ADAMA SCIENCE & TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

POSSIBILITY OF AMHARIC QUERY PROCESSING IN DATABASE USING NATURAL LANGUAGE INTERFACE

A thesis submitted to School of Graduate Studies of Adama Science and Technology University in partial fulfillment of the requirements for the degree of Master of Science in Information System

By

Yehitfenta Minuyelet

June 2016

ADAMA SCIENCE & TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

POSSIBILITY OF AMHARIC QUERY PROCESSING IN DATABASE USING NATURAL LANGUAGE INTERFACE

A thesis submitted to School of Graduate Studies of Adama Science and Technology University in partial fulfillment of the requirements for the degree of Master of Science in Information System

BY: Yehitfenta Minuyelet

Names and Signature of Members of the Examining Board

Role	Name	Signature
Chair person, Examining board		
Advisor	SOLOMON TEFERA (PhD)	
Internal examiner		
External examiner		

Acknowledgements

First and foremost I would like to warmly thank my research advisor, Dr. Solomon Tefera, for his critical and constructive comments of my work and for sharing his best research experience from the very beginning to the end of this paper. It was a pleasure and great opportunity to constantly advised and guided by such an intellectual advisor.

I also want to extent heartfelt gratitude to Dr Million Meshesha for providing me the technical support throughout the paper.

Special thanks go to my best friends, Atakiliti Kassu and Semegnew Asemay, for their technical support.

I would like to thank my friends, who were with me, for their moral and material support and many enjoyable moments we had to finish this research.

Last but not least I want to thank Wachemo University and Adama Science and Technology University for sponsorship.

Declaration

I, Yehitfenta Minuyelet, declare that, this thesis entitled: Possibility of Amharic Language interface in database: is my original work produced under the guidance of my advisor Dr. Solomon Tefera and has never been published and/or submitted for any award of Degree in any other University. Any source used is duly acknowledged in this study.

Signed: ----------

June, 2016

Abbreviations

NLI	Natural language interface
AI	Artificial Intelligence.
NLP	Natural language processing
NLIDB	Natural language interface in database
DBMS	Database management system.
SQL	Structural query language
RDBMS	Relational database management system.
PLSQL	Programming language with structural query language
IQ	Intermediate Query
HMM	Hidden Markov Models.
ALIDB	Amharic language interface in database
IE	Information Extraction.
IES	Information Extraction System.
IJCNLP	International Joint Conference on Natural Language Processing.
IR	Information Retrieval.
ECSA	Ethiopian Central Statistics Authority
IDBS	Intelligent database systems
GUI	Graphical User Interface
DBA	Database administrator
CFG	Context-Free Grammar

Contents

List of Table

List of Figure

Abstract

In the world, information plays an important role in our lives. One of the major sources of information is database. Database and database technology are having major impact on the growing use of computers. Almost all IT applications are storing and retrieving the information or data to and from the database. 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 with users. For storing and retrieving the information from database it requires the knowledge of database language like SQL. Structured Query Language (SQL) is an ANSI standard for accessing and manipulating the information stored in databases. Only few people who have knowledge of database structure and formal database language (such as Structured Query Language (SQL) can retrieve the desired information from databases.

The objective of this research is to explore the development of an efficient and user friendly Amharic Language Query Interface to Database (ALIDB) system that allows non-technical users to interact with database using Amharic language.

The prototype system can handle the following types of queries: List Queries, Condition Queries and Join queries is developed. It is tested with 60 Amharic query sentences which are collected from 12 users. The accuracy of the system is measured in term of precision percentage with two classes that identifies query response as: Correct and Incorrect. The results found are encouraging and the overall efficiency of system is observed to be 63.3%.

Keywords: database, Database management system, NLP, NLI, NLIDB, Amharic language interface in database.

Chapter One

Introduction

1.1 Background

Natural Language Processing (NLP) is a field of research and application that determines the way computers can be used to understand and manage natural language text or speech to do useful things. The term “natural” in the context of the language is used to distinguish human languages (such as Amharic, Tigrigna, AfanOromo, English and so on) from computer languages (such as C, C++, Java and Prolog). Natural language processing is becoming one of the most active areas in Human-computer Interaction.

The goal of NLP is to enable communication between people and computers without resorting to memorization of complex commands and procedures. In other words, NLP is a technique which can make the computer understand the languages naturally used by humans 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. Figure 1.1 represents a generic NLP system.

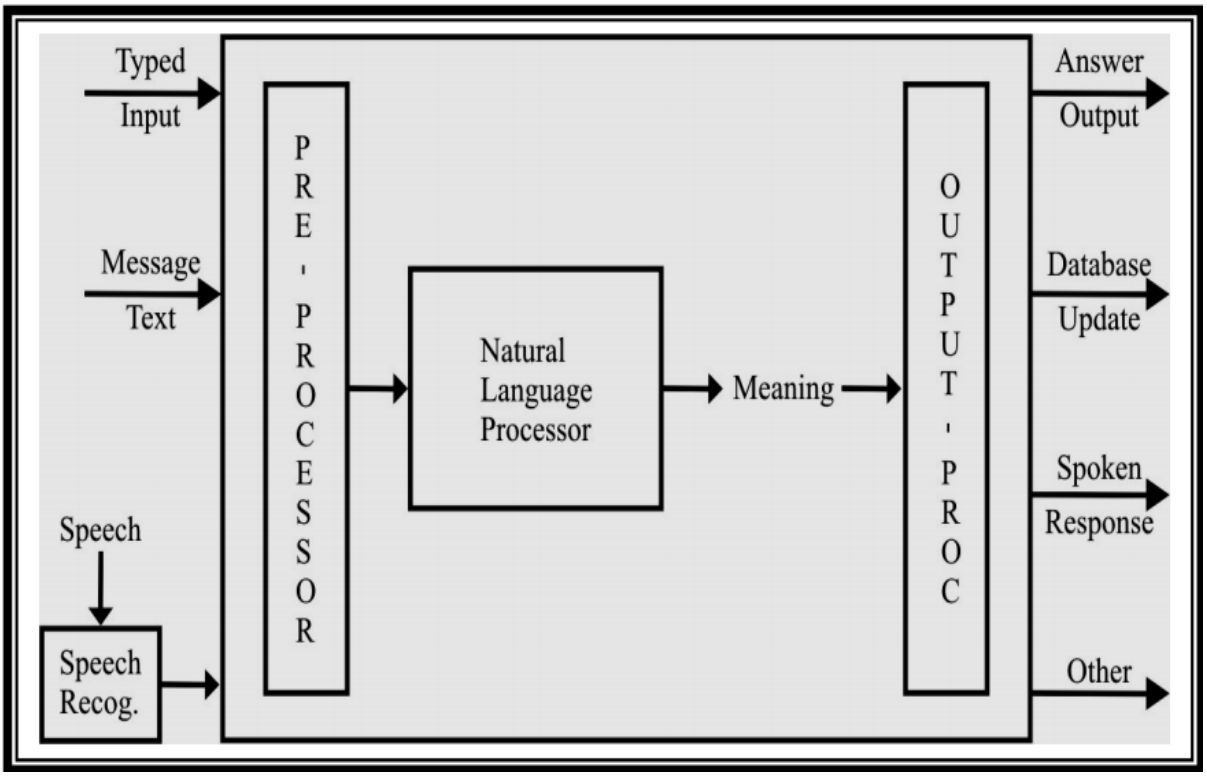

Figure 1.1: Generic NLP system [43]

The NLP can broadly be divided into various levels as shown in figure 1.2

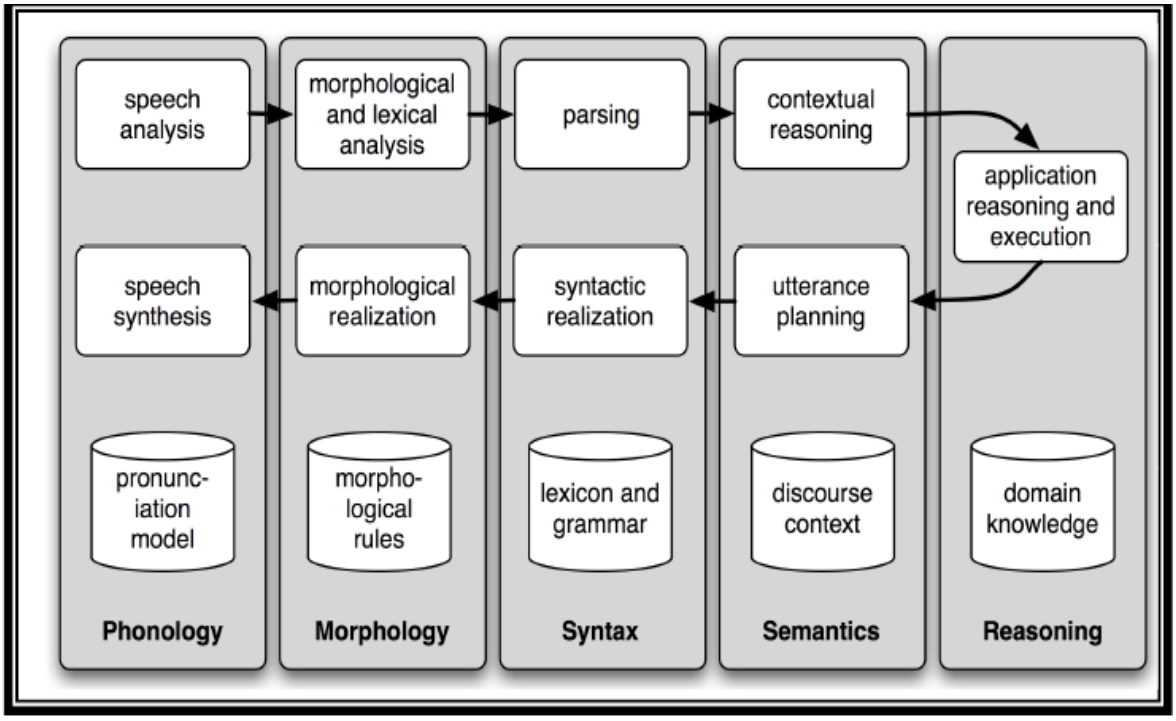

Figure 1.2 NLP levels [44]

Natural Language Interface to Database (NLIDB): It is one of the applications of natural language process and it is a process of finding answers from database by asking questions in natural language.

Natural language understanding is the major challenge for any NLP that is enabling the computers to extracts meanings from natural language query.

Processing natural language to extract query information will be done using keywords and statistical keyword disambiguation using a vector similarity measure. We defined keywords to be words or phrases that hold particular meaning within the domain [2].

One of the most wide and interesting area of Natural Language Processing (NLP) is the development of a natural language interface to database systems (NLIDB). In the last few decades many NLIDB systems have been developed. Through these systems, users can interact with database in a more convenient and flexible way [5].

We require information in our daily life. One of the major sources of information is database. A database is made up of three types of elements: relations, attributes and values. Each element is distinct and unique, an attribute element is a particular column in a particular relation and each value element is the value of a particular attribute. A value is compatible with its attribute and also with the relation containing this attributes [12].

Query processing in databases that provide low-level access routines such as flat file databases, the programmer must write code to perform the queries. With higher level database query languages such as SQL, a special component of the DBMS called the Query Processor takes care of arranging the underlying access routines to satisfy a given query. Thus queries can be specified in terms of the required results rather than in terms of how to achieve those results.

Database Management System is a collection of interrelated data and set of programs to access those data. Databases management systems quires are written in the form of complex language like SQL, SPARQL etc. which is difficult for the causal and non-technical users that limits the access to the databases whereas a Natural Language Interfaces to Databases (NLIDBs) provide mechanism for those users to access the database with their native language.

In Ethiopia most working areas that using Amharic language in day to day activities and that records their data to database in Amharic language but there is no knowledge of Query languages like SQL to retrieve their data from database and update and delete their data to database .The goal of this study is to develop Amharic language interface in database in order to show the possibility of Amharic query processing in database using natural language interface. It provides communication between user and computer without knowledge of SQL query or using their native language.

1.2 Statement of the Problem

Amharic language query processing on database is not used in most working areas but some experiments or researches have been conducted in other languages like Indian and English language. NLIDB starting from the year 1970 and still the work is going on, but now a day in Ethiopia is not conducted in Amharic language.

Amharic is an official working language of the Federal Democratic Republic of Ethiopia. According to a census report by ECSA (Ethiopian Central Statistics Authority) (1998), it is the first language for more than 17 million and second language for over 5 million people. But in 2007 there are 25 million native speakers, outside Ethiopia; Amharic is the language of some 2.7 million emigrants.

In working areas, Ethiopia that use Amharic language, for this large number of working areas that use the recorded data for their activities so, to easily access data from the recorded data or from the database, the user that should use query language. But Query languages like SQL are very difficult to use for general users and they find it hard to learn. So to make database applications easy to use for these users or general users, to make Amharic language interface to database has been developed for easily use. The query will be asked in the Amharic language for retrieving relevant information from database and there is a facility of updating and deleting the data from table or database by users. Users give the input in Amharic query sentences i.e. unstructured query sentences, the system that display structured query sentences and the result or the retrieved data from database in same language to the users on the graphical user interface. Some researchers conducted in related to natural language interface in database done in Hindi

language by BTKIT, Dwarahat, Almora, and Uttarakhand, and in English language Ambala and Haryana India.

Asking questions to databases in natural language is a very convenient and easy method of data access, especially for casual users who do not understand complicated database query languages such as SQL.

The method used in this research in Amharic query processing in database can contribute to the conversion or translation of unstructured query sentences in to structure query sentences and simplification of retrieving data from database for Amharic language.

1.3 Research Questions

In this study, the following questions have been answered with the result obtained from the experiment:

1. What are the challenges of developing for Amharic language interface in database?
2. How to convert unstructured query sentences into structure one?
3. How to map user requirement with proper structured query language?

1.4 Objective of the Study

To contribute towards addressing the problems described above, the study has the following general and specific objectives:

1.4.1 General Objective

The general objective of the study is to show the possibility of Amharic query processing for database using the natural language interface.

1.4.2 Specific Objectives

With the intention of providing a tool to the system for Amharic query processing in database using natural language interface have the following specific objectives:

- To review previous researches as related works

- To understand the concept of query processing in database using natural language interface.
- To develop a system that can handle queries for conversion of unstructured Amharic query sentences in to structure query sentences and retrieving the data from tables stored in database.
- To examine the effectiveness of the prototype using the queries prepared for the experimentation
- To discuss and report experimental results found and recommend further research works in the area.

1.5 Methodology

To realize the main goal of this study, different methods and techniques are used**.** The main procedure or methods of the study described on below.

