PUNJABI LANGUAGE INTERFACE TO DATABASE

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Software Engineering

By:

Amandeep Kaur

(800831027)

Under the supervision of:

Mr. Parteek Bhatia

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2010

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Punjabi Language Interface to Database"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Parteek Bhatia* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Amandeep Kaur
(Amandeep Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mr. Prateek Bhatia)
Assistant Professor
Computer Science and Engineering Department,
Thapar University, Patiala.

Countersigned by

(RAJESH BHATIA) 17/02/10
Head
Computer Science and Engineering, Department,
Thapar University,
Patiala.

(R.K.SHARMA) 1.2.10
Dean (Academic Affairs)
Thapar University,
Patiala.

Acknowledgement

I wish to express my deep gratitude to Mr. Parteek Bhatia, Assistant Professor and Dr. Rajesh Bhatia Assistant Professor & Head, Computer Science & Engineering Department for providing their uncanny guidance and support throughout the preparation of the thesis report.

I would also like to thank all the staff members and all my friends especially Rupinderdeep Kaur, Gagandeep Singh, Jarrar Alam Rana and Divya Pandove who were always there at the hour of need and provided all the help and support, which I required for the completion of the thesis.

Last but not the least; I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Amandeep Kaur

(Amandeep Kaur)

Abstract

Database management systems (DBMS) have been widely used for storing and retrieving data. However, databases are often hard to use since their interface is quite rigid in cooperating with users. End user is required to issue SQL query to retrieve information from the database. General public can issue query on the database in their natural language The purpose of natural language interfaces is to allow users to compose question in natural language and receive responses. Here we have given introduction to natural language processing (NLP) and natural language interfaces to database (NLIDB) as an application of NLP. Advantages and disadvantages of natural language interface to database are discussed as compared to other GUI systems.

In Punjab, natural language of people is Punjabi. Also Large number of e-governance applications use database. So, to use such database applications with ease, people who are more comfortable with Punjabi language, require these applications to accept a simple sentence in Punjabi, and process it to generate a SQL query, which is further executed on the database to produce the results. Therefore, any interface in Punjabi language will be an asset to these people. Here we present the design and implementation of natural language (Punjabi) interface to (agriculture) database. We have tested the system with different set of queries. This system provides an interface to the user to put query in Punjabi language, process that query, and provide output to the user in Punjabi language.

Contents

List of Figures

LIST OF TABLES

Chapter 1

Introduction

Internet is the largest data provider in today's date and it caters to users of all kinds. The vastness of data makes it mandatory that data is saved in an organized manner so that it is easy to search, retrieve and maintain. For this purpose the most logical and commonly used storage method is by the use of databases. But to efficiently use or maintain any database the knowledge of languages such as SQL becomes essential. This would limit the use of data to only those users who have the knowledge of these languages. Hence, an easy to use user interface comes into picture which would facilitate diverse users to access data. There is a need to design and develop an interface in the local language so that user can easily use that system without the knowledge of English language. So, in order to address these issues we have developed Natural language Interface to Database.

1.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. The foundations of NLP lie in a number of disciplines, computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, *etc*. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, natural language interfaces to database, multilingual and Cross Language Information Retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on [1].

1.2 Natural Language Interface to Database (NLIDB)

One may find interacting with a foreign person with no knowledge of English intricate and frustrating. Thus, a translator will have to come into the picture to allow one to communicate with the foreigner. Companies have related this problem to extracting data from a DataBase Management System (DBMS) such as MS Access, Oracle and others. A person with no knowledge of Structured Query Language (SQL) may find himself or herself handicapped in corresponding with the database. Therefore, companies like Elfsoft (English Language Frontend Software which has developed SQL Tutor) have analysed the abilities of Natural Language Processing to develop products for people to interact with the database in simple English. This enables a user to simply enter queries in English to the Natural Language Database Interface. This kind of application is known as a Natural Language Interface to a DataBase (NLIDB).

The system works by utilizing the use of syntactic knowledge and the knowledge it has been provided about the relevant database. Hence, it is able to implement the natural language input to the structure, scope and contents of the database. The program translates the whole query into the standard query language to extract the relevant information from the database. Thus, these products have created a revolution in extracting information from databases. They have discarded the fuss of learning SQL and time is also saved in learning this query language.

1.3 Sub Components of NLIDB

Computing scientists have divided the problem of natural language access to a database into two sub-components:

- Linguistic component
- Database component

1.3.1 Linguistic Component

It is responsible for translating natural language input into a formal query and generating a natural language response based on the results from the database search.

1.3.2 Database Component

It performs traditional Database Management functions. A lexicon is a table that is used to map the words of the natural input onto the formal objects (relation names, attribute names, *etc*.) of the database. Both parser and semantic interpreter make use of the lexicon.

A natural language generator takes the formal response as its input, and inspects the parse tree in order to generate adequate natural language response.

Natural language database systems make use of syntactic knowledge and knowledge about the actual database in order to properly relate natural language input to the structure and contents of that database. Syntactic knowledge usually resides in the linguistic component of the system, in particular in the syntax analyzer whereas knowledge about the actual database resides to some extent in the semantic data model used.

Questions entered in natural language translated into a statement in a formal query language. Once the statement unambiguously formed, the query is processed by the database management system in order to produce the required data. These data then passed back to the natural language component where generation routines produce a surface language version of the response.

1.4 NLIDB's Comparison with Other GUI

The use of NLIDB, however, is much less widespread than it was once predicted, mainly because of the development of alternative graphic and form-based database interfaces. But these alternative interfaces are less natural to interact with and queries that involve quantification, or that require multiple database tables to be consulted are very difficult to

formulate with graphic or form-based interfaces, whereas they can be expressed easily in natural language [2]

For example, queries that involve quantification like "*list the ten crops with highest prices*" or queries that involve multiple selections like "*list name, age, address and contact of the farmer*" or queries that involve multiple database tables like "*list the address of the farmers who got bonus greater than 10000 rupees for the crop of wheat*" are difficult to express in graphic or form based database interfaces.

1.5 Punjabi Language Interface to Database

Large number of e-governance applications use database. So, to use such database applications with ease, people who are more comfortable with Punjabi language, require these applications to accept a simple sentence in Punjabi, and process it to generate a SQL query, which is further executed on the database to produce the results. Therefore, any interface in Punjabi language will be an asset to these people. With the help of Punjabi interface, they will be able to make use of database applications very well. Punjabi Language is used in both parts of Punjab in India and Pakistan. In Eastern Punjab (India) Punjabi is written in Gurmukhi script. Punjabi is the mother tongue of more than 110 million people of Pakistan (66 million), India (44 million) and many millions in America, Canada and Europe. Punjabi is world's 12^{th} most widely spoken language and its free word language *i.e.* a single sentence can be written in many forms, with almost same meaning.

Chapter 2

Literature Survey

This chapter covers introduction about Natural Language Processing (NLP) and its applications in the first section. Second section covers information about Natural language Interface to Database (NLIDB). Third and fourth section includes the advantages and disadvantages of NLIDB system respectively. Fifth section covers some already developed NLIDB systems. Sixth section includes various approaches used for development of NLIDB systems. Seventh section covers techniques for developing NLIDB. Eighth section discuses about the most commonly used architecture of NLIDB system. Last section covers portability issues of NLIDB system.

Language is the primary means of communication used by humans. It is the tool we use to express the greater part of our ideas and emotions. Learning new concepts and expressing ideas through them is so natural that we hardly realize how we process natural language.

Natural Language Processing (NLP) is concerned with the development of computational models of aspects of human language processing. There are two main reasons of such development:

- To develop automated tools for language processing.
- To gain a better understanding of human communication.

Building computational models with human language processing abilities requires knowledge of how humans acquire store and process language. It also requires knowledge of the world and of language.

2.1 NLP Applications

The applications utilizing NLP include the following:

- **Machine Translation**

  This refers to automatic translation of text from one human language to another. In order to carry out this translation, it is necessary to have an understanding of words and phrases, grammars of two languages involved, semantics of the language, and world knowledge.

- **Speech Recognition**

  This is a process of mapping acoustic speech signals to a set of words. The difficulties arise due to wide variations in the pronunciations of words.

- **Speech Synthesis**

  This refers to automatic production of speech (utterance of natural language sentences). Such systems can read out your mails on telephone. Or even read out a storybook for you. In order to generate utterances, text has to be processed.

- **Natural Language Interfaces to Databases**

  It allows querying a database using natural language sentences. It is a system that allows the user to access information stored in a database by typing requests expressed in some natural language.

- **Information Retrieval (IR)**

  This is concerned with identifying documents relevant to user's query. Indexing, word sense disambiguation, query modification and knowledge base have also been used in IR system to enhance performance.

- **Information Extraction**

  This captures and outputs factual information contained within a document. Similar to IR system, it responds to user's information need. The information need is not expressed as a keyword query instead it is specified as pre-defined database schemas or templates.

- **Question Answering**

  In this, given a question and a set of documents, a question answering system attempts to find the precise answer. Or at least the precise portion of text in which the answer appears [3].

2.2 Natural Language Interface to Database

People via computer all around the world, access, accumulate and manipulate huge amount of data every second of the day. These huge amounts of data are located in private personal computers or remote locations. Mostly, data is stored in some kind of repository system such as database. Data in database is usually managed by DBMS and access to database is facilitated through a special interaction language called SQL or some version of it. To override the complexity of SQL for non-professionals, many researches have turned out to use Natural Language (NL). The idea of using NL has prompted the development of new type of processing method called Natural Language Interface to Database [4]. Here we are focusing on natural language interface to database, an application of natural language processing. A Natural Language Interface to a Database (NLIDB) is a system that allows the user to access information stored in a database by typing requests expressed in some natural language [5].

2.3 Advantages of NLIDB system

Advantages of Natural Language Interface to Database are as follows:

- **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 like SQL are difficult to learn and master, at least by non-computer-specialists.

- **No Need for Training**

  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.

- **Better for Some Questions**

  It has been argued that there are some kind of questions (*e.g*. 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.

- **Easy to Use for Multiple Database Tables**

  Queries that involve multiple database tables like "*list the address of the farmers who got bonus greater than 10000 rupees for the crop of wheat*", are difficult to form in graphical user interface as compared to natural language interface [8].

2.4 Disadvantages of NLIDB System

Disadvantages of natural language interface to database system are as follows:

- **Linguistic Coverage Not Obvious**

  A frequent complaint against NLIDBs is that the system's linguistic capabilities are not obvious to the user. 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.

  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.

For example, database doesn't contain information about profit of farmers and user inputs the query "*which farmer gets maximum profit*". This query is out of the linguistic coverage of the system.

- **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 (e.g. 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, user asks a query "*list the names of farmers who are 35 years old*" and the database has information about the age of the farmer but 'age' word is not there in query. So this query is not out of linguistic coverage but conceptually it is not right, because system does not understand what 35 is representing *i.e.* whether it is for age or for address.

- **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.

  For example, when user asks a query "*list the names of farmers who are 35 years old*", he/she is not specifying the word 'age', assuming that system will understand it automatically. But system is not so intelligent.

- **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. In natural language interface user has to type full sentence with all the connecters (articles, prepositions, etc) but in graphical or form based interfaces it is not required [8].

2.5 Some Already Developed NLIDB Systems

Work for developing Natural Language Interface to Database has started in early seventies. Since then many systems have been developed. Early systems have many flaws then some systems were developed to overcome these flaws. Some of the developed NLIDB systems are discussed below.

- **LUNAR System**

  This system comes in early seventies (1973). The system LUNAR [6] is a system that answers questions about samples of rocks brought back from the moon. The

meaning of system's name is that is in relation to the moon. The system was informally introduced in 1971. To accomplish its function the LUNAR system uses two databases; one for the chemical analysis and the other for literature references. The LUNAR system uses an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics. According to words [7], the LUNAR system performance was quite impressive; it managed to handle 78% of requests without any errors and this ratio rose to 90% when dictionary errors were corrected. But these figures may be misleading because the system was not subject to intensive use due to the limitation of its linguistic capabilities.

- **LADDER**

  This system was developed in 1978. It was designed as a natural language interface to a database of information about US Navy ships. According to Hendrix [8], the LADDER system uses semantic grammar to parse questions to query a distributed database. 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 [5]. The system uses semantic grammars technique that interleaves syntactic and semantic processing. The question answering is done via parsing the input and mapping the parse tree to a database query. The system LADDER was implemented in LISP. At the time of creation of the LADDER system, it was able to process a database that is equivalent to a relational database with 14 tables and 100 attributes.

