as quantifier scoping, PP attachment, and elliptical questions, current NLIDB has many shortcomings: First, as a frequent complaint, it is difficult for users to understand which kinds of questions are actually allowed or not. Second, the user assumes that the system is intelligent; he or she thinks NLIDB has common sense, and can deduce facts. Finally, users do not know whether a failure is caused by linguistic coverage or by conceptual mismatch. Nevertheless, natural language does not need training in any communication media or predefined access patterns. NLIDB systems, one of the first applications of natural language processing, including "LUNAR" were developed from the 1970s. In the 1980s, research focuses on intermediate representation and portability, and attempts to interface with various systems. CHAT-80

transforms an English query into PROLOG representation, and ASK teaches users new words and concepts. From 1990s, commercial systems based on linguistic theories such as GPSG, HPSG, and PATR-II appear, and some systems attempt to semi-automatically construct domain knowledge. LOQUI , a commercial system, adopts GPSG grammar. Meanwhile, Demers introduces a lexicalist approach for natural language to SQL translation , and as the CoBase project of UCLA, Meng and Chu combine information retrieval and a natural language interface. The major problems of the previous systems are as follows. First, they do not effectively reflect the vocabulary used in the description of database attributes into linguistic processing. Second, they require users to pose natural language queries at one time using a single sentence rather than give the flexibility by dialog-based query processing. The discordance between attribute vocabulary and linguistic processing vocabulary causes the portability problem of domain knowledge from knowledge acquisition bottleneck; the systems need extensive efforts by some experts who are highly experienced in linguistics as well as in the domain and the task.

Androutsopoulos, which are mainly referenced for this section, classifies NLIDB Systems into the following four major categories: pattern matching systems, syntax based systems, semantic grammar systems, and intermediate representation language systems.

2.1.3 Text Interpretation

Text interpretation can be categorized in following three types:

2.1.3.1 Information Retrieval (IR)

Information retrieval is a collection of methods which can be used to retrieve documents relevant to a query from a group of documents. A simple query is a list of words which describes the content of the documents which we are searching for. Every document has some sort of abstract associated with it. It may be the document title, a set of key words or even an n-dimensional vector. N-dimensional vectors are used in more modern information retrieval systems.

2.1.3.2 Text Categorization

Text categorization is the sorting of natural language texts into fixed topic categories. Natural language processing has been successful in this area categorizing over 90% of news stories into their correct categories. Natural language processing systems tend to be faster and more consistent than their human counterparts.

2.1.3.3 Data Extraction

Data extraction is the process of extracting useful data from a natural language text (which is often online), and placing this data in a structured database record or template. The SCISOR system developed by Jacobs and Rau in 1990 is an example of a data extraction system.

2.2 Advantages of NLIDBs

Following are some advantages of Natural Language Interfaces to databases:

2.2.1 No artificial language

One advantage of NLIDBS is supposed to be that the user is not required to learn an artificial communication language. Formal query languages are difficult to learn and master, at least by non-computer-specialists. Graphical interfaces and form-based interfaces are easier to use by occasional users still, invoking forms, linking frames, selecting restrictions from menus, etc. constitute artificial communication languages that have to be learned and mastered by the end-user. In contrast, an ideal NLIDB would allow queries to be formulated in the user's native language. This means that an ideal NLIDB would be more suitable for occasional users, since there would be no need for the user to spend time learning the system's communication language.

In practice, current NLIDBS can only understand limited subsets of natural language. Therefore, some training is still needed to teach the end-user what kinds of questions the NLIDB can or cannot understand. In some cases, it may be more difficult to understand what sort of questions an NLIDB can or cannot understand, than to learn how to use a formal query language, a form-based interface, or a graphical interface (see disadvantages below). One may also argue that a subset of natural language is no longer a natural language.

2.2.2 Better for some questions

It has been argued that there are kinds of questions (For example questions involving negation, or quantification) that can be easily expressed in natural language, but that seem difficult (or at least tedious) to express using graphical or form-based interfaces. For example, "Which department has no programmers?" (negation), or "Which company supplies every department?" (universal quantification), can be easily expressed in natural language, but they would be difficult to express in most graphical or form-based interfaces. Questions like the above can, of course, be expressed in database query languages like SQL, but complex database query language expressions may have to be written.

2.2.3 Discourse

Another advantage of NLIDBS, concerns natural language interfaces that support anaphoric and elliptical expressions. NLIDBS of this kind allow the use of very brief, underspecified questions, where the meaning of each question is complemented by the discourse context. In formal query languages, graphical Interfaces, and form-based interfaces this notion of discourse context is usually not supported.

2.3 Disadvantages of NLIDBs

Following are some disadvantages of Natural Language Interfaces to databases:

2.3.1 Linguistic coverage not obvious

A frequent complaint against NLIDBS is that the system's linguistic capabilities are not obvious to the user. As already mentioned, current NLIDBs can only cope with limited subsets of natural language. Users find it difficult to understand (and remember) what kinds of questions the NLIDB can or cannot cope with. For example, MASQUE [11] [12] is able to understand "What are the capitals of the countries bordering the Baltic and bordering Sweden?", which leads the user to assume that the system can handle all kinds of conjunctions (false positive expectation). However, the question "What are the capitals of the countries bordering the Baltic and Sweden?" cannot be handled. Similarly, a failure to answer a particular query can lead the user to assume that "equally difficult" queries cannot be answered, while in fact they can be answered (false negative expectation).

Formal query languages, form-based interfaces, and graphical interfaces typically do not suffer from these problems. In the case of formal query languages, the syntax of the query language is usually well-documented, and any syntactically correct query is guaranteed to be given an answer. In the case of form-based and graphical interfaces, the user can usually understand what sorts of questions can be input, by browsing the options offered on the screen and any query that can be input is guaranteed to be given an answer.

2.3.2 Linguistic vs. conceptual failures

When the NLIDB cannot understand a question, it is often not clear to the user whether the rejected question is outside the system's linguistic coverage, or whether it is outside the system's conceptual coverage. Thus, users often try to rephrase questions referring to concepts the system does not know (For example rephrasing questions about salaries towards a system that knows nothing about salaries), because they think that the problem is caused by the system's limited linguistic coverage. In other cases, users do not try to rephrase questions the system could conceptually handle, because they do not realize that the particular phrasing of the question is outside the linguistic coverage, and that an alternative phrasing of the same question could be answered. Some NLIDBs attempt to solve

this problem by providing diagnostic messages, showing the reason a question cannot be handled (For example unknown word, syntax too complex, unknown concept, etc.)

2.3.3 Users assume intelligence

NLIDB users are often misled by the system's ability to process natural language, and they assume that the system is intelligent, that it has common sense, or that it can deduce facts, while in fact most NLIDBs have no reasoning abilities. This problem does not arise in formal query languages, form-based interfaces, and graphical interfaces, where the capabilities of the system are more obvious to the user.

2.3.4 Inappropriate medium

It has been argued that natural language is not an appropriate medium for communicating with a computer system. Natural language is claimed to be too verbose or too ambiguous for human-computer interaction. NLIDB users have to type long questions, while in form-based interfaces only fields have to be filled in, and in graphical interfaces most of the work can be done by mouse-clicking. Natural language questions are also often ambiguous, while formal, form-based, or graphical queries never have multiple meanings.

2.3.5 Tedious configuration

NLIDBs usually require tedious and lengthy configuration phases before they can be used. In contrast, most commercial database systems have built-in formal query language interpreters, and the implementation of form-based interfaces is largely automated. Several experiments have been carried out, comparing how users cope with formal query languages, form-based interfaces, graphical interfaces, and NLIDBs.

2.4 Background History

One such experiment, during which fifty five subjects, ranging from computer novices to programmers, were asked to perform database queries using a formal query language (SQL). a graphical interface (SunSimplify), and a NLIDB (DATATALKER [38], later known as NATURAL LANGUAGE [43]). The subjects first received some training on how to use the three interfaces, and were then asked to perform database queries, most of which were similar to the queries the subjects had encountered during the training period. The experiment measured the number of queries the subjects managed to perform successfully, and the average time the subjects used to perform each successful query. None of the three interfaces could be said to be a winner. Each interface was better in some kinds of queries, and in most queries the subjects, performance was roughly the same, whether the subjects were asked to use SQL, the graphical interface, or the NLIDB. Roughly speaking, the NLIDB seemed to be better in queries where data from many tables had to be combined, and in queries that were not similar to the ones the users had encountered during the training period.

Various approaches that have been used in the evaluation of natural language systems, and describes an experiment where first a Wizard of Oz was used to collect sample user questions, and then the sample questions were used as input to an actual NLIDB. In a Wizard of Oz experiment, the user interacts with a person who pretends to be an NLIDB through a computer network. The user is not aware that he/she is not interacting with a real NLIDB.

Information about several experiments on the usability of NLIDBs, and description in detail, one such experiment that assessed the usability of Intellect. The authors conclude that "natural language is an effective method of interaction for casual users with a good knowledge of the database, who perform question-answering tasks, in a restricted domain".

The existence of commercial natural language interfaces (NLI's), such as INTELLECT [34] from Artificial Intelligence Corporation and Q&A [43] from Symantec,

shows that NLI technology provides utility as an interface to computer systems. The success of all NLI technology is predicated upon the availability of substantial knowledge bases containing information about the syntax and semantics of words, phrases, and idioms, as well as knowledge of the domain and of discourse context. A number of systems demonstrate a high degree of transportability, in the sense that software modules do not have to be changed when moving the technology to a new domain area; only the declarative, domain specific knowledge need be changed. However, creating the knowledge bases requires substantial effort, and therefore substantial cost. It is this assessment of the state of the art that causes us to conclude that *knowledge acquisition is one of the most fundamental problems to widespread* applicability of NLI technology.

Edite [53] is a ***multi-lingual*** (Portuguese, French, English, Spanish) natural language front-end for relational databases. It answers written questions about tourism resources by transforming them into SQL queries. The answer depends on the type of question. It can be a nominal list of resources, text, images or graphics. At present, the database contains 53000 tourism resources, arranged on 253 distinct types, corresponding to 209 tables. The main goal of a NLIDB is to provide users with the capability of, in an efficient alternative way, obtaining information stored in the database [5]. The user is not required to learn an artificial communication language being possible to formulate the question in his own native language. The system building up was driven by two main objectives: (a) the exploration of a new technology in an existing context, *i.e.* by exploring how this technology can be used to increase the efficiency of current processes. The technology should also work as a new motive of attraction. We could integrate this into multimedia kiosks, but we can foresee the adequacy to other information supports, like *For example* Internet; (b) the advantages NLIDBs have, compared to others interfaces like formal query languages, form-based interfaces and graphical interfaces: (1) the user is not required to learn an artificial communication language to use the system's potentiality. This does not mean that there is no need for some information (training) about the system's functionality (linguistic coverage, language's domain); (2) there are kinds of questions (*For example* questions involving negation or quantification) that can be easily expressed in natural language, but that seem difficult (or at least tedious) to express using graphical or form-based interfaces. For

example, "*What Lisbon's hotels are rated over 3 stars?*" (numerical quantification), or "*Which are the Algarve's golf courses without a driving range?*" (negation), can be easy expressed in natural language, but they would be difficult to express in most graphical or form-based interfaces; (3) the system will support anaphoric and elliptical expressions. NLIDBs of this kind allow the use of very brief, under-specified questions, where the meaning of each question is complemented by the discourse context. In other interfaces this notion of discourse context is usually not supported.

At the present moment the project is in the final stage of conception and prototyping. In fact the Portuguese prototype is ready to be integrated in the experimental version of the entire system. On the other hand, the dictionaries necessary for the multi language processing are being embedded in the application. Therefore, in the beginning of 1997 the product will be available for testing in the WEB and in the Portuguese Tourism Information Network. In the future, we intend to extend Edite in order to achieve the following goals: Resolution of several language phenomenons such as anaphora and ellipse with the purpose of increasing the communication speed. For example "*Which are the country clubs in Albufeira area? And which are the hotels?*".

- Treatment of negation. *For example "Which campings do not require a camping license?"*
- Handling temporal questions. In this type of questions, the numbers, dates and times have to be carefully treated. *For example "Is the Jeronimos monastery open on Saturdays?" , "In Lisbon, what are the churches open between 7 am. and 10 am.?"*
- Semi-automatization of the fulfillment data. The goal is construct a software tool, Domains Editor, that partially automates the information augmenting process; by information we mean dictionaries, conceptual model and mapping tables. This acquisition component is crucial to the success of a portable system. This tool allows a better management with fewer demands on (i) the knowledge of the system's internal workings; (ii) the intricacies of the grammar; (iii) computational linguistics in general.

- The development of techniques for generating co-operative responses. Nowadays, if the user asks "*What are the hotels in Marinhais?*", the system will return an empty list. In the future, we expect that the system answers with: "*There are no hotels in Marinhais.*"; next it can present information related to other types of lodging. The idea is to progress from the actual answers to co-operative ones.
- Natural language generation.
- Due to the evolution of signal processing technology, namely digital speech recognition, it will be possible to integrate, in the future, a voice analyzer in this system. This will allow the establishment of a direct oral dialogue between the final user and the system.

As mentioned earlier practical natural language processing systems have lexicons with 10,000 to 100,000 root word forms. To create a lexicon of this size is a major investment of time and money. Individual dictionary publishers and natural language processing system developers have not been willing to share with one another. If they collaborated they could produce significantly better dictionaries.

Several questions arise with respect to a menu-based approach to building natural language interfaces. First, can users successfully use an NLMENU interface in which they have only one way to state their query? We have run a series of pilot studies using Tennant's methodology for evaluating natural language understanding systems. All subjects were successfully able to solve all of their problems. Comments from subjects indicated that although the phrasing of a query is at times stilted, subjects were not bothered by this and could find the alternative phrasing without any difficulty.

A second question arises: Since the size of the lexicon determines the number of items that need to be displayed on an NLMENU screen, is menu size a problem? Menus must not become too big or the user will be swamped with choices and will be unable to find the right one. For most of the interfaces we have generated, this has not been a problem, since choices earlier in a sentence tend to restrict later choices to a manageable few. Only for interfaces with a large number of relations (over 10, say) or with relations with a large

number of attributes (over 20, say) do 'recognition problems, start to occur. All our menus are scrollable. Other interaction techniques can be used to put off the problem. But eventually, menu size does limit the sort of interfaces cne can use the NLMENU approach for.

Prototype NLIDBs had already appeared in the late sixties and early seventies. The best known NLIDB of that period is Lunar [51], a natural language interface to a database containing chemical analyses of moon rocks. Lunar and other early natural language interfaces were each built having a particular database in mind, and thus could not be easily modified to be used with different databases. (Although the internal representation methods used in Lunar were argued to facilitate independence between the database and other modules [50], the way that these were used was somewhat specific to that project's needs.)

By the late seventies several more NLIDBs had appeared. Rendezvous [22] engaged the user in dialogues to help him/her formulate his/her queries. LADDER [35] could be used with large databases, and it could be configured to interface to different underlying database management systems (DBMSs). Ladder used semantic grammars.

A technique that interleaves syntactic and semantic processing, although semantic grammars helped to implement systems with impressive characteristics, the resulting systems proved difficult to port to different application domains. Indeed, a different grammar had to be developed whenever LADDER was configured for a new application. As researchers started to focus on portable NLIDBS, semantic grammars were gradually abandoned. Planes [47] and PHILIQA1 [42] were some of the other NLIDBS that appeared in the late seventies.

CHAT-80 [48] is one of the best-known NLIDBS of the early eighties. CHAT-80 was implemented entirely in Prolog. It transformed English questions into Prolog expressions, which were evaluated against the Prolog database. The code of CHAT-80 was circulated widely, and formed the basis of several other experimental NLIDBs (For example MASQUE [11] [12] [10]).

Masque is a powerful natural language query interface for databases, developed at the University of Edinburgh. The system accepts written English questions and transforms them into Prolog queries, that are evaluated against the Prolog database.

Masque is currently implemented in SICStus Prolog. It can be easily configured for a variety of knowledge domains, and in most cases it manages to generated answers to complex written English questions in a couple of seconds. However, until this project was carried out the system could only be used as a front-end to Prolog databases: Masque answers each question by transforming it into a Prolog query which is evaluated against the Prolog database. This method does not allow Masque to be used as a front-end to real-world (usually relational) commercial databases.

Masque is a descendant of CHAT-80, a system created by Warren and Pereira in the early eighties [Warren & Pereira 82]. CHAT-80 was designed as a general and portable natural language interface to an arbitrary database. Although its ability to cope with non-trivial English questions and its efficiency were impressive, it proved to be in many ways idiosyncratic and hard to port between databases and knowledge domains. Since then, several efforts have made at the University of Edinburgh, to redesign the system, so that it is genuinely portable and efficient. Masque is the outcome of these efforts.

In the mid-eighties NLIDBs were a very popular area of research, and numerous prototype systems were being implemented. A large part of the research of that time was devoted to portability issues. For example, TEAM [27] [28] [39] was designed to be easily configurable by database administrators with no knowledge of NLIDBS. Systems that can be adapted to provide access to databases for which they were not specifically hand tailored. It describes an initial version of *a transportable* system, called TEAM (for Transportable English Access Data manager). The hypothesis underlying the research described in this paper is *that* the information required for the *adaptation* can be obtained through an Interactive dialogue with database management personnel who are not familiar with natural language processing techniques.

Information about words, lexical information, includes the syntactic properties of the words that will be used in querying the database and semantic information about the kind of concept to which a particular word refers. TEAM records the lexical information specific to a given domain in a lexicon. Conceptual information includes information about taxonomic relationships, about the kinds of objects that can serve as arguments to a predicate , and about the kinds of properties an object can have. In TEAM, the internal representation of information about the entities in the domain of discourse and the relationships that can hold among them is provided by a conceptual schema. This schema includes a sort hierarchy encoding the taxonomic relationships among objects in the domain, information about constraints on arguments to predicates, and information about relationships among certain types of predicates. A database schema encodes information about how concepts in the conceptual schema map onto the structures of a *particular* database. In particular, it links conceptual-schema representations of entities and relationships in the domain to their realization in a particular database. TEAM currently assumes a relational database with a number of files. (No language processing related problems are entailed in moving TEAM to other database models.) Each file is about some kind of object (For example, employees, students, ships, processor chips); the fields of the file record properties of the object (For example, department, age, length).

JANUS [36] [41] [49] [20] had similar abilities to interface to multiple underlying systems (databases, expert systems, graphics devices, etc). All the underlying systems could participate in the evaluation of a natural language request, without the user ever becoming aware of the heterogeneity of the overall system. JANUS is also one of the few systems to support temporal questions.

Systems that also appeared in the mid-eighties were DATALOG [29] [30], EUFID [44], LDC [14] [15], TQA [24] [25], TELI [13], and many others. The following are some of the commercially available NLIDBs:

- INTELLECT [34] from Trinzic (formed by the merger of AICorp and Aion). This system is based on experience from ROBOT [31] [32] [33].

- BBN's PARLANCE [17] built on experience from the development of the RUS [19] and IRUS [16] systems.
- IBM's LANGUAGEACCESS [40]. This system stopped being commercially available in October 1992.
- Q&A from Symantec.
- NATURAL LANGUAGE from Natural Language Inc. According to [23], this system was previously known as DATATALKER, it is described in [38], and it is derived from the system described in [26].
- LOQUI [18] from BIM.
- ENGLISH WIZARD from Linguistic Technology Corporation. The company was founded by the author of AICorp's original INTELLECT.

Some aspects of the linguistic capabilities of INTELLECT, Q&A, and NATURAL LANGUAGE are reviewed in [43].

It should be noted that when some researchers use the term "database", they often just mean "a lot of data". In this paper, we mean quite a lot more than that. Most importantly, the data is assumed to represent some coherent attempt to model a part of the world. Furthermore, the database is structured according to some model of data, and the NLIDB is designed to work with that data model. Database systems have also evolved a lot during the last decades. The term "database system" now denotes (at least in computer science) much more complex and principled systems than it used to denote in the past. Many of the underlying "database systems" of early NLIDBs would not deserve to be called database systems with today's standards. In the early days of database systems, there was no concept of naive end-users accessing the data directly this was done by an expert programmer writing a special computer program. The reason for this was the 'navigational' nature of the data model used by these early database systems. Not only did the user need to know about the structure of the data in the application. He/she also needed to know many programming tricks to get at the data. The development of the relational model of data in the 1970's (Codd [21]) had a major impact on database systems. In the relational model, the only storage structure is the table, and this was something that even naive users could understand.

Relatively simple declarative query languages, such as SQL, were developed for this class of user.

Currently, there are two major developments in database technology that will have an impact on NLIDBS. The first is the growing importance of object-oriented database systems, and the second is the trend in relational database technology towards more complex storage structures to facilitate advanced data modeling. We note that both of these trends could make it harder to produce an NLIDB. They both reflect a tendency to concentrate on new complex database application areas, such as network management and computer-aided design, where the user is anything but naïve and the immediate access to the database will often be carried out by a layer of application software.