1.5.1 Method of Data Collection

The data was collected from educational institutions by using observation for record data in database. The repositories of natural language queries are prepared based on various question categories which is list query, single conditional query and simple joining table query within single condition and 60 queries collected from 12 different users.

1.5.2 Research Design

To convert the natural language query into a structured query form, it has to go through various phases or levels. We have analysed and designed the architecture of Amharic language interface in database for the generation and execution of SQL query by taking the user's input in natural language form (Amharic).The Amharic language interface in database system uses tools such as: java, NetBean and oracle. The various components used in research, algorithm designed and prototype implementation are discussed in chapter 4 and 5.

1.6 Scope and limitation of the Study

The study focused on developing the prototype of natural language interface for database in Amharic language. For query processing in Amharic language that only include the activities for conversion of Amharic query sentences into structured query and retrieving data from the database. However, the research was limited to the development of a prototype Amharic language interface in database since it is impossible to develop a full-fledged system within the given time and resources available for the research. Developing a complete system demands to construct maps of different levels for easy exploration, which in turn requires a long period of time.

1.7 Significance of the Study

The major contribution of the study is to developing Amharic language interface in database to facilitate user access to the data from relational databases without forcing them to have knowledge of formal database language such as SQL. So that this work will have an immense effect on future researches of query processing in database systems in improving their performance by understanding the context of queries. Benefits of the system:

- The user can query using Amharic language without spending time to learn a formal database language such as SQL.

•And at the end of this work, the result will help future researchers that aim to work on developing query processing in database in other language like: Tigrigna, AfanOromo etc.

1.8 Organization of the study

The documentation is organized in the form of the following chapters

Chapter-1 Deals with introduction to Natural Language Processing: It describes an overview of natural language processing in the form of its level such as phonology, morphology, lexical, syntactic and semantic. This chapter also discusses a brief history of Natural Language Processing, Besides, Natural Language Processing applications like information retrieval, information extraction, question answering, summarization, machine translation; dialogue

systems etc. are also discussed. The chapter moreover discusses the aim and objective of the Natural Language Interface to Database (NLIDB), benefits of NLIDB with different interfaces. Objective and scope of research work are also discussed in this chapter

Chapter-2 Deals with the study related to Natural Language Interface to Database: It describes survey related to Natural Language Interface to Database, available systems and commercially available NLP tools.

Chapter-3 Deals with the methodologies of the study: It describes the tools that are used in this IDE – NetBeans is used for developing application which discusses various Java API and library used and for GUI. It also discusses the database tool **oracle**.

Chapter-4 Deals with analysis & Design of Amharic Language Interface in Database: It describes the various phases of Amharic language interface to database based on their input sentence, and mapping into structured query language, executing the same and providing the query result to user in the form of graphical user interface. The chapter also discusses the architecture or model and algorithm for various methodology designed in the work.

Chapter-5 Deals with the prototype implementation and Summary of Findings, It describes the prototype of the proposed system and their results with the database and there Amharic query sentences collected from different end-users.

Chapter-6 Deals with the conclusions and future works of the study: Based on the results of the experiment, the last chapter (chapter six) presents the conclusion and Future Works of the research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The previous chapter discussed the problem addressed by this study and the method used. This chapter provided a review of related systems and the lessons learnt from such systems. In addition, the natural language interface in database (NLIDB) system, advantages and disadvantage, components, architecture and approaches are described in the subsections of this chapter.

2.2 NLIDB Systems

One of the most interesting application areas of Natural Language Processing (NLP) is natural language interface to database systems (NLIDB).A lot of work has been done in the area of NLIDB. Asking questions in natural language is very convenient and easy method of data access, especially for casual users who do not understand complex database query language such as SQL.

The purpose of Natural language Interface to Database System is to accept requests in Amharic or any other natural language and attempts to 'understand' them or we can say that Natural language interfaces to databases (NLIDB) are systems that translate a natural language sentence into a database query. NLIDB is a step towards the development of intelligent database systems (IDBS) to enhance the users in performing flexible querying in databases [24].A complete NLIDB system will benefit us in many ways. Anyone can gather information from the database by using such systems. Additionally, it may change our awareness about the information in a database. Traditionally, people are used to working with a form; their expectations depend heavily on the capabilities of the form. NLIDB makes the entire approach more flexible, therefore will maximize the use of a database.

The area of NLIDB research is still very experimental and systems so far have been limited to small domains, where only certain types of statements can be used. When the systems are scaled up to cover larger domains, it becomes difficult due to vast amount of information that needs to

be incorporated in order to parse statements [28]. Although the earliest research has started since the late sixties, but in Ethiopian context there is no researches conducted in NLIDB.

2.3 Advantages and disadvantages of NLIDB

Advantages of NLIDB

The NLIDB systems allow the user to communicate with database in their native language. The main advantages of Natural language Interface to Database are given below [1]:

- The user that want to retrieve the data or that process the query from database that simply access the data from database their native languages.
- Simple and easy to use: The natural language interface is very simple and easy to use because the end users write the query in their native language
- No need of Training: No special training is required before using the natural language interface. It is highly user friendly and easy to use by the end users.

Disadvantages of NLIDB

- Linguistics coverage is not obvious: Currently all NLIDB systems can understand some subsets of a natural language but it is quite difficult to define these subsets. Even some NLIDB systems can't handle certain query belong to their own subsets. This is not the case of formal language like SQL. Because the formal language coverage is obvious and provide the corresponding answers of any statements that follow the given rules [5].
- Confusion for users: when the NLIDB does not understand the 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. Some NLIDB attempt to solve this problem by providing diagnostic messages, show the reason a question cannot be handled.

2.4 Components of NLIDB

In NLIDB there are two major components can be considered as a classical problem in the field of natural language processing and can be solved in this two stages [23]. Although, the earliest research has started since the late sixties, NLIDB remains as an open research problem.

2.4.1 Linguistic component:

It handles input as natural language, translate it into formal query and generate output in natural language from the result which comes after execution of query [23]. 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.

2.4.2 Database component:

Database component performs traditional database management functions. A lexicon composed of a number of tables that store natural language words and their corresponding mapping to formal objects that are used to create a formal query. These tables can have entries of relations name; attribute names, etc. questions entered in natural language translated into a statement with the help of lexicon. Then a formal query is formed by mapping these tokens into lexicon tables, which is executed and the result in natural language is given to user. Database management and access is performed and the SQL query is executed by the system [23]. 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 [24].

In this study includes these two stages which are linguistic components for user input query tokenization to logical query execution and database components for SQL generation to SQL execution to users.

2.5 Various Approaches Used for Development of NLIDBs

Several researchers applied different techniques to deal with language. Neelu Nihalani et al [6] Mainly introduced or described three types of approaches regarding the development of NLIDB, Those are symbolic, empirical and Connectionist approaches. Regardless of these approaches, the development of complete and suitable model is still difficult for certain applications like speech understanding.

2.5.1 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 [24]. 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 for example in query processing the meaning of the token word is depend on the lexicon dictionary this dictionary is prepare depend on the case study database i.e. employee database . It tries to capture the meaning of the language based on these rules.

2.5.2 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 to developing robust, efficient natural language processing (NLP) systems [24]. 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 [24].

Given the successes of empirical NLP methods, researchers have recently begun to apply learning methods to the construction of information extraction systems [24].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. Majumder, experimented N-gram based language

modeling and claimed to develop language independent approach to IR and Natural Language Processing [24].

2.5.3 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 on 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 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 [24]. 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 [24] system is neutrally inspired system for event modeling and temporal processing at a connectionist level.

2.6 Architecture of NLIDB systems

There are various architectures that are used by different researchers. Researchers have applied in a number of architectures in different cases these are: pattern matching system, syntax-based system, semantic grammar system and intermediate representation language. Which is mainly introduced or described those four architecture by Himani Jain [8], Androutsopoulos, et al [10] and Neelu Nihalani et al [24] .

2.6.1 Pattern matching system

In pattern matching system patters and rules are given and that patterns and rules are fixed. The rules are, if input sentence or word is match with given pattern, the action has been taken and that actions are also mention in the database. But it is for some limited database and to the number of complexities of its pattern [24]. This technique also helps the systems to come up with reasonable answers even if the questions are out of the range for which the systems are designed [4]. The advantage of this system is no parsing and module needed, system can be easily

implemented other systems and it is easy to add or remove features within the system .Consider this table below: [26].

Table: 2.1 pattern matching system

Name	Department	Salary
አለማየሁ አበራ	**አካውንቲንግ**	4,000
ኤልያስ ግዛው	**ኮምፒውተር ሳይንስ**	9,000
ሰውነት አበበ	**ሶሾሎጂ**	5,000

A simple pattern matching system may develop a rule like:
pattern: ... 'department' ... [employee]

answer : SELECT department FROM employee WHERE name=[employee] The above rule specifies that if there is a sentence containing a keyword "department" followed by a employee name, then the related answer is SELECT department FROM employee WHERE name=[employee] .The employee name can be obtained by looking for a certain value in the table. Certainly a pattern matching system is not necessarily to be that simple. Instead of just seeking an exact match with the keyword, the system could also consider additional conditions. For example: a single term in a pattern can consist of several features , such as the stemmed form of the keyword, the part of speech (POS), the synonym, the hypernym, the position in the sentence, and many others. Some systems are working effectively but some would lead to be failed. SANVY is the best example of the pattern matching system [10].

2.6.2 Syntax based system

In syntax-based systems the users question is parsed (i.e. analyzed syntactically) and the resulting parse tree is directly mapped to an expression in some database query language. It uses a grammar that describes the possible syntactic structures of the users' questions. One of the examples of syntax based system is LUNAR [25].

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) [24]. The main

advantage of using syntax based approaches is that they provide detailed information about the structure of a sentence. A parse tree contains a lot of information_about the sentence structure; starting from a single word and its part of speech, how words can be grouped together to form a phrase, how phrases can be grouped together to_form more complex phrases, until a complete sentence is built. Having this information, we can map the semantic meanings to certain production rules (or nodes in a parse tree).

Syntax-based systems use a grammar that describes the possible syntactic structures of the user's questions. The following example shows an over-simplistic grammar in a Lunar-like system [10].

S → NP VP
NP → Det N
Det → "what" | "which"| "the"
N → "rock" | "specimen" | "magnesium" | "radiation" | "light"
VP → V N
V → "contains" | "emits"

The grammar above says that a sentence (S) consists of a noun phrase (NP) followed by a verb phrase (VP), that a noun phrase consists of a determiner (Det) followed by a noun (N), that a determiner may be "what" or "which", etc. Using this grammar, a NLIDB could figure out that the syntactic structure of the question "which rock contains magnesium" is as shown in the parse tree of figure 2.1

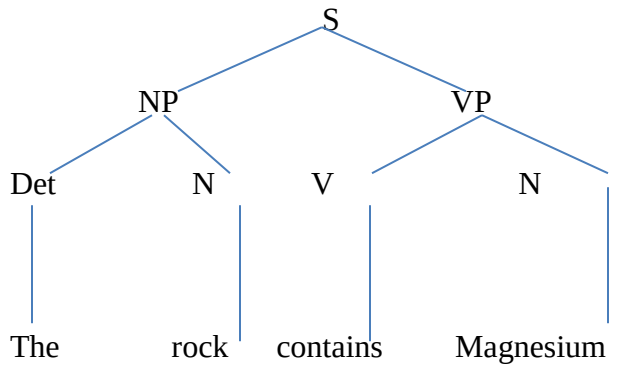

Figure 2.1: Parse tree in a syntax-based system

2.6.3 Semantic grammar system

A semantic grammar NLIDB system is similar to a syntax-based NLIDB system. The user inputs will be translated into semantic tree by semantic analysis, then it will be translated into SQL and a typical example is PLANES [23] and LADDER [23].example of semantic grammar [23] shown below: **example of the semantic grammar**

S -> Specimen question | Spacecraft question
Specimen question -> Specimen Emits info | Specimen Contains info
Specimen -> “which rock” | “which specimen”
Emits info -> “emits” Radiation’, Radiation -> “radiation” | “light”
Contains info -> “contains” Substance, Substance -> “magnesium” | “calcium”

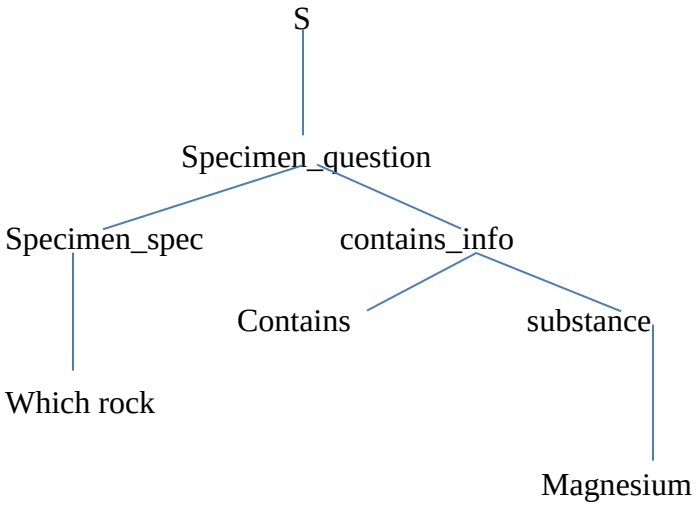

Figure 2.2: Semantic Tree

The basic idea of a semantic grammar system is to simplify the parse tree as much as possible, by removing unnecessary nodes or combining some nodes together. Based on this idea, the semantic grammar system can better reflect the semantic representation without having complex parse tree structures [26]. Therefore, a production rule in a semantic grammar system does not necessarily correspond to the general syntactic concepts.

Compared to syntax analysis, semantic analysis uses semantic domain instead of grammar concept. The grammar’s categories (i.e. the non-leaf nodes that will appear in the parse tree) do not necessarily correspond to syntactic concepts.

The main drawback of semantic grammar approach is that it requires some prior- knowledge of the elements in the domain, therefore making it difficult to port to other domains. In addition, a

parse tree in a semantic grammar system has specific structures and unique node labels, which could hardly be useful for other applications. Regardless, there are on-going attempts to automatically build the grammar rules by obtaining the prior-knowledge based on user interaction or by automatically extracting it from a corpus.

2.6.4 Intermediate Representation Languages

Due to the difficulties of directly translating a sentence into a general database query languages using a syntax based approach, the intermediate representation systems were proposed. The idea is to map a sentence into a logical query language first, and then further translate this logical query language into a general database query language, such as SQL. In the process there can be more than one intermediate meaning representation language [24]. Figure 2.3 shows a possible architecture of an intermediate representation language system.