- **RENDEZVOUS System**

  This system appeared in late seventies. In this, users could access databases via relatively unrestricted natural language. In this Codd's system, special emphasis is placed on query paraphrasing and in engaging users in clarification dialogs when there is difficulty in parsing user input [9].

- **PLANES**

  This was developed in late seventies for (Programmed LANguage-based Enquiry System) at the University of Illinois Coordinated Science Laboratory. PLANES include an English language front end with the ability to understand and explicitly answer user requests. It carries out clarifying dialogues with the user as well as answer vague or poorly defined questions. This work is being carried out using database based upon information of the U.S. Navy 3-M (Maintenance and Material Management), it is a database of aircraft maintenance and flight data, although the ideas can be directly applied to other non-hierarchic record-based databases [10].

- **PHILIQA**

  This was developed in 1977 and was known as Philips Question Answering System [11], uses a syntactic parser which runs as a separate pass from the semantic understanding passes. This system is mainly involved with problems of semantics and has three separate layers of semantic understanding. The layers are called "English Formal *Language",* "World Model Language", and "Data Base Language" and appear to correspond roughly to the "external", "conceptual", and "internal" views of data.

- **CHAT-80**

  This 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. The database of CHAT-80 consists of facts (*i.e*. oceans, major seas, major rivers and major cities) about 150 of the countries world and a small set of English language vocabulary that are enough for querying the database [5].

- **TEAM**

  It was developed in 1987. A large part of the research of that time was devoted to portability issues. TEAM was designed to be easily configurable by database administrators with no knowledge of NLIDBs [12, 13].

- **ASK**

  This system developed in 1983, allowed end-users to teach the system new words and concepts at any point during the interaction. ASK was actually a complete information management system, providing its own built-in database and the ability to interact with multiple external databases, electronic mail programs and other computer applications. All the applications connected to ASK were accessible to the end-user through natural language requests. The user stated his/her requests in English and Ask transparently generated suitable requests to the appropriate underlying systems [5].

- **JANUS**

  It 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 [14].

- **EUFID**

  The EUFID system consists of three major modules, not counting the DBMS. First is analyzer module, second is mapper module and third is translator module. [15]

- **DATALOG**

  It is an English database query system based on Cascaded ATN grammar. By providing separate representation schemes for linguistic knowledge, general

world knowledge, and application domain knowledge, DATALOG achieves a high degree of portability and extendibility [16].

Systems that also appeared in mid-eighties were LDC [17], TQA [18], TELI [19] and many others.

2.5.1 Commercially Available Systems

Some of the commercially available NLDBs are as follows:

- **IBM'S LANGUAGE ACCESS:** This system stopped being commercially available in October 1992.
- **SQL Tutor:** It was developed in 1998. SQL can be very difficult for beginner users to understand. The SQL-Tutor program tutors students by assisting the students through a number of database questions from four different databases. A student model is kept for each student based on query constraints (each constraint represents a part of the query that is necessary to answer the question). Each time a particular query constraint is used, SQL-Tutor records whether it was used successfully or unsuccessfully. In this way a model of a student's strengths and weaknesses is generated and SQL-Tutor can select questions which re-enforce problem areas or introduce new query concepts [20].
- **INTELLECT:** It was from Trinzic (formed by the merger of AICorp and Aion). This system is based on experience from ROBOT.
- **BBN'S PARLANCE:** It was built on experience from the development of the Rus and Irus systems.
- **ENGLISH WIZARD:** It was developed by Linguistic Technology Corporation. The company was founded by the author of AICorp's original Intellect [5].

2.6 Various Approaches Used for Development of NLIDB Systems

Natural language is the topic of interest from computational viewpoint due to the implicit ambiguity that language possesses. Several researchers applied different techniques to deal with language.

2.6.1 Symbolic Approach (Rule Based Approach)

Natural Language Processing appears to be a strongly symbolic activity. Words are symbols that stand for objects and concepts in real worlds, and they are put together into sentences that obey well specified grammar rules. Hence, for several decades Natural Language Processing research has been dominated by the symbolic approach. Knowledge about language is explicitly encoded in rules or other forms of representation. Language is analyzed at various levels to obtain information. On this obtained information certain rules are applied to achieve linguistic functionality. As Human Language capabilities include rule-base reasoning, it is supported well by symbolic processing. In symbolic processing rules are formed for every level of linguistic analysis. It tries to capture the meaning of the language based on these rules.

2.6.2 Empirical Approach (Corpus Based Approach)

Empirical approaches are based on statistical analysis as well as other data driven analysis, of raw data which is in the form of text corpora. A corpus is collections of machine readable text. The approach has been around since NLP began in the early 1950s. Only in the last 10 years or so empirical NLP has emerged as a major alternative to rationalist rule-based Natural Language Processing. Corpora are primarily used as a source of information about language and a number of techniques have emerged to enable the analysis of corpus data. Syntactic analysis can be achieved on the basis of statistical probabilities estimated from a training corpus. Lexical ambiguities can be resolved by considering the likelihood of one or another interpretation on the basis of context. Recent research in computational linguistics indicates that empirical or corpus –based methods are currently the most promising approach for developing robust, efficient Natural Language Processing (NLP) systems. These methods automate the acquisition of much of the complex knowledge required for NLP by training on suitably annotated natural language corpora, *e.g.* tree-banks of parsed sentences.

Most of the empirical NLP methods employ statistical techniques such as n-Gram models, Hidden Markov Models (HMMs), and Probabilistic Context Free Grammars

(PCFGs). Given the successes of empirical NLP methods, researchers have recently begun to apply learning methods to the construction of information extraction systems. Several different symbolic and statistical methods have been employed, but most of them are used to generate one part of a larger information extraction system. Experimented n-Gram based language modeling and claimed to develop language independent approach to IR and Natural Language Processing.

2.6.3 Connectionist Approach (Using Neural Network)

Since human language capabilities are based on neural network in the brain, Artificial Neural Networks (also called as connectionist network) provides an essential starting point for modeling language processing. In the recent years, the field of connectionist processing has seen a remarkable development. The sub-symbolic neural network approach holds a lot of promise for modeling the cognitive foundations of *Natural Language Interface using Shallow Parsing* language processing. Instead of symbols, the approach is based on distributed representations that correspond to statistical regularities in language. There has also been significant research applying neural-network methods to language processing. However, there has been relatively little recent language research using sub-symbolic learning, although some recent systems have successfully employed decision trees transformation rules, and other symbolic methods. SHRUTI system is a neurally inspired system for event modeling and temporal processing at a connectionist level [5].

2.7 Techniques Used for Developing Natural Language Interface to Database (NLIDB)

The techniques followed by various developed NLIDB's systems are as follows

- **Pattern-Matching Systems**

  Some of the early NLIDBs relied on pattern-matching techniques to answer the user's questions. The main advantage of the pattern-matching approach is its simplicity: no elaborate parsing and interpretation modules are needed, and the

systems are easy to implement. Also, pattern-matching systems often manage to come up with some reasonable answer, even if the input is out of the range of sentences the patterns were designed to handle. One of the best natural language processing system that role in this style is ELIZA. ELIZA functions by processing users, by these responses to the scripts. It typically says differently and rephrased the statements of the users as questions and replies the answers of those questions to the 'patient. ELIZA was programmed by Mr. Joseph Weizenbaum in nearly from 1964 to 1966 .

- **Syntax-Based Systems**

  In syntax-based systems the user's question is parsed (*i.e.* analyzed syntactically) and the resulting parse tree is directly mapped to an expression in some database query language. Syntax-based systems use a grammar that describes the possible syntactic structures of the user's questions. Syntax-based NLIDBs usually interface to application-specific database systems that provide database query languages carefully designed to facilitate the mapping from the parse tree to the database query. It is usually difficult to devise mapping rules that will transform directly the parse tree into some expression in a real-life database query language (*e.g.* SQL).

- **Semantic Grammar Systems**

  In semantic grammar systems, the question-answering is still done by parsing the input and mapping the parse tree to a database query. The difference, in this case, is that the grammar's categories do not necessarily correspond to syntactic concepts. Semantic information about the knowledge domain is hard-wired into the semantic grammar that's why systems based on this approach are very difficult to port to other knowledge domains a new semantic grammar has to be written whenever the NLIDB is configured for a new knowledge domain. Semantic grammar categories are usually chosen to enforce semantic constraints [8]. Much of the systems developed till now like LUNAR, LADDER, use this approach of semantic grammar.

2.8 Most Commonly Used Architecture of NLIDB System

Architecture used by some of the NLIDB systems is shown in the Figure 2.1 which uses both the semantic grammar system architecture and syntactic grammar system architecture. Various modules of this architecture are discussed as follows:

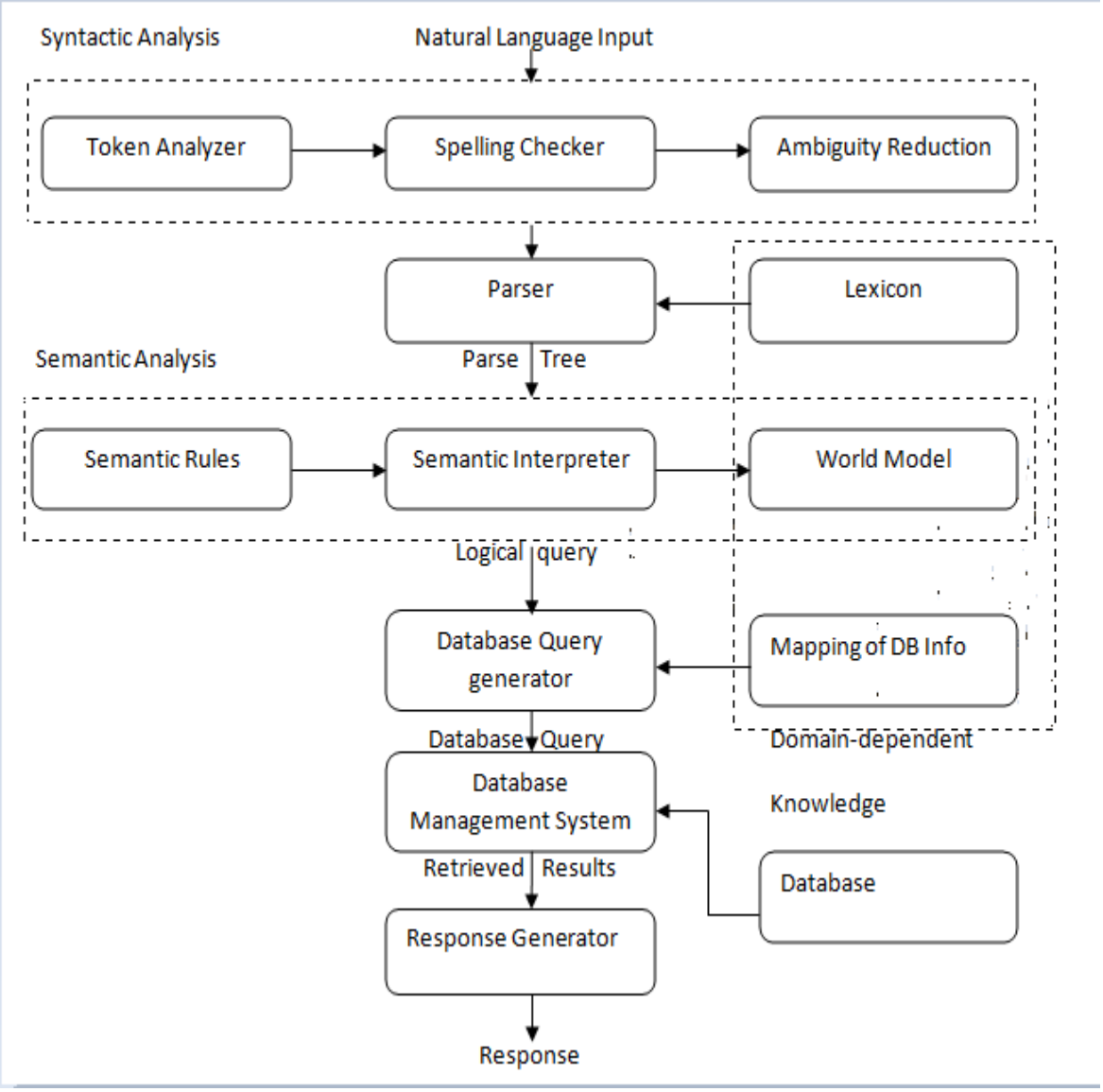

Figure 2.1: Commonly Used Architecture of Natural Language Interface to Database [5]

- **Syntactic Analysis**

  The word syntax means grammatical arrangements of words in a sentence and their relationship with each other. The objective of syntactic analysis is to find the syntactic structure of the sentence. This step (as referred in figure in 2.1) divides the sentence into simpler elements that are called tokens. **Token Analyzing** function is used to split the input string into a sequence of primitive units called tokens that is treated as a single logical unit.