Chapter # 3

Basic Concepts

3. Basic Concepts

In the early days of database systems, there was no concept of naive end-users accessing the data directly. This was done by an expert programmer writing a special computer program. The reason for this was the 'navigational' nature of the data model used by these early database systems. Not only the user need to know about the structure of the data in the application. He/she also needed to know many programming tricks to get at the data. The development of the relational model of data in the 1970's had a major impact on database systems. In the relational model, the only storage structure is the table, and this was something that was even simpler declarative model.

3.1 Natural Language

The term: "natural" languages refer to the languages that people speak, like English, French and Chinese, as opposed to artificial languages like programming languages or logic. As natural languages are the result of evolution as apposed to invention, like computer languages, therefore these languages have a very large lexicons and very vast grammars.

3.2 Natural Language Processing

Natural Language Processing is known as programs that deal with natural language in some way or another. There are different type of programs which are used for the processing of different natural languages depending upon the requirement and the limitations of different languages.

Children learn language by discovering patterns and templates. We learn how to express plural or singular and how to match those forms in verbs and nouns. We learn how to put together a sentence, a question, or a command. Natural Language Processing assumes that if we can define those patterns and describe them to a computer then we can teach a

machine something of how we speak and understand each other. Much of this work is based on research in linguistics and cognitive science.

3.2.1 Computational Linguistics

Computational Linguistics is doing linguistics with computers. Related to NLP, but sometimes explicitly linguistic, for example building models of linguistic theories to test their properties, without any real desire to use them for interfaces or any other application.

3.2.2 User interfaces

User Interfaces are btter than obscure command languages. It would be nice if you could just tell the computer what you want it to do. Of course we are talking about a textual interface – not speech.

3.2.3 Knowledge-Acquisition

Knowledge-Acquisition is program that could read books and manuals or the newspaper. So you do not have to explicitly encode all of the knowledge they need to solve problems or do whatever they do.

3.2.4 Information Retrieval

Information Retrieval finds articles about a given topic. Program has to be able somehow to determine whether the articles match a given query.

3.2.5 Translation

It sure would be nice if machines could automatically translate from one language to another. This was one of the first tasks the researches and computer programmers tried applying computers to but it is not easy to do this.

3.3 Levels of Natural Language Processing

NLP research pursues the elusive question of how we understand the meaning of a sentence or a document. What are the clues we use to understand who did what to whom, or when something happened, or what is fact and what is supposition or prediction? While words--nouns, verbs, adjectives and adverbs--are the building blocks of meaning, it is their relationship to each other within the structure of a sentence, within a document, and within the context of what we already know about the world, that conveys the true meaning of a text. People extract meaning from text or spoken language on at least seven levels. In order to understand Natural Language Processing, it is important to be able to distinguish among these, since not all "NLP" systems use every level.

3.3.1 Phonetic or Phonological Level

Phonetics refers to the way the words are pronounced. This level is not important for written text in information retrieval systems. It is crucial to understanding in spoken language and in voice recognition systems.

3.3.2 Morphological Level

The morpheme is a linguistics term for the smallest piece of a word to carry meaning. Examples are word stems like child (the stem for childlike, childish, children) or prefixes and suffixes like un-, or -ation, or -s. Many new search engines are able to determine word stems on a rudimentary level. This usually means that they can automatically offer both the singular and plural form of a word, which is a nice feature.

3.3.3 Syntactic Level

When we parsed a sentence in grade school, we were identifying the role that each word played in it, and that word's relationships to the other words in the sentence. Position can determine whether a word is the subject or the object of an action.

NLP systems, in their fullest implementation, make elegant use of this kind of structural information. They may store a representation of either of these sentences or they may also store not only the fact that a word is a verb, but the kind of verb that it is. They characterize dozens of different kinds of relationships, such as AGENT, POSSESSOR OF, or IS A.

The structure of a sentence conveys meanings and relationships between words even if we don't know what they mean.

3.3.4 Semantic Level

The semantic level examines words for their dictionary meaning, but also for the meaning they derive from the context of the sentence. Semantics recognizes that most words have more than one meaning but that we can identify the appropriate one by looking at the rest of the sentence.

In English, precise meaning is often carried by noun phrases--two nouns together that mean something quite different from their constituent words. By using both syntactic and semantic levels, NLP can identify automatically phrases such as box office, carbon copy, dress circle, walking stick, blind date, or reference book.

3.3.5 Discourse Level

This level examines the structure of different kinds of text and uses document structure to extract additional meaning. For instance, a newspaper article typically reports the most important facts--who, what, when, where, how--at the beginning, usually in the first paragraph. It makes predictions about the impact of the events towards the end. In contrast, a mystery novel never tells you who did it and how it was done until the end. A technical document starts with an abstract, describing the entire contents of the document in a single paragraph and then enlarging on all these points in the body of the work. NLP uses this

predictable structure "to understand what the specific role of a piece of information is in a document, for example-- is this a conclusion, is this an opinion, is this a prediction or is this a fact ?"

3.3.6 Pragmatic Level

The practical or pragmatic level depends on a body of knowledge about the world that comes from outside the contents of the document. Some information retrieval researchers feel that the only way to add this level of outside knowledge is to gather everything we know about the world and use it as a reference guide or knowledge base for information systems. The problem with building such a vast knowledge base is that it takes too long, and it looks backward to what we knew while it is not very good at adding new information quickly.

3.4 Practical Applications of NLP

Natural Language Processing is being applied in different fields to facilitate the naïve-user. It is not very easy for every person to learn and apply the computer languages just like a professional programmer can do. It has also made it easy to search and retrieve the required documents and information from the bulk of data sources by just giving the input in simple natural language.

3.4.1 Machine Translation

Machine translation uses natural language processing techniques to translate from one natural language to another. A typical translation system has a lexicon of 20,000 to 100,000 words and a grammar of 100 to 10,000 rules. It is very difficult to perform 100% accurate machine translation because it requires a detailed understanding of the text and of the situation to which the text refers.

3.4.2 Database Access

Database access via natural language allows the user to interact with the database without having to learn a formal language such as SQL, Access, C/C++, etc. There is a disadvantage to this. It can be confusing and frustrating for the user if their query fails, because it is outside of the system's competence.

3.4.3 Information Retrieval (IR)

Information retrieval is a collection of methods which can be used to retrieve documents relevant to a query from a group of documents. A simple query is a list of words which describes the content of the documents which we are searching for. Every document has some sort of abstract associated with it. It may be the document title or a set of key words. Early information retrieval systems performed simple comparisons between the user's queries and document abstracts. Modern information retrieval systems translate the user's query into a vector which can then be compared to the vectors of each document in the group of documents we are searching. This allows us to determine how good of a match a given document is.

The natural language processing techniques used in information retrieval are usually quite simple, in that they only examine individual words and not the relationships between them. More advanced information retrieval systems may perform limited syntactic and semantic analysis. Understand the meanings of information retrieval, query and abstract. Natural language processing techniques used in information retrieval are usually quite simple.

3.4.4 Text Categorization

Text categorization is the sorting of natural language texts into fixed topic categories. Natural language processing has been successful in this area categorizing over 90% of news

stories into their correct categories. Natural language processing systems tend to be faster and more consistent than their human counterparts.

3.4.5 Data Extraction

Data extraction is the process of extracting useful data from a natural language text and placing this data in a structured database record or template.

3.5 Software Agent

An agent is simply another kind of software abstraction, an abstraction in the same way that methods, functions, and objects are software abstractions. An object is a high-level abstraction that describes methods and attributes of a software component. An agent, however, is an extremely high-level software abstraction which provides a convenient and powerful way to describe a complex software entity. Rather than being defined in terms of methods and attributes, an agent is defined in terms of its behavior. This is important because programming an agent-based system is primarily a matter of specifying agent behavior instead of identifying classes, methods and attributes. It is much easier and more natural to specify behavior than to write code.

3.5.1 Types of Software Agents

There is a minimum set of common features that typify a software agent.

3.5.1.1 Autonomous

A software agent is called autonomous when the agent is capable of operating as a standalone process and performing actions without user intervention.

3.5.1.2 Communicative

A software agent is communicative when it communicates with the user, other software agents, or other software processes.

3.5.1.3 Perceptive

A software agent is perceptive if it is able to perceive and respond to changes in its environment.

3.6 Efficient Parsing

When a string of natural language text is presented to the agent it parses the string so that it can correctly identify the meaning of different parts of the input. We do not want the time taken by the agent to be prohibitively long so our parsing algorithm must be efficient.

Most natural language processing systems perform four tasks to process the input string. These four tasks are tokenization, morphological analysis, dictionary lookup and error recovery.

3.6.1 Tokenization

Tokenization is the process of breaking the input up into distinct tokens. In languages like English, tokens are simply the words and punctuation which comprise the input string. For some languages such as Japanese or Chinese, tokenization is very difficult because there are no spaces between words (and placing them there would make the input meaningless).

Tokenization routines are optimized for speed rather than accuracy with the idea that if they are consistent in breaking up the input, ambiguity can be dealt with at a later stage of input processing.

3.6.2 Morphological Analysis

Morphological analysis identifies the prefixes, suffixes, and root words that compose a token of the input string. We perform morphological analysis in order to identify words which we cannot immediately find in the dictionary. Although the words themselves may not be in the dictionary, their root words (and prefixes and suffixes) may be. Using these we can understand and identify the words themselves. There are three major types of morphological analysis:

3.6.2.1 Inflectional Morphology

Inflectional morphology refers to the changes required of a word in a particular grammatical context. Consider the suffix "s" which is added to most nouns in order to make them plural.

3.6.2.2 Derivational Morphology

Derivational morphology refers to the changes required to change a word of one category to a similar word of another category. For example, the suffix "ness" is added in the adjective "sick" to form a noun "sickness".

3.6.2.3 Compounding

Compounding takes two words and combines them to form a new word. For example, the word "rainbow" is a compound of the words "rain" and "bow".

3.6.3 Dictionary Lookup

Dictionary lookup is the process of looking up every token (except for punctuation) to find its definition and meaning in context.

3.6.4 Scaling Up the Lexicon

What do we do when an agent does not understand a word? Scaling up the lexicon means to increase the size of the lexicon. In essence that is what we want to do, we want to teach the agent the word which it does not understand, or at least give the agent some way to deal with it.

There are many different methods of error recovery. These methods are the procedures by which the lexicon is "scaled up", they must be applied whenever a word cannot be found in the dictionary. The course text presents four methods of error recovery:

3.6.4.1 Morphological Analysis

Morphological analysis is done by looking up the word's category based on its prefixes, suffixes, and root word forms.

3.6.4.2 Capitalization

Capitalization often indicates a proper noun such as a person's name, a place, a noun or a pronoun.

3.6.4.3 Special Format

Special format clues often indicate dates, times etc. There are many special formats which can be easily recognized. For example, 12/21/1990 is recognized as a date.

3.6.4.4 Spelling Correction Routines

Sometimes the user may make a spelling mistake while entering his/her input (either because they do not know how to spell a word, or they simply mistype it). There are two models of spelling correction closeness and sound-based.

- The closeness model assumes that a small error has been made in entry such as an extra or missing character, transposition of two characters, etc.
- The sound-based model looks up the word in a phonetic dictionary to see if it can find a word which has similar pronunciation.

Chapter # 4

System Analysis

4. System Analysis

Analysis is the foremost part of project development. Most of the time spent in project development is dedicated to analysis. System analysis was done using prototyping and object oriented methodology.

4.1 Analysis

Analysis plays a significant role in making of a software. There are two main parts of the analysis.

- Requirement Analysis
- Domain Analysis

4.1.1 Requirement Analysis

The rationale of analysis is to provide a model of the system's behavior. In conducting the project, Object Oriented approach is adopted. Object oriented analysis is a method of analysis that examines the requirements from the perspective of the classes and objects found in the vocabulary of the problem domain. In requirement analysis we define use cases diagram containing use cases, actors. A first step in analysis is to extract scenarios, or *use cases* that describe the behavior of a system from an external user's perspective.

4.1.2 Domain Analysis

Conceptual domain analysis yields common ground for each specific analysis. Object Oriented (OO) analysis notions lend themselves for capturing generic concepts at multiple levels of granularity. Ensembles, sub ensembles, classes, and generic relationships are all candidates for describing an application domain. A requirements domain analysis may lead to an OO domain engineering effort. This entails the construction of design fragments of the

generic elements identified by a requirements domain analysis. These designs can be implemented and added to a domain-specific code library.

4.2 Steps for Object Oriented Analysis:

The following are the steps for object oriented analysis:

- Obtain ``complete" requirements.
- Describe system-context interaction.
- Delineate subsystems.
- Develop vocabulary by identifying instances with their classes, ensembles, and relationships.
- Elaborate classes and relationships by defining their generic static structure and describing their generic dynamic dimension.
- Construct a model in which the dynamics of objects are wired together.

These steps are connected by transformation -- elaboration relationships. The output of the last step, the model, feeds naturally into the design phase.

4.3 Use Cases

A use case is a specific way of using the system by using some part of the functionality. Each use case constitutes a complete course of events initiated by an actor and it specifies the interaction that takes place between an actor and the system. A use case is thus a special sequence of related transactions performed by an actor and the system in a dialogue. The collected use cases specify all the existing ways of using the system.

4.3.1 Use Case Analysis

Use case analysis is performed to identify portion of system performing specific task. In use case analysis use cases, actors interacting with those use cases, and boundaries are identified. A use case comprises a course of events begun by an actor, and it specifies the

interaction between actor and the system. All the use cases specify existing ways to use the whole system. It is interaction of actors with external or other system with system being designed in order to achieve a goal. Use case describes the functionality of the product to be constructed.

4.3.2 Actors

There is one actor in this use case diagram

- User

4.3.3 Use Cases

There are eighteen use cases in this domain each of which shows its functionality. These are as follows.

- Login
- Client Database Details
- Fill Synonym Dictionary
- Extract Table Constraints
- Manipulate Table Synonyms
- Add Table Synonyms
- Edit Table Synonyms
- Delete Table Synonyms
- Manipulate Column Synonyms
- Add Column Synonyms
- Edit Column Synonyms
- Delete Column Synonyms
- Manipulate Keyword Synonyms
- Add Keyword Synonyms
- Edit Keyword Synonyms

- Delete Keyword Synonyms
- Analyze Query
- Log Off

Use case diagram of IDA is given in Figure 4.1. the detailed information is given in section 4.3.4.

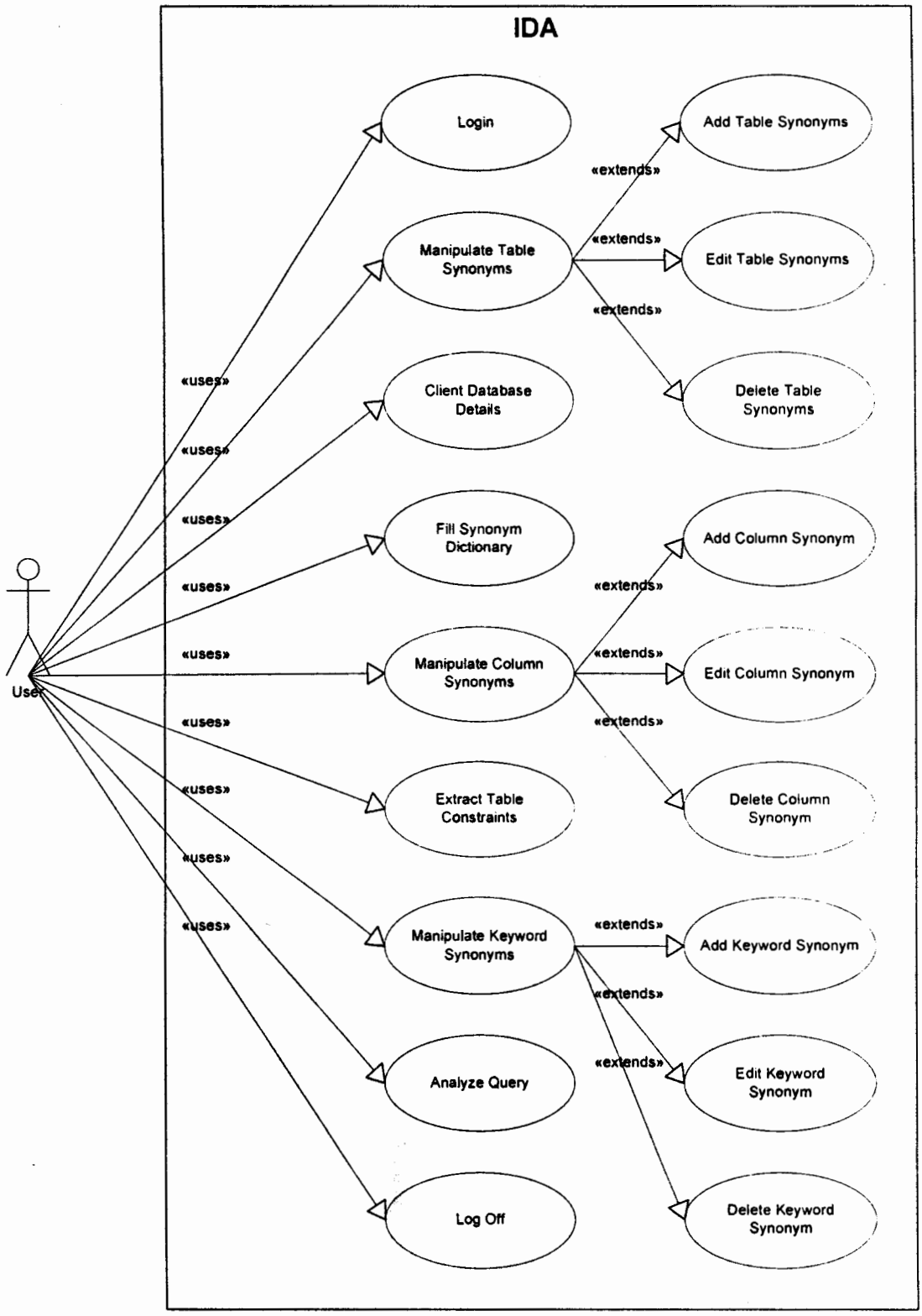

Figure 4.1 Use Case diagram of IDA

4.3.4 Use Case Expanded Format

4.3.4.1 Use Case Login

Actors: User

Purpose: To Login to the system.

Overview: User input the login name and password and then enter to the system.

Type: Real and Primary

Preconditions

Login name and password is entered.

Post Conditions

User is allowed to sign in to the system.

Initiation

This use case is initiated when user double-click the software's executable file.

Navigation

Action	System Response
1. User double clicks the left mouse button on the executable file.	2. Login screen appears to input the Login name and password.
3. User enters the Login name and password. 4. Click on the login button.	5. Validity of Login name and password will be checked.

Alternative Courses

4.a User presses the close button on the login screen and system will be terminated.

4.3.4.2 Use Case Client Database Details

Actors: User

Purpose: To get the details of the Client Database.

Overview: User give the details of the Client database.

Type: Real and Primary

Preconditions

User is signed in to the system.

Post Conditions

Client Database Details are given to the system.

Initiation

This use case is initiated when user click the menu button.

Navigation

Action	System Response
1. User clicks on the menu item "Client Database Details".	2. Details of the connecting database will be shown.

4.3.4.3 Use Case Fill Synonym Dictionary

Actors: User

Purpose: To Fill the Synonyms of the table and column names by consulting the dictionary.

Overview: User Automates the Process of Synonym Filling.

Type: Real and Primary

Preconditions

Database is connected to the system.

Post Conditions

Table and column synonyms are filled in the internal database.

Initiation

This use case is initiated when user presses the generate button.

Navigation

Action	System Response
1. User clicks on the menu item "Fill Synonym Dictionaries".	2. A dialogue appears with generate and close buttons.
3. Click on Generate button.	4. Synonym Dictionaries will be generated.

Alternative Courses

3.a User presses the close button and dialogue will be disappeared.

4.3.4.4 Use Case Extract Table Constraints

Actors: User

Purpose: To Extract all the table constraints of the connected database.

Overview: User Automates the Process of extracting the constraints.

Type: Real and Primary

Preconditions

Database is connected to the system.

Post Conditions

Constraints of the connected database are extracted.

Initiation

This use case is initiated when user presses the extract button.

Navigation

Action	System Response
1. User clicks on the menu item "Extract Table Constraints".	2. A dialogue appears with extract and close buttons.
3. Click on Extract button.	4. Database constraints will be extracted.

4.3.4.5 Use Case Manipulate Table Synonyms

Actors: User

Purpose: Manipulation of table synonyms.

Overview: User manipulates the table synonyms.

Type: Real and Primary

Preconditions

Synonym dictionary is filled with all the possible synonyms.

Post Conditions

Table synonyms are manipulated and stored in the database.