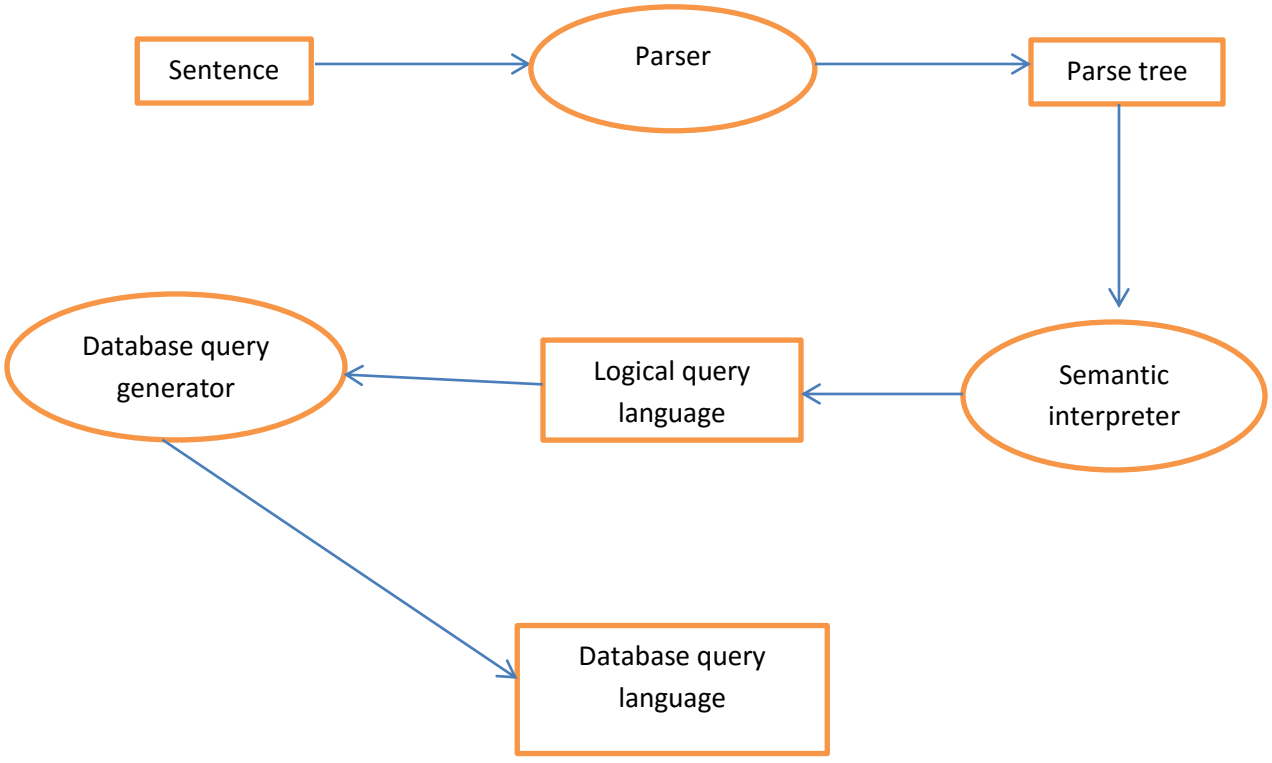

Figure 2.3 Intermediate Representation Language possible Architecture

The transformation from a logical query language to a database query language does not need to be made in one step. As an example, an NLIDB system developed at the University of Essex uses a multi-stage transformation process [23]. The first logic query is in the form of λ-calculus,

which is then transformed to a first-order predicate logic, universal domain relational calculus, domain relational calculus, tuple relational calculus, and finally SQL.

In the intermediate representation language approach, the system can be divided into two parts. One part starts from a sentence up to the generation of a logical query. The other part starts from a logical query until the generation of a database query. In the part one, the use of logic query languages makes it possible to add reasoning capabilities to the system by embedding the reasoning part inside a logic statement. In addition, because the logic query languages are independent from the database, it can be ported to different database query languages as well as to other domains, such as expert systems and operating systems [23].

Most current NLIDBs first transform the natural language question into an intermediate logical query, expressed in some internal meaning representation language. One typical example of this type is MASQUE/SQL [23].

2.7 Existing NLIDB systems

There are large numbers of developments in the field of natural language interface in databases has been done. Asking questions in natural language is very convenient and easy method of data access, especially for casual users who do not understand complicated database query language such as SQL.

2.7.1 Recently developed NLIDBS in English language

N nihalani *et al*, [28]*:* developed an intelligent interface for relational databases. The system was developed using JAVA with MySQL and MS-Access as databases and tested using supplier-part database and North wind database of SQL server 7.0. A group of five students was asked to formulate the queries in English for the two databases. It has one main characteristic on that interface it was domain-independence, and it has also composed of two main modules, which are: a pre-processor and a run time processor.

Neelu Nihalani et al [25] discussed the concept of Natural Language Interface to Database using Semantic Matching. The system was developed by using the MYSQL server and MS access in database and system was tested for Northwind Database and compared with MS English Query

product. The main functionality is based on semantics and rules. It is composed of two modules: a pre-processor and run time processor.

2.7.2 Natural language interfaces other than English language

Himani Jain [8] developed Hindi Language Interface to Database. Hindi Shallow Parser which uses Shakti Standard Format is considered for parsing a sentence. The system was developed in Java as front end and MySQL in database. For testing of the developed system, employee database and 50 Hindi input sentences is used. The system is directly maps user keywords to database entity names and the performances of the system is measured by number of query sentences.

Amardeep Kaur [27] presented the design and implementation of Punjabi language interface to agriculture database. The database consists of three tables those are farmer, crop and sale tables but the author is not considered joining of these table and consider limited words. The system uses MS Access database and Graphical User Interface with NetBeans platform. The system accepts input in specific template and test the result based on the three types of query sentences which include selection of whole table, selection of certain columns of a table and selection of certain rows of the table.

Ashish Kumar et al, [1] developed Hindi language interface to database system. The system also developed using their semantic matching. Their system architecture had three phase that is 1.tokinization 2.semantic matching and SQL generator and 3. SQL query execution. For testing of the developed system, student database is used. Do not improve the linguistic coverage.

Rashid Ahmad et al,[32] is developed An algorithm that efficiently maps a natural language query, entered in Urdu, the algorithm has been implemented in Visual C#.NET and tested on a database containing Student Information System and Employee Information System. The interface that is proposed for this system is based on formal semantics and use AV (attribute value) mapping algorithm to deal with the tokens. This algorithm maps the particular value of token into corresponding attribute token. This converts the query from Urdu language to SQL query. There is about 85 % accuracy in results produces by this system.

Rajender Kumar et al, [2] Developed Domain-independent Hindi language interface in relational database. To make the system was domain independent that use the language processing module and database module. The authors discuss more about language processing module (query analyzer, POS tagger, morphological analyzer and semantic analyzer), Graphical User Interface database module (Domain Identifier, SQL Query Generator and SQL executor). Indian Shallow Parser which uses for tokenizing and parsing a Hindi sentence. It focuses for the system separates the domain-oriented knowledge from the linguistic data. The system was developed in MYSQL for database and JAVA swings as front end. MYSQL is connected to the JAVA using JDBC (Java database connectivity). For testing of the developed system, Employee Payroll, Railway Enquiry and Student Information database and 80 queries are used.

Nikita Bhati et al, [29] discussed Analysis of Hindi language Graphical user Interface. Hindi Shallow Parser which uses Shakti Standard Format is considered for parsing and tokenizing a sentence. To design the Hindi language interface to databases that includes the description of architecture of system and structure of EMPLOYEE database used in the system. Their architecture was focus on five phases those are:

- Parse and tokenize their input sentence using Hindi Shallow Parser.
- find the English word corresponding to Hindi tokens from a dictionary
- Map input Hindi query with database query.
- Generate SQL query with the help of database query generator and underlying DBMS. And
- Execute the query and give the response to user.

The system was developed in MYSQL as database and HTML and CSS for as front end. MYSQL is connected with ASP.net using mysqldata.dll. That system was handled retrieving updating and deleting the queries. For testing the developed system employee database and 50 Hindi input sentences used.

Looking at the various NLIDB systems developed by large number of researcher in this field, we have designed and developed Amharic Language Interface to Database (ALIDB) system that partially fulfills the knowledge and show the possibility of Amharic query processing in database. Some of the silent features of ALIDB are: can enter the Amharic query sentence in to

the form, direct mapping to table name and column name with user words, semantic of words are considered, lexicon analysis is done, system that only consider converting the Amharic query sentences into structure query sentences i.e. converting into SQL and retrieve the data from datasets or that consider select statements.

CHAPTER 3

Methodology of the study

3.1 Introduction

To realize the main goal of this study, different methods and techniques are used**.** The main procedure or methods of the study are described below:

- Tokenize the input provided in Amharic query sentences.
- Identify feature of queries
- Extract the table, column, condition information from input Amharic sentences with the help of mapping of token with the database values and their lexicon or systems dictionaries used.
- Generating SQL query mapping of input query with the value stored in database tables
- Execute SQL query and give output to user in Amharic language

3.2 Data Collection

The researches gathered data from academic employee in excel form and normalized the collected data in the form of three tables; those are **employee table**, **department table** and **employee on education table**. 60 Amharic query sentences were collected by telling about the database and ,collect from 12 users which is 6 for who have the knowledge of structural query language and 6 for who are not aware of structural query language and testing from different end users, those users are:

1. The end users who have the knowledge of structural query language for 30 Amharic query sentences and,
2. The end users who are not aware about database or structural query languages for 30 Amharic query sentences.

3.3 Related tools and technologies

In this study we use different tools to develop our system those are java for platform NetBean, database that is oracle and SQL developer as graphical interface of oracle database for the reason is discussed on below.

3.3.1 Java

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. As of 2015, Java is one of the most popular programming languages in use and a safe language to allow within the database. Today, Java not only permeates the Internet, but also it is the invisible force behind many of the applications and devices that power our day-to-day lives. Java is everywhere. Java Applets are also used for improved and secured environment while browsing on the World Wide Web using desktop computers. The Java platform is not specific to any one operating system but has an execution engine (called virtual machine) and a compiler with a set of libraries, which can be executed in various hardware and software systems [30]. Java is a unique technology that runs across all platforms. The library includes java card, java macro edition, java standard edition and java enterprise edition.

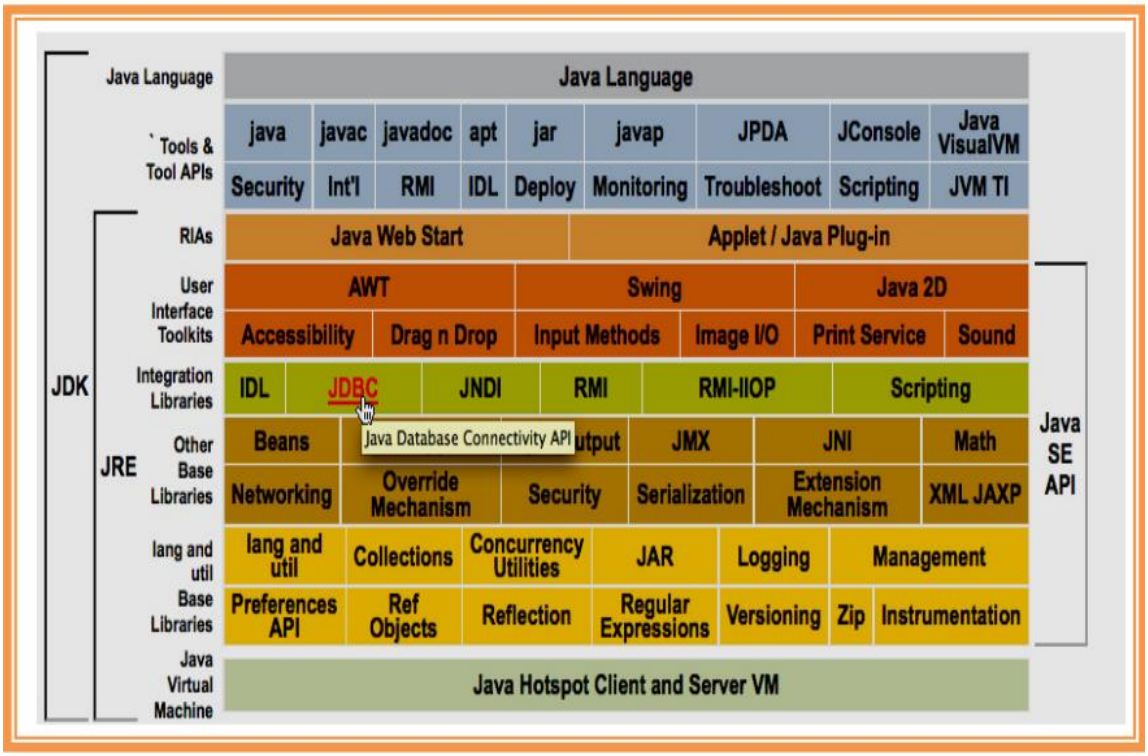

Figure3.1. JSE platform [30]

The Java platform consists of several programs, each of which provides a portion of its overall capabilities. For example, a Java compiler converts Java source code into Java bytecode (intermediate language) for Java Virtual Machine (JVM). The Java Runtime Environment (JRE), complementing the JVM, with Just-In-Time (JIT) compiler converts intermediate bytecode into native code. The working strategy of JAVA is shown in the figure 3.2 an executable set of libraries are also part of the Java Platform. The essential components in this platform are Java language compiler, the libraries and the runtime environment in which Java intermediate bytecode executes according to the rules laid out in the virtual machine specification [42].