  For example, we have query *"list names of all the farmers",* Token Analyzing function will tokenize this query and form tokens as 'list', 'names', 'of', 'all', 'the', 'farmers'. Then **Spelling Checker** function makes sure that each token is in the system's dictionary (lexicon) and if this is not the case then the spelling correction is performed or new words are added to the systems' vocabulary. **Ambiguity reduction** function reduces the ambiguity in a sentence and simplifies the task of the parser.

  For example, we have query *"list city of kamal"*, now 'kamal' can be name of farmer or name of crop. So, ambiguity reduction function will reduce this ambiguity by finding the 'city' token and clears that 'kamal' relates to the name of farmer.

- **Parse Tree**

  Output of syntactic analysis is a parse tree (as referred in figure in 2.1). It represents the syntactic structure of a sentence according to some formal grammar. A parse tree is composed of nodes and branches; each node is either a root node, a branch node, or a leaf node. In a parse tree, an interior node is a phrase and is called a non-terminal of the grammar, while a leaf node is a word and is called a terminal of the grammar.

  For example, we have a query *"List me all employees having salary greater than or equal to 300 dinars"* and the parse tree for this query is in figure 2.2.

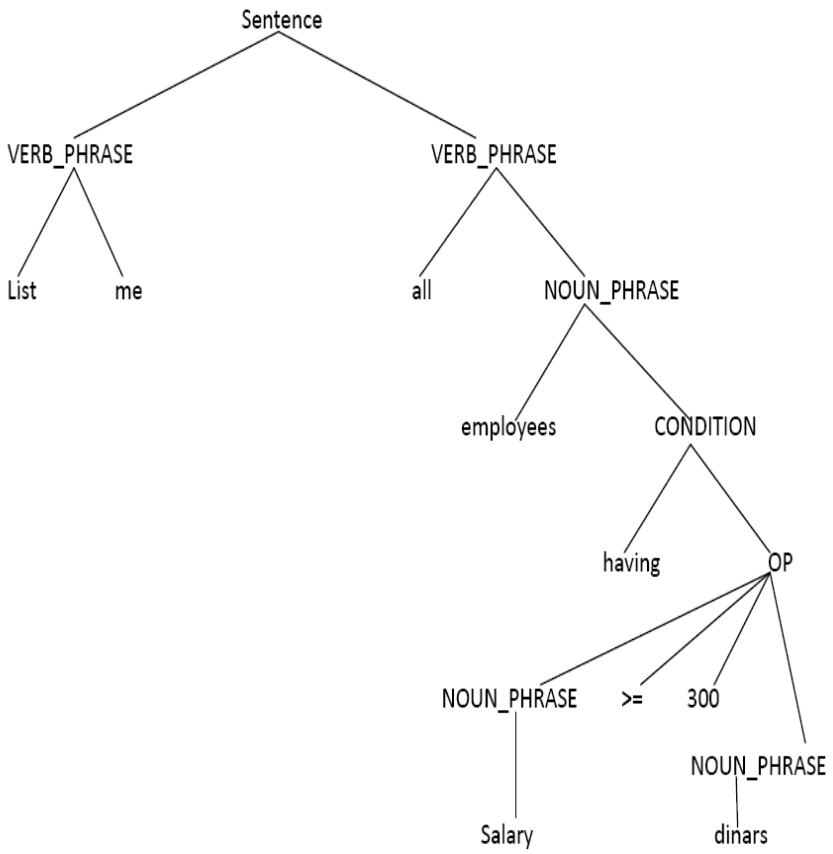

Figure 2.2: Parse Tree

- **Semantic Analysis**

  Semantics is associated with the meaning of language. Semantic analysis (as referred in figure in 2.1) is concerned with creating representations for meaning of linguistic inputs. In this focus is on logical words, and no attention is paid to non-logical words. It deals with how the meaning of sentence is determined from the meanings of its parts. And thus, it generates a logical query which is fed as input to the database query generator.

- **Database Query Generator**

  The task of the Database Query Generator (as referred in figure in 2.1) is to map the elements of the logical query to the actual elements of the used databases. The query generator uses four routines, each of which manipulates only one specific

part of the query. The overall database query statement is constructed from the concatenation of the output of the four routines. The first routine selects the part of the natural language query that corresponds to the appropriate DML command with the attributes' names (*i.e*. SELECT * clause). The second routine selects the part of the query that would mapped to a table's name or a group of tables' names to construct the FROM clause. The third routine selects the part of the query that would be mapped to the WHERE clause (condition). The fourth routine selects the part of the natural language query that corresponds to the order of displaying the result (ORDER BY clause with the name of the column).

- **Database Management System**

  The purpose of this system (as referred in figure in 2.1) is to get the required results from the used database. In order to achieve this, the generated database query would be tested to verify correctness before applied to the used database and then represent the result to the user. It executes that query on the database and produces the results required by the user. Response generator directs that results to user [5].

2.9 Portability Issues of NLIDB System

Early NLIDBs were each designed for a particular database application. Lunar [6], for example, was designed to support English questions, referring to a database of a particular structure, holding data about moon rocks. Such application-tailored NLIDBs were very difficult to port to different applications. In recent years a large part of the research in NLIDBs has been devoted to portability, *i.e*. to the design of NLIDBs that can be used in different knowledge-domains, with different underlying DBMSs, or even with different natural languages. This section discusses the different kinds of NLIDB portability, and presents some methods that have been used to approach the goal of portability.

2.9.1 Knowledge-Domain Portability

Existing NLIDBs can only cope with questions referring to a particular *knowledge domain* (*e.g*. questions about train services, questions about bank accounts). A NLIDB provides knowledge-domain portability, if it can be configured for use in a wide variety of knowledge domains.

Typically, whenever a NLIDB is being reconfigured for a new knowledge-domain, someone has to "teach" the system, the words and concepts used in the new domain, and how these concepts relate to the information stored in the database. Different NLIDBs assume that the knowledge-domain configuration will be carried out by people possessing different skills:

- **Programmer**

  In some systems a part (usually small and well defined) of the NLIDB's code has to be rewritten during the knowledge domain configuration. It is, therefore, assumed that the person that will carry out the configuration will be a programmer, preferably familiar with the NLIDB's code.

- **Knowledge Engineer**

  Other NLIDBs provide tools that can be used to configure the system for a new knowledge domain, without the need to do any programming. It is still assumed, however, that the tools will be used by a knowledge engineer, or at least a person familiar with basic knowledge representation techniques, databases, and linguistic concepts.

  Some systems have built-in dictionaries, listing the most common words. The dictionaries can be customized by the person configuring the system to include domain-specific terminology.

- **DataBase Administrator (DBA)**

  According to the designers of Team [12, 13], a realistic scenario is to assume that the NLIDB will be bought by a company already using a database to which the

NLIDB is to be linked. In such situations, the responsibility of installing and configuring the NLIDB would probably be assigned to the local database administrator. Therefore, the configuration tools should be designed, so that they can be used by persons with a good understanding of database concepts, familiar with the particular database to which the NLIDB is to be linked, but with no knowledge of AI, logics, or linguistics.

2.9.2 DBMS Portability

A NLIDB provides DBMS portability, if it can be easily modified to be used with different underlying Database Management Systems (DBMSs).

In the case of NLIDBs that generate queries, supported by number of DBMS, it may be possible to port the NLIDB from one DBMS to another, with only minor modifications.

If the original database query language is not supported by the new DBMS, then a NLIDB can be ported to the new DBMS by rewriting the module that translates the intermediate logic queries to database queries. In contrast, NLIDBs that do not use intermediate representation languages may require extensive modifications

In the case of NLIDBs that clearly separate the linguistic front-end from the rest of the system, it should be possible to port a NLIDB configured for a particular knowledge domain to a different underlying DBMS containing data about the same knowledge-domain, without modifying the modules of the linguistic front-end.

2.9.3 Natural Language Portability

The largest part of the research that has been carried out till now assumes that the natural language requests will be written in English. Modifying an existing NLIDB to be used with a different natural language can be a difficult task. Modifying the NLIDB for a different natural language typically requires rewriting/modifying the syntax rules, the semantic rules, and the lexicon.

2.9.4 Hardware and Programming Language Portability

Up to the early eighties, Some NLIDBs could be used only on expensive research computing systems, supporting "exotic" programming languages (*e.g*. LISP machines), thus making NLIDBs almost impossible to use in real-life applications. But in recent years, NLIDB has become easy to port across computing platforms and to use in real life applications [5].

Chapter 3

Problem Statement

Natural Language Interface to database system provides an interface to the user which helps him/her to query the database in his/her natural language. As the query languages like SQL are very difficult to use for the common people and they find it very hard to learn. So, to make database applications easy to use for these people who don't know query languages, Natural Language Interface for various databases has been developed.

3.1 Problem Statement

The main problem is to design a Natural Language Interface of Database for Punjabi language using Agriculture database. The query is asked in the Punjabi language for retrieving the relevant information from the database. The format of the queries asked by the user must be simple, not complex. Hence, there is need of developing a Punjabi language interface for agriculture database to query the database in Punjabi language.

3.2 Objectives

Objectives of the system are:

- Extract the table information from input Punjabi sentence with the use of table mapping.
- Extract the column information from input Punjabi sentence with the use of column mapping.
- Extract the condition information from input Punjabi sentence with the use of condition mapping.
- Generate SQL query.
- Execute the SQL query and retrieve output in Punjabi.

Chapter 4

Design and Implementation of Punjabi Language Interface to Agriculture Database

As Punjab is agriculture rich state and people have to deal a lot with crops marketing. So, in order to make crop marketing easy for the farmers, we have used an agriculture database and the area of discussion is Punjabi language interface for this database. Here we present the Design and implementation of the Punjabi language interface to agriculture database.

- **Agriculture Database**

  There are three tables in this database: **FARMER** table, **CROP** table, **SALE** table, which contains actual information, the user wants to retrieve from the database. **FARMER** table contains the personal information about the farmer *i.e.* farmer name, address, age, contact. **CROP** table contains information about the crops *i.e.* crop name, season, selling price for that crop. **SALE** table contains sales information about crops *i.e.* selling price, labour amount, and bonus amount.

4.1 Architecture of the System

The architecture of the Punjabi language interface to agriculture database system is shown in the figure 4.1.

Architecture of the system is divided into three phases:

- Tokenizing the query
- Mapping and formation of SQL query
- Execution of query.

Here is the description of these phases.

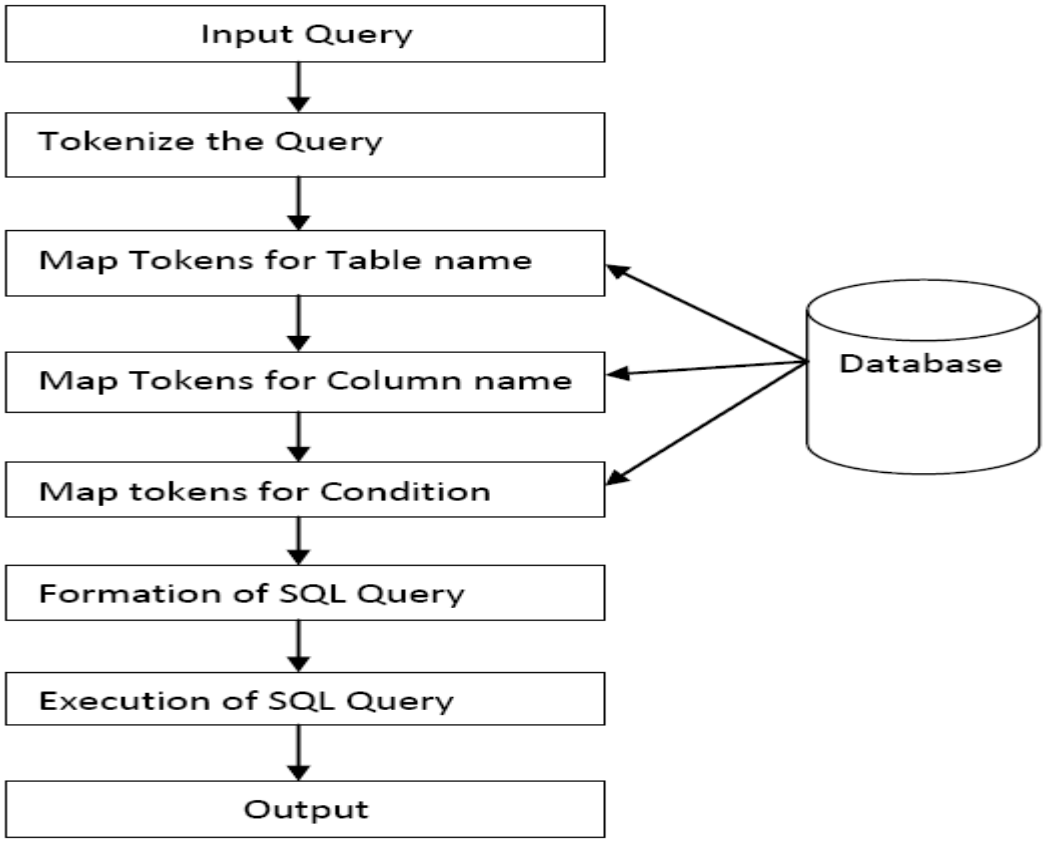

Figure 4.1: Architecture of the Punjabi Interface to Agriculture Database System

- **Tokenizing**

  User is providing input in the form of simple sentence to the system in Punjabi language. System tokenizes that sentence and produce tokens that are to be searched in the database for mapping.