Initiation

This use case is initiated when user presses the Enter/Edit Table Synonyms menu item.

Navigation

Action	System Response
1. User clicks on the menu item "Enter/Edit Table Synonyms".	2. A Form appears with all the available table synonyms.
3. Select the table name. 4. Select the operation to perform.	5. Click the Close button.

4.3.4.6 Use Case Add Table Synonyms

Actors: User

Purpose: Adding the table synonyms.

Overview: User adds the table synonyms.

Type: Real and Secondary

Preconditions

User clicked on the Enter/Edit Table Synonyms.

Post Conditions

Table synonyms are added and stored.

Initiation

This use case is initiated when user presses the Add button on the Enter/Edit Table Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. All the synonyms of the selected table will be displayed.
3. Enter the synonym of the table. 4. Click Add button. 5. Click Save button.	6. Table synonym will be saved.

Alternative courses

2.a Table synonym already exists, click cancel button.

4.3.4.7 Use Case Edit Table Synonyms

Actors: User

Purpose: Edit the existing table synonyms.

Overview: User edits the table synonyms.

Type: Real and Secondary

Preconditions

Synonym is already exist in the systems internal database.

Post Conditions

Table synonyms are edited and saved.

Initiation

This use case is initiated when user presses the edit button on the Enter/Edit Table Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. All the synonyms of the selected table will be displayed.
3. Select the synonym. 4. Edit the synonym of the table. 5. Click Edit button. 6. Click Save button.	7. Table synonym will be saved.

4.3.4.8 Use Case Delete Table Synonyms

Actors: User

Purpose: Delete the existing table synonyms.

Overview: User deletes the table synonyms.

Type: Real and Secondary

Preconditions

Synonym already exists in the systems internal database.

Post Conditions

Table synonyms are deleted.

Initiation

This use case is initiated when user presses the delete button on the Enter/Edit Table Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. All the synonyms of the selected table will be displayed.
3. Select the synonym. 4. Click delete button. 5. Click Save button.	 6. Table synonym will be deleted. 7. Database is updated.

Alternative courses

2.a Synonym does not exist in the database.

4.3.4.9 Use Case Manipulate Column Synonyms

Actors: User

Purpose: Manipulation of column synonyms.

Overview: User manipulates the column synonyms.

Type: Real and Primary

Preconditions

Synonym dictionary is filled with all the possible synonyms.

Post Conditions

Column synonyms are manipulated and stored in the database.

Initiation

This use case is initiated when user presses the Enter/Edit Column Synonyms menu item.

Navigation

Action	System Response
1. User clicks on the menu item "Enter/Edit Column Synonyms".	2. A Form appears with all the available column synonyms.
3. Select the table name. 4. Select the column name. 5. Select the operation to perform.	6. Click the Close button.

4.3.4.10 Use Case Add Column Synonyms

Actors: User

Purpose: Adding the column synonyms.

Overview: User adds the column synonyms.

Type: Real and Secondary

Preconditions

Synonym does not already exist in the systems internal database.

Post Conditions

Column synonyms are added and stored.

Initiation

This use case is initiated when user presses the Add button on the Enter/Edit Column Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. System displays all the columns of the selected table.
3. Select the relevant column name. 4. Enter the synonym of the column. 5. Click Add button. 6. Click Save button.	7. Table synonym will be saved.

Alternative courses

4.a Column synonym already exists, click cancel button.

4.3.4.11 Use Case Edit Column Synonyms

Actors: User

Purpose: Edit the existing column synonyms.

Overview: User edits the column synonyms.

Type: Real and Secondary

Preconditions

Column synonym already exists in the systems internal database.

Post Conditions

Column synonyms are edited and saved.

Initiation

This use case is initiated when user presses the edit button on the Enter/Edit Column Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. All the columns of the selected table will be displayed.
3. Select the column name.	4. All synonyms of the selected column will be displayed.
5. Select the synonym. 6. Edit the synonym of the column. 7. Click Edit button. 8. Click Save button.	9. Synonym will be saved.

4.3.4.12 Use Case Delete Column Synonyms

Actors: User

Purpose: Delete the existing column synonyms.

Overview: User deletes the column synonyms.

Type: Real and Secondary

Preconditions

Synonym already exists in the systems internal database.

Post Conditions

Column synonyms are deleted.

Initiation

This use case is initiated when user presses the delete button on the Enter/Edit Column Synonyms Form.

Navigation

Action	System Response
1. User selects the table name.	2. All the columns of the selected table will be displayed.
3. Select the column name.	4. All synonyms of the selected column will be displayed.
5. Select the synonym. 6. Delete the synonym of the column. 7. Click Delete button. 8. Click Save button.	9. Synonym will be deleted and database is updated.

Alternative courses

2.a Synonym does not exist in the database.

4.3.4.13 Use Case Manipulate Keyword Synonyms

Actors: User

Purpose: Manipulation of Keyword synonyms.

Overview: User manipulates the Keyword synonyms.

Type: Real and Primary

Preconditions

Synonym dictionary is filled with all the possible synonyms.

Post Conditions

Keyword synonyms are manipulated and stored in the database.

Initiation

This use case is initiated when user presses the Enter/Edit Keyword Synonyms menu item.

Navigation

Action	System Response
1. User clicks on the menu item "Enter/Edit Keyword Synonyms".	2. A Form appears with all the available Keyword synonyms.
3. Select the keyword. 4. Select the operation to perform. 5. Click Save button.	6. Keyword manipulated and database is updated.

4.3.4.14 Use Case Add Keyword Synonyms

Actors: User

Purpose: Adding the keyword synonyms.

Overview: User adds the keyword synonyms.

Type: Real and Secondary

Preconditions

Synonym does not already exist in the systems internal database.

Post Conditions

Keyword synonyms are added and stored.

Initiation

This use case is initiated when user presses the Add button on the Enter/Edit Keyword Synonyms Form.

Navigation

Action	System Response
1. User selects the keyword.	2. System displays all the synonyms of the selected keyword.
4. Enter the synonym of the keyword. 5. Click Add button. 6. Click Save button.	 7. Keyword synonym will be saved.

Alternative courses

4.a Keyword synonym already exists, click cancel button.

4.3.4.15 Use Case Edit Keyword Synonyms

Actors: User

Purpose: Edit the existing keyword synonyms.

Overview: User edits the keyword synonyms.

Type: Real and Secondary

Preconditions

Keyword synonym already exists in the systems internal database.

Post Conditions

Keyword synonyms are edited and saved.

Initiation

This use case is initiated when user presses the edit button on the Enter/Edit keyword Synonyms Form.

Navigation

Action	System Response
1. User selects the keyword.	2. All synonyms of the selected keyword will be displayed.
3. Select the synonym. 6. Edit the synonym of the keyword. 7. Click Edit button. 8. Click Save button.	9. Synonym will be saved.

4.3.4.16 Use Case Delete Keyword Synonyms

Actors: User

Purpose: Delete the existing keyword synonyms.

Overview: User deletes the keyword synonyms.

Type: Real and Secondary

Preconditions

Synonym already exists in the systems internal database.

Post Conditions

Column synonyms are deleted.

Initiation

This use case is initiated when user presses the delete button on the Enter/Edit keyword Synonyms Form.

Navigation

Action	System Response
1. User selects the keyword.	2. All synonyms of the selected keyword will be displayed.
3. Select the synonym. 6. Delete the synonym of the keyword. 7. Click Delete button. 8. Click Save button.	9. Synonym will be deleted and database is updated.

Alternative courses

2.a Synonym does not exist in the database.

4.3.4.17 Use Case Analyze Query

Actors: User

Purpose: To analyze the natural language input and to convert it into SQL Query.

Overview: User input in natural language and system analyze it to convert it in SQL.

Type: Real and Primary

Preconditions

User is logged in and input is given in natural language.

Post Conditions

Natural language input is analyzed and converted into SQL query.

Initiation

This use case is initiated when user click on the Analyze button on the Natural Language Analyzer form.

Navigation

Action	System Response
1. User enters the input in natural language. 2. Click on Analyze Button.	3. System checks the input string and break it into tokens to form the SQL query.

Alternative courses

2.a no input is given for the conversion.

4.3.4.18 Use Case Log Off

Actors: User

Purpose: To sign off from the system when working is complete.

Overview: User completes his/her work and then exit from the system.

Type: Real and Primary

Preconditions

User is logged in to the system.

Post Conditions

User is logged off and program is closed.

Initiation

This use case is initiated when user click on the Log off menu item.

Navigation

Action	System Response
1. User click on the "Log Off" menu item.	2. System displays the Log Off confirmation dialogue.
3. User Click on the OK button.	4. System execution is terminated.

Alternative courses

3.a User does not confirm the Log off.

4.4 Domain Analysis

In domain analysis we represent concepts of our project. Discuses the functionality of the project by representing conceptual diagram, which contains main concepts, their relations and their attributes.

4.4.1 Conceptual Diagram

Conceptual Model is a quintessential step in analysis or investigation, it is a decomposition of the problem into individual Objects (called concepts), and things we are aware of. In addition to that creating a Conceptual Model also aids in clarifying the terminology or vocabulary of the domain. A Conceptual Model is a description of things in a real world problem domain, that is, it is not a Model of software design. Conceptual diagram is presented in figure 4.2.

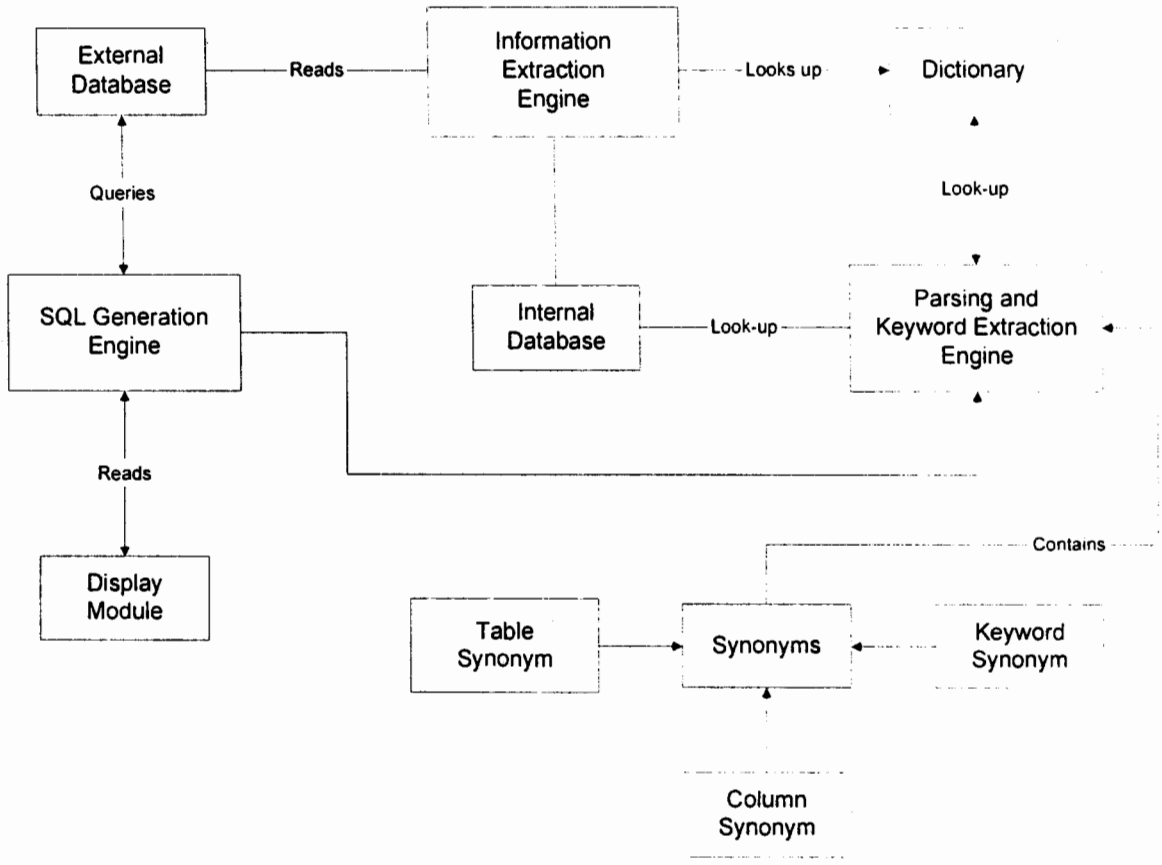

Figure 4.2: Conceptual Diagram of IDA

Chapter # 5

System Design

5. System Design

The purpose of design is to create architecture for the evolving implementations. Object oriented design is a method of design encompassing the process of objects oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of system under design.

The design phase focuses on defining the software to implement the application. The design object is to produce a model of the system, which can be used later to build the system. The design goal is to find the best possible design within the limitations imposed by the requirement and the physical and social environment in which the system will operate .

5.1 Activity Diagrams

It gives the pictorial representation for algorithm for function. Activity diagram is used to represent activities present in use cases. Basic need is that we want to make procedural design in UML. Operations in use cases in sequence are represented in activity diagram. Activity diagram are useful when we want to describe a behavior which is parallel, or when we want to show how behaviors in several use cases interact.

Following processes are shown in the figures:

1. Configuring Database in figure 5.1
2. Adding Synonym in figure 5.2
3. Editing Synonym in figure 5.3
4. Deleting Synonym in figure 5.4
5. Generating Query in figure 5.5

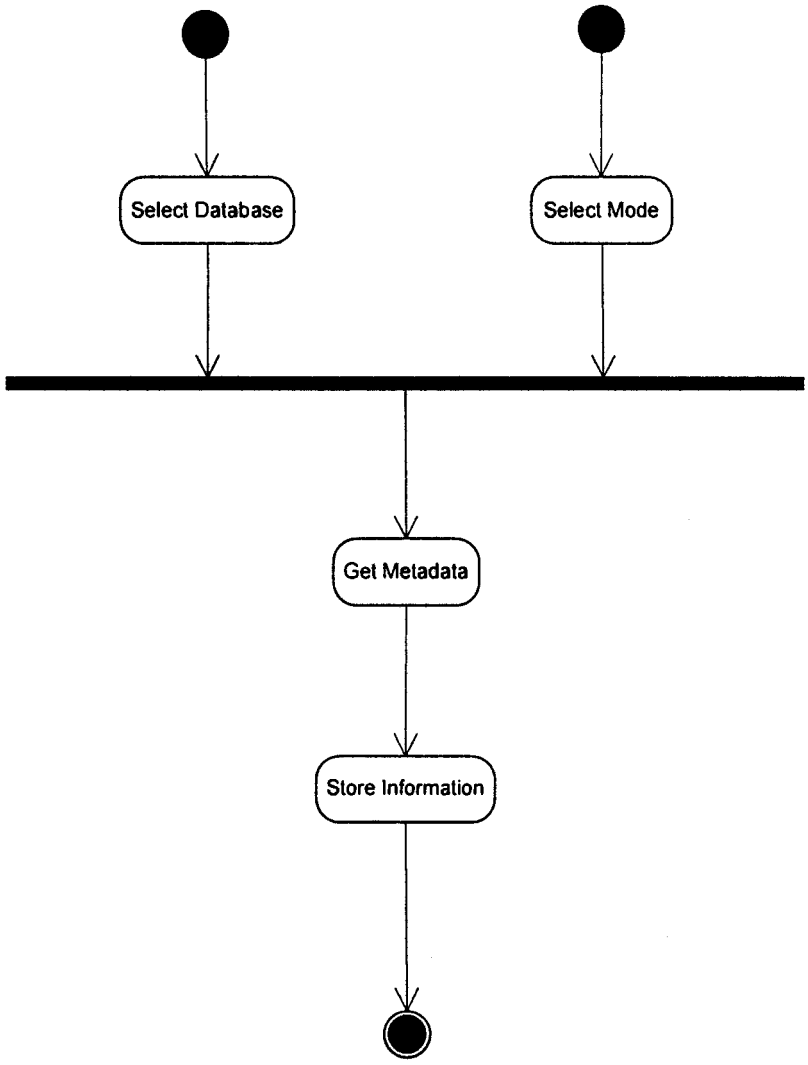

Figure 5.1: Activity Diagram of Configuring Database

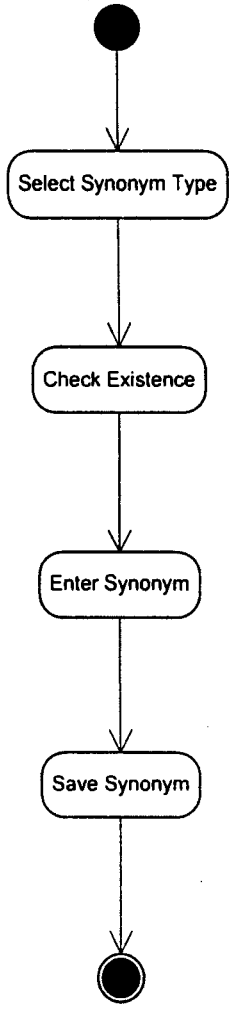

Figure 5.2: Activity Diagram of Adding a Synonym

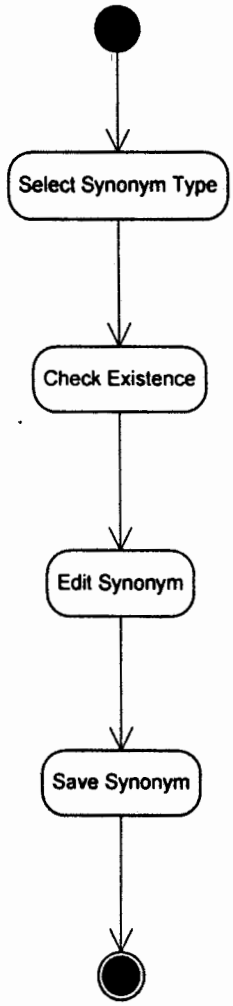

Figure 5.3: Activity Diagram of Editing a Synonym

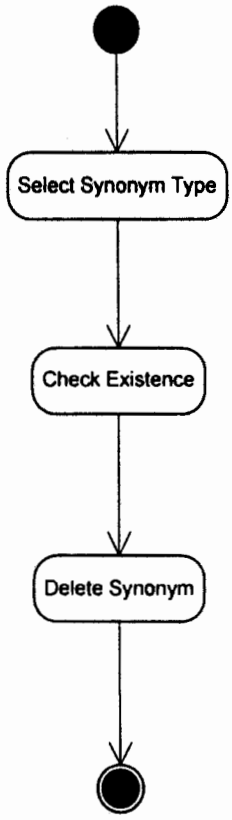

Figure 5.4: Activity Diagram of Deleting Synonyms

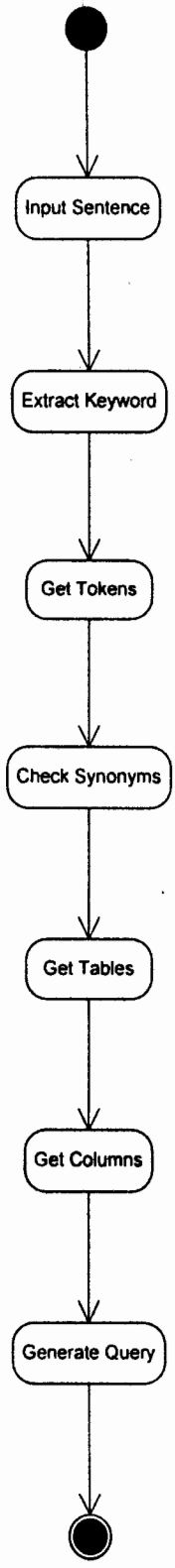

Figure 5.5: Activity Diagram of Generating Query

5.2 Class

A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations. class stands for a family of objects that have something in common. A class is not to be equated with a set of objects, although at any moment we can consider the set of instances that belong to the class. A class may be seen as what all these sets have in common. In technical terminology, a class stands for the intension of a particular characterization of entities, while the set of objects that conform to such a characterization in a certain period is known as the extension.

5.3 Class Diagrams

The development phase produces candidate classes and relationships. After selecting concise and evocative names we must describe each class with attributes. Although each class must have a unique name, classes should be distinguishable on the basis of their attribute characterizations. A rule of thumb is if two classes have identical attributes, then they are most likely the same. Class diagram of Intelligent Database Agent is shown in figure 5.6.

5.4 Attribute

An attribute expresses an essential definitional feature that is shared by all instances of a class. A minimal characterization of an attribute consists of the value domain of the attribute and a name that explains the role or relationship that an attribute value has with respect to the instance to which it belongs. Multi valued attributes may be annotated with multiplicity characterizations. Defaults for an attribute value and/or multiplicity description can be formulated in this phase as well. Constraints can restrict attribute value combinations and/or refer to multiplicity descriptions.

Real-life entities are often described with words that indicate stable features. Most physical objects have features such as shape, weight, color, and type of material. Sometimes it is useful to indicate a default initial value for an attribute.