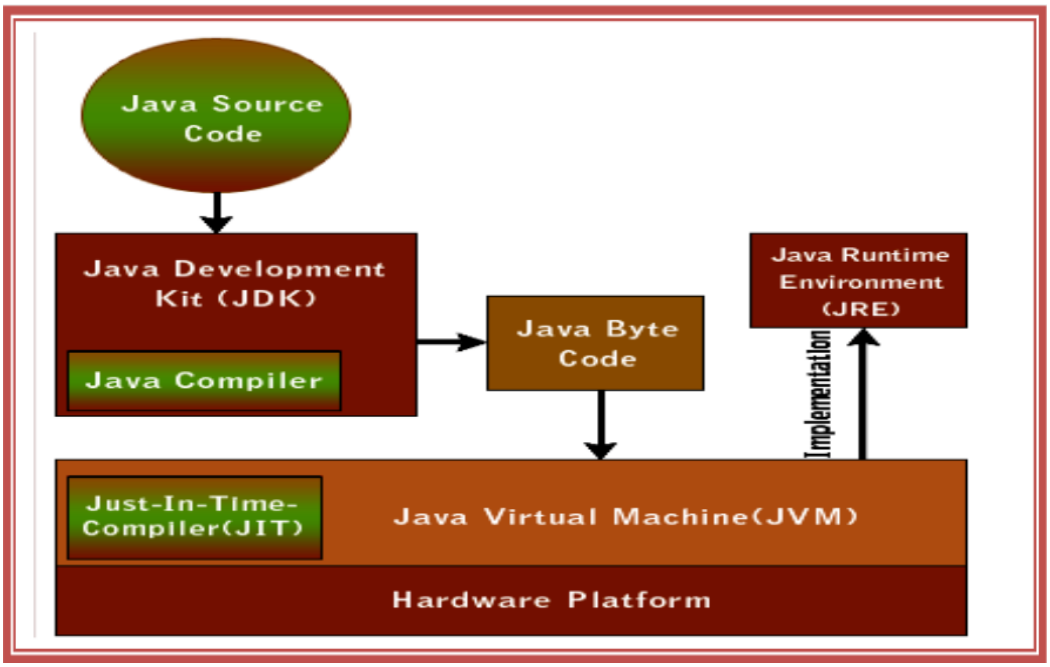

Figure 3.2.how java works (source: www.google.images.com)

3.3.1.1 Java development kit (JDK)

The JDK has as its primary components as a collection of programming tools, some components of the JDK are:

- applet viewer – this tool can be used to run and debug Java applets without a web browser
- apt – the annotation-processing tool
- extcheck – a utility which can detect JAR-file conflicts
- Idlj – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- Jabswitch – the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- Java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
- javac – the Java compiler, which converts source code into Java bytecode

- javadoc – the documentation generator, which automatically generates documentation from source code comments
- Jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- Javafxpackager – tool to package and sign JavaFX applications etc.

3.3.1.2 Java APIs

The Java API is the set of classes included with the Java Development Environment (JDE). These classes are written using the Java language and run on the java virtual machine (JVM). The Java API includes everything from collection classes to GUI classes. For example, if we include the statement in the Java program "import javax.swing.∗", then the swing API provide access to graphical component used to build Graphical User Interface (GUI).

3.3.1.3 Java JAR Files

JAVA JAR (Java Archives) is a package format typically used to aggregate many Java class files and associated metadata and resources such as text, images, etc. JAR files are archive files built in ZIP file format and have jar file extension. A JAR file allows the Java runtime to efficiently deploy a set of classes and their associated resources.in the present work for implementation of the prototype use the following jar files:

- JDK 1.8.0-05
- Jre8

3.3.2 NetBeans

NetBeans Integrated Development Environment (IDE) is a free, open source and has a worldwide community of users and developers, it is used to easily and quickly develop desktop, mobile, and web applications with Java, HTML, PHP, C/C++, and more. The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. NetBeans IDE is written in java and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM. Features of the NetBeans ide platform are follows:

- User Interface Management (e.g. menus and toolbars)
- User Setting Management
- Storage Management (saving and loading any kind of data)
- Windows Management
- Wizard Framework (supports step-by-step dialogues)
- NetBeans Visual Library
- Integrated development tools, etc.

3.3.3 DATABASE

3.3.3.1 Oracle

Oracle is open source relational database management system software in Oracle Corporation. It is a full featured database engine that has successfully passed stringent security tests and has excelled in performance benchmarks [31]. With built-in support for PL/SQL and Java, developers can build complex stored procedures, functions, and triggers that are stored and executed within the database. Views, sub queries, bi-directional replication, clustering, role-based security, and native support for Internet-based computing with the included Apache server and XML tools are just a few examples of how Oracle Database is well suited for managing mission-critical data for any size organization [31].

Key Features of Oracle which are selected from other databases are:

- It works on any platforms
- It supports or handles very large database
- It is also have high performance
- Oracle offers inline views, role based security, advanced replication, etc
- Oracle supports the creation of programs that are embedded within the database by way of a procedural language and can be executed independently or triggered by certain events.
- Oracle is better for large scale deployments as it has extensive capabilities.
- Most editions of oracle have high licensing costs, which are needed in order to use the software. Oracle Express is freely available.
- It Support Pl-SQL, in addition to SQL

- It provides security features
- It requires username, password, and profile validation at the time of logging
- It Database supports the use of temporary tables for an individual session, or global to all users.
- It has Tablespace, Role management, snapshots, synonym and packages

We have used SQL developer as GUI which is free and open source community edition in Oracle Corporation.

3.3.3.2 SQL developer

SQL developer is a Graphical User Interface (GUI) tool for oracle relational database management system. It was developed by Chuck Murray. It is available both as a freely available version and also as paid version. Figure 3.3 represents the user interface of SQL developer.

Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Oracle Database in both traditional and Cloud deployments. SQL Developer offers complete end-to-end development of PL/SQL applications, a worksheet for running queries and scripts, a DBA console for managing the database, a reports interface, a complete data modeling solution, and a migration platform for moving 3rd party databases to Oracle.

Features of oracle SQL developer are:

- One of SQL Developer's most popular features has undergone a significant upgrade. Users can quickly define and recall delimited or excels files to be imported to a new or existing oracle table. Data preview and validation is provided for each column, as well as 'best guess' data type and date format mask mapping. This process can now be automated via the SQL Developer command line interface (SDCLI) 'Import' command i.e. .import data from other database
- offers multi-cursor editing, remembering frequently used files and directories in open and save file dialogs, and verbose SQL recording for all statements sent to the database
- It supports data manipulation i.e. (select, update and delete) using spreadsheet like interface. It has an editor with syntax highlighting and various automatic formatting options.

- Supports Oracle NoSQL database. Support includes Developer and DBA roles, VLite Store deployment and seeding, Store Reporting, Store Statistics and Administration. Etc.

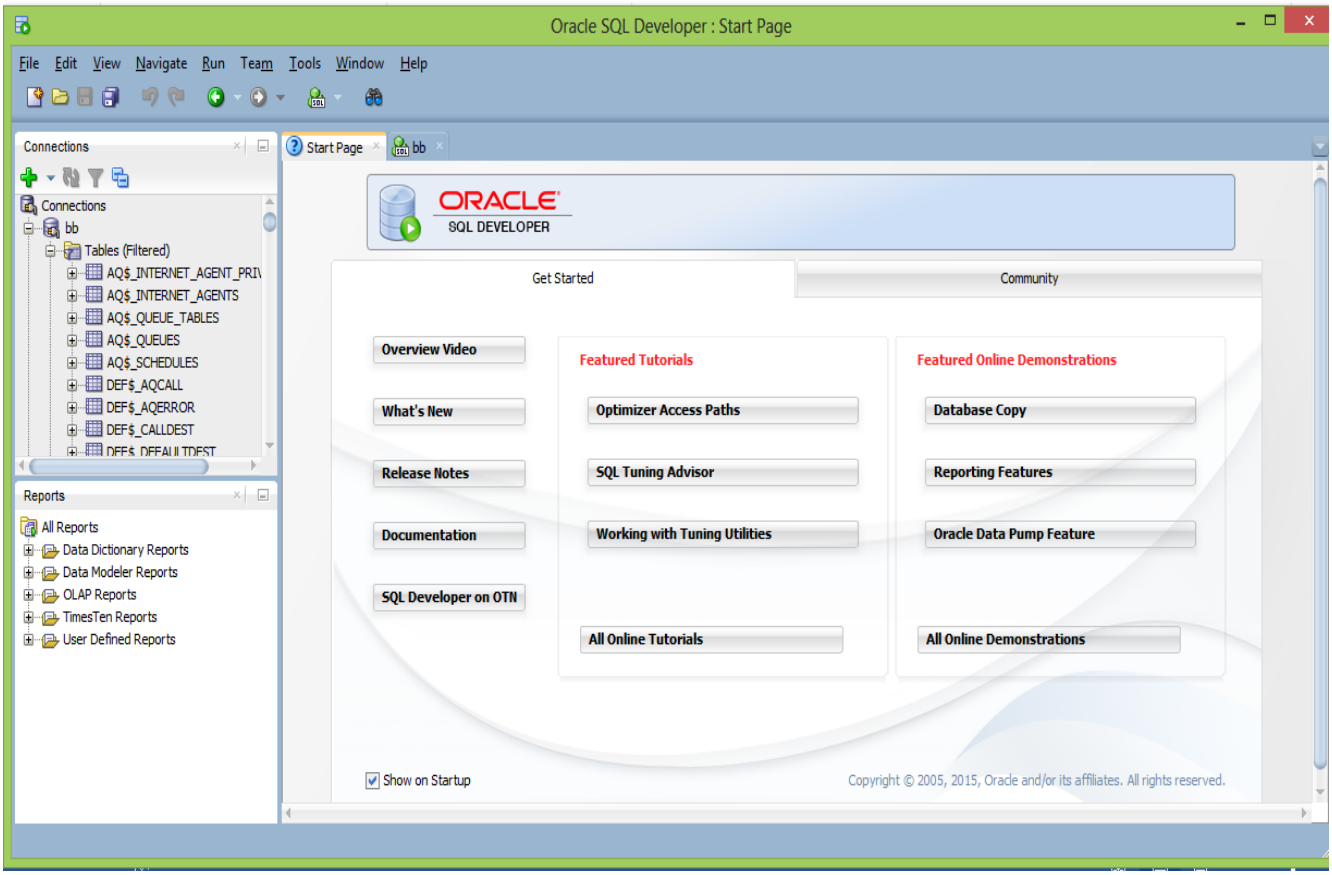

Figure 3.3 SQL Developer user interface

3.4 Summary

This chapter generally discussed about data collection technique, the concepts of related tools i.e. Java programming language, its features, Jar files, Java API, Netbean etc. Finally, it discusses about the concept of database i.e. DBMS, RDBMS, Oracle database and SQL developer.

Chapter-4

Analysis & Design of Amharic Language Interface in Database

4.1. Introduction

The computer era has begun since late 1970, bringing awareness into different working groups of society regarding the usages and benefits of computers. Not only industry, but also educational institutes, sectors like service, manufacturing, etc. and also started using computers to store, process and update the information or data. The huge amount of data stored in repository is called database. In order to query or retrieve the data or information from the database by a beginner, natural language that will run correct and precise information without knowing the depth of structured query language was the need of the time. The idea of using natural language in the context of database prompted the need for the development of Amharic language interface in database.

4.2 Architecture of Amharic language interface to database (ALIDB)

We have proposed the architecture of the NLIDB system which shows the possibility of Amharic query processing in database. Figure 4.1 represents the architecture of the ALIDB. The System has two major components: (a) Linguistic Component, and (b) Database Component. The Linguistic Component translates the natural language input to an expression of Intermediate Query (IQ), which is subsequently passed to Database Component for generation of Structured Query Language (SQL) statement. The resulting SQL statement is then executed by relational database management system. Various phase of processing the system can be described as follows.

4.2.1 Graphical User Interface

Our system has a user-friendly graphical user interface which is; user can easily enter their query in Amharic language and after processing and executing the query, results are displayed to the user.

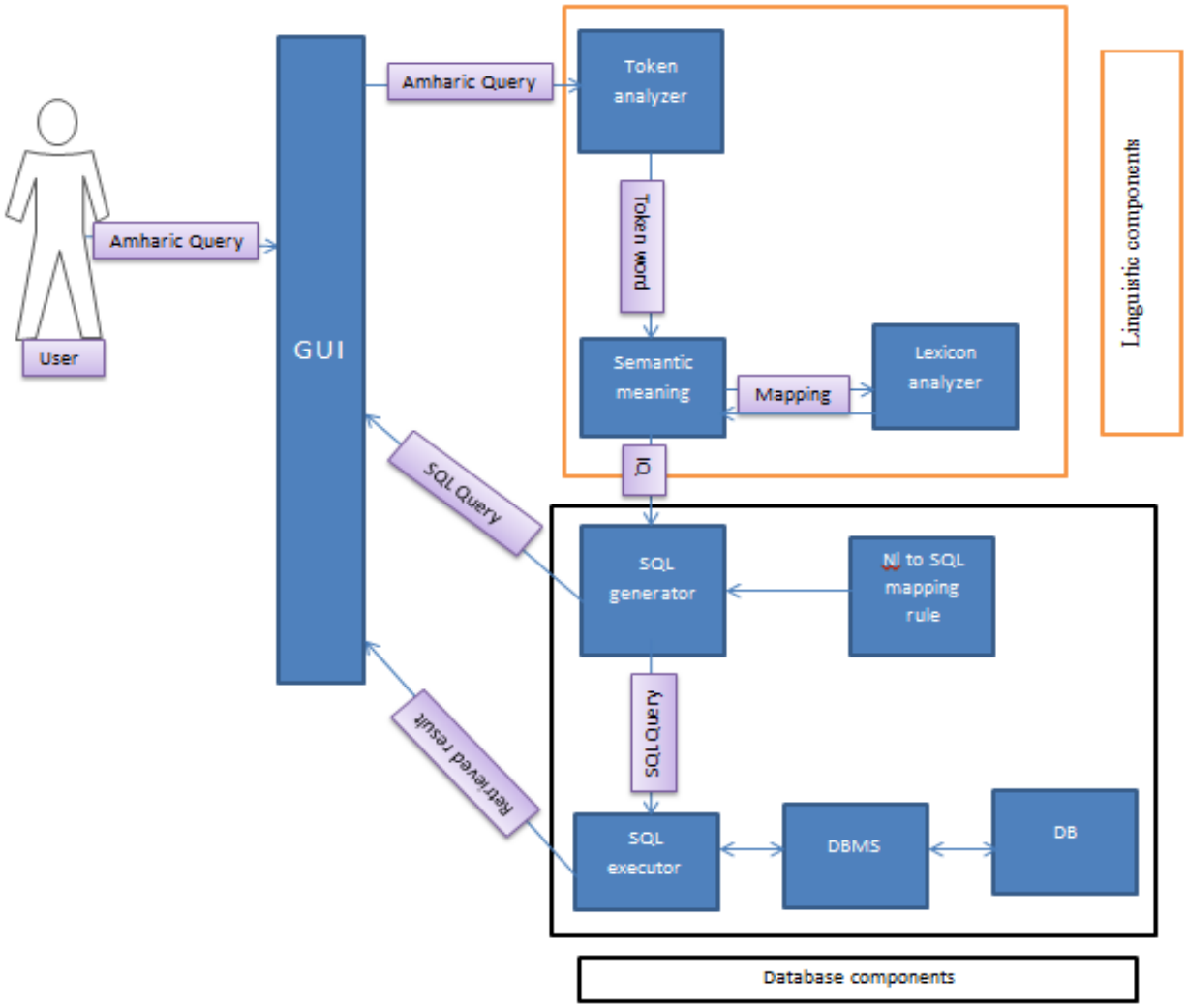

Figure 4.1 Architecture of Amharic language interface in database

4.2.2. Linguistic components

The Linguistic Component studies about language. 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. It has mainly three aspects to study which includes language form (syntax), language meaning and language context [42] which consist of Lexical analyses, token analyses and semantic meaning analyses.