- **Mapping and Formation of SQL Query**

  This phase is based on database and is explained in the flowchart given in figure 4.2. The database used for the system is divided into two parts: Agriculture database and Administrator's database. But for mapping we are only using administrator's database.

  - **Administrator's Database:** it consists of four tables: **TABLE_HANDLING** table, **COLUMN_HANDLING** table, **CONDITION** table, **CONDITIONALWORDS** table, that are used for

mapping of Punjabi query to SQL query. **TABLE_HANDLING** table and **COLUMN_HANDLING** table contains data required for mapping of table name and column name, found in Punjabi query, to their corresponding English words. **TABLE_HANDLING** table contains mapping for all names of available tables and **COLUMN_HANDLING** table contains mapping for all names of columns of all available tables. **CONDITIONALWORDS** contain mapping for conditional symbols *i.e.* less than (<) and greater than (>). **CONDITION** table contains mapping for "where" keyword of SQL query. All the words in Punjabi language that correspond to where in input query are added in this table [5].

Tables 4.1, 4.2, 4.3, 4.4 describe the mapping used in the system.

Table 4.1: TABLE_HANDLING Table

Token word	**Mapped word**
ਕਿਸਾਨਾਂ | ਕਿਸਾਨ	Farmer
ਫ਼ਸਲ | ਫ਼ਸਲਾਂ	Crop
ਵਿਕਰੀ	Sale

Table 4.2:COLUMN_HANDLING Table

Token word	**Mapped word**
ਨਾਮ | ਨਾਂ	Name
ਉਮਰ	Age
ਪਤਾ | ਸਿਰਨਾਵਾਂ	Address
ਸ਼ਹਿਰ	City

ਮੋਬਾਇਲ | ਮੋਬਾਇਲ-ਨੰਬਰ | ਨੰਬਰ | ਨੰ	Contact
ਕੀਮਤ | ਮੁੱਲ	Price
ਰੁੱਤ | ਮੌਸਮ	Season
ਮਜ਼ਦੂਰੀ	Labour
ਬੋਨਸ	Bonus

Table 4.3: CONDITION Table

Token word	**Mapped word**
ਜਿਹੜਾ	Where
ਜਿਹੜੀ	Where
ਜਿਸਦਾ	Where
ਜਿਸਦੀ	Where
ਜਿੰਨ੍ਹਾ	Where

Table 4.4: CONDITIONALWORDS Table

Word	Representation
ਤੋਂ ਜਿਆਦਾ	>
ਤੋਂ ਵੱਧ	>
ਤੋਂ ਅੱਗੇ	>
ਤੋਂ ਵੱਡਾ	>
ਤੋਂ ਉੱਪਰ	>
ਤੋਂ ਘੱਟ	<
ਤੋਂ ਥੋੜਾ	<
ਤੋਂ ਪਿੱਛੇ	<
ਤੋਂ ਛੋਟਾ	<
ਤੋਂ ਥੱਲੇ	<
ਬਰਾਬਰ ਨਹੀਂ	<>
ਬਰਾਬਰ	=
ਜਾਂ ਇਸ ਤੋਂ ਜਿਆਦਾ	>=
ਬਰਾਬਰ ਜਾਂ ਜਿਆਦਾ	>=
ਬਰਾਬਰ ਜਾਂ ਘੱਟ	<=
ਜਾਂ ਇਸ ਤੋਂ ਘੱਟ	<=

System searches the token first in **TABLE_HANDLING** table in the database as shown in figure 4.2. If any token matches with the entries in this table, then system will map that entry with the table name and will find the table to which that query is associated. If the system does not find the table name then query must be invalid.

After finding the table name, system searches rest of the tokens of the query in **COLUMN_HANDLING** table. If the token matches with the entries in this table, then system will map that entry with the column name and finds that column name to which that query is associated. If column name is not mapped then system is executing the query "select * from tablename;".

When column name is mapped, tokens are further searched in the database. Now if any of the tokens map with the where keyword of the SQL query, when searched in **CONDITION** table, then tokens are further searched in **CONDITIONALWORDS** table and query is executed with less than or greater than sign if conditional word found. Otherwise query with equal to condition is executed. If system does not map any token with where keyword then it will execute query "select columnname from tablename".

- **Execution of SQL Query**

  The SQL query generated is executed on the database. The data retrieved from the execution is provided to the user as output.

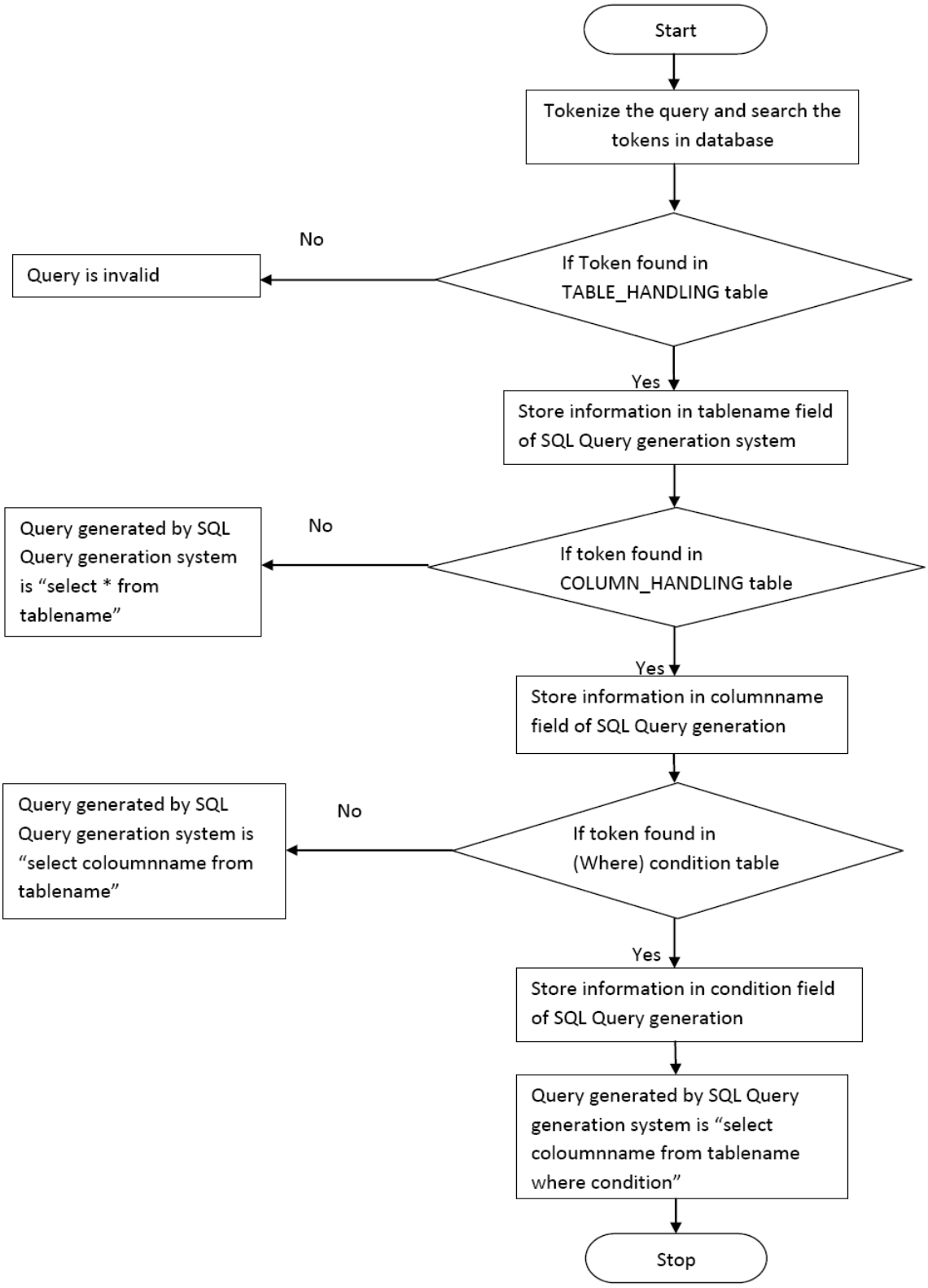

Figure 4.2: Flowchart of Punjabi Interface to Agriculture Database System

4.2 Implementation and Working of the System

We have developed the database in MS Access, and Graphical User Interface with NetBeans platform. For providing the output in Punjabi language we have used Unicode driver in MS Access and stored data in Punjabi language itself. Here, we describe the functionalities of the designed system (shown in figure 4.3) in three parts [21].

- Input query
- Execution of query
- Showing SQL query

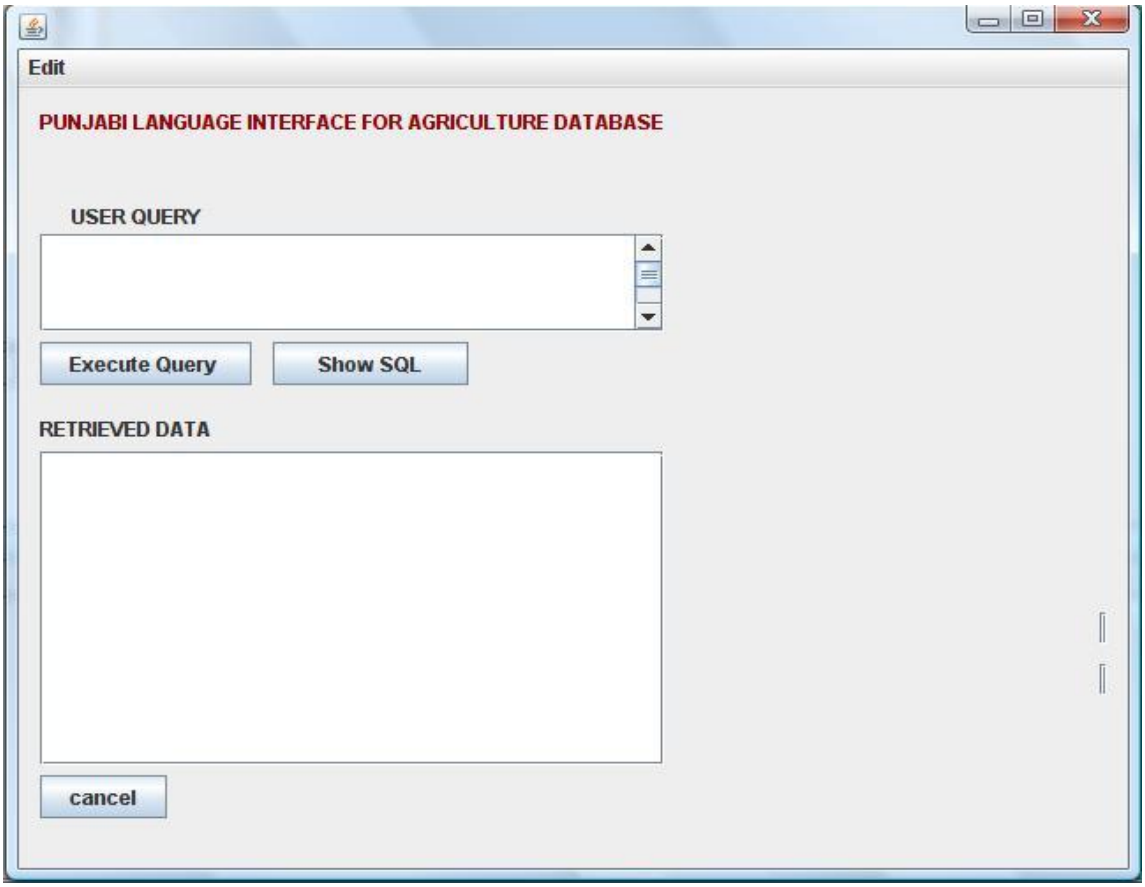

Figure 4.3: User Interface of the Punjabi Language Interface to Database

4.2.1 Input Query: User provides input in the form of simple sentence to the system in Punjabi language in the text area. User can ask queries of three different types:

- Queries for selection of whole table.
- Queries for selection of certain columns.
- Queries for selection of certain rows from certain columns *i.e.* queries with 'where' condition.
    - **Queries for Selection of Whole Table:** When user wants to retrieve data of the whole table then he inputs the query like presented in figure 4.4.