5.5 Relationships

Relationships help capture target system-specific knowledge by describing connections among different objects. Relationships may also be used to modify descriptions in the previous step. For example, when an attribute has a multiplicity range that includes zero, one may eliminate the attribute and represent this information as a relationship instead.

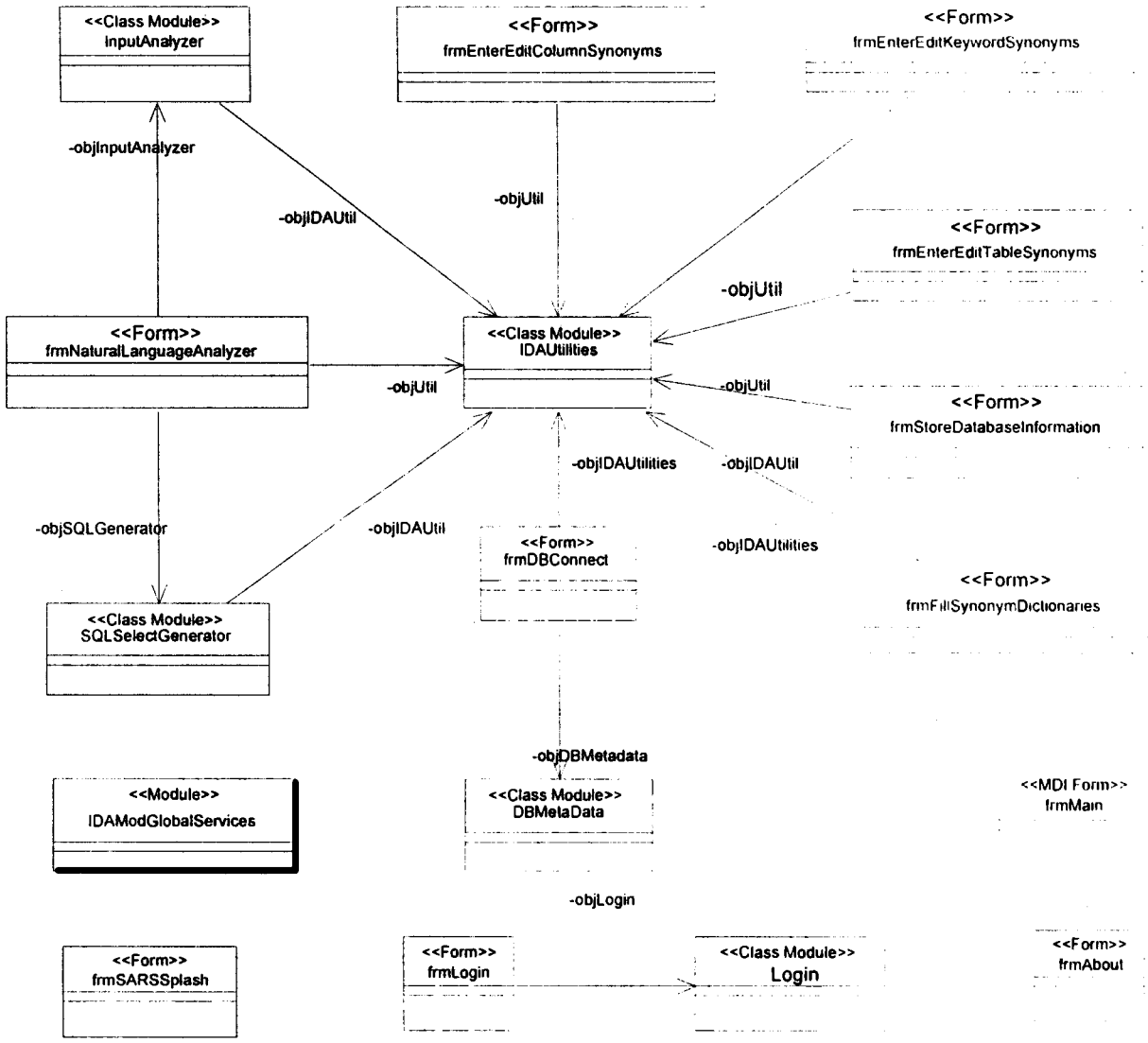

Figure 5.6: Class Diagram of IDA

Now we will show the detailed view of each class. Class IDAModGlobalServices is shown in Figure 5.7.

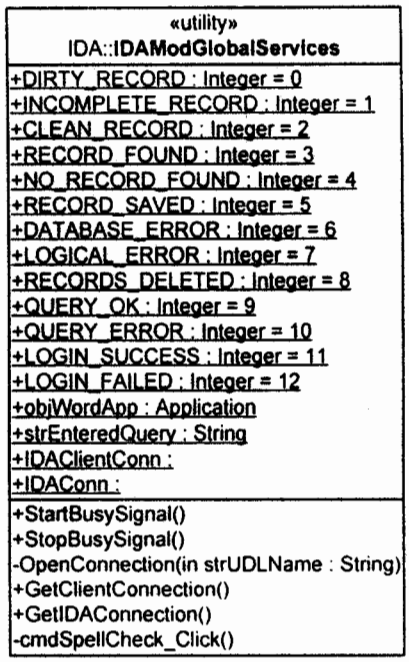

Figure 5.7: Class IDAModGlobalServices

Class frmDBConnect is shown in Figure 5.8.

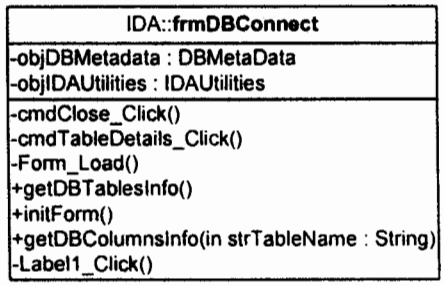

Figure 5.8: Class frmDBConnect

Class frmEnterEditColumnSynonyms is shown in Figure 5.9.

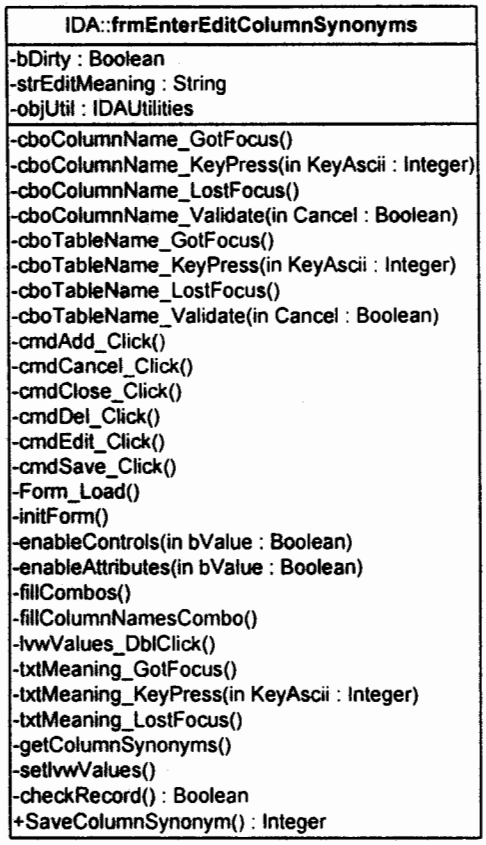

Figure 5.9: Class frmEnterEditColumnSynonyms

Class frmAbout is shown in Figure 5.10.

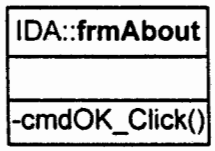

Figure 5.10: Class frmAbout

Class frmMain is shown in Figure 5.11.

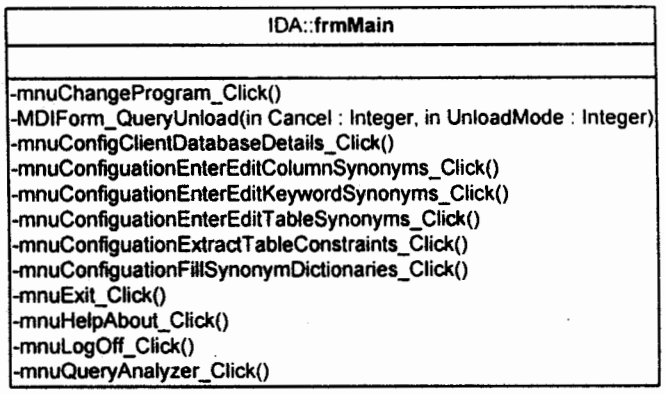

Figure 5.11: Class frmMain

Class frmNaturalLanguageAnalyzer is shown in Figure 5.12.

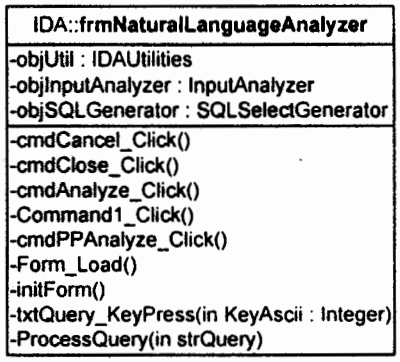

Figure 5.12: Class frmNaturalLanguageAnalyzer

Class DBMetaData is shown in Figure 5.13.

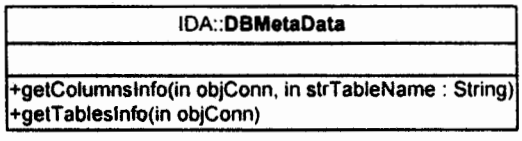

Figure 5.13: Class DBMetaData

Class frmEnterEditTableSynonyms is shown in Figure 5.14.

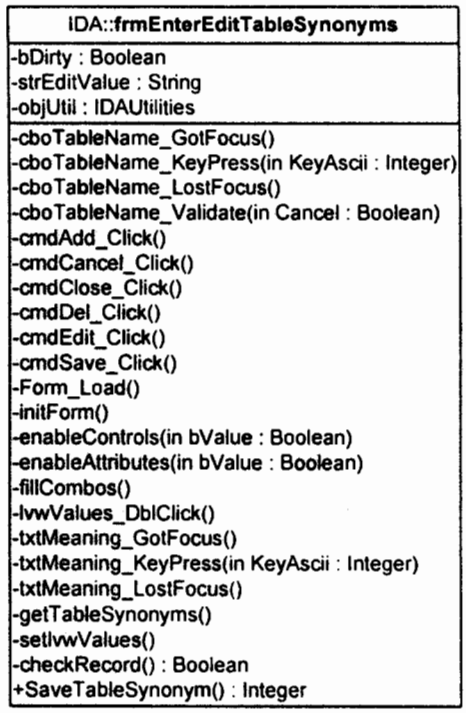

Figure 5.14: Class frmEnterEditTableSynonyms

Class SQLSelectGenerator is shown in Figure 5.15.

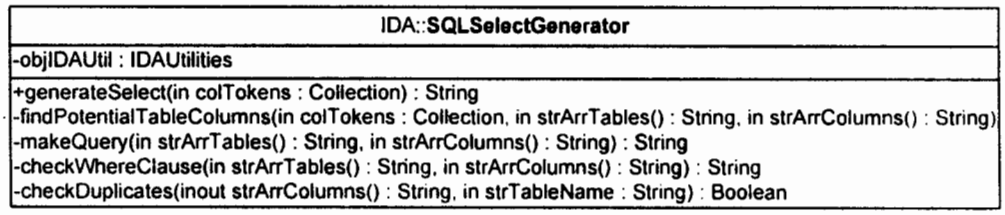

Figure 5.15: Class SQLSelectGenerator

Class frmFillSynonymDictionaries is shown in Figure 5.16.

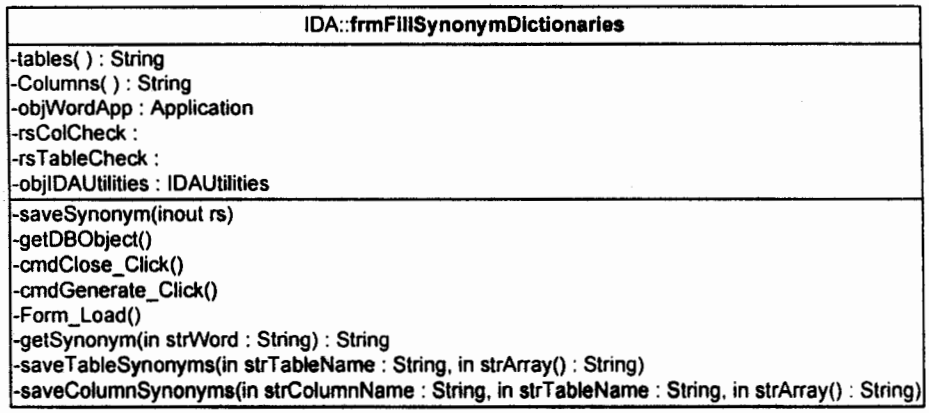

Figure 5.16: Class frmFillSynonymDictionaries

Class frmStoreDatabaseInformation is shown in Figure 5.17.

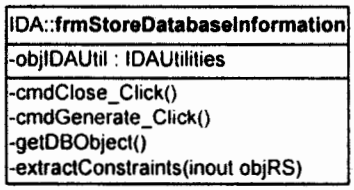

Figure 5.17: Class frmStoreDatabaseInformation

Class Login is shown in Figure 5.18.

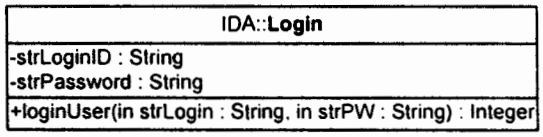

Figure 5.18: Class Login

Class InputAnalyzer is shown in Figure 5.19.

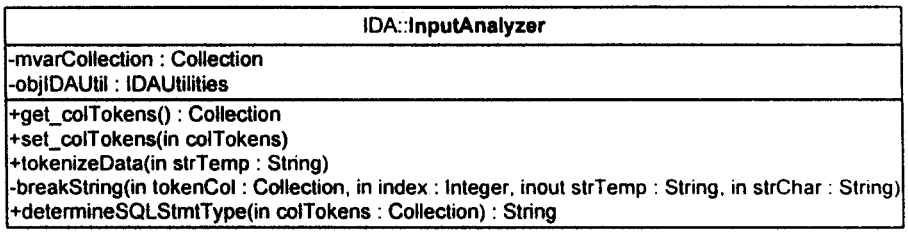

Figure 5.19: Class InputAnalyzer

Class IDAUtilities is shown in Figure 5.20.

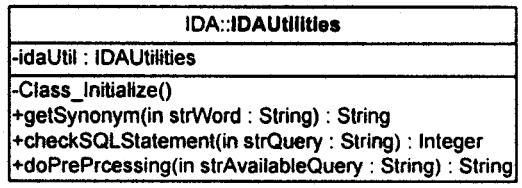

Figure 5.20: Class IDAUtilities

Class frmLogin is shown in Figure 5.21.

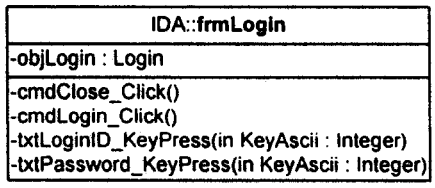

Figure 5.21: Class frmLogin

Class frmEnterEditKeywordSynonyms is shown in Figure 5.22.

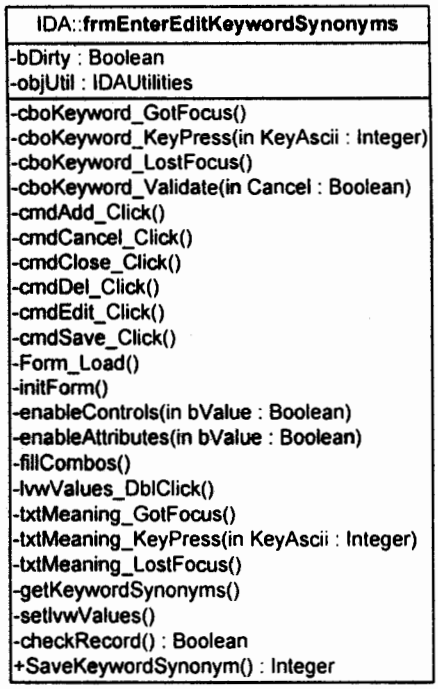

Figure 5.22: Class frmEnterEditKeywordSynonyms

Class frmSARSSplash is shown in Figure 5.23.

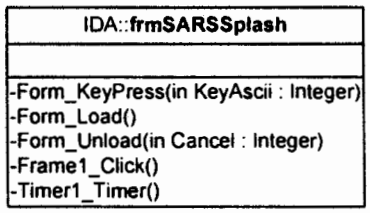

Figure 5.23: Class frmSARSSplash

5.6 Sequence Diagram

Sequence diagrams are used to show the flow of functionality through a use case .For one use case diagram there can be multiple sequence diagrams. For alternate course of actions there are separate sequence diagrams. Sequence diagrams are time dependent and tell which operation will be executed first .Sequence diagrams define a pattern of interaction among objects arranged in chronological order. These show the objects participating in interaction by the order of their life times and the messages being sent from one object to the other. The following are the sequence diagrams:

5.6.1 Enter/Edit Table Synonyms

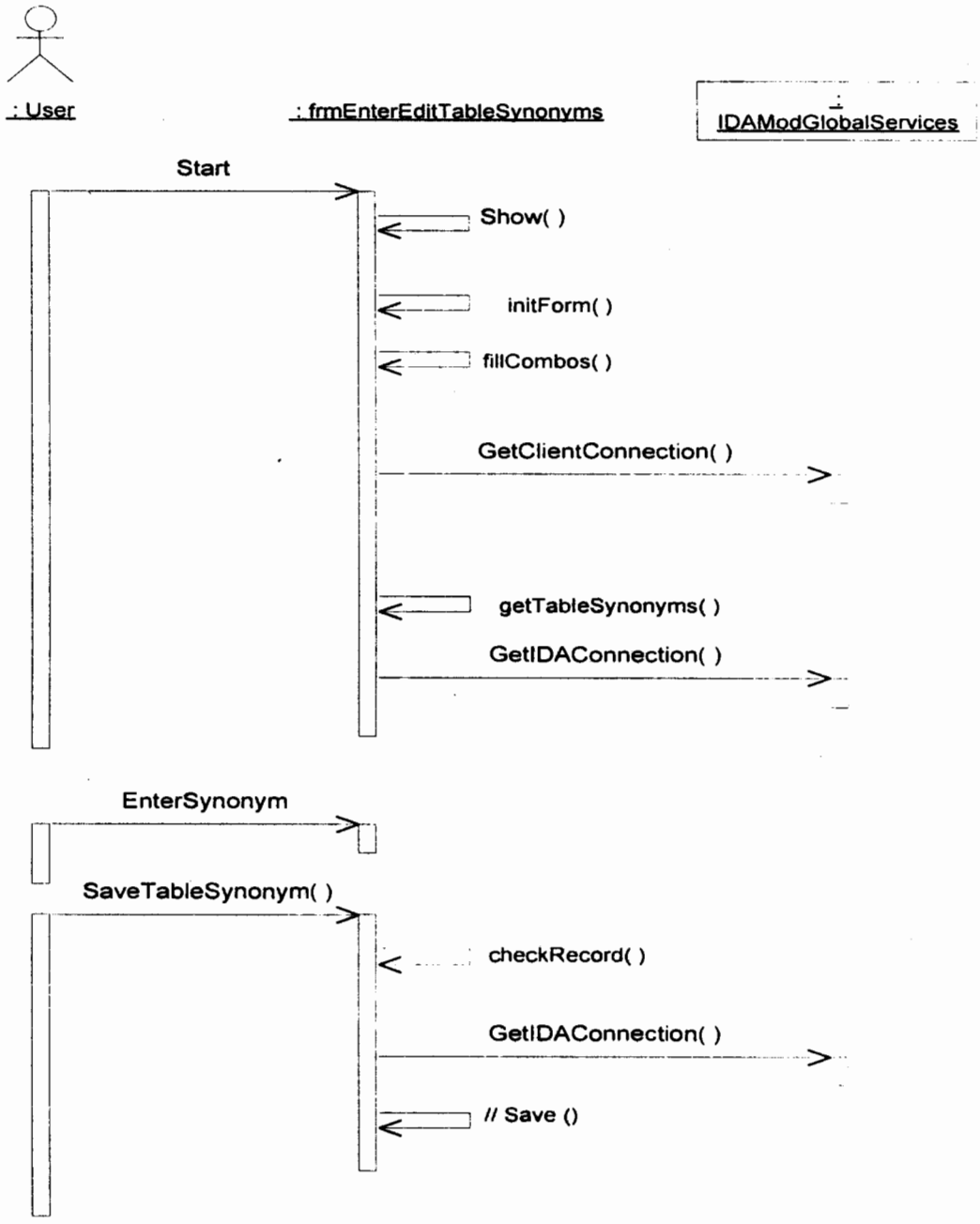

Figure 5.24: Sequence diagram of Enter/Edit Table Synonyms

5.6.2 Enter/Edit Column Synonyms

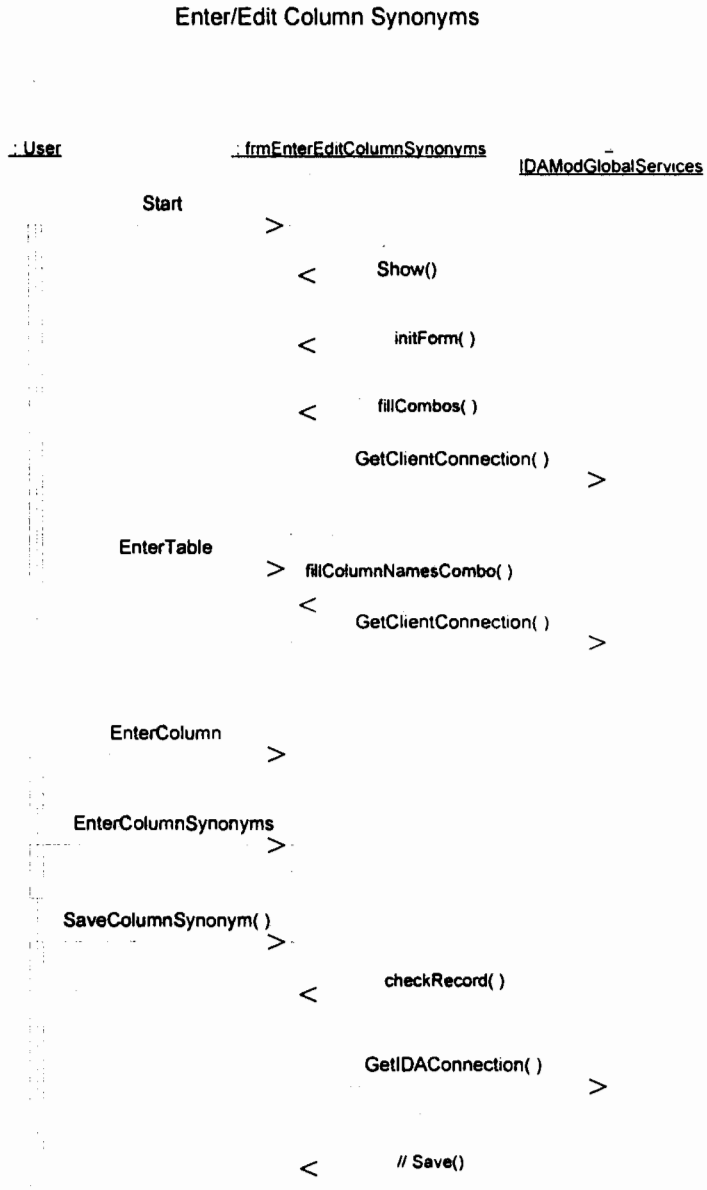

Figure 5.25: Sequence diagram of Enter/Edit Column Synonyms

5.6.3 Enter/Edit Keyword Synonyms

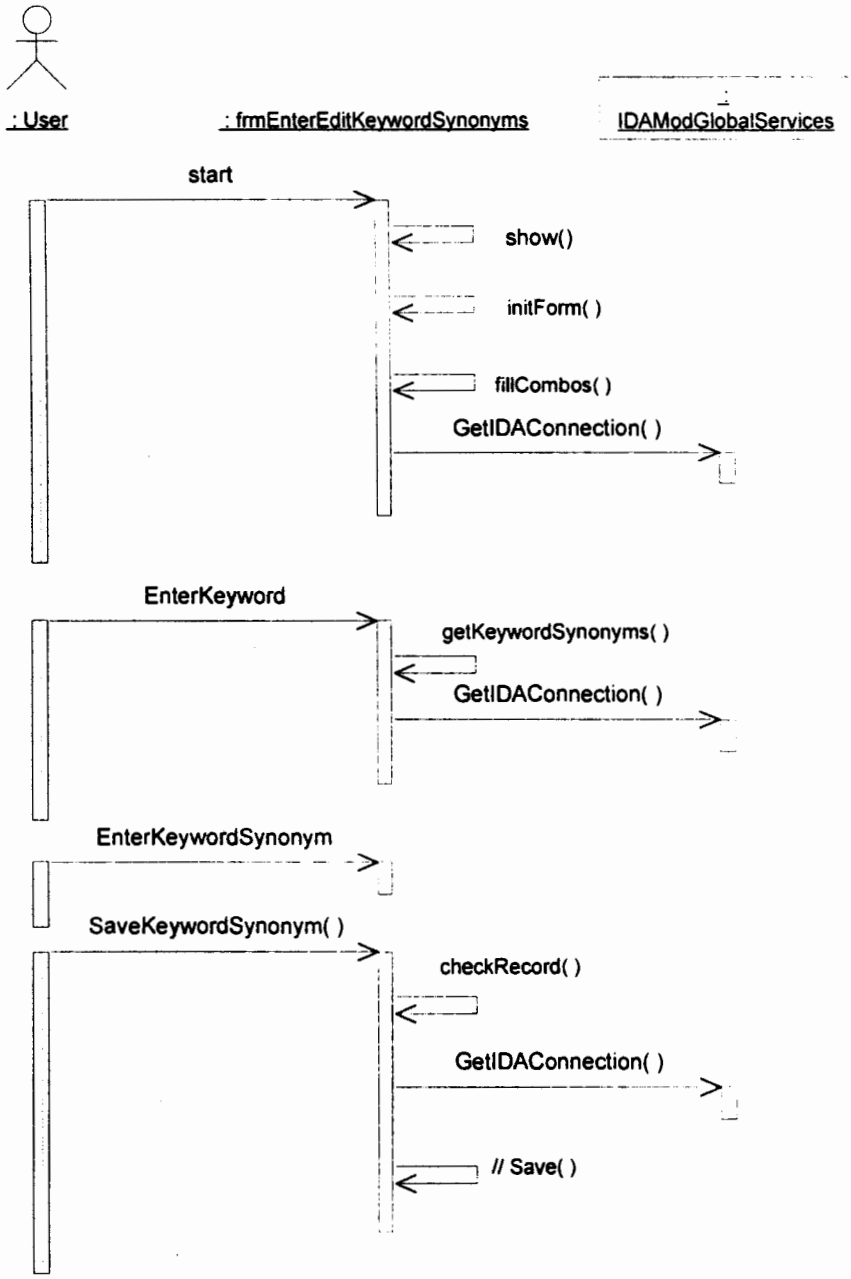

Figure 5.26: Sequence diagram of Enter/Edit Keyword Synonyms

5.6.4 Database Information

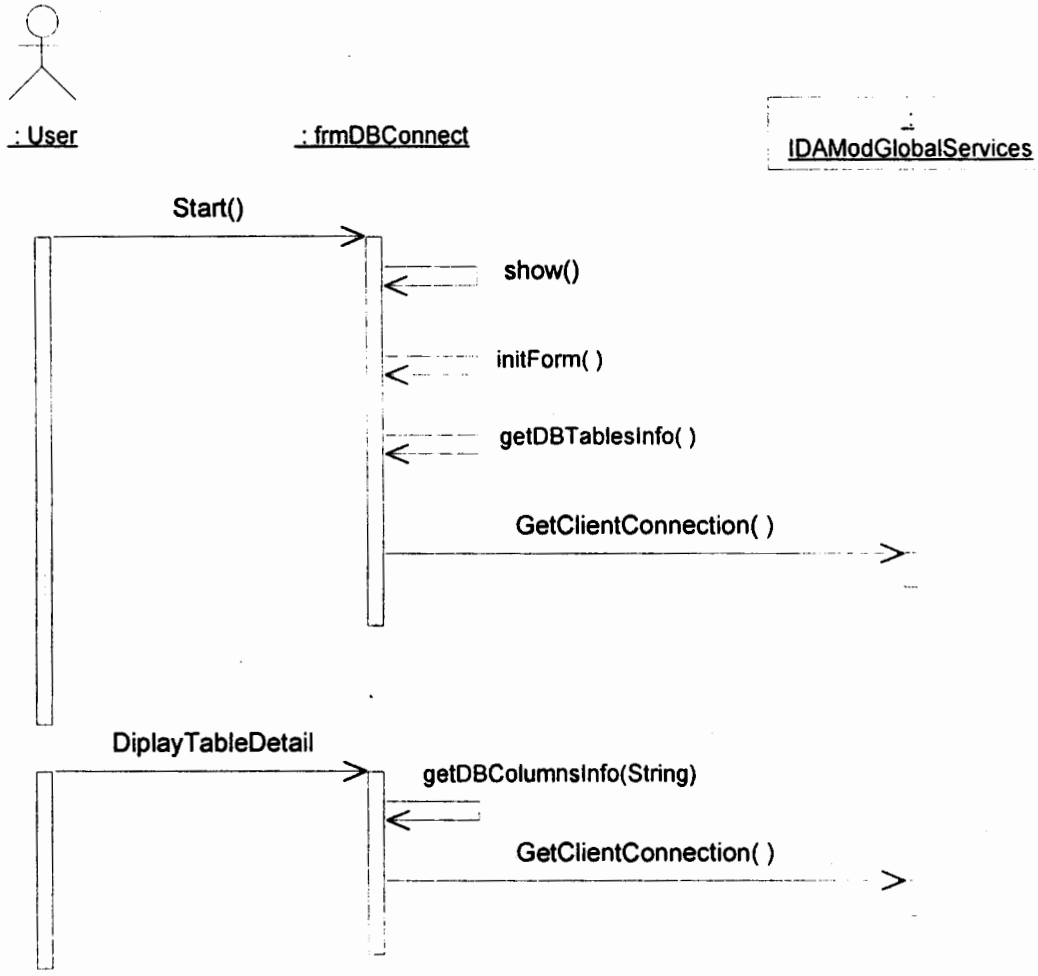

Figure 5.26: Sequence diagram of Database Information

Chapter # 6

Implementation

6. Implementation

The Intelligent Database Agent is a simple Database front end based on artificial intelligence and natural language processing. The basic idea behind the creation of the Intelligent Database Agent is to communicate with the database without having in depth knowledge of the database like the database tables and relations etc. and without the help of specialized database query languages like SQL instead using simple English.

The database Agent converts the English query into structured query language understandable by the underlying database by applying knowledge of the database and the English language.

The theme behind Intelligent Database Agent is the effective use of natural language based input given by the user through identifying and using the keywords present in the user's entered query. This information in the form of keywords could then be combined with database information and knowledge present in the system to make a language. Structured Query Language in this case to talk to the underlying database.

6.1 Structured Query Language

Structured Query Language is just like any other programming language and is specialized in Database Querying. Currently SQL is the generally accepted, most commonly used database query language. However SQL has some disadvantages:

- The language is difficult to use and apply by any person who is not well conversant to this computer language.
- There is no real standard (each database has its own SQL dialect).
- There are too many syntax rules.
- Some functionality seem outdated (e.g. length restriction for text; this was probably inherited from COBOL – modern languages like Java do not know such restrictions).
- "Null" is not equivalent to "null" - this is not obvious for many developer.

- It is difficult to integrate SQL in some languages like Java.
- The SQL language is very old and the language was never revised, but extends in each case.

The Intelligent Database Agent diagram is shown in figure 6.1:

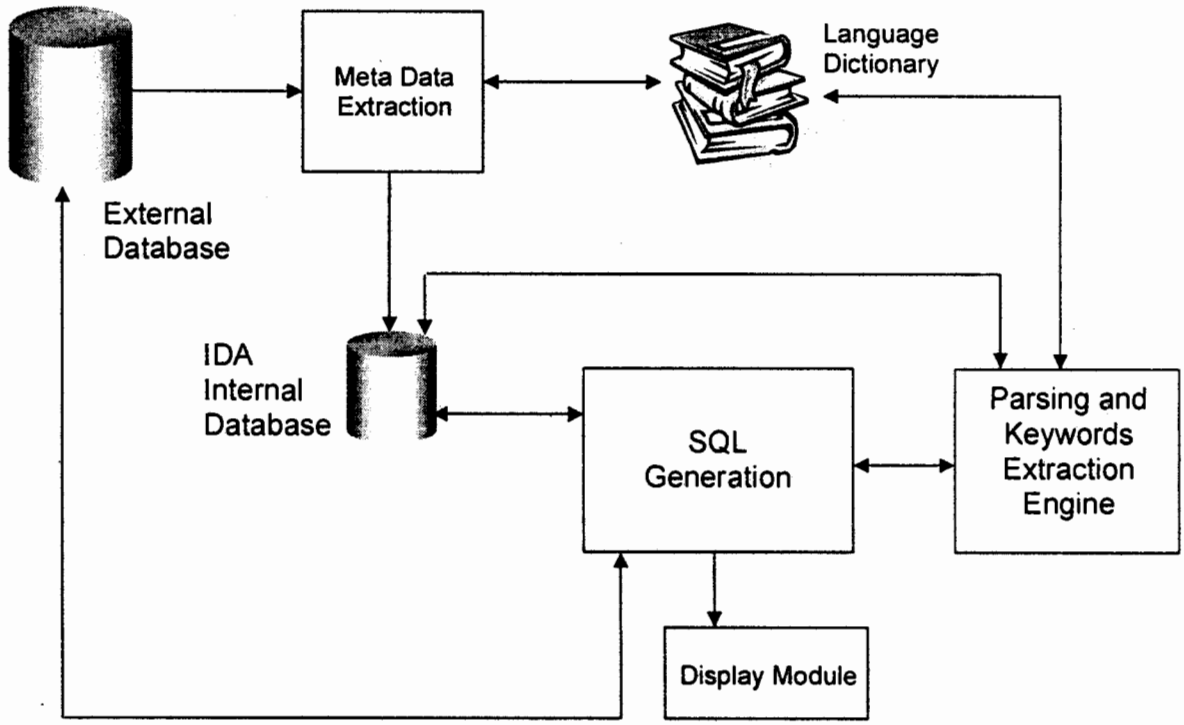

Figure 6.1: Intelligent Database Agent

6.2 Modules of Intelligent Database Agent

The Intelligent Database Agent is composed of the following modules:

- Database Connection with NLP system
- English Language dictionary interface.
- Database Information Extraction
- IDA internal Database
- Parsing and keywords extraction engine.
- SQL generation Engine
- User Interface (input and help regarding result generation and the output of the result)

6.2.1 Database Connection with NLP system

Most of the systems which aim at providing a natural language interface to the database are restricted to a particular domain or database. This is due to the fact that as the domain of the system gets bigger it will give rise to ambiguities and it would be even harder to deal with the natural language queries which would be of a great variety. So the resulting system would be more complex requiring greater lexicon, flexibility, automatic configuration capability and also more computation.

Intelligent Database Agent is aimed at providing database interfacing not only to a particular domain or database but it is tried to make it open so that any database could be connected to it irrespective of it's domain and the system would be able to talk to it with little configuration.

At one time the Intelligent Database Agent would be connected to two databases, at the most. These are

- External Database
- IDA's Internal database

External database is the database which is connected to the Intelligent Database Agent whose interface it has to provide to the user to retrieve the required information and IDA's internal database is the database which is used to store the lexicon and other configuration information of the external database.

Both the databases are developed in Oracle. IDA basically focuses on the Structured Query Language for Oracle which has a little difference but most of it would also run on SQL Server and other databases using the same dialect of SQL.

6.2.1.1 Universal Data Link

Universal Data Link (UDL) uses file to connect to the databases. The UDL file connection is just like the DSN based connection but it is a relatively newer technique and can be ported along with the application. With the change in the Oracle client version or change in Oracle Database we have to reconfigure the UDL file easily with the Windows based wizard like interface.

We are using two UDLs here :

- IDA
- IDAClient

Incase of changing the client database the IDA Client UDL will have to be changed accordingly based on the Oracle Database and version. Once the IDA Client UDL is changed there would be little configuration that would be needed and the system is ready to go.

There is a need that the connection with the database would be reliable and smooth ensuring the usage of the system without hassles. This technique of data access is used

widely today in enterprises to connect to remote database servers and is also easy to run and maintain incase changes occur.

6.2.1.2 Client Database Details

The system is provided with the interface where the system shows some details of the connection. This includes some database driver related information, user information, version information and so on that might be of interest to the user. It also provide the list of the various database objects and table's column's details, types, lengths and null constraint to the user with the confirmation and basic details of the connected database.

6.2.2 English Language Dictionary Interface

As the Intelligent Database Agent is a database interface agent who has to provide database interface to the user and is not much conversant with Structured Query Language and natural language, English in this case would be the source of communication between the IDA and the user. Such type of natural language interfaces could be very useful as they are relatively open thing when it comes to better understanding of the user input and it is also updated so it is a good idea to use any such dictionary. To take advantage of such facilities Intelligent Database Agent is connected to the Microsoft Word's internal English language dictionary which shipped with Microsoft Word by default. This dictionary comes with MS Word so it does not have to be explicitly installed on to the system. It is connected to IDA using automation of Microsoft Word.

The thesaurus and synonyms serves in a very useful way of better understanding and interpreting the user input. This dictionary is also used during the configuration phase of the system as well as during the understanding the user input for the transformation of Structured Query Language from the user's input.

6.2.3 Database Information Extraction

Most small-scale NLP systems use a semantic grammar. These are different to normal English grammar parsers in the way they classify words and phrases. For example, in normal English the word 'movie' is just a *noun*, but in a semantic grammar it can be classified differently depending on its meaning, or semantics.

6.2.3.1 Database Independent Information

The first, lowest-level type of information is a combination of the two languages involved, English and SQL. These languages are common to any similar NLP system. At this level we have represented the basics of queries, no matter which database we will eventually use. For example the "greater than or equal to" in English is the representation of >= in SQL. This type of information is called the database independent information.

This information is very important because it helps in the generation of keywords in the SQL from user input. Apart from key words it also helps the system understand particular type of user phrases. The greater the information the wider will be the domain. More information of this sort can be added into IDA with the help of provided interface time to time as the need arise and thus making the system more dynamic.

6.2.3.2 Database Related Information

This level of information includes the information related to the database for which the IDA acts as an interface. This includes the information related to the database structure for example how database tables are joined together, what type of information is stored in the different attributes, as well as table and attribute names and the database information content.

This information is needed when the database is connected to the IDA system. Some information require manual configuration. IDA shows the user, the database related

information including the connection related information, the tables along with their attributes and the data types for allowing the user to get first hand information about the database in a user friendly manner. IDA has the capability to extract database related information of the tables their relationships and keys (primary key and foreign keys).

6.2.3.3 Database Semantics Information

The third level of information can extend, or alter the other layers, depending on how the database is referred to in English. Sometimes simple assumptions may have to be made in order to represent the database in English; for example the FNAME attribute is not referred to as 'fname', but rather 'first name'. This sort of information is not always obvious from the database itself, so some other source of information must be found. For the NLP systems the human author serves as this source of information. This is the main weakness of general NLP systems – there is no extra source of third level information.

IDA provides mechanism for obtaining this information from the user which will be used for the generation of SQL. As there is no other source of this information the user has to provide this information with the help of appropriate forms provided.

Most systems that provide some kind of Natural interface to the databases or any other source of information must have in depth knowledge of the domain in which they are providing any kind of service. Introducing this kind of knowledge is very beneficial and effective in the case of any computer based system, not necessarily a Natural Language based system. As on one hand this type of intelligence is very helpful, but on the other hand it has its own price in the form of storage and computation needs. The computer being a dumb machine requires a lot of information to show a little intelligence. Generally Natural Language based systems have huge lexicons, rules and grammars but still they do not perfectly answer user's queries. Even if the domain of the system is restricted the ambiguities that arise from natural language are far greater. This is the basic cause of the low market

share of the natural language based database interfaces even though the research started decades before.

In the case of this system it is not necessary that the user will be fully aware of the database related issues. It is tried that the system should behave in such a way that it does not require user knowledge of the database as much as possible. In the ideal condition it would be such that the system will be intelligent enough to know about all the details of the database and it should not require human need in any way and instead it will guide the user in any kind of database interaction, but such type of systems are quite difficult to make as the domain of the system would become very large and hence very hard to manage.

6.2.3.4 IDA Configuration

Most of the system that deals with natural language requires user configuration to enable it to work in its required manner. Actually Natural Language based database interfaces normally require tedious and lengthy configuration phases before they can be used. Where as most of the commercially available database systems have built-in formal query language interpreters, and the implementation of form-based interfaces is largely automated.

In practice natural language database interfaces are often configured by the administrators of existing databases. As most of the natural language systems focus on a particular type of area or database to be specific, most of them would not work on any other domain or database. These systems also require much information from the end user who is familiar with the database, for their working. There are systems that work on a variety of domains but as the same system is used for a variety of domains, with the change in the underlying database, a lot of configuration has to be made in the system to adjust according to the new database in order to work properly.