4.2.2.1. Token Analyzer

It splits the input string into a sequence of simple units called tokens that is treated as a single logical unit. To make a sentence able to be processed by the computer, it is necessary to divide it into chunks or tokens to understand its meaning and structure [32]. System tokenizes that sentence and produce tokens that are to be searched in the lexicon for mapping. In Amharic, most of the words are separated by space as it is in English except some compound words and the words that are separate in English comma',' in Amharic called ne-te-la serze (፣).There are different token representation rules used by our prototype some of the rules are:

1. Core token type:
    - If the sentences contain only single words and strings without any compound words and each word is separated with spaces and 'ne-te-la serze' (፣) in English comma. e.g. **የሰራተኞችን ስም፣ጾታ፣ደመወዝ እና ሃላፊነት አውጣልኝ**
2. Numeric Token:
    - If token contains digits only or digits that are separated by forward slash or in Amharic ezi bar (/), then it is a numeric token.eg.23000 and 122/05
3. Sentence ending marker:
    - Sentence can be terminated with 'arate nteb' (::) or question mark (?) or an exclamation mark (!).
4. Special value token:
    - If token contains numeric data with, forward slash or in Amharic 'ezi bar' (/) followed by numeric data then, it is a value token. For example, 112/05 is a value token.
    - If token contain numeric data eg.2400.
5. Noise token:
    - A sentence contains words which has no importance in the context which is to be removed. e.g. **እና** (which combine column name), **የሁሉንም**, **ዝርዝር** etc.
6. Abbreviated token:
    - If the sentence contains a string with a forward slash or in Amharic ezi bar (/) followed by word or string, it is considered as a single token. For example **ት/ክፍል** etc.

4.2.2.2. Lexical analyzer:

Lexicon is system dictionary that store all the Amharic tokens with their corresponding English word and their SQL words, and type of token whether it is table name, column name, condition, conditional words, SQL command or something else. The lexicon dictionary prepared in different table forms that is **table_name table** that handle all the table names contain in the database, **column_name table** that handle all name of columns of all available tables contains in the database tables**, condition table** that handle where condition**, conditional word table** that handle all conditional words that is >,<,=,<=.>= etc.**, SQL word table** that handle the SQL statement words i.e. Select etc. Sample example of lexicon dictionary in **conditional word** handler and **where condition**

E.g. 1: Conditional word handler

Table 4.1 operator word handler

Amharic Token word	Mapped word
በታች	<
ያነሰ	<
ያነሱ	<
ያነሱትን	<
በላይ	>
የበለጠ	>
የበለጡ	>
እና ከዚያ	=

E.g. 2: Where Condition handler

Table4.2. Where condition handler

Token word	Mapped word
ያላቸው	Where

የነበረ	Where
የሆኑ	Where
የሆኑት	Where
የሆኑትን	Where

4.2.2.3. Semantic meaning:

Semantics is the study of meaning in language. It can be applied to entire texts or to single word. It is about the manner in which lexical meaning is combined morphologically and syntactically to form the meaning of a sentence. Mostly, this is regular, productive and rule-governed. Within these steps that finds the meaning of words or token words from lexicon and match their corresponding English words.

Lexicon store all the Amharic tokens, their corresponding English word and type of token whether it is table name, column name, any value, operation, command or something else. Tokens which we extracted in above step are matched with the tokens stored in lexicon one by one. If it matches then its corresponding English word is saved along with its type. This is the most important phase. All the useless tokens discarded in this phase only useful tokens are stored. After this step we will have with table name, field name (columns name), conditions, and conditional words. This will be further used to make SQL query.

The output of the semantic meaning interpretation module gives a logical expression of the words in the lexicon, to generate the logical query. The approach used here is an Intermediate Query (IQ) representation which can express the meaning of user input in terms of high level concepts, independent of database structure.

E.g. If the users input in the natural language query as: **የሰራተኞችን ስም ዝርዝር አውጣልኝ** :: it tokenize and mapping from lexicon as:

[የሰራተኞችን] employee

[ስም] emp_name

[ዝርዝር] noise word

[አውጣልኝ] select

[::] discard

From the above example, the semantic meaning matches the Amharic token with their corresponding English words or mapped words from the lexicon dictionary and identifies their table name, column name etc. The word ‘ዝርዝር’ or ‘zi re zi re’ is the noise word that has no meaning in the lexicon or system dictionary which described in the above section in token representation rules.

Intermediate Query (IQ): It is the logical query interpreter which is the midway of the semantic matching and the SQL generator or transitional of logical query to the SQL query.

4.2.3 Database components

The database component consists of SQL Generation SQL Execution and database management system (DBMS). The SQL Generation component takes an Intermediate Query (IQ) as an input from Linguistic component and generates an equivalent SQL query as output. The SQL Execution component executes the generated SQL query and displays appropriate query output or message on GUI to user.

4.2.3.1. SQL Generation

The task SQL Query Generator is to map the elements of the logical query to the corresponding elements of the used databases. The query generator uses four routines in these systems, each of which manipulates only one specific part of the query. The first routine the part query that corresponds to the appropriate DML command with the attribute’s names (SELECT). The second routine part of the query that would mapped to a column name if there is specified if not use all or * clause, the third routine mapped table’s name or a group of tables names to construct the FROM clause and the forth routine selects the part of the query that would be mapped to the WHERE clause (condition).

The steps to generate SQL query are: (i) to analyze the user words from an input sentence (ii) to recognize the specific patterns of database query language (iii) to determine the analysis of answer and (iv) by using an IQ, the system generates structured query. I.e.IQ that handle the

column name, table name, where clause, values and operators and generate the SQL query e.g. **የሁሉንም ሰራተኛ ስም፣የቅጥር ቀን እና ደመወዝ አውጣልኝ።።**

ሰራተኛ Table name **employee**

ስም Column name **Emp_name**

የቅጥር ቀን column name **Hire_date**

ደመወዝ Column name **Salary**

አውጣልኝ SQL word **Select** by using is that generate SQL query that is, Select column name from table name: Select Emp_name, Hire_date, Salary from employee.

4.2.3.2. SQL executor

The task of query executor is to map the generated SQL query to database management system and retrieve the relevant answer by connecting to the appropriate database tool. We have used an Oracle relational database for query execution. The SQL query which is executed by relational database giving the following kinds of responses:

- The retrieved database tuples contain answers and are stored in file rather than displaying directly in grid format as specified by the SQL database output.
- When the system is not able to search the particular tuples, then the appropriate response is sent to the user. For example, no record found.
- When the system may not understand the question, then no tuples will be retrieved and system may send appropriate response to the user. For example, the query cannot be generated or no domain found.

4.2.3.3. DBMS

The purpose of this system is to get the correct result from the database. It executes the query on the database and produces the results required by the user.

4.3 Algorithm for ALIDB

Input: user inter Amharic query sentence in GUI

Output: execution of SQL query with appropriate out or message on GUI

Step1. Tokenize input sentences from GUI

DO while to tokenize the sentences

Use the rule the word separated by ne te la serz, arat netib question mark and exclamation mark and store all token word into lsttokonized, and

If token word is table name

add on lstbl

If else token word is column name

add on lscol

If else token word is where

add on lswhere

If else token word is select

add on lsselect

If else token word is wherecolumnname

add on lswherecolumnname

Else add on noise

Special value token

If the token is numeric it is value token

Add on lsnumber

Else

Add on lscharactor

Step 2.handle table name from step 1

if number of table is one

table =lstbl1

else if number of table is two

table=tbl1 inner join lstbl2 on lstbl2.fk_lstbl2=lstbl1.pk_lstbl1

else

table=lstbl3 inner join lstbl2 on lstbl3.fk=lstbl2.pk inner join lstbl1 on lstbl2.fk=lstbl1.pk

End if

Step 3.handle column name from step 2

If column name is one

Column=lscol1

Else if column name is more than one

Column = lscol1, lscol2, … lscoln

Else

Column =”*”

End if

Step 4 handle SELECT keyword from step 1

Step 5 handle WHERE keyword from step 1

If where keyword is found in step1

Handle wherecolumn, operator and value from step 1

If wherecolumn =(salary or age or id or hire date)

Value =lsnumber

Else

Value =lscharacter

SQL statement = select column from table where wherecolumn operator value

Else

SQL statement = select column from table

Step 6 If SQL statement successfully executed then

If record exists then

Display SQL statement output on GUI.

Else

Display the appropriate message on GUI

(say, record not found).

End If

End If

Step 7: Exit

Chapter 5

Prototype implementation and Summary of Findings

5.1. ALIDB System

We have developed a prototype of Amharic Language Interface to Database (ALIDB) system using a Graphical User Interface (GUI) to convert unstructured query sentences to structure one and retrieving the result from database system. Within this system there must be some requirement specifications those are: system and software specifications. In system requirement specification:

1. Amharic query sentences or unstructured Amharic sentences and

2. Employee databases.

In software requirement and specification:

1. NetBeans 8.0.2 IDE which supports JDK1.8.0-05 and Jre8 and

2. Oracle database with GUI interface i.e. SQLDeveloper 4.0.

5.1.1 User Interface

Figure 5.1 shows the original display of the system which acts as GUI in which the user will input a query in Amharic language and the system will display the SQL query and its corresponding output.

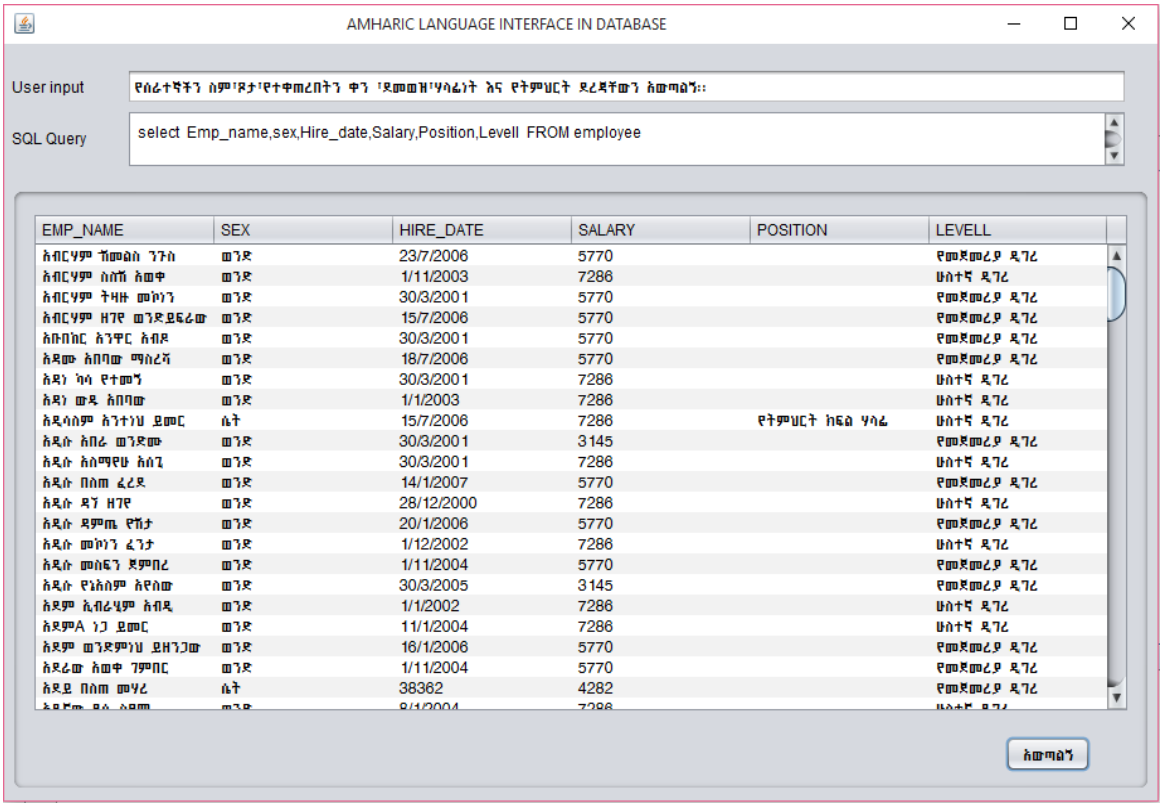

Figure 5.1 GUI of Amharic language query processing

5.1.2 Amharic language query supported in ALIDB

There are three types of queries which are supported in the system. Those are 1.List queries without condition, 2.conditional queries i.e. with single condition and more than one condition and 3. Join Queries.

5.1.2.1. List queries without condition

These types of queries require basic information to be selected from the table which specified or listed in the database. The query may be paraphrase in different form such as: **የሰራተኞችን ስም እና ደመወዝ አውጣልኝ።፣ የሰራተኛ ሥም እና ወርሃዊ ክፍያ አውጣልኝ።፣የሰራተኞችን ስማቸውን እና ደመወዛቸውን አውጣልኝ።፣ የሰራተኛዎችን ስም እና ደሞዝ አውጣልኝ።፣ የሰራተኛውን ስም እና ወርሃዊ ክፍያ አውጣልኝ።** etc. and figure 5.2 shows the prototype of select or list queries without condition.