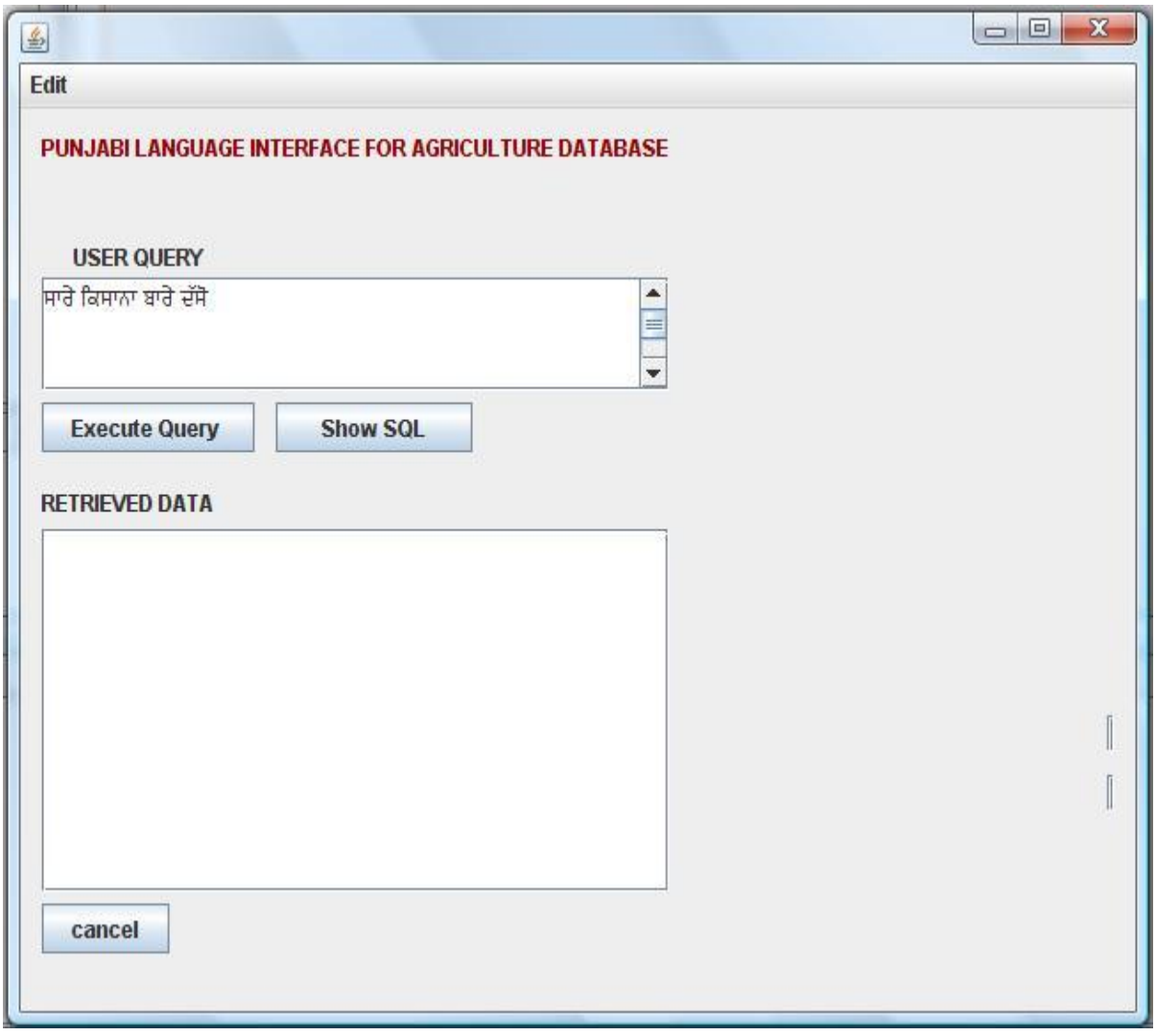

Figure 4.4: Example of Query for Selection of Whole Table

- **Queries for Selection of Certain Column:** When user wants to retrieve data from certain columns only, not of the whole table then he inputs the query like presented in figure 4.5.

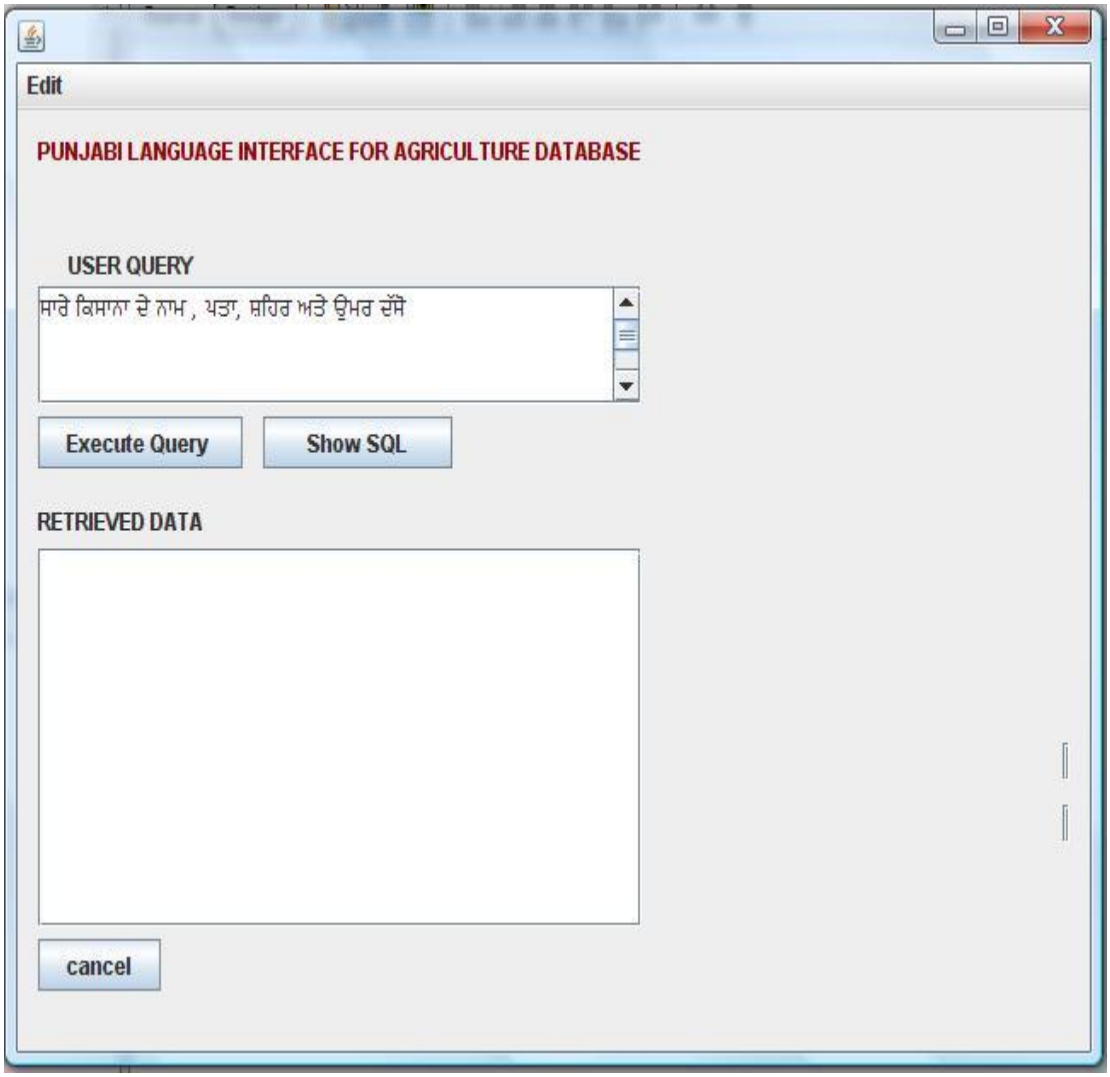

Figure 4.5: Example of Query for Selection of Certain Columns

- **Queries for Selection of Certain Rows from Certain Columns *i.e.* Queries with 'where' Condition:** When user wants to retrieve data of only certain rows from certain columns table then he inputs the query like presented in figure 4.6.

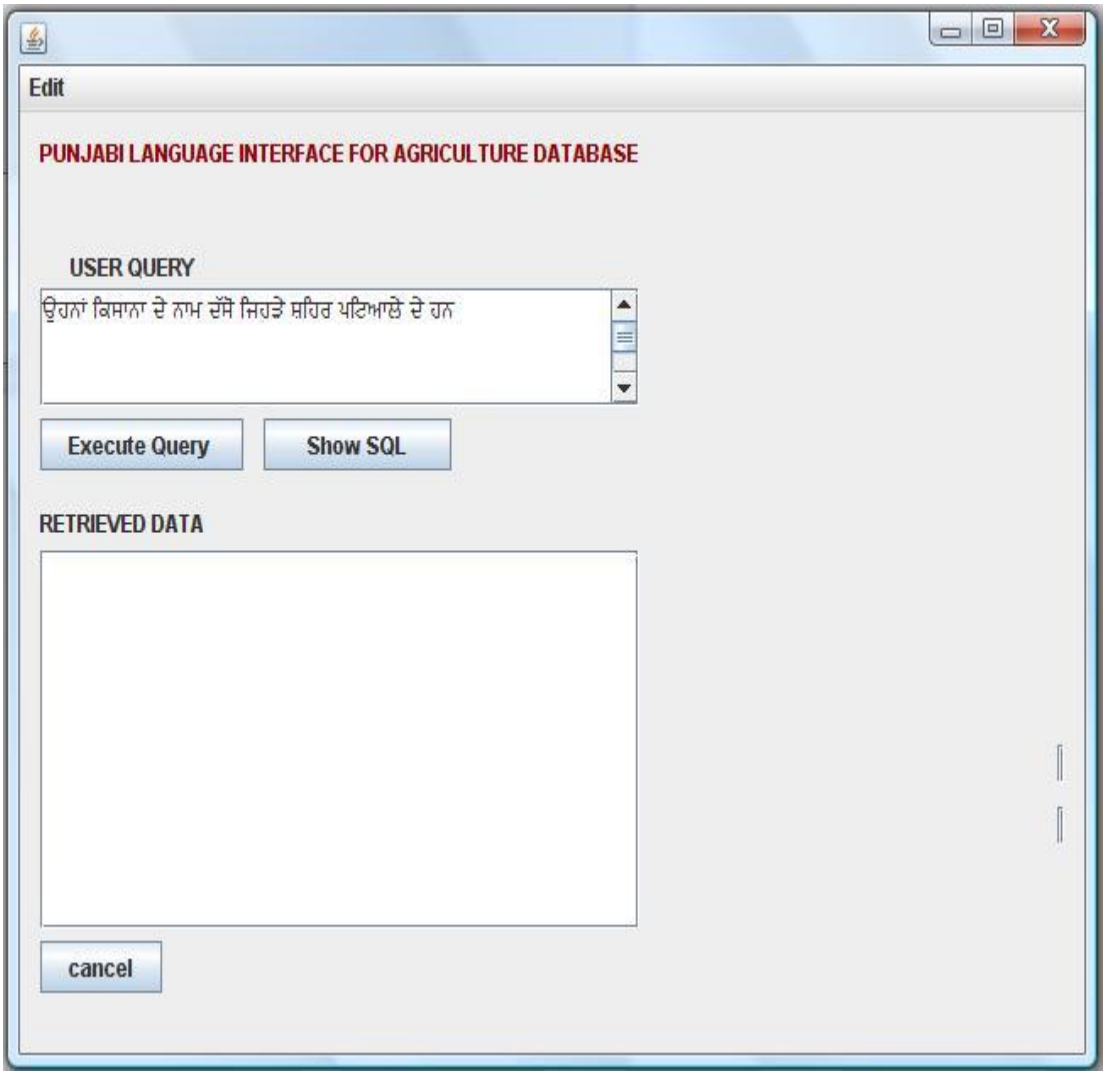

Figure 4.6: Example of Query with 'where' Condition

4.2.2 Executing the query: When user clicks on "Execute Query" button, the system maps the query with data provided in the database, according to the method explained in flowchart. After processing the query, system provides output to the user in the text area. Corresponding to three types of input, output is as follows:

- **Execution of Queries for Selection of Whole Table:** When user query is for the selection of whole table then after execution of that query, results are like shown in figure 4.7.