IDA also requires some configuration that has to be done by the user of the system who is bit familiar with the database tables etc. This configuration is necessary for the working of the system. In a database where the column and the table names are descriptive

and properly named, fewer configurations would be needed. The use of meaningful names and correct spelling increases readability and maintainability of the database.

The configuration of the system is key to the better results obtained from the system. The systems which are related to artificial intelligence even to some extent maintain their knowledge base and set of rules with the help of which they fetch related information and make some sort of decision out of it. The modern NLP systems have huge lexicons to support them because if the system has less data then it would have less information with the help of which it can do something meaningful and failure rate increases and so is the desperation on the part of user.

6.2.4 IDA Internal Database

Intelligent Database Agent requires some information of the database to be used later on in processing the user input in natural language. The system requires this simple configuration only once a new client database is connected to the IDA system for running the system. This activity may also be required when there is a change in the database tables structure, like a new column is added to the database or entirely new table(s) are added to the database so that all the changes could be incorporated in the IDA and the system could handle these newly made changes.

The information is stored in the table of the IDA system. The information stored is given as follows

Table Name: IDATableConstraints

Column Names:

1. TableName
2. ColumnName
3. KeyType
4. ReferencedTable

The field TableName is used to store the name of the table of the client database connected which is under consideration. This will help the system to understand and relate the rest of the information in the table to the table so that a complete picture of the structure could be made.

The ColumnName field tells about the field which is under consideration. This will clear that which column of the table the information stored is about.

The field KeyType stores the information about the key of the table. The key could be primary key or foreign key. Primary key would tell that the corresponding column is the primary key of the table whose name is in the TableName field, which would guarantee that it would be unique and rest of the columns would be fully functionally dependent on this field (if the table is normalized to 3rd Normal Form). The KeyType field can also contain foreign key which would tell about the parent child relationship between the two tables. This information would be useful incase a join condition is to be established between two tables which have parent, child relationship between them.

The field ColumnName is related to the previously described field which the KeyType. If the KeyType contains a foreign key then this column would tell about the name of the column on which the parent child relationship is made between the two tables. Without this field, information about the foreign key would be virtually useless as it could not be used for any constructive purpose.

Table Name: IDAColumnSynonyms

Column Names:

1. ColumnName
2. ColumnSynonym
3. TableName

The field ColumnName in this table stores the name of all the columns existing in the client database. This information is used when system matches extracts the synonyms for the existing columns by checking the dictionary and analyzing the user input.

The field ColumnSynonym is used to store all the synonyms of the column stored in ColumnName field. These synonyms will be used to compare the input of the user for the generation of SQL query.

TableName field stores the name of the referenced tables of the columns stored in the ColumnName field. This will be helpful to identify the tables required to reference in the SQL statement.

Table Name: IDALexicon

Column Names:

1. Symbol
2. Meaning

In SQL statements, there are different symbols which are used to compare, equalize or negate some conditions, but a naïve user does not know about these standards. So there is a need to store the synonyms for all these symbols which are often used in the queries. This field stores all those symbols required to use in the database retrieval statement.

The field Meaning stores all the possible meanings of the symbol stored in the symbol field. This is done in the configuration mode so that our agent will be capable to understand the natural language input. These meanings are basically different words of the natural language.

Table Name: IDATableSynonyms

Column Names:

1. TableName
2. TableSynonym

This table of IDA internal database stores all the table names and their possible synonyms of the client database. This is done by the information extraction engine which extracts all the table names of the external database. The field TableName stores the names of the tables

TableSynonym stores all the possible synonyms of the external database tables. These synonyms are generated automatically by the system as well as manually given during the configuration mode.

Table Name: IDALogin

Column Names:

1. UserID
2. Password

UserID keep the unique username of all the users allowed to use the system, and Password field stores the password of all the users.

5.2.5 Parsing and keywords extraction engine

User gives the input in natural language which is not understandable by the underlying system. To make it understandable by the computer, it is converted in the computer understandable language. It is not easy to effectively and accurately convert the natural language input into the machine understandable language.

When it comes the conversion process then there is a way to do all this job in a formal way. First of all, the input given by the user is broken into small parts known as tokens. These tokens are nothing but the words composed that user input. For example, if user will input the sentence: "Who is the head of finance department?" then tokens for this input are "who", "is", "the", "head", "of", "finance", and "department". This process is called tokenization.

The query input by the user needs to be analyzed and parsed accordingly keeping in view the database related information and the information of the English language available. (using the database independent, database dependent and the database semantics information). The structure of the SQL queries will be analyzed and keywords will be extracted from the user input keeping in view this information.

IDA analyses the user entered data and tokenizes the input string. Later it searches this user input for SQL query type. Then it extracts possible column names and tables that are involved in the query after consulting the internal database and MS Word dictionary.

6.2.5.1 Tokenization

Tokenization is the process of breaking the input up into distinct tokens. In languages like English, tokens are simply the words and punctuation which comprise the input string. For some languages such as Japanese or Chinese, tokenization is very difficult because there are no spaces between words (and placing them there would make the input meaningless). Tokenization routines are optimized for speed rather than accuracy with the idea that if they are consistent in breaking up the input, ambiguity can be dealt with at a later stage of input processing.

6.2.6 SQL generation Engine

Input given by the user in natural language is converted to SQL by this engine with the help of the keywords, the query information along with necessary changes that might be necessary using the tables info, database info and language thesaurus etc the query is generated for the underlying database for the user query (Natural Language query).

IDA uses the columns and tables information that is extracted from the user input along with SQL keywords to generate the required query based on the limitations and standards of Structured Query Language as understandable by ORACLE database.

6.2.7 User Interface

IDA provides a user friendly and simple interface with the help of which it extracts some database information and metadata, which would be used by it in the process of query generation.

6.3 Sample Code

Here the sample code is given for parsing the data and converting it into tokens.

Making the tokens of the Input

```
Dim strChar As String
Dim index As Integer
index = Len(strTemp)

While index <> 0
   index = Len(strTemp)
   If InStr(1, strTemp, " ") <= index And InStr(1, strTemp, " ") <> 0 Then
      index = InStr(1, strTemp, " ")
      strChar = " "
   End If
   If InStr(1, strTemp, vbCrLf) <= index And InStr(1, strTemp, vbCrLf) <> 0 Then
      index = InStr(1, strTemp, vbCrLf)
      strChar = vbCrLf
   End If
   If InStr(1, strTemp, ",") <= index And InStr(1, strTemp, ",") <> 0 Then
      index = InStr(1, strTemp, ",")
```

```
        strChar = ","
      End If
      If InStr(1, strTemp, "(") <= index And InStr(1, strTemp, "(") <> 0 Then
        index = InStr(1, strTemp, "(")
        strChar = "("
      End If
      If InStr(1, strTemp, ")") <= index And InStr(1, strTemp, ")") <> 0 Then
        index = InStr(1, strTemp, ")")
        strChar = ")"
      End If

      If InStr(1, strTemp, "<") <= index And InStr(1, strTemp, "<") <> 0 Then
        index = InStr(1, strTemp, "<")
        strChar = "<"
      End If

      If InStr(1, strTemp, ">") <= index And InStr(1, strTemp, ">") <> 0 Then
        index = InStr(1, strTemp, ">")
        strChar = ">"
      End If

      If InStr(1, strTemp, "=") <= index And InStr(1, strTemp, "=") <> 0 Then
        index = InStr(1, strTemp, "=")
        strChar = "="
      End If

      If index = 0 Or Len(strTemp) = 1 Then
        breakString tokenCol, index, strTemp, ""
      Else
        breakString tokenCol, index, strTemp, strChar
        strChar = ""
      End If

    Wend

    Set mvarCollection = tokenCol
```

```
    Exit Sub
Error_Handler:
    MsgBox Err.Number & " " & Err.Description & "IDA"
End Sub


Private Sub breakString(tokenCol As Collection, index As Integer, ByRef strTemp As String, strChar As
String)
        If strChar = " " Then
            index = InStr(1, strTemp, " ")
            If Trim(Left(strTemp, index)) <> "" And index > 1 Then
                tokenCol.Add Trim(Left(strTemp, index))
            End If
            strTemp = Trim(Right(strTemp, Len(strTemp) - index))

        ElseIf strChar = vbCrLf Then
            index = InStr(1, strTemp, vbCrLf)
            If Trim(Left(strTemp, index)) <> "" And index > 1 Then
                tokenCol.Add Trim(Left(strTemp, index - 1))
            End If
            strTemp = Trim(Right(strTemp, Len(strTemp) - index - 1))

        ElseIf strChar = "," Then
            index = InStr(1, strTemp, ",")
            If Trim(Left(strTemp, index)) <> "" Then
```

If the first alphabet is "," then else is executed

```
                If index > 1 Then
                    tokenCol.Add Trim(Left(strTemp, index - 1))
                    tokenCol.Add Trim(Mid(strTemp, index, 1))
                Else
                    tokenCol.Add Trim(Mid(strTemp, index, 1))
                End If
            End If
            strTemp = Trim(Right(strTemp, Len(strTemp) - index))
```

```
ElseIf strChar = "(" Then
   index = InStr(1, strTemp, "(")
   If Trim(Left(strTemp, index)) <> "" Then
      If index > 1 Then
         tokenCol.Add Trim(Left(strTemp, index - 1))
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      Else
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      End If
   End If
   strTemp = Trim(Right(strTemp, Len(strTemp) - index))

ElseIf strChar = ")" Then
   index = InStr(1, strTemp, ")")
   If Trim(Left(strTemp, index)) <> "" Then
      If index > 1 Then
         tokenCol.Add Trim(Left(strTemp, index - 1))
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      Else
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      End If
   End If
   strTemp = Trim(Right(strTemp, Len(strTemp) - index))

ElseIf strChar = "<" Then
   index = InStr(1, strTemp, "<")
   If Trim(Left(strTemp, index)) <> "" Then
      If index > 1 Then
         tokenCol.Add Trim(Left(strTemp, index - 1))
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      Else
         tokenCol.Add Trim(Mid(strTemp, index, 1))
      End If
   End If
   strTemp = Trim(Right(strTemp, Len(strTemp) - index))
```

```
    ElseIf strChar = ">" Then
      index = InStr(1, strTemp, ">")
      If Trim(Left(strTemp, index)) <> "" Then
        If index > 1 Then
          tokenCol.Add Trim(Left(strTemp, index - 1))
          tokenCol.Add Trim(Mid(strTemp, index, 1))
        Else
          tokenCol.Add Trim(Mid(strTemp, index, 1))
        End If
      End If
      strTemp = Trim(Right(strTemp, Len(strTemp) - index))

    ElseIf strChar = "=" Then
      index = InStr(1, strTemp, "=")
      If Trim(Left(strTemp, index)) <> "" Then
        If index > 1 Then
          tokenCol.Add Trim(Left(strTemp, index - 1))
          tokenCol.Add Trim(Mid(strTemp, index, 1))
        Else
          tokenCol.Add Trim(Mid(strTemp, index, 1))
        End If
      End If
      strTemp = Trim(Right(strTemp, Len(strTemp) - index))

    Else
      index = InStr(1, strTemp, " ")
      If Len(strTemp) > 0 Then
        tokenCol.Add Trim(Right(strTemp, Len(strTemp) - index))
      End If
    End If
End Sub
```

Checking if any of the natural language or it's meaning entered by the user resembles any SQL statement

```
Public Function determineSQLStmtType(ByVal colTokens As Collection) As String
On Error GoTo Error_Handler


   Set objIDAUtil = New IDAUtilities
   Dim objRS As ADODB.Recordset
   Set objRS = New ADODB.Recordset
```

Checking for the select statement

```
   objRS.Open "Select meaning from idalexicon where upper(symbol) = 'SELECT'",
GetIDAConnection, adOpenStatic, adLockReadOnly
   Dim i As Integer
   i = 1
   Dim arrSynonyms() As String
   While i <= colTokens.Count
      arrSynonyms = objIDAUtil.getSynonym(colTokens.Item(i))
```

Checking it against the select synonyms

```
      If objRS.EOF Then objRS.MoveFirst
      While Not objRS.EOF
         Dim j As Integer
         For j = LBound(arrSynonyms) To UBound(arrSynonyms)
```

Compare select meaning with token meaning and select meaning with token itself and select = token and select = token synonym

```
            If UCase(objRS.Fields("meaning")) = UCase(arrSynonyms(j)) Or
UCase(objRS.Fields("Meaning")) = UCase(colTokens.Item(i)) Or "SELECT" = UCase(colTokens.Item(i)) Or
"SELECT" = UCase(arrSynonyms(j)) Then
               determineSQLStmtType = "SELECT"
               Exit Function
            End If
         Next j
         objRS.MoveNext
```

```
        Wend
        i = i + 1
    Wend
    determineSQLStmtType = "Not Found"
    Exit Function

Error_Handler:
    MsgBox Err.Number & " " & Err.Description & "IDA"
End Function
```

Chapter # 7

Testing

7. Testing

Testing is an important phase during software development life cycle, and shows the stability of the product. It also helps in comparing the final product with the objectives. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

Testing should focus upon the system's internal and external behavior, a secondary purpose of testing is pushing the limits of the system in order to understand how it fails under certain conditions. A design must be testable. An implementation must be tested.

7.1 Objective of Testing

The overall objective of the testing is to find the maximum number of errors in minimum amount of effort.

7.2 Object Oriented Testing Strategies

Testing begins with unit testing, then progress towards integration testing, and culminates with validation and system testing. In unit testing single modules are tested first. Once each module is tested individually, it is integrated into a program structures while a series of regression tests are run to uncover errors due to interfacing of modules and side effects caused by addition of new units. Finally the system as a whole is tested.

7.3 Types of Testing Done

We conducted various types of testing to make the software stable and error free.

- **Code Inspection**

  Reviews and walk through of the program code.

- **Unit Testing**

  All the modules of the project were first tested individually by inserting invalid values. Exceptions thrown were properly handled.

- **Integration Testing**

  After the modules were tested individually, they were combined to form the final product. All the links and paths were tested. Invalid values were also checked and measure taken to handle them successfully.

  Tests of inter object and inter process coordination should be built at several granularity levels. For example, tests of two or three interacting objects, dozens of objects, and thousands of them are all needed.

- **Black Box Testing**

  The software was tested for graphical user interface and measures taken that expected output is generated on input.

- **White Box Testing**

  Prior testing is part of white box testing in which we look inside the code. Here we can often find errors, Tests that force most or all computation paths to be visited, and especially those that place components near the edges of their operating conditions form classic test strategies.

- **Beta testing**

  Use by outsiders rather than developers often makes up for lack of imagination about possible error paths by testers.

- **System Testing**

  The software was checked under different operating system like Windows NT and 2000.

- **Portability testing**

  Tests should be applied across the range of systems on which the software may execute. Tests may employ suspected non-portable constructions at the compiler, language, tool, operating system, or machine level.

- **Regression testing**

  Tests should never be thrown out (unless the tests are wrong). Any changes in classes, etc., should be accompanied by a rerun of tests. Most regression tests begin their lives as bug reports.

7.4 Evaluation

Evaluation of the software is carried out to check the stability and usability of the product being developed .We took measures to ensure that the developed software becomes effective and easy to use. Some of the features of the software are given below.

- **Efficiency and Effectiveness**

  The product developed is effective and efficient.

- **Accuracy**

  The software provided reliable results. Format which is not supported can not be opened.

- **Data security and integrity**

  Data is secured, as everyone does not know watermark.

- **Easy to use graphical user interface**

  The graphical user interface used is simple and steps taken that no problems arise during option finding.

- **Data and Design Consistency**

  Throughout uniform notations are used .Only images of given formats can be processed.

Appendix - A

A. Intelligent Database Agent

Intelligent Database Agent is a simple natural language interface to database, based on the simple and efficient keyword extraction technique. A naïve user can easily use it without having the in depth knowledge of database architecture and Standard Query Language. It converts the user input in English language to Standard Query Language understandable by the underlying ORACLE database.

A.1 Description

With Intelligent Database Agent you can perform these functions on an image:

- Configure Database Details
- Fill synonym Dictionaries
- Extract Table Constraints
- Add Table Synonyms
- Edit Table Synonyms
- Delete Table Synonyms
- Add Column Synonyms
- Edit Column Synonyms
- Delete Column Synonyms
- Add Keyword Synonyms
- Edit Keyword Synonyms
- Delete Keyword Synonyms
- Input in Natural Language
- Analyze Natural Language
- Generate SQL query

A.2 Menu Items

Three menu items are there in Intelligent Database Agent: Configuration, Query and Help. These are shown in Figure A.1.

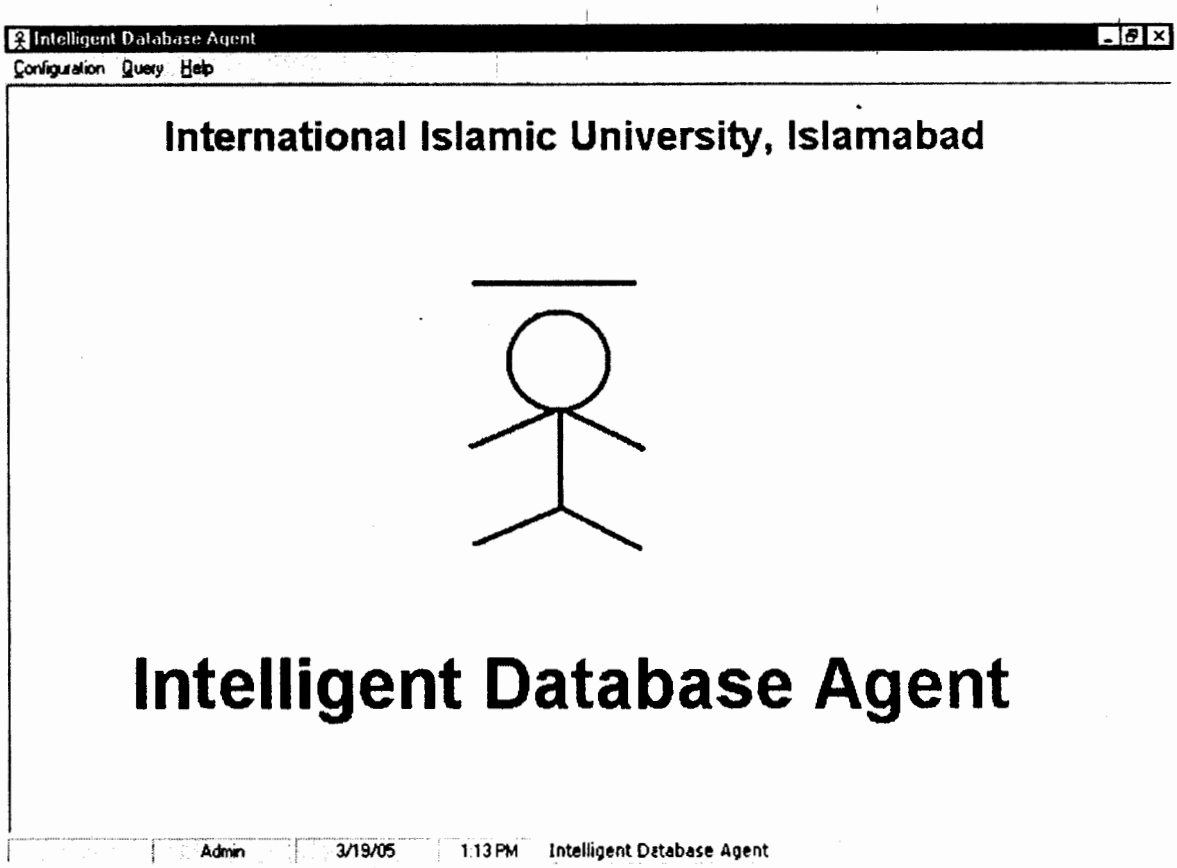

Figure A.1: Main Window of Intelligent Database Agent

A.2.1 Configuration

Under the configuration menu, we have the following options needed to initially configure the database to make it intelligent:

- Client Database Details
- Fill synonym Dictionaries

- Extract Table Constraints
- Enter/Edit Table Synonyms
- Enter/Edit Column Synonyms
- Enter/Edit Keyword Synonyms
- Log Off
- Exit

A.2.1.1 Client Database Details

This option is given for the user to view the detailed information of the connected database, for example tables, columns, data types etc. It also show the user name of the database and driver used to connect the database. The Client Database Details is shown in Figure A.2.