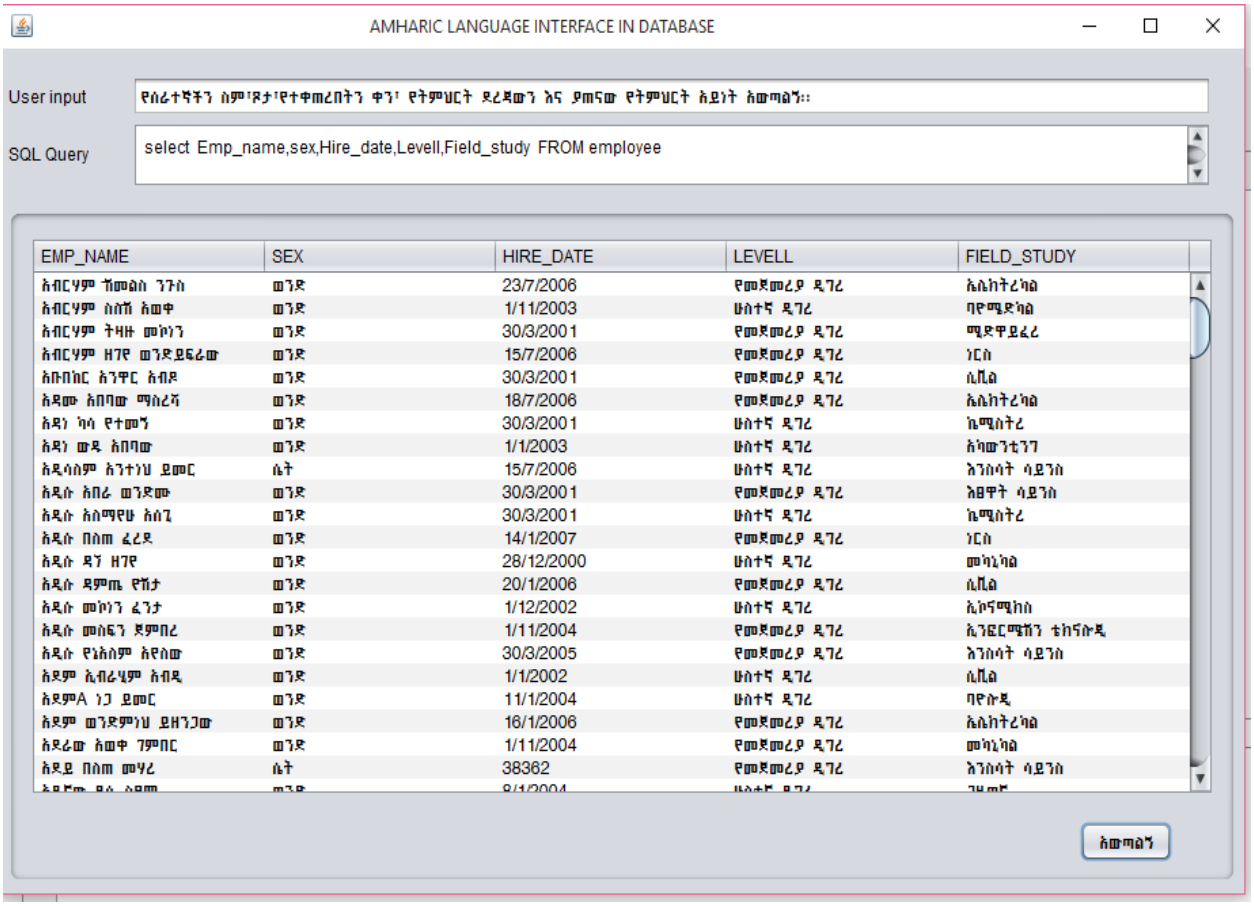

Figure 5.2 list query screenshot

5.1.2.2. Conditional queries

These types of queries require information to be listed from table which is specified or listed into the database based on specified condition. The condition can be single or composite conditions which are categorized as: (i) single query with a single condition as shown in figure 5.3 and (ii) single query with composite conditions. But only single query with a single condition includes in these study.

5.1.2.2.1. Single query with single condition

This type of query is requiring information to be selected from the table which is specified in the database. In this type of query have single condition or where condition with single query. The query may be paraphrase in different form such as: **የአካውንቲንግ ትምህርት ክፍል ሰራተኛ የሆኑትን ስም አውጣልኝ።, የአካውንቲንግ ትምህርት ክፍል ሰራተኞች የሆኑትን ሥም አውጣልኝ።ደመወዛቸው ከ**

3500 በላይ የሆኑ የሰራተኞችን ስም አውጣልኝ።, ወርሃዊ ክፍያቸው ከ 3500 በላይ የሆኑ የሰራተኞችን ስም አውጣልኝ።, ጾታቸው ወንድ የሆኑ ሰራተኞችን ስም እና የመታወቂያ ቁጥራቸውን አውጣልኝ።,ወንድ ሰራተኞችን ስም እና መስያ ቁጥራቸውን አውጣልኝ።። etc. and figure 5.3 shows the prototype of select queries with single condition.

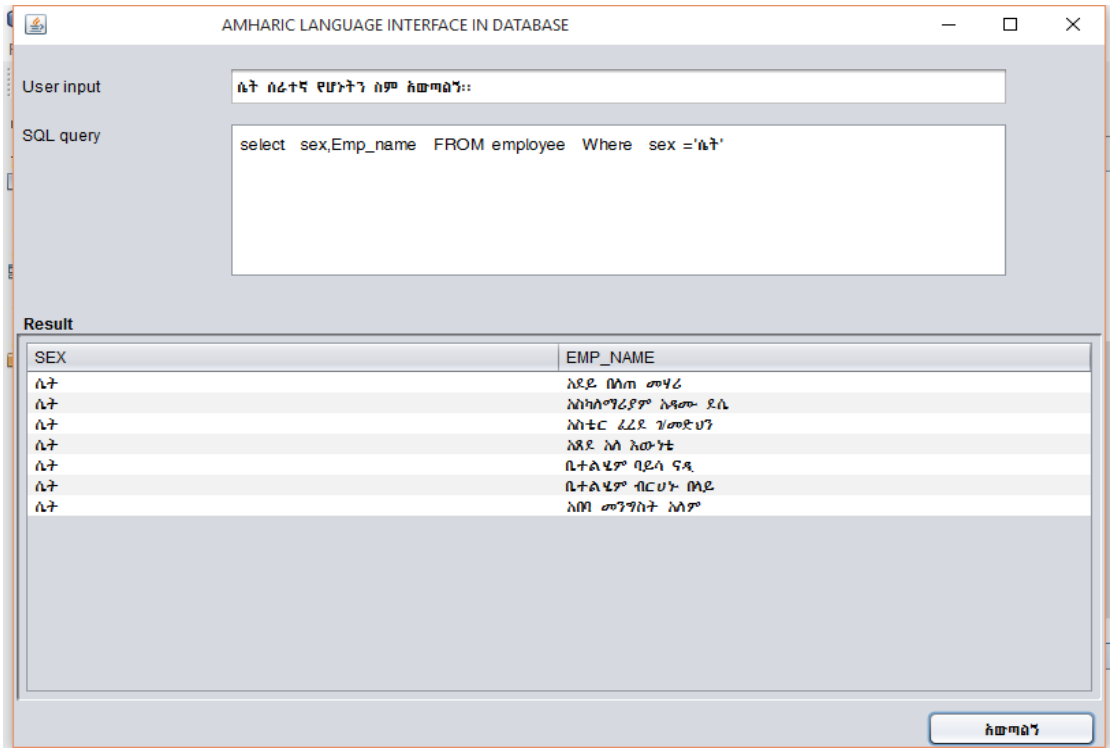

Figure 5.3.Single condition query screenshot

5.1.2.3. Join Queries:

These types of queries include joining of two or three tables and retrieving the relevant data from the database. The different paraphrases of such queries are: የሰራተኞችን ስም፣ጾታ፣የተቀጠረበትን ቀን ፣ትምህርት ክፍል፣የሚማርበትን ዩኒቨርሲቲ እና የትምህርት ደረጃቸውን አውጣልኝ።፣የሰራተኛዎችን ሥም፣ፃታ፣የቅጥር ቀን ፣ትምህርት ክፍል፣የሚማርበትን ዩኒቨርሲቲ እና የትምህርት ደረጃቸውን አሳየኝ።፣ የሰራተኞችን ስም ዝርዝር እና ትምህርት ክፍላቸውን አውጣልኝ።, ሰራተኞች ስራ የጀመሩበት ቀን፣ ስም፣ጾታ፣ዲፓርትመንት እና የትምህርት ደረጃቸውን አውጣልኝ።, የሰራተኞችን ስም ዝርዝር ከነትምህርት ክፍላቸው አውጣልኝ።, የትምህርት ክፍል ሃላፊዎች የሆኑትን ስም እና ትምህርት ክፍል አውጣልኝ።, አዲስ

አበባ ዩኒቨርሲቲ የሚማሩ የኮምፒውተር ሳይንስ ተማሪዎችን የሆኑትን ስም አውጣልኝ።። etc. and figure 5.4 shows the prototype of select queries by joining different tables.

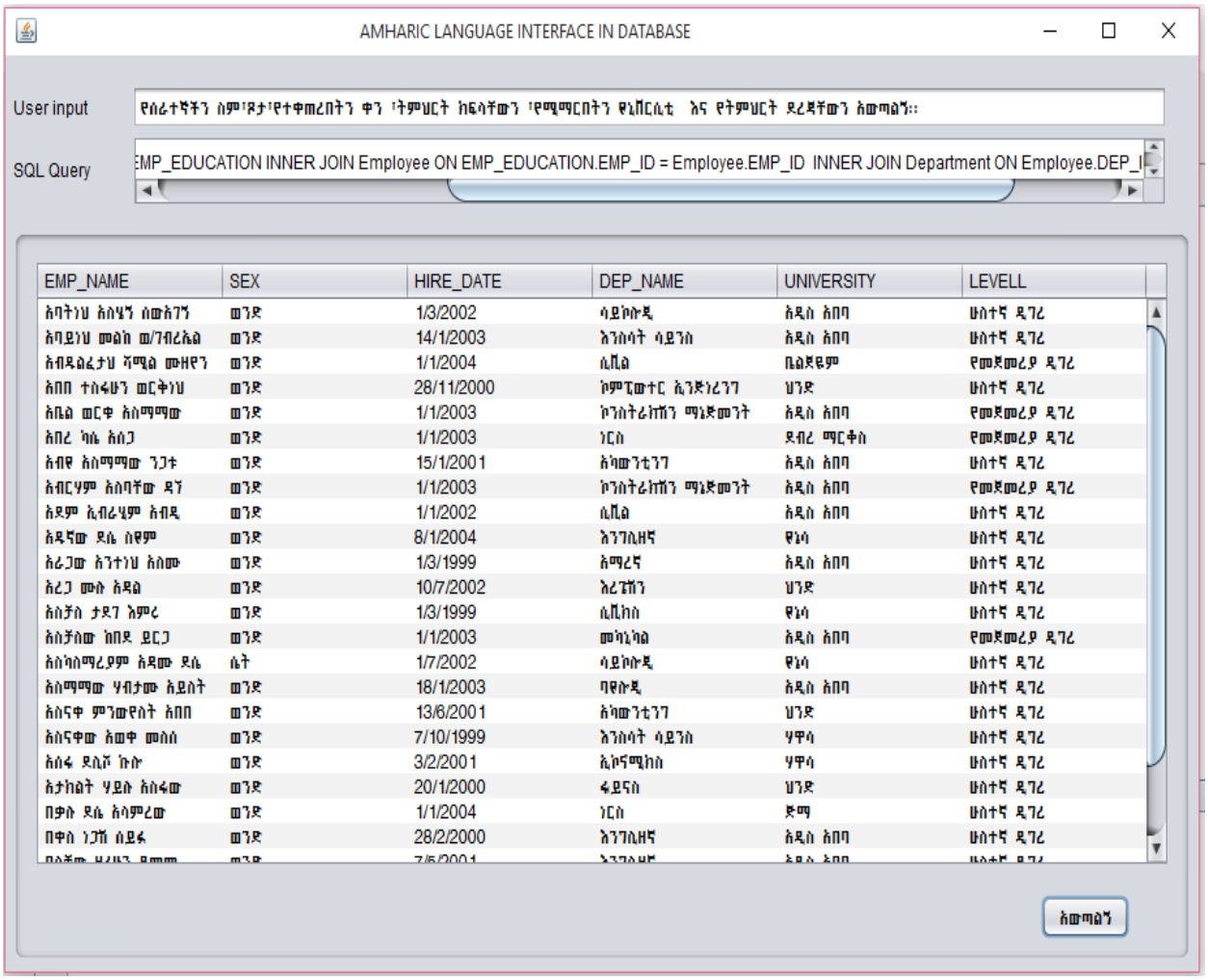

Figure 5.4.Joining query screenshot

5.2 Evaluation of proposed system

We have tested the system taking two different groups of users which is: (i) users who are not aware of database concepts, and (ii) users who are aware of database concepts. The following analyses are observed:

A. Allow relaxation of grammar rules: The list of queries includes different paraphrases of same query or sentence. The system does not follow any particular template for input query. The result supports the claim of the processing of ALIDB as flexible.

B. Easy to understand and maintain: The prototype system supports the user query even when the user enters any letters like (ጸ፡ዐ፡ሰ፡ሠ) etc., for the query which is not understood by the system, gives appropriate error message for the user. The result supports the claim of processing ALIDB as user friendly.

5.3 Analysis of results

5.3.1 Databases used for evaluation:

We have used Academic employee database to verify or validate the developed prototype or ALIDB system. This database contains three tables namely, Employee, Department and Employee on education. The structures of the database or tables are shown in table 5.1, table 5.2 and table 5.3.

Table5.1 Employee table structure

COLUMN_NAME	DATA _TYPE	COMMENTS
EMP_ID	VARCHAR2(200 BYTE)	Employee identification number
EMP_NAME	VARCHAR2(200 BYTE)	name of Employees
SEX	VARCHAR2(60 BYTE)	Sex of employees
LEVELL	VARCHAR2(34 BYTE)	Label of employees
HIRE_DATE	VARCHAR2(200 BYTE)	Hire date of employees
DEP_ID	VARCHAR2(200 BYTE)	Department identification number
FIELD_STUDY	VARCHAR2(200 BYTE)	Field of study of employees
POSITION	VARCHAR2(200 BYTE)	Position of employees
SALARY	NUMBER	Salary of employees

Table5.2 Department table structure

COLUMN_NAME	DATA _TYPE	COMMENTS
DEP_ID	VARCHAR2(200 BYTE)	Department identification

		number
DEP_NAME	VARCHAR2(200 BYTE)	Name of department
COLLAGE	VARCHAR2(200 BYTE)	Collage of employees

Table 5.3 Employee on education table structure

COLUMN_NAME	DATA _TYPE	COMMENTS
EMP_ID	VARCHAR2(200 BYTE)	Employee identification number
UNIVERSITY	VARCHAR2(200 BYTE)	the university where employee studying on
FIELD_STUDYING	VARCHAR2(200 BYTE)	The employees field of studying
STARTING_YEAR	VARCHAR2(200 BYTE)	The year employees started studying

5.3.2 Analysis according to Question Category

The analysis which is done related to question categories such as: list queries, condition queries which include single condition queries and Join queries. The category wise query description is shown in table 5.4. Moreover, the accuracy measurement of system performance in terms of Correct (C) and Incorrect (I) is carried out through ALIDB system because there is no common accuracy measurement of NLIDB system.