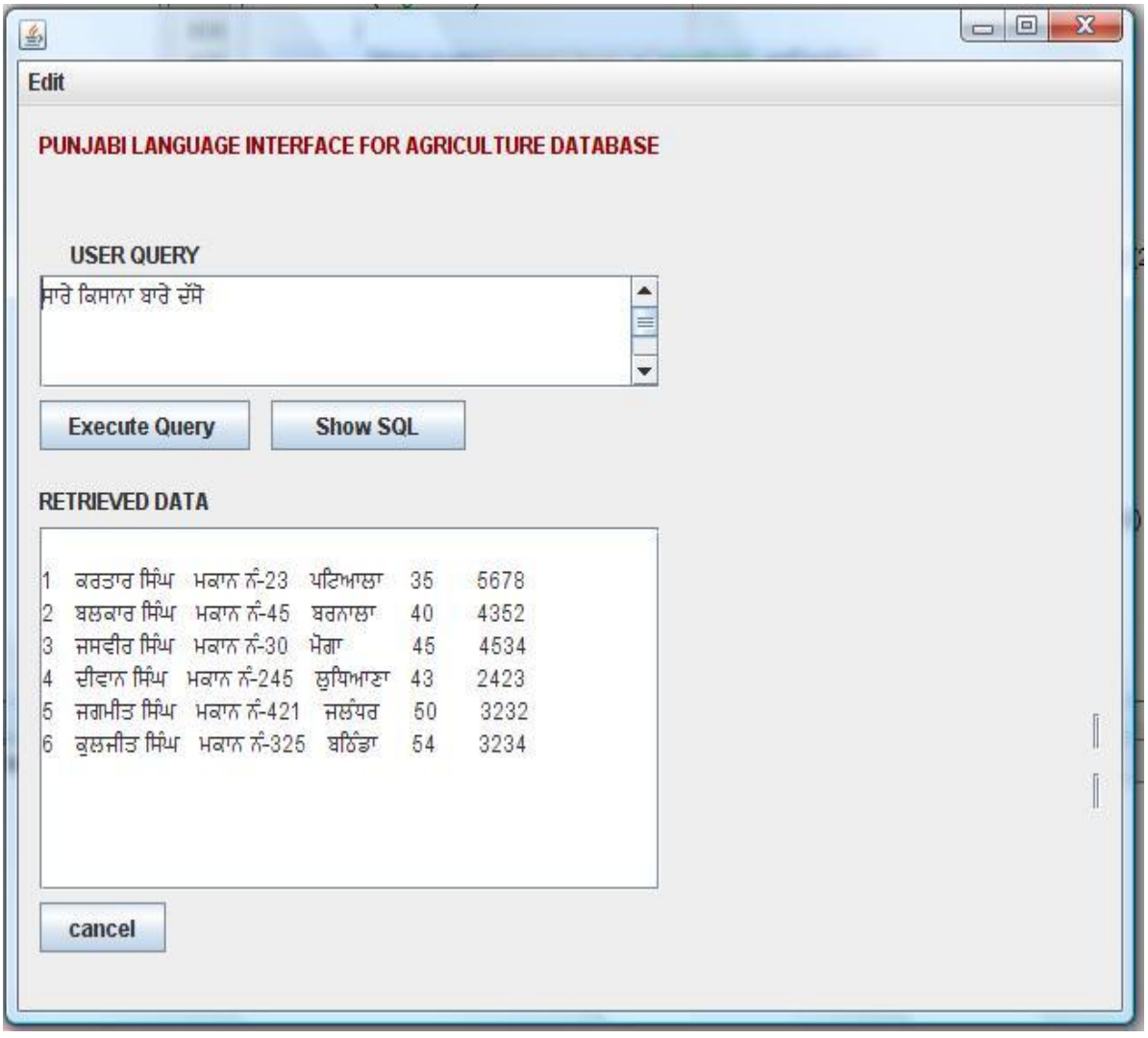

Figure 4.7: Results after Executing Query for Selection of Whole Table

- **Execution of Queries for Selection of Certain Columns of Table:** When user query is for the selection of certain columns of the table then after execution of that query, results are like shown in figure 4.8.

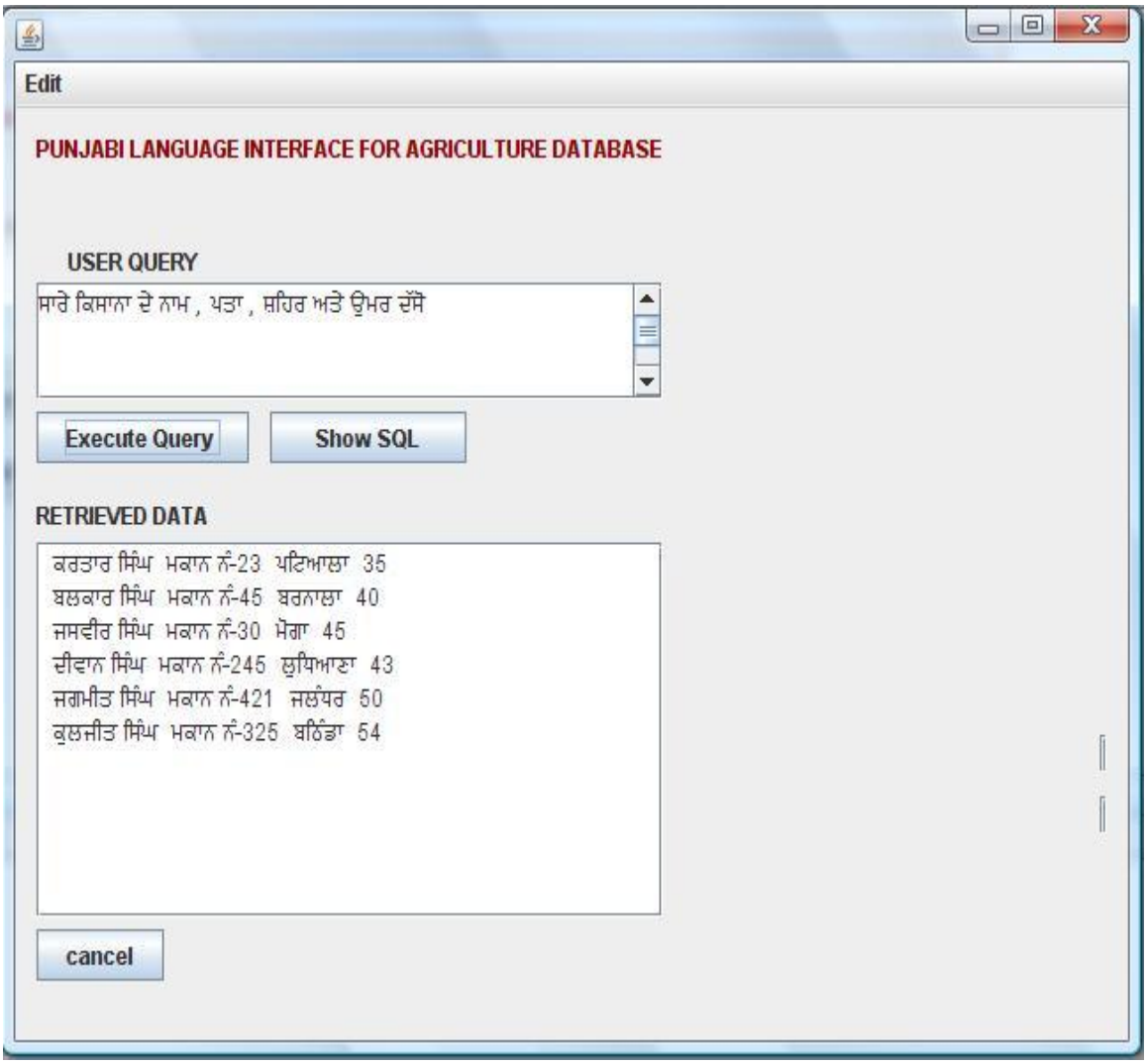

Figure 4.8: Results after Executing Query for Selection of Certain Columns

- **Execution of Queries for Selection of Certain Rows from Certain Columns:** When user query is for the selection of only certain rows from certain columns then, after execution of that query, results are like shown in figure 4.9.

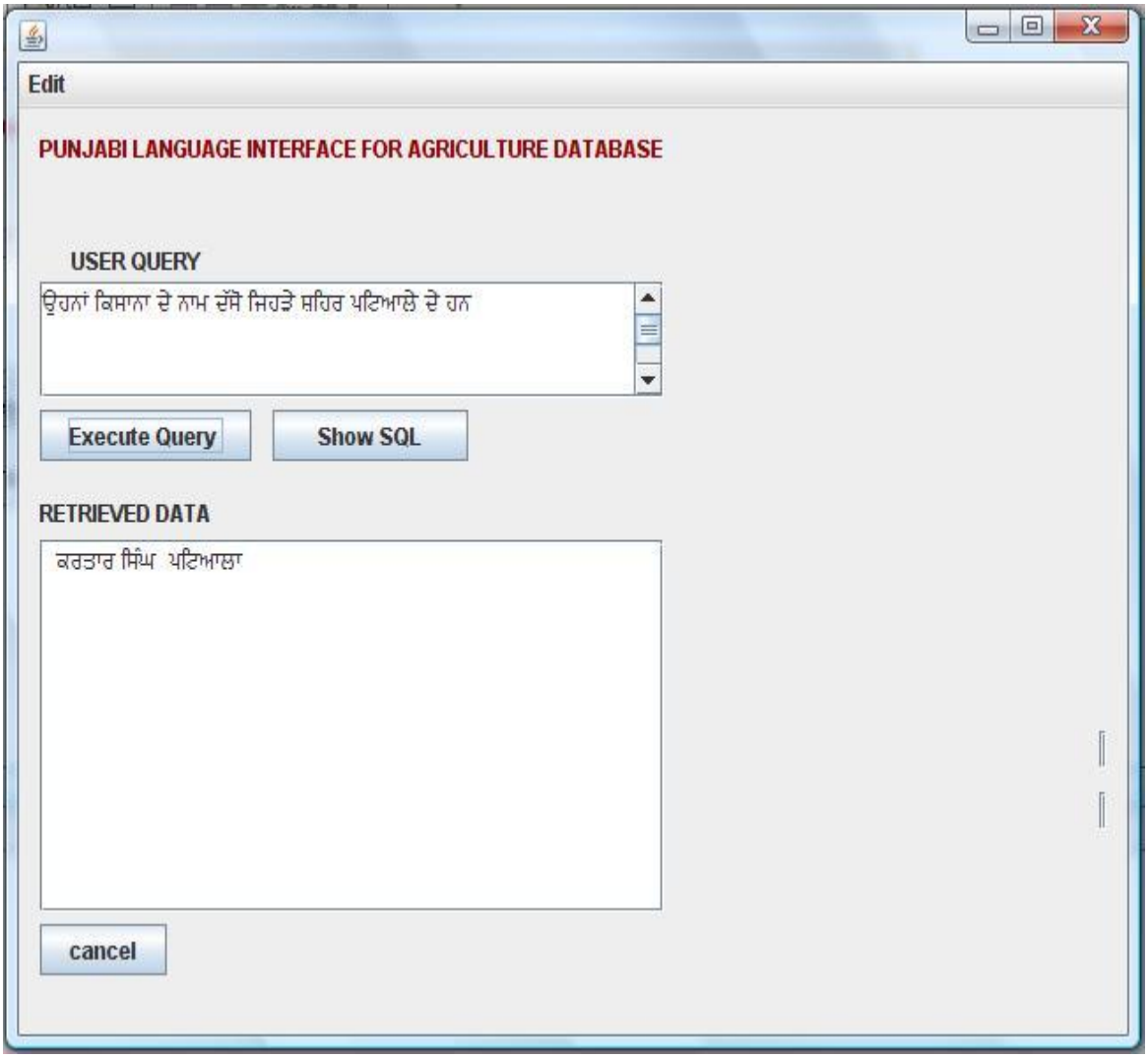

Figure 4.9: Results after Executing Query with 'where' Condition

4.2.3 Showing SQL query: When user clicks on the "Show SQL" button, system will show SQL query generated with respect to the input Punjabi query. According to the input queries, the corresponding SQL queries generated are as follows:

- **SQL Query for Selection of Whole Table:** When user query is for the selection of whole table then the corresponding SQL query generated by the system is shown in figure 4.10.