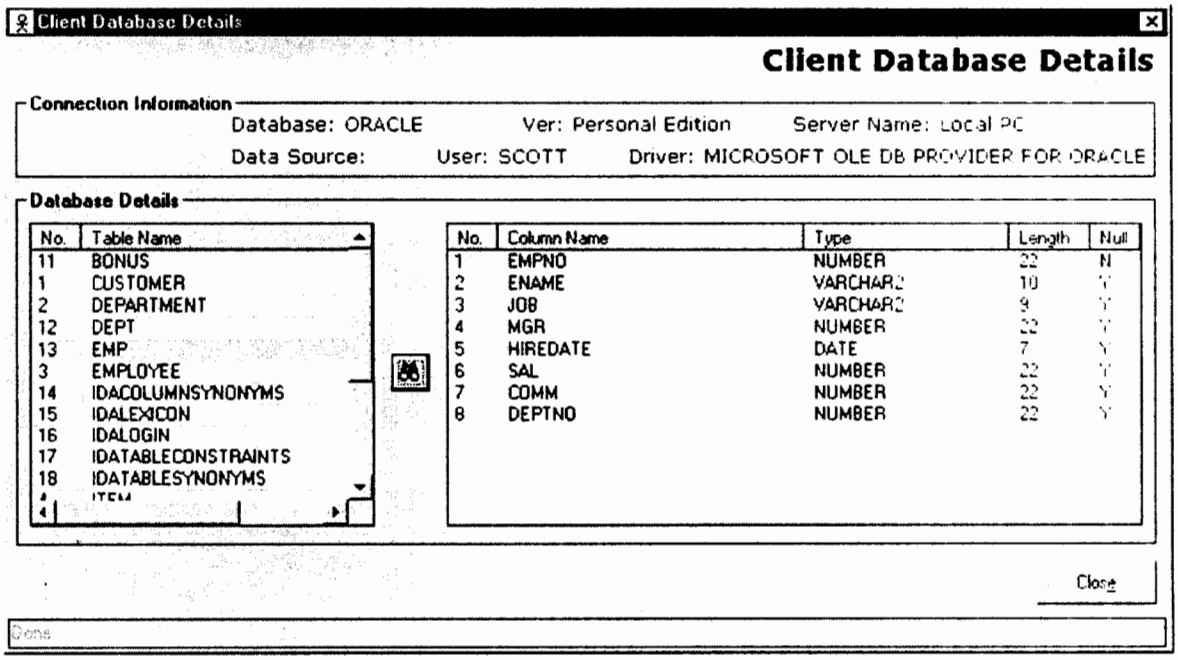

Figure A.2: Client Database Details

A.2.1.2 Fill synonym Dictionaries

It generated all the available synonyms of the tables and columns of the connected database, by looking up the dictionary and the consulting to the internal database of the system. It is shown in Figure A.3.

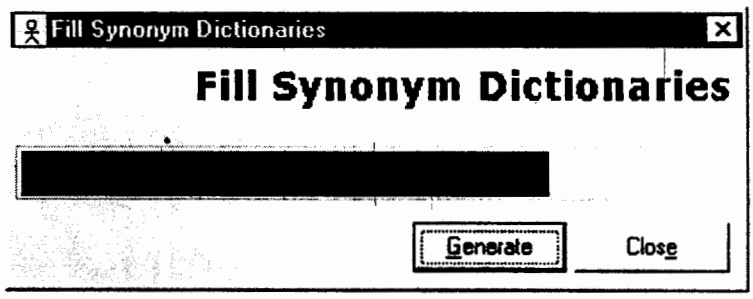

Figure A.3: Generating Synonym Dictionaries

A.2.1.3 Extract Table Constraints

In configuration mode, initially user has to get the constraints of the connected database and store them in the systems internal database. These are used later in SQL generation process. For example Store Database Information is shown in Figure A.4.

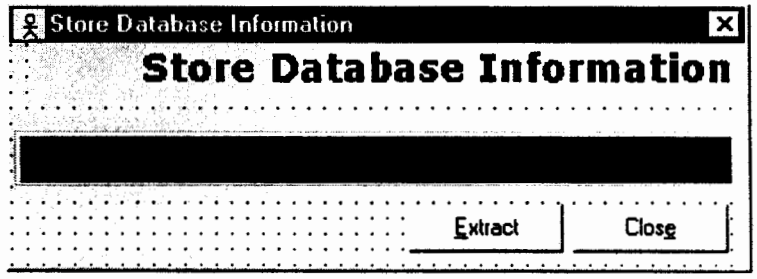

Figure A.4: Extract and save Database Information

A.2.1.4 Enter/Edit Table Synonyms

User has the option to add, edit or delete the synonyms of the tables of the client database. These synonyms are stored in the internal database of the system. This is shown in Figure A.5.

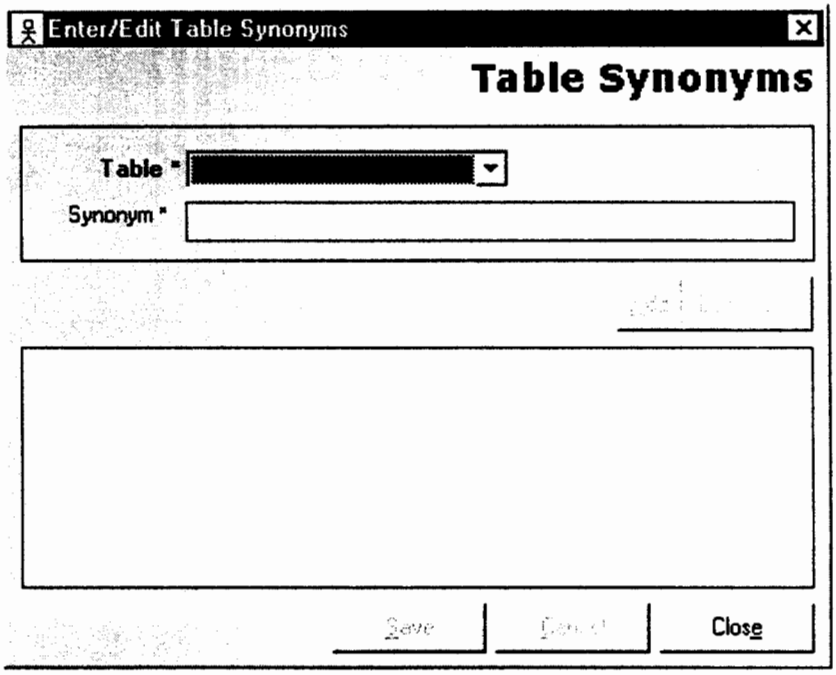

Figure A.5: Enter/Edit Table Synonyms

A.2.1.5 Enter/Edit Column Synonyms

User has the option to add, edit or delete the synonyms of the columns of the related tables of the client database. These synonyms are stored in the internal database of the system. First user selects the table name and then relevant column name. All the synonyms of the selected column will be displayed. The Enter/Edit Column Synonyms is shown in Figure A.6.

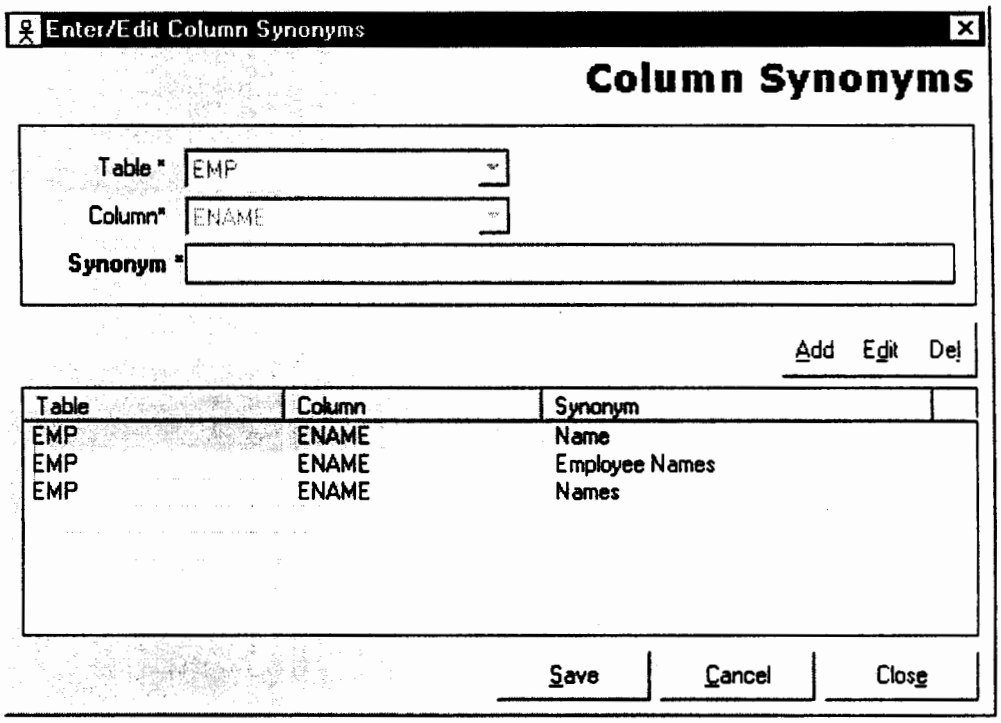

Figure A.6: Enter/Edit Column Synonyms

A.2.1.6 Enter/Edit Keyword Synonyms

In SQL different symbols are used to build a query, but in natural language user can not understand this requirement, so these symbols are configured for their synonyms. These keywords are necessary to be used for different database retrieval operations.

A.2.1.7 Log Off

When user finishes his/her work, he surely has to log out of the system, and terminated the application.

A.2.1.8 Exit

Close the system and exit.

A.2.2 Query

Under this menu item we have only one sub-menu item which is analyzer.

A.2.2.1 Analyzer

This is the main part of the system. By clicking on this sub-menu item, user is shown a form. In this form user inputs the natural language sentences, and then click on the Analyze button to analyze the given input. Then system checks the input, break it into tokens, compare with all the synonym information available and then generate the SQL query as shown in Figure A.7.

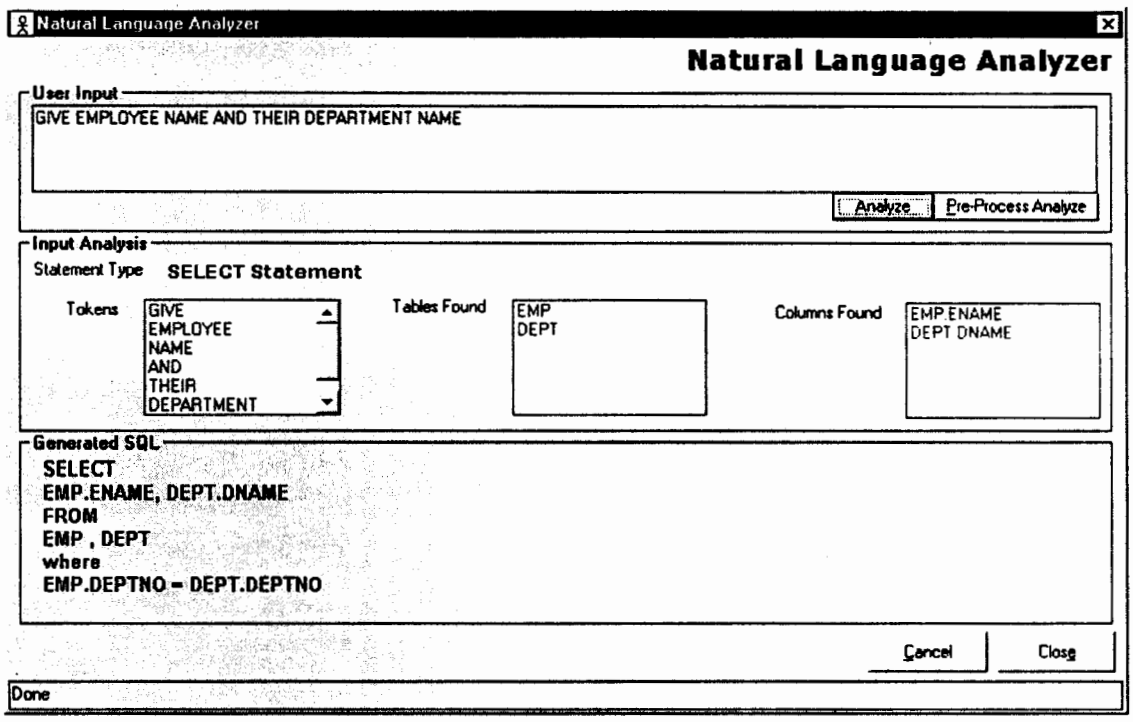

Figure A.7: Natural Language Analyzer

A.2.3 Help

In this main menu we have only one sub-menu item which is about.

A.2.3.1 About

This is about the contributions and work of the developed system shown in Figure A.8.

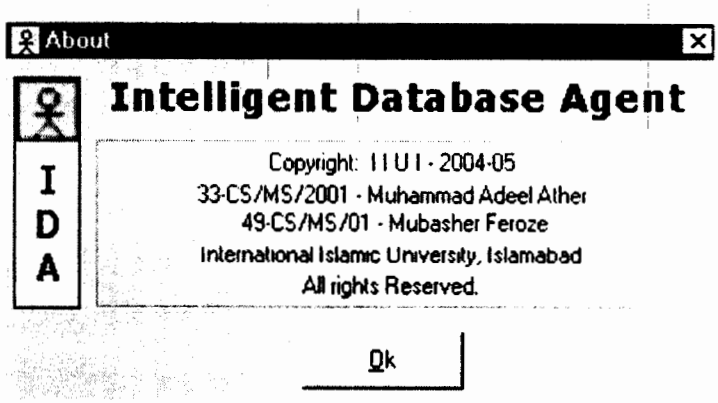

Figure A.8: About Window

Appendix - B

B. Figures Use in Project Documentation

Figure No. **Page No.**

Figure 4.1: Use Case diagram of IDA 41
Figure 4.2: Conceptual Diagram of IDA 61
Figure 5.1: Activity Diagram of Configuring Database 64
Figure 5.2: Activity Diagram of Adding a Synonym 65
Figure 5.3: Activity Diagram of Editing a Synonym 66
Figure 5.4: Activity Diagram of Deleting Synonyms 67
Figure 5.5: Activity Diagram of Generating Query 68
Figure 5.6: Class Diagram of IDA 71
Figure 5.7: Class IDAModGlobalServices 72
Figure 5.8: Class frmDBConnect 72
Figure 5.9: Class frmEnterEditColumnSynonyms 73
Figure 5.10: Class frmAbout 73
Figure 5.11: Class frmMain 74
Figure 5.12: Class frmNaturalLanguageAnalyzer 74
Figure 5.13: Class DBMetaData 74
Figure 5.14: Class frmEnterEditTableSynonyms 75
Figure 5.15: Class SQLSelectGenerator 75
Figure 5.16: Class frmFillSynonymDictionaries 76
Figure 5.17: Class frmStoreDatabaseInformation 76
Figure 5.18: Class Login 76
Figure 5.19: Class InputAnalyzer 77
Figure 5.20: Class IDAUtilities 77
Figure 5.21: Class frmLogin 77
Figure 5.22: Class frmEnterEditKeywordSynonyms 78
Figure 5.23: Class frmSARSSplash 78

Figure 5.24: Sequence diagram of Enter/Edit Table Synonyms 80
Figure 5.25: Sequence diagram of Enter/Edit Column Synonyms 81
Figure 5.26: Sequence diagram of Enter/Edit Keyword Synonyms 82
Figure 5.26: Sequence diagram of Database Information 83
Figure 6.1: Intelligent Database Agent 86
Figure A.1: Main Window of Intelligent Database Agent 112
Figure A.2: Client Database Details 113
Figure A.3: Generating Synonym Dictionaries 114
Figure A.4: Extract and save Database Information 114
Figure A.5: Enter/Edit Table Synonyms 115
Figure A.6: Enter/Edit Column Synonyms 116
Figure A.7: Natural Language Analyzer 117
Figure A.8: About Window 118

Appendix - C

C. Glossary

Compounding takes two words and combines them to form a new word.

Computational Linguistics is doing linguistics with computers.

Data extraction is the process of extracting useful data from a natural language text and placing this data in a structured database record or template.

Discourse level examines the structure of different kinds of text and uses document structure to extract additional meaning.

Information retrieval is a collection of methods which can be used to retrieve documents relevant to a query from a group of documents.

Knowledge-Acquisition is the program that could read books and manuals or the newspaper.

Machine translation uses natural language processing techniques to translate from one natural language to another.

Morpheme is a linguistics term for the smallest piece of a word to carry meaning.

Morphological analysis identifies the prefixes, suffixes, and root words that compose a token of the input string.

Natural Language is a language which humans use when communicating with one another.

Natural Language Processing is the way to convert the Natural Language into the Computer understandable language.

Parsing means breaking a string into parts so that correctly identifying the meaning of different parts of the input.

Phonetics refers to the way the words are pronounced.

Pragmatic level depends on a body of knowledge about the world that comes from outside the contents of the document.

Semantic level examines words for their dictionary meaning,

Software Agent is simply a kind of software abstraction.

Structured Query Language is just like any other programming language and is specialized in Database Querying.

Text categorization is the sorting of natural language texts into fixed topic categories.

Token is the small part of the input sentence.

Tokenization is the process of breaking the input up into distinct tokens.

UDL (Universal Data Link) file connection is used to connect to the database.

Use case is a specific way of using the system by using some part of the functionality.

Bibliography and References

Bibliography and References

[1] Craig. W. Thompson, Kenneth M. Ross, Harry R. Tennant and Richard M. Saenz. Building Usable Menu-Based Natural Language Interfaces To Databases, Central Research Laboratories, Texas Instruments Incorporated, Dallas, Texas.

[2] Roger Curry and Vanessa Le. Artificial Intelligence A Modern Approach (pages 691 - 705), 1999.

[3] Russell S.J. and Norvig, P. (1995; 1998, January) Artificial Intelligence: A Modern Approach. (1999, March 27).

[4] Systran Translation Software. (1999, January).

[5] Communications Research Centre - The CHAT Natural Language System. (1999, March 23).

[6] Becker, K. (1998, November) Signatures. (1999, March 27).

[7] Obtaining Wordnet - (1999, March 27).

[8] H. Alshawi. The Core Language Engine. MIT Press, Cambridge, Massachusetts. 1983.

[9] H. Alshawi, D. Carter, R. Crouch, S. Pulman, M. Rayner, and A. Smith. CLARE - A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. Final report, SRI International, December 1992.

[10] I. Androutsopoulos, G. Ritchie, and P. Thanisch. An Efficient and Portable Natural Language Query Interface for Relational Databases. In P.W. Chung, G. Lovegrove,

and M. Ali, editors, Proceedings of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, U.K., pages 327-330. Gordon and Breach Publishers Inc., Langhorne, PA, U.S.A., June 1993.
ISBN 2-88124-604-4.

[11] P. Auxerre. MASQUE Modular Answering System for Queries in English – Programmer's Manual. Technical Report AIAI/SR/11, Artificial Intelligence Applications Institute, University of Edinburgh. March 1986.

[12] P. Auxerre and R. Inder. MASQUE Modular Answering System for Queries in English – User's Manual. Technical Report AIAI/SR/10, Artificial Intelligence Applications Institute, University of Edinburgh, June 1986.

[13] B. Ballard and D. Stumberger. Semantic Acquisition in TELI. In Proceedings of the 24th Annual Meeting of ACL, New York, pages 20-29, 1986.

[14] B.W. Ballard, The Syntax and Semantics of User-Defined Modifiers in a Transportable Natural Language Processor. In Proceedings of the 22nd Annual Meeting of ACL, Stanford, California, pages 52-56, 1984.

[15] B.W. Ballard, J.C. Lusth, and N.L. Tinkham. LDC-1: A Transportable, Knowledge-based Natural Language Processor for Office Environments. ACM Transactions on Office Information Systems, 2(1):1-25, January 1984.

[16] M. Bates, M.G. Moser, and D. Stallard. The IRUS transportable natural language database interface. In L. Kerschberg, editor, Expert Database Systems, pages 617-630 Benjamin/Cummings, Menlo Park, CA., 1986.

[17] BBN Systems and Technologies. BBN Parlance Interface Software _ System Overview, 1989.

[18] J.-L. Binot, L. Debille, D. Sedlock, and B. Vandecapelle. Natural Language Interfaces: A New Philosophy. SunExpert Magazine, pages 67-73 January 1991.

[19] R.J. Bobrow. The RUS System. In Research in Natural Language Understanding, BBN Report 3878. Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1978.

[20] R.J. Bobrow, P. Resnik, and R.M. Weischedel. Multiple Underlying Systems: Translating User Requests into Programs to Produce Answers. In Proceedings of the 28th Annual Meeting of ACL, Pittsburgh, Pennsylvania, pages 227-234, 1990.

[21] E.F. Codd. A Relational Model for Large Shared Data Banks. Communications of the ACM, 13(6):377-387, 1970.

[22] E.F. Codd. Seven Steps to RENDEZVOUS with the Casual User. In J. Kimbie and K. Koffeman, editors, Data Base Management. North-Holland Publishers, 1974.

[23] A. Copestake and K. Sparck Jones. Natural Language Interfaces to Databases. The Knowledge Engineering Review, 5(4):225-249,1990.

[24] F. Damerau. Operating statistics for the transformational question answering system. American Journal of Computational Linguistics, 7:30-42,1981.