Table 5.4: Description of Category wise Question

Category of Query	Description
L	List Queries
S	Single condition Queries
J	Join Queries

I. **(L) List queries** that include the single table query without condition (all rows will be retrieved or only listed row in user queries will be retrieved) and columns can be in a form of C_1, C_2… C_n or count (*) or C_1 and C_2… C_n or different C_1 etc., as shown in table 5.5, table 5.6. Figure 5.5 shows accuracy of system for list queries.

Table5.5 Category List (L) Queries

No.	User input query	System generated query	C/I
1	የሰራተኞችን ዝርዝር አውጣልኝ።	Select * from employee	C
2	የሠራተኞችን ሙሉ መረጃ አሳየኝ።	select * FROM employee	C
3	የሁሉንም ሠራተኞች ሙሉ መረጃ አውጣ።	select * FROM employee	C
4	የሰራተኞችን መረጃ አውጣ።	select * FROM employee	C
5	የሰራተኛወችን ዝርዝር አሣይ።	select * FROM employee	C
6	የሰራተኞችን ስም፣ጾታ፣የተቀጠረበትን ቀን እና የትምህርት ደረጃቸውን አውጣልኝ።	Select Emp_name,sex,Hire_date,levell from employee	C
7	የሰራተኞችን ሥም፣ፃታ፣የቅጥር ቀን እና ትምህርት ደረጃ አውጣ።	select sex,Hire_date,Levell FROM employee	I
8	የሠራተኛዎችን ስም፣ፃታ፣ስራ የጀመረበትን ቀን እና የትምህርት ደረጃ አሣይ።	select Emp_name,sex,Hire_date,Levell FROM employee	C
9	ሠራተኞች የተቀጠሩበትን ቀን፣ስም፣ጾታ እና ትምህርት ደረጃቸውን አውጣልኝ።	select Hire_date,Emp_name,sex,Levell FROM employee	C
10	የሰራተኛዎችን ትምህርት ደረጃ፣የቅጥር ቀን፣ጾታ እና ስም አሣየን።	select Levell,Hire_date,sex,Emp_name FROM employee	C
11	የሁሉንም ሰራተኛ ስም፣የቅጥር ቀን እና ደመወዝ አውጣልኝ።	select Emp_name,Hire_date,Salary FROM employee	C
12	ሠራተኞች ሥራ የጀመሩበትን ቀን፣ስም እና ደሞዝ አሣይ።	select Hire_date,Emp_name,Salary FROM employee	C
13	የሠራተኛዎችን ሥም፣ወርሃዊ ክፍያ እና የቅጥር ቀን አውጣልን።	select Hire_date FROM employee	I
14	የሠራተኛዎችን ደመወዛቸውን፣ስማቸውን እና ስራ የጀመሩበትን ቀን አሳየን።	select Hire_date FROM employee	I
15	የሁሉንም ሰራተኛ ወርሃዊ ክፍያ፣የቅጥር ቀን እና ሥም አውጣ።	select Hire_date FROM employee	I
16	የሰራተኞችን ስም፣ሃላፊነት እና ደመወዝ አውጣልኝ።	select Emp_name,Position,Salary FROM employee	C
17	የሰራተኛዎችን ስም፣የስራ ድርሻ እና ደሞዝ አውጣ።	select Emp_name,Position,Salary FROM employee	C
18	የሠራተኛ ስም፣ሐላፊነት እና ወርሃዊ ክፍያ አሣይ።	select Emp_name,Salary FROM employee	I
19	የሠራተኛዎችን ስማቸውን፣ኃላፊነታቸውን እና ወርሃዊ ክፍያቸውን አሣየን።	select Emp_name,Position,Salary FROM employee	C
20	የሰራተኞችን የስራ ድረሻ፣ስም እና ወርሃዊ ክፍያ አሳየን።	select Position, Emp_name, Salary FROM employee	C

Table 5.6 Accuracy of system for Category L Query

Category L Accuracy	Total queries	Correct queries (C)	Incorrect queries(I)
75%	20	15	5

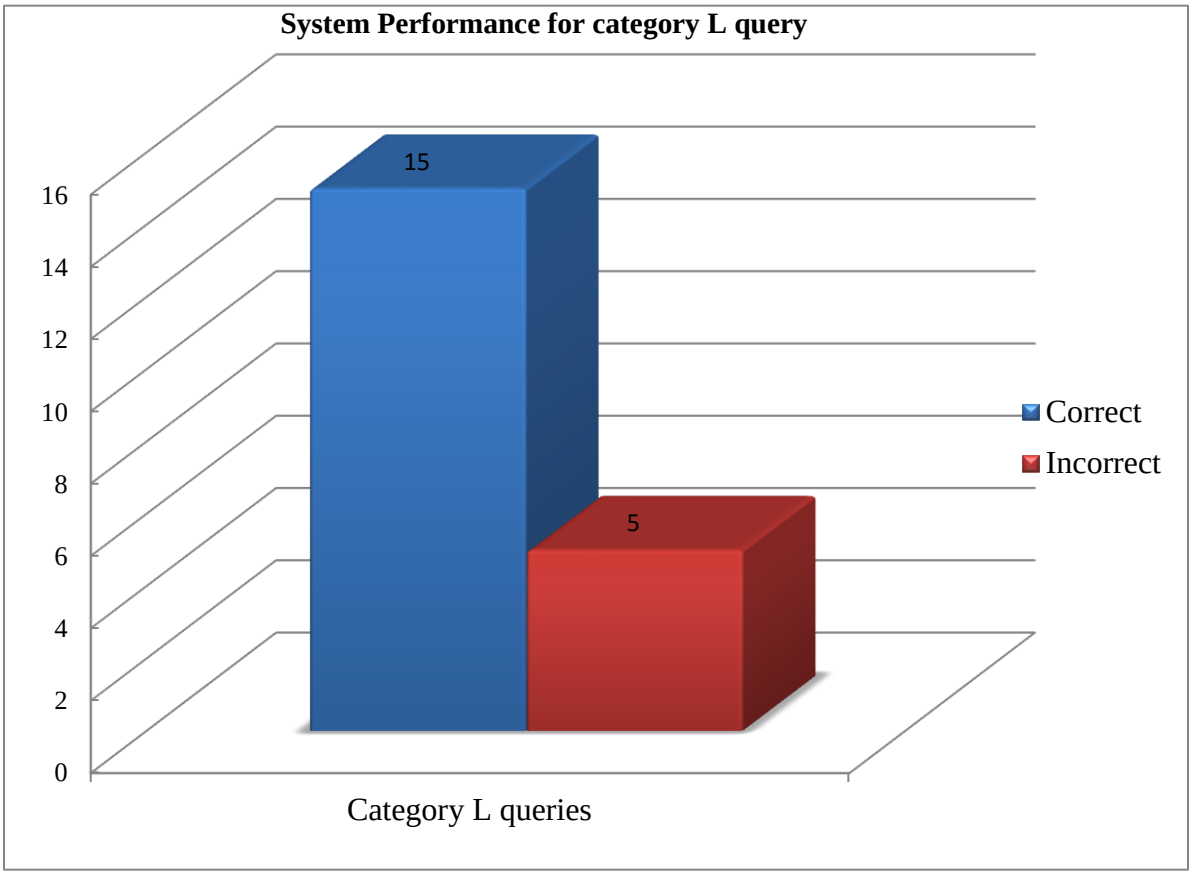

Figure 5.5: System Performance for category L query

II. **Conditional Query:** This query includes single query with single condition and single query with composite or multiple conditions. But composite or multiple conditions is not

including in the study with the shortage of time.

C. Single query with single condition(S): includes the rows and columns to be retrieved based on single condition from a single table as shown in table 5.7 and table 5.8. The figure 5.6 shows accuracy of system for single condition queries.

Table 5.7 Category single condition(S) queries

No.	User input query	System generated query	C/I
1	ወንድ ሰራተኛ የሆኑትን ስም፣የትምህርት ደረጃቸውን እና ደመወዝ አውጣልኝ።	select Emp_name,Levell,Salary FROM employee Where sex ='ወንድ'	C
2	ወንድ ሰራተኛ የሆኑትን ስም፣የትምህርት ደረጃቸውን እና ደሞዝ አውጣ።	select Emp_name,Levell,Salary FROM employee Where sex ='ወንድ'	C
3	ፃታቸው ወንድ የሆኑ ሰራተኞችን ስም፣ደሞዝ እና ትምህርት ደረጃቸውን አሳየኝ።	select sex,Emp_name,Salary,Levell FROM employee Where sex ='ወንድ'	C
4	ወንድ ሰራተኛ የሆኑትን ሥም፣የትምህርት ደረጃቸውን እና ወርሃዊ ክፍያ አውጣ።	select Levell,Salary FROM employee Where sex ='ወንድ'	I
5	የወንድ ሠራተኛ የሆኑትን ሥም፣የትምህርት ደረጃቸውን እና ወርሃዊ ክፍያ አውጣ።	select Levell,Salary FROM employee Where sex ='የወንድ'	I
6	ሴት የሆኑ ሰራተኞችን ስም፣ወርሃዊ ክፍያ እና ትምህርት ደረጃቸውን አሳይ።	select sex,Emp_name,Salary,Levell FROM employee Where sex ='ሴት'	C
7	ደመወዛቸው 3500 ብር እና በላይ የሆኑ ሰራተኞችን ስም አውጣልኝ።	select Salary,Emp_name FROM employee Where salary >=3500.0	C
8	ደሞዛቸው ከ 3500 በታች የሆነ ሰራተኞችን ስም እና ሥራ የጀመሩበትን ቀን አውጣልኝ።	select Emp_name,Hire_date FROM employee Where Hire_date <='ደሞዛቸው'	I
9	ወርሃዊ ክፍያቸው 5700 የሆኑትን ሰራተኞችን ስም አሳይ።	select Emp_name FROM employee Where salary =5700.0	C
10	ወርሃዊ ክፍያቸው 5700 የሆኑትን ሰራተኞችን ስም እና ሃላፊነት አውጣል።	select Emp_name,Position FROM employee Where salary =5700.0	C
11	ደሞዛቸው 5700 ብር የሆኑትን ሰራተኞችን ስም እና የስራ	select Emp_name,Position FROM	C

	ድረሻ አሳየን።	employee Where salary =5700.0	
12	ወርሃዊ ክፍያቸው 3500 እና ከዚያ በላይ የሆኑ ሰራተኞችን ስም እና ሃላፊነት አሳይ።	select Emp_name,Position FROM employee Where salary >=3500.0	C
13	ሃላፊነታቸው የኮሌጅ ዲን የሆኑትን ሰራተኞች ስም አውጣልኝ።	select Position,Collage,Emp_name FROM employee Where Position ='ሃላፊነታቸው'	I
14	የትምህርት ደረጃቸው ሁለተኛ ዲግሪ የሆነ ሁሉንም ሰራተኞችን ስም አውጣልኝ።	select Levell,Emp_name FROM employee Where levell ='የትምህርት'	I
15	የትምህርት ደረጃቸው ሦስተኛ ዲግሪ የሆኑ ሁሉንም ሰራተኞችን ስም አውጣልኝ።	select Levell,Emp_name FROM employee Where levell ='አውጣልኝ'	I
16	የትምህርት ክፍል ሃላፊዎች የሆኑትን ስም እና ደመወዝ አውጣልኝ።	select Dep_name, Position, Emp_name, Salary FROM department Where Position ='የትምህርት'	I
17	ደመወዛቸው ከ 4500 ብር በላይ የሆነ ሰራተኞችን ስም እና የመታወቂያ ቁጥር አውጣልኝ።	select Salary,Emp_name,Emp_id FROM employee Where salary >4500.0	C
18	የትምህርት ደረጃቸው አንደኛ ዲግሪ የሆኑ ሁሉንም ሰራተኞችን ስም አውጣልን።	select Levell, Emp_name FROM employee Where levell ='አውጣልን'	I
19	የጥናት ዘርፋቸው ህገ የሆኑ ሰራተኞችን ስም እና ደመወዝ አውጣልኝ።	select Emp_name, Salary FROM employee Where 'የጥናት'	I
20	ደመወዛቸው ከ 4500 ብር በላይ የሆነ ሰራተኞችን ሥም እና የመታወቂያ ቁጥር አውጣልኝ።	select Salary,Emp_id FROM employee Where salary >=4500.0	I

Table 5.8 Accuracy of system for Category S Query

Category S Accuracy	Total queries	Correct queries (C)	Incorrect queries(I)
50%	20	10	10

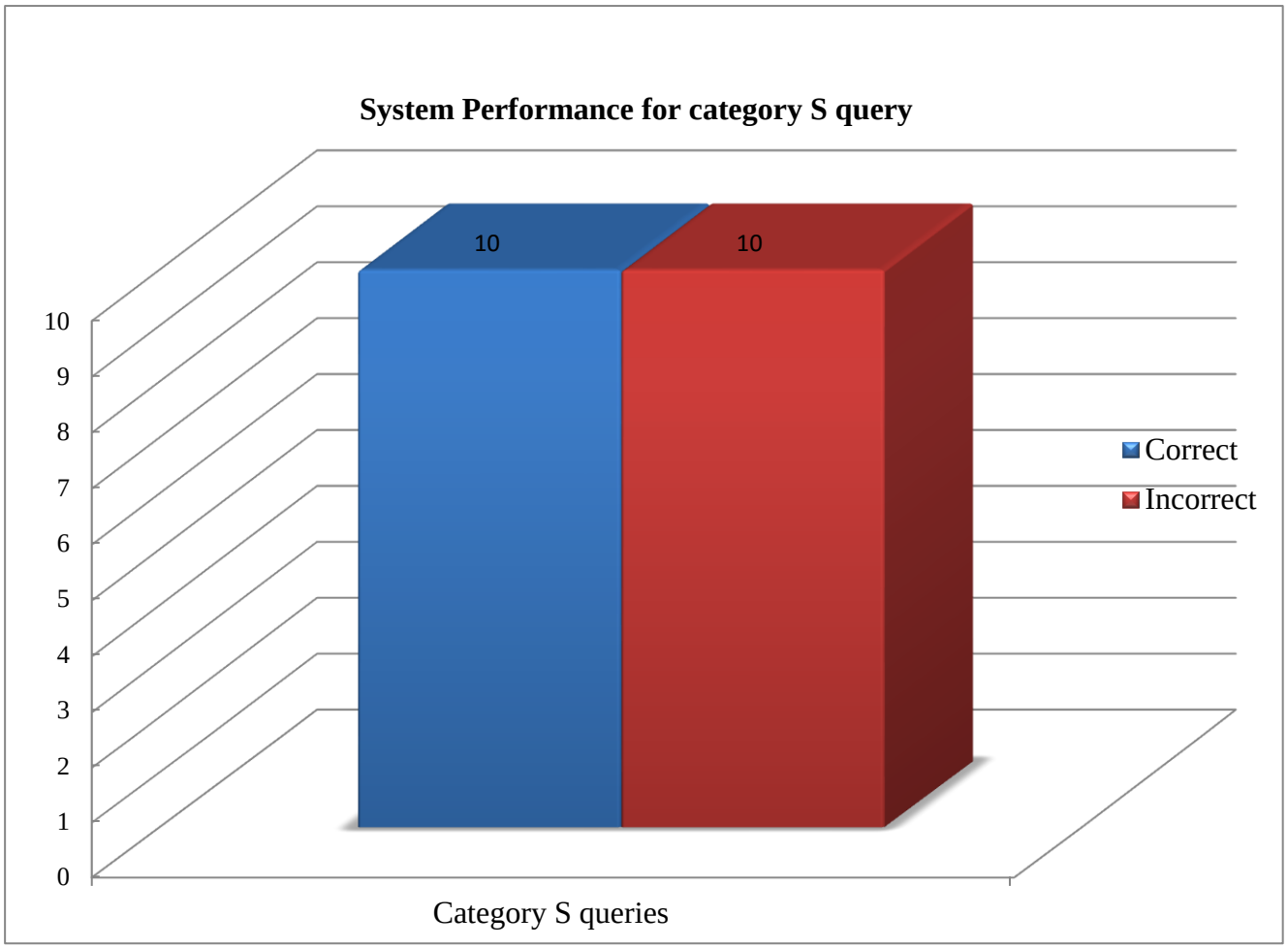

Figure 5.6 System Performance for category S query

III. **Join Queries** (J): Join queries include joining of two or three tables. The query can have only single conditions as shown in table 5.9 and table 5.10 and, figure 5.7 shows the performance and accuracy of the system for Join queries.