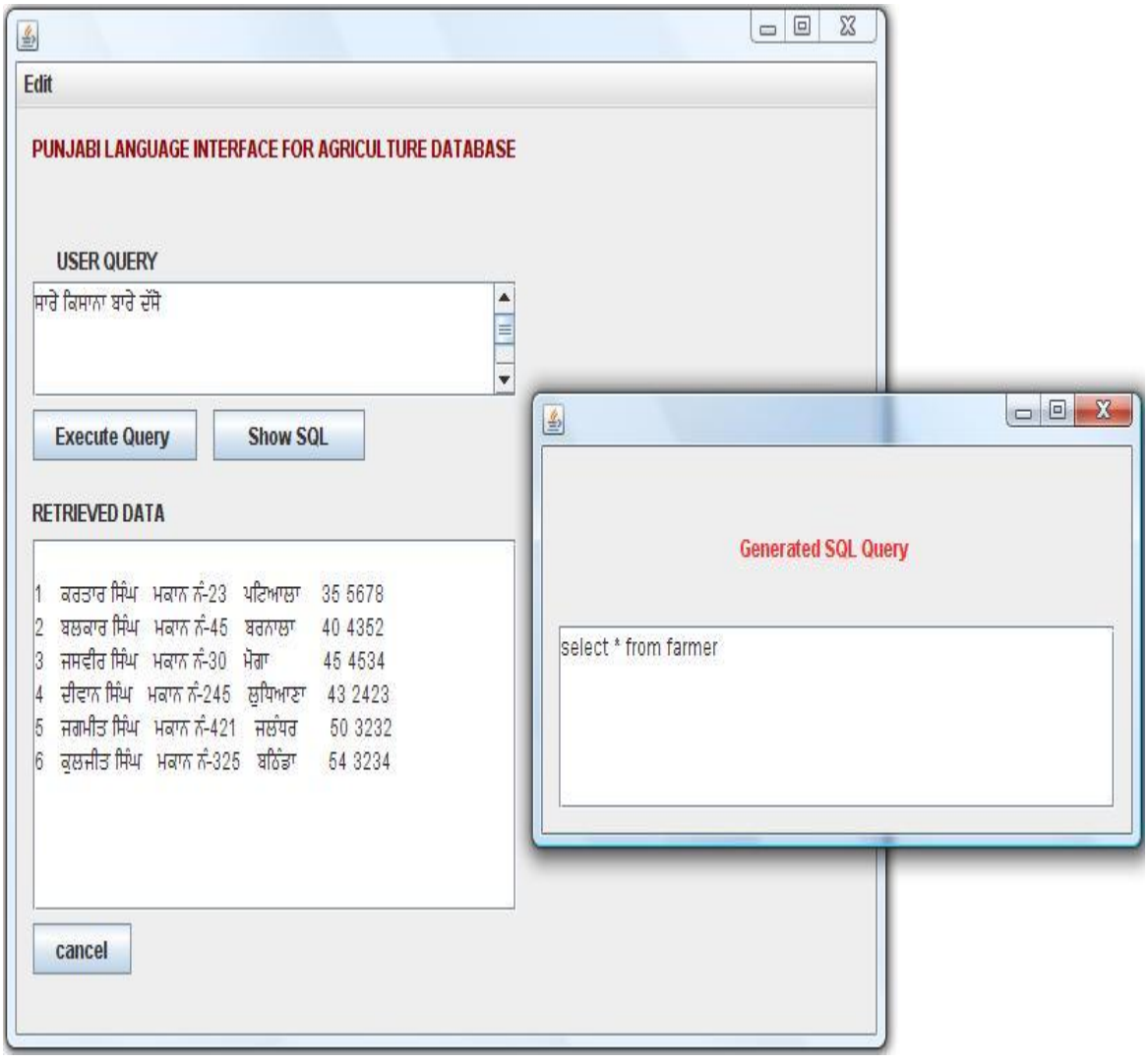

Figure 4.10: SQL Query Shown for Selection of Whole Table

- **SQL Query for Selection of Certain Columns:** When user query is for the selection of certain columns of the table then the corresponding SQL query generated by the system is shown in figure 4.11.

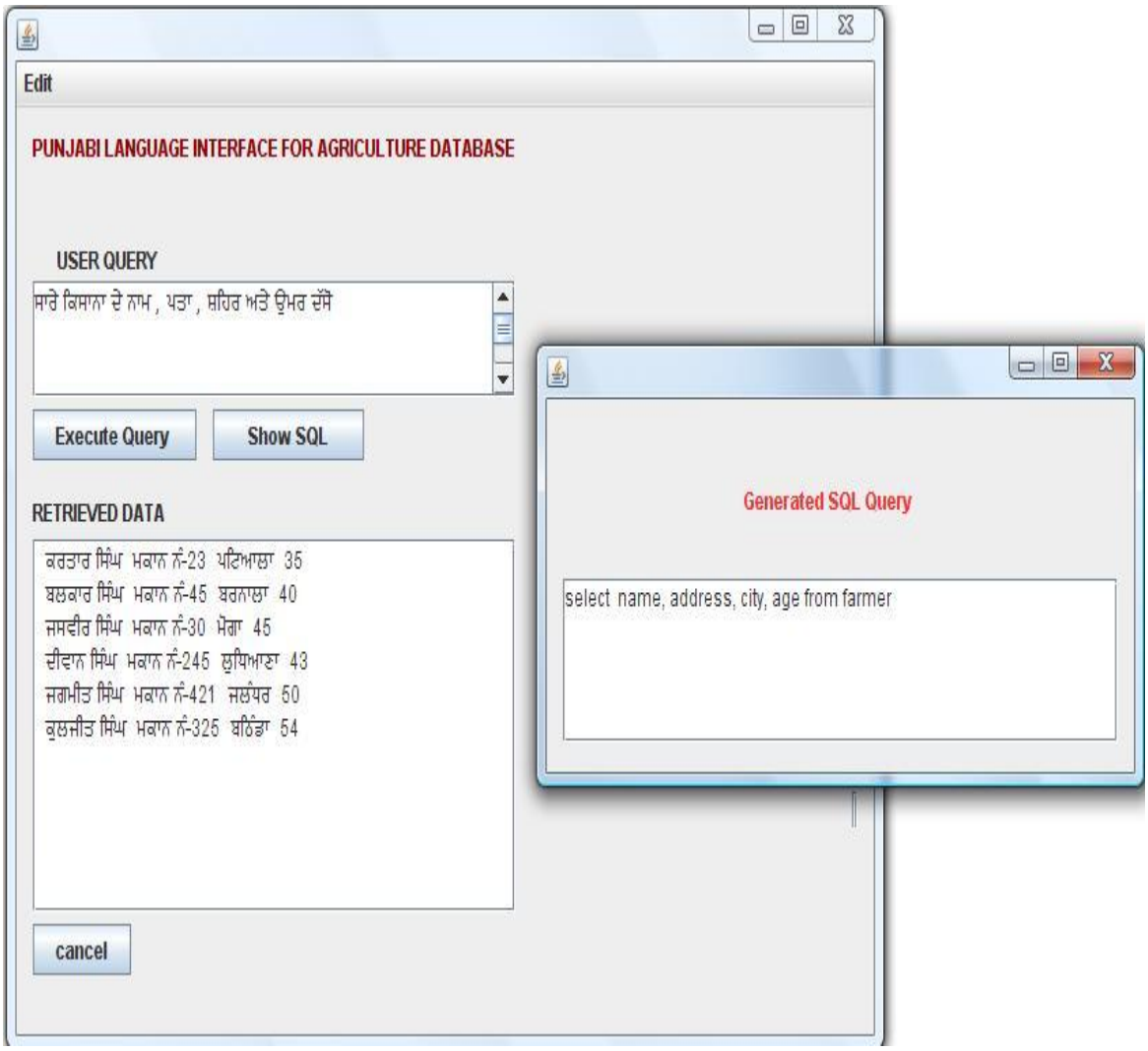

Figure 4.11: SQL Query Shown for Selection of Certain Columns

- **SQL Query for Selection of Certain Rows from Certain Columns of the Table:** When user query is for the selection only certain rows from certain columns then the corresponding SQL query generated by the system is shown in figure 4.12.

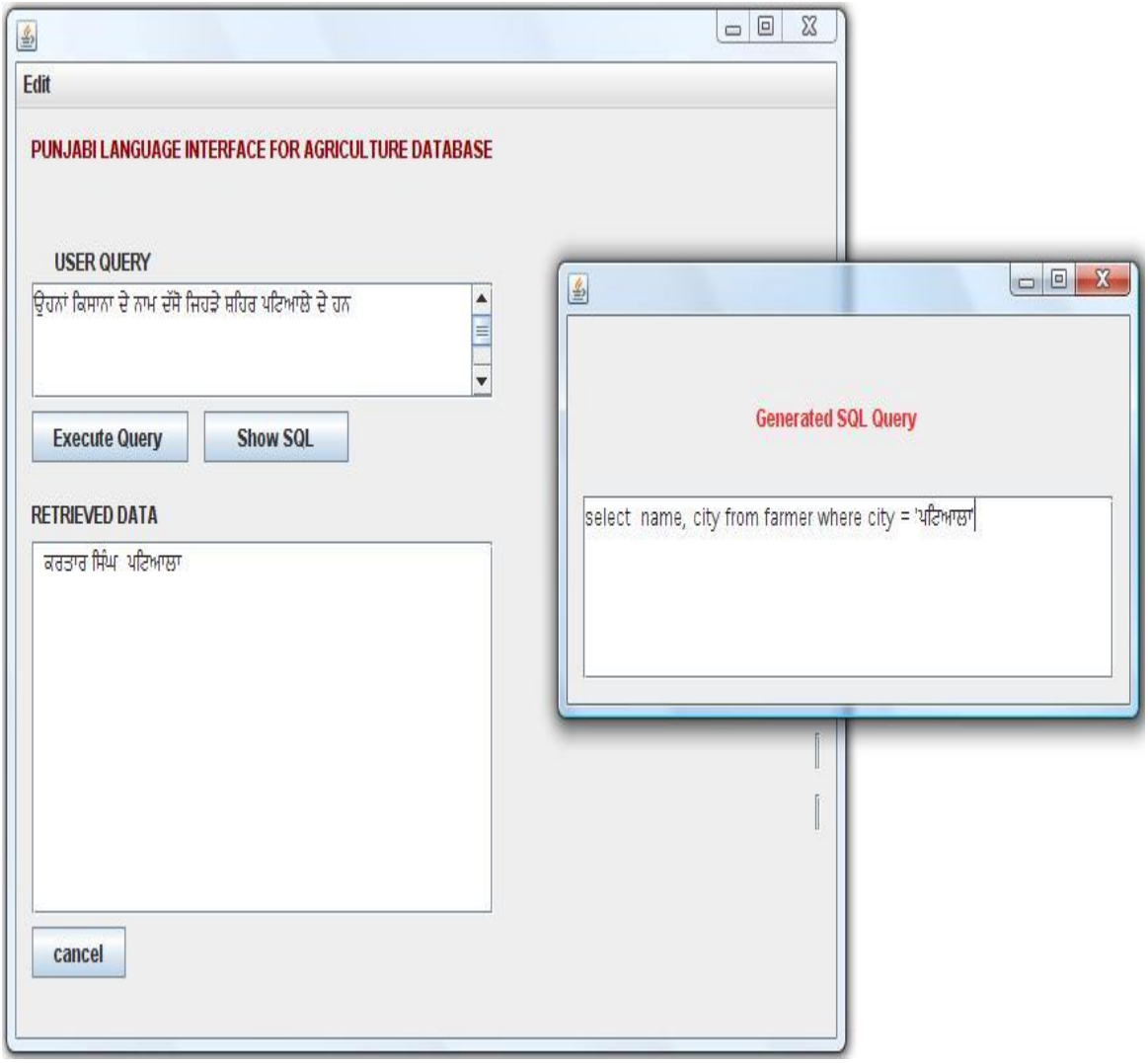

Figure 4.12: SQL Query Shown with 'where' Condition

Chapter 5

Testing of Punjabi Language Interface to Agriculture Database

We have tested the system for 3 types of queries. These are as follows:

- Queries for selection of all the columns.
- Queries for selection of certain columns.
- Queries for selection of certain rows from certain columns.