[25] F. Damerau. Problems and Some Solutions in Customization of Natural Language Front Ends. ACM Transactions on Office Information Systems, 3(2):165-184, April 1985.

[26] J.M. Ginsparg. A Robust Portable Natural Language Database Interface. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 25-30, 1983.

[27] B.J. Grosz. TEAM: A Transportable Natural-Language Interface System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 39-45, 1983.

[28] B.J. Grosz. D.E. Appelt, P.A. Martin, and F.C.N. Pereira. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. Artificial Intelligence, 32:173-243,1987.

[29] C.D. Hafner. Interaction of Knowledge Sources in a Portable Natural Language Interface. In Proceedings of the 22nd Annual Meeting of ACL, Stanford, California, pages 57-60, 1984.

[30] C.D Hafner and K. Godden. Portability of Syntax and Semantics in Datalog. ACM Transactions on Office Information Systems, 3(2):141-164, April 1985.

[31] L.R. Harris. User-oriented Data Base Query with the ROBOT Natural Language Query System. International Journal of Man-Machine Studies, 9:697-713, 1977.

[32] L.R. Harris. The ROBOT System: Natural Language Processing Applied to Data Base Query. In Proceedings of the ACM'78 Annual Conference, 1978.

[33] L.R. Harris. Experience with ROBOT in 12 Commercial Natural Language Data Base Query Applications. In Proceedings of the 6th International Joint Conference on Artificial Intelligence, Tokyo, Japan, pages 365-368, 1979.

[34] L.R. Harris. Experience with INTELLECT: Artificial Intelligence Technology Transfer. The AI Magazine, 5(2):43-50, 1984.

[35] G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a Natural Language Interface to Complex Data. ACM Transactions on Database Systems, 3(2):105-147, 1978.

[36] E.W. Hinrichs. Tense, Quantifiers, and Contexts. Computational Linguistics, 14(2):3-14, June 1988.

[37] T. Johnson. Natural Language Computing: The Commercial Applications. Ovum Ltd., London, 1985.

[38] J.L. Manferdelli. Natural Languages. Sun Technology, pages 122-129, Summer 1989.

[39] P. Martin, D. Appelt, and F. Pereira. Transportability and Generality in a Natural Language Interface System. In B.J. Grosz, K. Sparck Jones, and B.L. Webber, editors, Readings in Natural Language Processing, pages 585-593. Morgan Kaufmann Publishers, California, 1986.

[40] N. Ott. Aspects of the Automatic Generation of SQL Statements in a Natural Language Query Interface. Information Systems, 17(2):147-159, 1992.

[41] P. Resnik. Access to Multiple Underlying Systems in JANUS. BBN report 7142, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, September 1989.

[42] R.J.H. Scha. Philips Question Answering System PHILIQA1. In SIGART Newsletter, no.61. ACM, New York, February 1977.

[43] W. Sijtsma and O. Zweekhorst. Comparison and Review of Commercial Natural Language Interfaces. In F.M.G. de Jong and A. Nijholt, editors, Natural Language Interfaces, From Laboratory to Commercial and User Environments - Proceedings of the 5th Twente Workshop on Language Technology, Enschede, Twente University, NL, June 1993. Also MMC Preprint no. 13, Institute for Language Technology and Artificial Intelligence (ITK), Tilburg University, NL.

[44] M. Templeton and J. Burger. Problems in Natural Language Interface to DBMS with Examples from EUFID. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 3-16, 1983.

[45] B.H. Thompson and F.B. Thompson. Introducing ASK, A Simple Knowledgeable System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 17-24, 1983.

[46] B.H. Thompson and F.B. Thompson. ASK is Transportable in Half a Dozen Ways. ACM Transactions on Office Information Systems, 3(2):185-203, April 1985.

[47] D.L. Waltz. An English Language Question Answering System for a Large Relational Database. Communications of the ACM, 21(7):526-539, July 1978.

[48] D. Warren and F. Pereira. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. Computational Linguistics, 8(3-4):110-122, July-December 1982.

[49] R. Weischedel. A Hybrid Approach to Representation in the JANUS Natural Language Processor. In Proceedings of the 27th Annual Meeting of ACL, Vancouver, British Columbia, pages 193-202, 1989.

[50] W.A. Woods. Procedural Semantics for a Question-Answering Machine. In Proceedings of the Fall Joint Computer Conference, pages 457-471, New York, NY, 1968. AFIPS.

[51] W.A. Woods, R.M. Kaplan, and B.N. Webber. The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[52] Leo Obrst and Krishna Nanda Jha. NLP and Industry: 'Transfer and Reuse of Technologies', Philadelphia.

[53] Paulo Reis João Matias Nuno Mamede. Edite - A Natural Language Interface to Databases. A new dimension for an old approach. Portugal.

IDA: INTELLIGENT DATABASE AGENT
A KEYWORD BASED NATURAL LANGUAGE DATABASE INTERFACE

Muhammad Adeel Ather, Mubasher Feroze, Malik Sikandar Hayat Khiyal
Department of Computer Science, Faculty of Applied Sciences,
International Islamic University, Sector H-10,
Islamabad, Pakistan.

ABSTRACT

This research paper includes the study about Natural Language Processing based database interfacing. Normally, different databases are accessed by using their standard formal query languages like Structured Query Language (SQL) which is used only by specialized people, hence restricting the direct accessibility of the database by a lay man. Intelligent Database Agent provides a way to access the database using Natural Language i-e English by using the technique of keyword extraction, thus reducing the need of using complex natural language processing algorithms and linguistic specifics.

Keywords: Natural Language Processing, Database Agent, SQL, linguistics, Artificial Intelligence.

1. INTRODUCTION

By Natural Language we mean any language that is used for communication between human beings. It can be any language like Arabic, English, Urdu etc. These languages are formed through evolution process rather than invention [1]. On the other hand, languages such as COBOL, C++ and SQL were created in a relatively short period of time. Natural languages tend to have very large lexicons and highly complex grammars [2]. Computers do not understand natural languages. This produces a gap between man and machine. To overcome this gap we use Natural Language Processing (NLP).

In Natural Language Processing, user input in natural language is converted to the computer understandable language so that the required results can be obtained from the computer. NLP has many advantages. It can be used to perform a variety of useful tasks; ultimately it offers a more natural mode of communication between the system and user. In conjunction with speech synthesis and speech recognition NLP allows people with physical handicaps (such as paralysis) to use computers more effectively [2].

Natural language processing is being applied in more and more fields each day. The main applications of natural language processing are machine translation, database access, and text interpretation. The more successful applications of natural language processing have two common properties. First, they are focused on a particular domain instead of allowing discussion of any topic. Second, they focus on a particular aspect of comprehension instead of trying to understand the language completely [3].

1.1 Language Translation Engines

Natural Language Processing can be used to translate materials in one language to another or helping in filling communication gap between people knowing different languages.

Taum-Meteo is one successful application of machine translation. It translates weather reports from English to French. This system is successful because the language used in weather reports is quite regular and consistent (a restricted domain) [4].

SPANAM is another natural language translation system which translates between Spanish and English [5]. Although SPANAM is less accurate, it operates in a broader domain. The translation program runs on an IBM mainframe computer (4341 DOS/VSE), which is used for many other purposes as well. Texts are submitted and retrieved using the ordinary word-processing workstation (Wang OIS/140) as a remote job-entry terminal. The post-edited output is ready for delivery to the user with no further preparation required. The SPANAM program is written in PL/I. It is executed on the mainframe at speeds as high as 700 words per minute in clock time (172,800 words an hour in CPU time), and it runs with a size parameter of 215 K. Its source and target dictionaries (60,150 and 57,315 entries, respectively, as of May 1984) are on permanently mounted disks and occupy about 9 MB each.

1.2 Database Access

Database access via natural language allows the user to interact with the database without having to learn a formal language such as SQL, Access, C/C++, etc. There is a disadvantage to this. It can be confusing and frustrating for the user if their query fails, because it is outside of the system's competence. For example a natural language interface for a database may understand "south of the equator", but possibly not "in the southern hemisphere" even though both phrases have the same meaning [2].

As a natural language interface, question answering on relational databases allows users to access information stored in databases by requests in natural language, and generates as output natural language sentences, tables, and graphical representation [6]. The NL interface can be combined with other interfaces to databases: a formal query language interface directly using SQL, a form-based interface with fields to input query patterns, and a graphical interface using a keyboard and a mouse to access tables. The NL interface does not require the learning of formal query languages, and it easily represents negation and quantification, and provides discourse processing.

The use of natural language has both advantages and disadvantages. Including general NLP problems such as quantifier scoping, PP attachment, and elliptical questions, current Natural Language Interfaces to Databases (NLIDB) has many shortcomings: First, as a frequent complaint, it is difficult for users to understand which kinds of questions are actually allowed or not. Second, the user assumes that the system is intelligent; he or she thinks NLIDB has common sense, and can deduce facts [7]. Finally, users do not know whether a failure is caused by linguistic coverage or by conceptual mismatch. Nevertheless, natural language does not need training in any communication media or predefined access patterns.

NLIDB systems, one of the first applications of natural language processing, including "LUNAR" were developed from the 1970s [8]. In the 1980s, research focuses on intermediate representation and portability, and attempts to interface with various systems. CHAT-80 transforms English query into PROLOG representation [9], and ASK teaches users new words and concepts. LOQUI , a commercial system, adopts GPSG grammar. Meanwhile, Demers introduces a lexicalist approach for natural language to SQL translation, and as the CoBase project of UCLA, Meng and Chu combine information retrieval and a natural language interface. The major problems of the previous systems are as follows. First, they do not effectively reflect the vocabulary used in the description of database attributes into linguistic processing. Second, they require users to pose natural language queries at one time using a single sentence rather than give the flexibility by dialog-based query processing. The discordance between attribute vocabulary and linguistic processing vocabulary causes the portability problem of domain knowledge from knowledge acquisition bottleneck; the systems need extensive efforts by some experts who are highly experienced in linguistics as well as in the domain and the task.

1.3 Text Interpretation

If the computer could understand what the user actually wants, it could easily find articles or other information relating to the user's demand, thus producing more relevant information neglecting all unwanted material.

Information retrieval is a collection of methods which can be used to retrieve documents relevant to a query from a group of documents [10]. Every document has some sort of abstract associated with it. It may be the document title, a set of key words or even an n-dimensional vector. N-dimensional vectors are used in more modern information retrieval systems [11].

Text categorization is the sorting of natural language texts into fixed topic categories. Natural language processing has been successful in this area categorizing over 90% of news stories into their correct categories. Natural language processing systems tend to be faster and more consistent than their human counterparts.

Data extraction is the process of extracting useful data from a natural language text (which is often online), and placing this data in a structured database record or template. The SCISOR system developed by Jacobs and Rau in 1990 is an example of a data extraction system. A computer that can understand Natural Language could read large amounts of information available in the form of books, journals, newspapers thus increasing their knowledge bases to greater extent without any explicit effort.

The above mentioned areas are only broad categories but in fact the applications of Natural Language are as wide spread as the use of computer itself. A matured Natural Language technology could be used in one form or the other in virtually every computer application.

2. PROBLEMS

Research has been going on Natural Language Processing, but still we do not see fully accurate and matured products. A Natural Language Interface does not mean that the system would understand each and every word uttered, this is because of the vastness of the domain and the ambiguities of natural language itself. There are many Natural Language based interfaces to databases. But there are some problems with the existing interfaces. First of all, most of them are made for specific domains, i-e those can not be used for general purpose. For example, if a Natural Language Interface is developed for tourism industry then that will only cover the tourism domain and cannot be used for any other database.

Another problem is that Natural Language Database Interfaces usually require tedious and lengthy configuration phases before they can be used. Also the interfaces given by such systems are not simple and user friendly. CHAT-80 [9] is one of the best-known NLIDBS of the early eighties. CHAT-80 was implemented entirely in Prolog. It transformed English questions into Prolog expressions, which were evaluated against the Prolog database. The code of CHAT-80 was circulated widely, and formed the basis of several other experimental NLIDBs, for example MASQUE [12].

The user interface mechanisms being developed including command line interfaces (CLI) to graphical user interfaces (GUI). More recently, even 3-dimensional representations become quite sophisticated, and as all these are artificial. Users have to learn the meaning of the various commands and icons, which parameters they take, and the varied forms how to interact with them. In short, computer systems haven't yet been able to satisfactorily communicate with the user in the most human way: natural language.

Masque is a descendant of CHAT-80, a system created by Warren and Pereira in the early eighties [Warren & Pereira 82] [13]. CHAT-80 was designed as a general and portable natural language

interface to an arbitrary database. Its ability to cope with non-trivial English questions and its efficiency were impressive, but it proved to be in many ways idiosyncratic and hard to port between databases and knowledge domains. Since then, several efforts have made at the University of Edinburgh, to redesign the system, so that it is genuinely portable and efficient.

Moreover the Natural Language techniques used are quite complex converting the input into some meaning representation language (MRL) then afterwards converted according to the specific database attached after applying sophisticated algorithms and natural language processing techniques.

3. OBJECTIVE

The objective of the Intelligent Database Agent is to provide a simple technique for utilizing the users input in Natural Language to generate Structured Query Language for the underlying database. The system should serve as a generic interface to different databases and does not have lengthy and cumbersome configuration requirements. System would be simple for the users to understand and would also reduce the effects of over and undershooting problems.

This paper primarily focuses on Natural Language interfaces to Databases. A database interface does not require any formal language like SQL to get information from it. This type of system will use end user input in English to talk to the database thus making the database more accessible and useful for the users especially those who are not conversant with SQL. Non-technical persons, management staff as well as technical people will benefit from this as it will enable them to get information from the database without any detailed knowledge of the database tables and constraints. A very simple and efficient technique is used to convert the user input from natural language to formal query language i-e SQL.

4. OUR APPROACH

A relatively very simple approach is used in the construction of Intelligent Database Agent, which basically focuses on the effective utilization of user's entered information by the use of keyword extraction. With the use of the extracted keywords combined with the information gathered from the underlying database, the lexicon information and the language dictionary interface, Structured Query Language is generated which is used to talk to the connected database. Intelligent Database Agent is not targeted to any particular database and therefore it is kept in mind that it would be domain independent and any database could be connected and used with this interface agent without hassle.

At one time the Intelligent Database Agent would be connected to two databases. One of the databases, which are connected to the Intelligent Database Agent is the one whose interface it has to provide and the other one is IDA's internal database that uses to store the lexicon and other configuration information of the attached database.

As the mode of communication between the user and IDA would be English therefore an English dictionary interface is provided to better understand the keywords that are extracted from input. This also makes the system more open to new meanings of words that are being entered by the user. This dictionary interface is also used in the configuration phase of IDA.

The Intelligent Database Agent provides a simple menu based user interface. It has different forms for performing various operations like enter/edit information, configuration processes and input to SQL conversion screen.

4.1 One Time Configuration

The Intelligent Database Agent requires some configuration which is made easy with the help of processes run through simple form interfaces. The processes stores attached database's constraints information as well as database tables and columns synonym information in the IDA internal database. The user will have to enter any database semantics information in the form of synonyms to tell IDA system about what type of information is stored in the table. The more natural the table names, easier it would be to configure it.

4.2 Algorithm

Our simple and efficient algorithm uses some steps to identify and convert the user input in natural language to Standard Query Language (SQL). These are six simple steps to be followed. Following is the algorithm:

```
Start
        Step 1;
                Get user input
                Break into tokens
                If no tokens found then
                        Prompt user for empty input
                        Go to step 1
                Else
                Go to step 2
        Step 2;
                Match tokens with available synonyms to identify the statement type
                If no synonyms found then
                        Prompt user for illogical input
                        Exit
                Else
                        Go to step 3
        Step 3;
                Compare tokens with column and table synonyms
                Go to step 4
        Step 4;
                Check the column occurrence
                Go to step 5
        Step 5;
                Identify table relationship by using internally stored database constraints information.
                Go to step 6
        Step 6;
                Check for the where clause by synonym matching
End;
```

Our simplified approach requires following steps to successfully translate the natural language input into SQL. By using the following input in natural language we express working of our technique:

"Give employee name and their department name"

1st Step: Tokenization

The basic functionality of the system is like that first the user enters the input in the provided interface form. After that the tokenization process is performed on the English language input where it is broken to individual words by considering certain delimiters like space, commas etc. so we get the tokens "give", "employee", "name", "and", 'their", "department" and "name".

2nd Step: Token Synonyms

From these tokens the statement type is judged by simple synonym lookup. So the first token "give" tells that it is the select statement. The next step is a need to identify the information desired by user and whether the information is present in the database or not. To do that the client database tables and columns synonym information present in the IDA database is used with the English language dictionary interface. Synonyms of all the tokens are acquired by dictionary lookup and stored in the IDA's internal database.

3rd Step: Comparison with table and column names

The individual tokens and their synonyms are matched with the tables and columns information stored in the IDA database. With the help of these keywords, potential database columns and tables are identified that possibly contain the user's desired information. All the table synonyms are stored in the table IDATableSynonym and column synonyms are stored in the IDAColumnSynonyms. These identified potential column and table names provide the basis for the generation of SQL. The columns serve as column names in the query to be built. Full qualified names of the columns are taken here in order to remove any ambiguity regarding their tables. Tables that were found earlier by the keyword extraction engine and those of the columns are considered as table names for the query. Now comparison is made between all the keywords and their synonyms with the stored table and column synonyms, so we found the columns and tables of the external database:

Columns: ename, dname

Tables: emp, dept

4th Step: Column Occurrence

The statement is generated by considering whether the columns are present in single table or from multiple tables. If all the columns are present in single table then any other tables identified would be rejected. It is found that ename and dname are present in more than one tables so we have to see the relationship between the tables.

5th Step: Identifying tables relationship

In case where the tables are more than one, the IDA database is consulted for checking the relation between the tables. If there is parent child relation between the identified tables they are joined by equi-join on the column forming the relation between the tables. This is done by consulting the IDATableConstraints table of the internal database. Parent-child relationship is found on deptno.

6th step: Checking Where clause

The system then checks for a where clause in the user's entered query tokens through synonym matching. If there is a where clause present then "where clause" is generated using the column names and synonyms information. The where clause entered by the user are matched with the keywords

present in IDA database. If found then phrases are replaced by the keywords to make the where clause for the resulting SQL query.

4.3 Result

Finally, after successful completion of all the steps, SQL generation engine use all above information and build a standard SQL statement that is understandable by the underlying database. So the result we get by applying all of the above steps to our selected example is:

"select emp.ename, dept.dname from emp,dept where emp.deptno = dept.deptno;"

The query formed is then displayed on the user interface. If the system could not figure out some thing meaningful from the user input appropriate messages are displayed to make necessary changes.

5. SUMMARY

This paper describes the effort made towards developing a simple Natural Language based database interface agent that could generate Structured Query Language from the user's entered input by using keywords extraction technique. The solution is not targeted to a specific database and hence tried to make it a simple and generic solution where complexities could be reduced as far as possible.

In third world countries there is lack of resources and appropriate guidance for carrying out research in new and emerging technologies. We have tried to work in this area and put in effort to move in the direction of making simpler solutions.

REFERENCES

[1] Gerald Gazdar, Chris Mellish. Natural Language Processing in Prolog/Pop11/Lisp.
[2] Russell S.J. and Norvig, P. (1995; 1998, January) Artificial Intelligence: A Modern Approach. (1999, March 27).
[3] Communications Research Centre - The CHAT Natural Language System. (1999, March 23).
[4] Chevalier, Monique; Dansereau, Jules; and Poulin, 1978. TAUM-METEO: System Description. TAUM Group, University of Montreal, Montreal, Canada.
[5] SPANAM AND ENGSPAN: MACHINE TRANSLATION AT THE PAN AMERICAN HEALTH ORGANIZATION. Muriel Vasconcellos and Marjorie Le6n 1, Pan American Health Organization, 525 Twenty-third Street, N.W. Washington, D.C. 20037.
[6] A Natural Language Database Interface For SQL-Tutor by Seymour Knowles
[7] Ana Maria Popescu, Oren Etzioni, Henry Kautz Towards a Theory of Natural Language Interfaces to Databases.
[8] W.A. Woods, R.M. Kaplan, and B.N. Webber. The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.
[9] D. Warren and F. Pereira. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. Computational Linguistics, 8(3-4):110-122, July-December 1982.
[10] Susan Feldman. Natural Language Processing in Information Retrieval.
[11] Asanee Kawtrakul. An Overview of a Role of Natural Language Processing in An Intelligent Information Retrieval System.

[12] P. Auxerre. MASQUE Modular Answering System for Queries in English – Programmer's Manual. Technical Report AIAI/SR/11, Artificial Intelligence Applications Institute, University of Edinburgh. March 1986.
[13] P. Auxerre and R. Inder. MASQUE Modular Answering System for Queries in English – User's Manual. Technical Report AIAI/SR/10, Artificial Intelligence Applications Institute, University of Edinburgh, June 1986.