These queries and their SQL formation is given in table 5.1, 5.2 and 5.3. Font used for transliteration is according to APPENDIX A.

Table 5.1: Queries for Selection of All Columns

Punjabi Query	**Transliteration of Query**	**Translation of Query in English**	**SQL Query**
ਸਾਰੇ ਕਿਸਾਨਾ ਬਾਰੇ ਦੱਸੋ	sare kYsana bare dWsU	List all the farmers	Select * from farmer;
ਸਾਰੇ ਕਿਸਾਨਾ ਦੀ ਜਾਣਕਾਰੀ ਦੇਵੋ	sare kYsana dYI jankarYI dewU	List details of all the farmers	Select * from farmer;
ਸਾਰੇ ਕਿਸਾਨਾ ਦਾ ਵੇਰਵਾ ਦੱਸੋ	sare kYsana da werwa dWsU	List details of all the farmers	Select * from farmer;
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਬਾਰੇ ਦੱਸੋ	sarYIAN phslaN bare dWsU	List all the crops	Select * from crop;
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਦੀ ਜਾਣਕਾਰੀ ਦੇਵੋ	sarYIAN phslaN dYI jankarYI devo	List all the crops	Select * from crop;
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਦਾ	sarYIAN phslaN da	List all the crops	Select * from crop;

ਵੇਰਵਾ ਦੱਸੋ	werwa dWsU		

Table 5.2: Queries for Selection of Certain Columns

Punjabi Query	Transliteration of Query	Translation of Query in English	SQL Query
ਸਾਰੇ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਦੱਸੋ	sare kYsana de nam dWsU	List names of all the farmers	Select name from farmer;
ਸਾਰੇ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਅਤੇ ਸ਼ਹਿਰ ਦੱਸੋ	Sare kYsana de nam ate shhYr dWsU	List name and city of all the farmers	Select name, city from farmer;
ਸਾਰੇ ਕਿਸਾਨਾ ਦੇ ਨਾਮ , ਸ਼ਹਿਰ ਅਤੇ ਉਮਰ ਦੱਸੋ	sare kYsana de nam , shhYr ate umr dWsU	List name, city and age of all the farmers	Select name, city, age from farmer;
ਸਾਰੇ ਕਿਸਾਨਾ ਦੇ ਨਾਮ , ਪਤਾ , ਮੋਬਾਇਲ , ਸ਼ਹਿਰ ਅਤੇ ਉਮਰ ਦੱਸੋ	sare kYsana de nam, pta, mUbaaYl, shhYr ate umr dWsU	List name, address, mobile, city, and age of all the farmers	Select name, address, mobile, city, age from farmer;
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਦੇ	sarYIAN phslaN de nam dWsU	List names of all the crops	Select name from crop;

ਨਾਮ ਦੱਸੋ			
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਦੇ ਨਾਮ ਅਤੇ ਰੁੱਤ ਦੱਸੋ	sarYIAN phslaN de nam ate ruWt dWsU	List names and seasons of all the crops	Select name, season from crop;
ਸਾਰੀਆਂ ਫ਼ਸਲਾਂ ਦੇ ਨਾਮ , ਰੁੱਤ ਅਤੇ ਕੀਮਤ ਦੱਸੋ	sarYIAN phslaN de nam, ruWt ate kYImt dWsU	List name , season and price of all the crops	Select name, season, price from crop

Table 5.3: Queries for Selection of Certain Rows from Certain Columns

Punjabi Query	**Transliteration of Query**	**Translation of Query in English**	**SQL Query**
ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਨ੍ਹਾ ਦੀ ਉਮਰ ੩੫ ਸਾਲ ਹੈ	kYsana de nam dWsU jYnHa dYI umr 35 sal hI	List name of farmers whose age is 35 years	Select name from farmer where age = 35;
ਸਾਰੇ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਅਤੇ ਸ਼ਹਿਰ ਦੱਸੋ ਜਿਨ੍ਹਾ ਦੀ ਉਮਰ ੩੫ ਸਾਲ ਹੈ	sare kYsana de nam ate shhYr dWsU jYnHa dYI umr 35 sal hI	List name and city of all the farmers whose age is 35 years	Select name, city from farmer where age = 35;
ਉਹਨਾਂ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਹੜੇ	uhnaN kYsana de nam dWsU jYhRe pTYAle shhYr de hn	List name of farmer whose city is	Select name from farmer where

ਪਟਿਆਲੇ ਸ਼ਹਿਰ ਦੇ ਹਨ		ਪਟਿਆਲਾ	city = ਪਟਿਆਲਾ ;
ਉਹਨਾਂ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਹੜੇ ਸ਼ਹਿਰ ਪਟਿਆਲੇ ਦੇ ਹਨ	uhnaN kYsana de nam dWsU jYhRe pTYAle shhYr de hn	list name of farmers who are from city ਪਟਿਆਲਾ	Select name from farmer where city = ਪਟਿਆਲਾ ;
ਉਹਨਾਂ ਕਿਸਾਨਾ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਹੜੇ ਪਟਿਆਲਾ ਸ਼ਹਿਰ ਦੇ ਹਨ	uhnaN kYsana de nam dWsU jYhRe pTYAle shhYr de hn	List name of farmers who belongs to city ਪਟਿਆਲਾ	Select name from farmer where city = ਪਟਿਆਲਾ ;
ਫ਼ਸਲਾਂ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਹੜੀਆਂ ਹਾੜੀ ਰੁੱਤ ਦੀਆਂ ਹਨ	phslaN de nam dWsU jYhRYIAN haRYI ruWt dYIAN hn	List name of crops that are of ਹਾੜੀ season	Select name from crop where season= ਹਾੜੀ;
ਫ਼ਸਲਾਂ ਦੇ ਨਾਮ ਅਤੇ ਕੀਮਤ ਦੱਸੋ ਜਿਹੜੀਆਂ ਹਾੜੀ ਰੁੱਤ ਦੀਆਂ ਹਨ	phslaN de nam ate kYImt dWsU jYhRYIAN haRYI ruWt dYIAN hn	List name and price of crops that are of ਹਾੜੀ season	Select name,price from crop where season= ਹਾੜੀ;

ਫ਼ਸਲਾਂ ਦੇ ਨਾਮ ਦੱਸੋ ਜਿਸਦੀ ਕੀਮਤ 1000 ਰੁਪਏ ਹੈ	phslaN de nam dWsU jYsdYI kYImt 1000 rupe hI	List name of crops which have price of rs.1000	Select name from crop where price=1000;

We have tested the Punjabi Language Interface for Agriculture database for different queries, which include selection of whole table, selection of certain columns of a table and selection of certain rows of the table. These queries successfully get converted into corresponding SQL queries and then execute on database to produce the desired results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The Natural Language Interface to Database system, for Punjabi language using agriculture database, accepts query in Punjabi language that is translated into SQL query, by mapping the Punjabi language words, with their corresponding English words with the help of database maintained. Then this SQL query is executed on database to provide output to the user. Major concepts in this system are:

- Tokenizing the input query and searching that token in the database whether it belongs to tablename or columnname.
- After analyzing query whether its simple query or a query with condition, system will form appropriate SQL query.
- Execute SQL query on the database to retrieve required information. And provide output to the user.

5.2 Future Work

In future the system can be extended to perform the following tasks

- It can execute Queries that involve two or more tables, *i.e.* queries involving the joining of tables
- Development of interface for administrator as till now administrator has to make entries in the database itself.
- Using the system with another database.
- Using the system with DBMS other than MS Access.
- Punjabi is a free word language, but our system accept queries only in a particular format. So to make the system to accept all the queries we can use the system with Punjabi parser.

References

[1] Gobinda G. Chowdhury, "Natural Language processing", Strath cis publication, (2001), pp 1

[2] Rajendra Akerkar, Manish Joshi, **"**Natural Language Interface Using Shallow Parsing", International Journal of Computer Science and Applications, Vol. 5, No. 3, pp 70 – 90

[3] Tanveer Siddiqui, U.S. Tiwary, "Natural Language Processing and Information Retrieval" Oxford university press, (2008)

[4] Faraj A. El-Mouadib, Zakaria Suliman Zubi, Ahmed A. Almagrous, I. El-Feghi, "Interactive Natural Language Interface", WSEAS transactions on computers, issue 4, volume 8, (2009), pp 5-9

[5] I. Androutsopoulos, G.D. Ritchie, P. Thanisch, "Natural Language Interfaces to Databases – An Introduction", Journal of Natural Language Engineering, vol. 1, No. 1. Cambridge University Press, (1995)

[6] W. Woods, R. Kaplan and B. Webber, B., "The Lunar Sciences Natural Language Information System", Final Report. B. B. N. Report No 2378, USA, (1972)

[7] W. Woods, "An experimental parsing system for transition network grammars. In Natural Language Processing", Algorithmic Press, New York, USA, (1973)

[8] G. Hendrix, E. Sacrdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data", ACM Transactions on Database Systems, Volume 3, No. 2, USA, (1978), pp. 105 – 147

[9] E.F. Codd, "Seven steps to rendezvous with the casual user", In IFIP Working Conference Data Base Management, (1974), pp 179–200

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

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

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

[13] 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: (1987), pp 173–243

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

[15] 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, (1983), pp 3–16

[16] C.D. Hafner, "Interaction of Knowledge Sources in a Portable Natural Language Interface", In Proceedings of the 22nd Annual Meeting of ACL, Stanford, California, (1984) pp 57–60

[17] 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): (January 1984), pp 1–25

[18] F. Damerau, "Operating statistics for the transformational question answering system", American Journal of Computational Linguistics, 7: (1981), pp 30–42

[19] B. Ballard and D. Stumberger, "Semantic Acquisition in TELI", In Proceedings of the 24th Annual Meeting of ACL, New York, (1986), pp 20–29

[20] Seymour Knowles, “A Natural Language Database Interface For SQL-Tutor”, 1999

[21] Faraj A. El-Mouadib, Zakaria S. Zubi, Ahmed A. Almagrous, and Irdess S. El-Feghi “Generic Interactive Natural Language Interface to Databases (GINLIDB)”, international journal of computers(2009).

Research Publications

Research Papers Published

- Amandeep Kaur, Parteek Bhatia, "Retrieval of Information from Database with Natural Language Interface" published at NCACCET-10 approved by AICTE, CCET Chandigarh. (Jan 29-30, 2010)

- Amandeep Kaur, Parteek Bhatia, "Natural Language Interface to Database" published at National Conference on NGC-2010 GITM, Gurgaon. (Mar 20, 2010)

Research Papers Communicated

- Amandeep Kaur, Parteek Bhatia, "Punjabi Language Interface to Database" communicated to International journal of computer science, WASET (World Academy of Science and Technology)

APPENDIX A

Punjabi Alphabet	English mapping
ਸ	S
ਹ	h
ਕ	k
ਖ	kh
ਗ	g
ਘ	gh
ਙ	Mg
ਚ	ch
ਛ	chh
ਜ	j
ਝ	jh
ਞ	Mj
ਟ	T
ਠ	Th
ਡ	D
ਢ	Dh
ਣ	n

ਤ	t
ਥ	th
ਦ	d
ਧ	dh
ਨ	n
ਪ	p
ਫ	ph
ਬ	b
ਭ	bh
ਮ	M
ਯ	Y
ਰ	R
ਲ	L
ਵ	W
ੜ	R
ਸ਼	Sh
ਖ਼	Kh
ਗ਼	G
ਜ਼	Z

ਫ਼	F
ਸ	S
ਹ	H
ਉ	Au
ਊ	auU
ਈ	aYI
ਇ	aY
ਆ	A
ਏ	e
ਐ	ay
ਔ	a/U
ਓ	aU
ਅ	a
ਾ	a
ਿ	Y
ੀ	YI
ੇ	e
ੈ	I
ੋ	U

◌ُ	/U
◌ُ	u
◌ُ	uU