Table 5.9 category of J Queries

No.	User input query	System generated query	C/I
1	አካውንቲንግ ት/ክፍል ሠራተኛ የሆኑትን ስም አውጣ።	select Dep_name,Emp_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='አካውንቲንግ'	C
2	ትምህርት ክፍላቸው አካውንቲንግ የሆኑ ሠራተኞችን ስም አሳይ።	select Dep_name,Emp_name FROM	C

		Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where dep_name ='አካውንቲንግ'	
3	አካውንቲንግ ትምህርት ዘርፍ የሚገኙ ሠራተኞች የሆኑትን ስም ዝርዝር አሳየን፡፡	select Emp_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='አካውንቲንግ'	C
4	አካውንቲንግ ትምህርት ክፍል ሰራተኛ የሆኑትን ስም አውጣልኝ፡፡	select Dep_name,Emp_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='አካውንቲንግ'	C
5	ትምህርት ክፍላቸው አካውንቲንግ የሆኑ የሠራተኞችን ሥም ዝርዝር አውጣልን፡፡	select Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where dep_name ='አካውንቲንግ'	I
6	የአካውንቲግ ትምህርት ክፍል ሰራተኛ የሆኑትን ሃላፊነት፣ደመወዝ እና ስም አውጣልኝ፡፡	select Dep_name,Position,Salary,Emp_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='የአካውንቲግ'	I
7	የግብርና እና ተፈጥሮ ሳይንስ ኮሌጅ ዲን የሆኑትን፣ስም፣ጾታ፣የተቀጠረበትን ቀን እና የትምህርት ደረጃውን አውጣልኝ፡፡	select Collage, Emp_name, sex, Hire_date,Levell FROM Where Hire_date ='የግብርና'	I
8	ማኔጅመንት ትምህርት ክፍል ሰራተኛ የሆኑትን ስም እና ትምህርት ደረጃቸውን አውጣልኝ፡፡	select Dep_name,Emp_name,Levell FROM Department INNER JOIN	C

		Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='ማኔጅመንት'	
9	የሰራተኞችን ስም ዝርዝር እና ትምህርት ክፍላቸውን አውጣልኝ።	select Emp_name,Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID	C
10	የሰራተኞችን ስም፣ጾታ፣የተቀጠረበትን ቀን ፣ዲፓርትመንት፣የሚማርበትን ዩኒቨርሲቲ እና የትምህርት ደረጃቸውን አውጣልኝ።	select Emp_name, sex, Hire_date, University, Levell FROM EMP_EDUCATION INNER JOIN Employee ON EMP_EDUCATION.EMP_ID = Employee.EMP_ID INNER JOIN Department ON Employee.DEP_ID = Department.DEP_ID	C
11	የሰራተኞችን ስም፣ትምህርት ክፍል እና ደመወዝ አውጣልኝ።	select Emp_name,Dep_name,Salary FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID	C
12	የትምህርት ደረጃቸው አንደኛ ዲግሪ የሆኑ ሁሉንም ሰራተኞችን ስም እና ትምህርት ክፍል አውጣልን።	select Levell, Emp_name, Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where Dep_name ='አውጣልን'	I
13	የሰራተኞችን ስም ዝርዝር ከነትምህርት ክፍላቸው አውጣልኝ።	select Emp_name,Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID	C
14	የሰራተኞችን ስም፣ጾታ፣የተቀጠረበትን ቀን ፣ዲፓርትመንት እና የትምህርት ደረጃቸውን አውጣልኝ።	select Emp_name,sex,Hire_date,Levell FROM Department INNER JOIN	C

		Employee ON Employee.DEP_ID = Department.DEP_ID	
15	**የሰራተኞችን ስም እና ትምህርት ክፍላቸውን አውጣልኝ፡፡**	select Emp_name,Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID	C
16	**ደመወዛቸው 4500 ብር የሆነ የሰራተኞችን ሥም እና የመታወቂያ ቁጥር አውጣልኝ፡፡**	select Salary,Emp_id FROM employee Where salary =4500.0	I
17	**ደመወዛቸው ከ 4500 ብር በላይ የሆነ ሰራተኞችን ስም፣የመታወቂያ ቁጥር እና ትምህርት ክፍል አውጣልኝ፡፡**	select Salary, Emp_name, Emp_id, Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID Where salary >4500.0	C
18	**የሰራተኞችን ሙሉ ሥም እና ትምህርት ክፍል አውጣልኝ፡፡**	select Dep_name FROM Department INNER JOIN Employee ON Employee.DEP_ID = Department.DEP_ID	I
19	**የሰራተኞችን ሙሉ ስም ፣የሚማሩበትን ዩኒቨርሲቲ እና ትምህርት ክፍል አውጣልኝ፡፡**	select Emp_name, University, Dep_name FROM EMP_EDUCATION INNER JOIN Employee ON EMP_EDUCATION.EMP_ID = Employee.EMP_ID INNER JOIN Department ON Employee.DEP_ID = Department.DEP_ID	C
20	**የሰራተኞችን ሙሉ ስም ፣የሚማሩበትን ዩኒቨርሲቲ እና ኮሌጅ አውጣልኝ፡፡**	select Emp_name,University,Collage FROM EMP_EDUCATION INNER JOIN Employee ON EMP_EDUCATION.EMP_ID =	I

		Employee.EMP_ID	

Table 5.10 Accuracy of system for Category J Query

Category L Accuracy	Total queries	Correct queries (C)	Incorrect queries(I)
65%	20	13	7

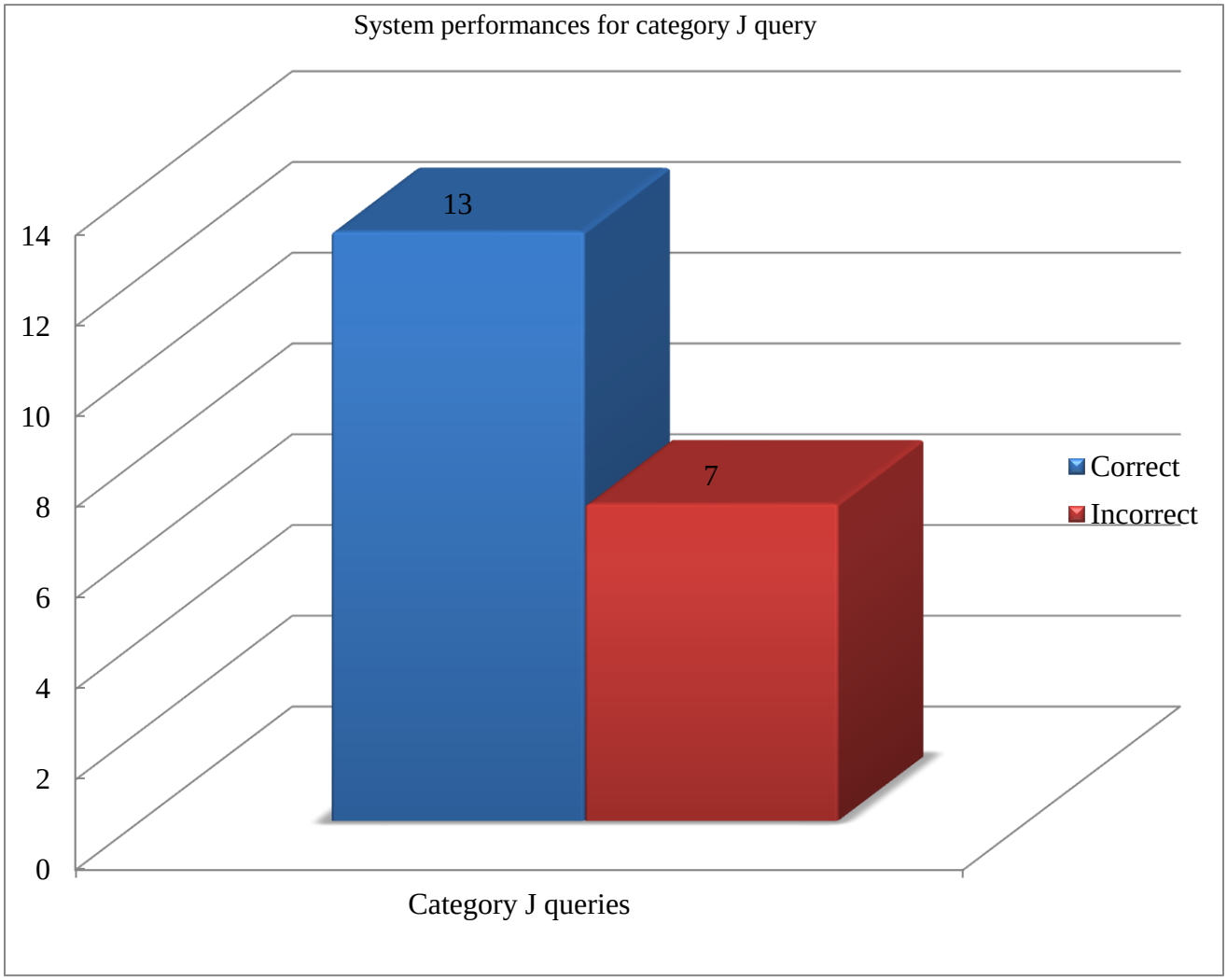

Figure 5.7 System performances for category J query

5.3.3 Category wise Accuracy

The effectiveness of the category wise question system is measured in terms of accuracy measurement as shown in figure 5.8.

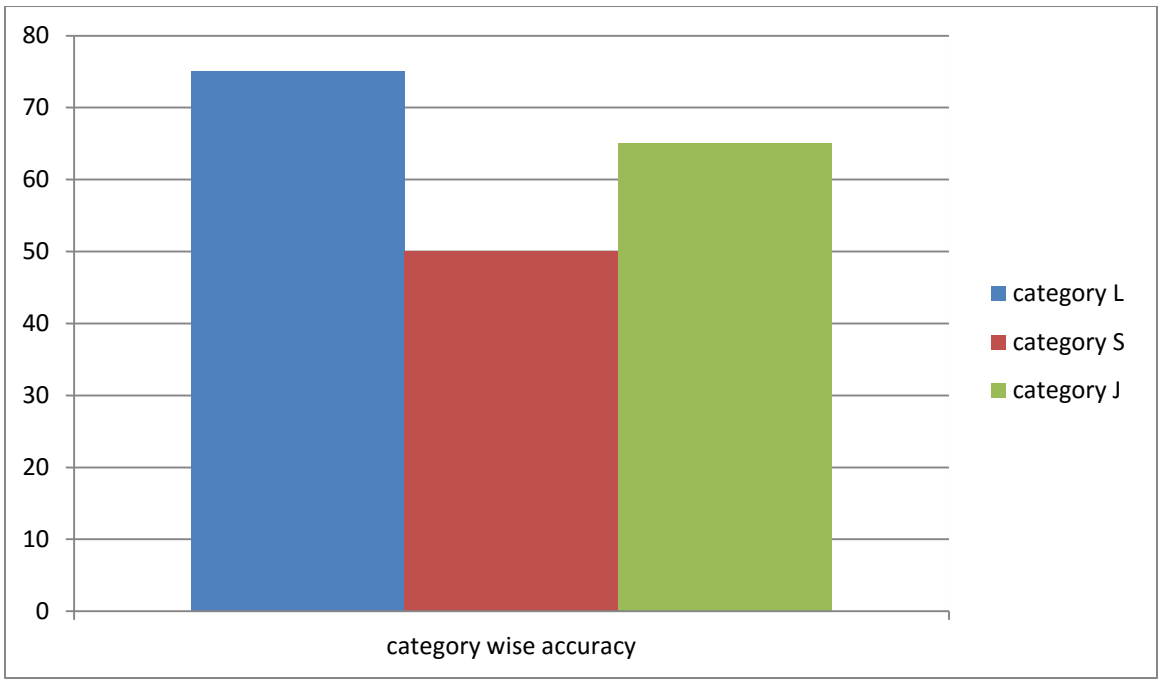

Figure 5.8 System performances for category wise accuracy

5.3.4 Analysis according to user category

We have tested our prototype system taking different users who are not aware of database concepts and also the users who are aware of database concepts. The figure 9 shows the accuracy level of system in terms of correct queries, incorrect queries and partially correct queries according to user groups.

Figure 5.9 System performances for User category

5.3.5 Overall measurement

The overall accuracy of the ALIDB system is based on how many SQL queries are generated by system is correct as per user's judgment. It is given by

Overall accuracy of correct queries=$total\ number\ of\ correct\ queries \div total\ number\ of\ inputted\ queries$

=38/60

=0.633

=63.3%

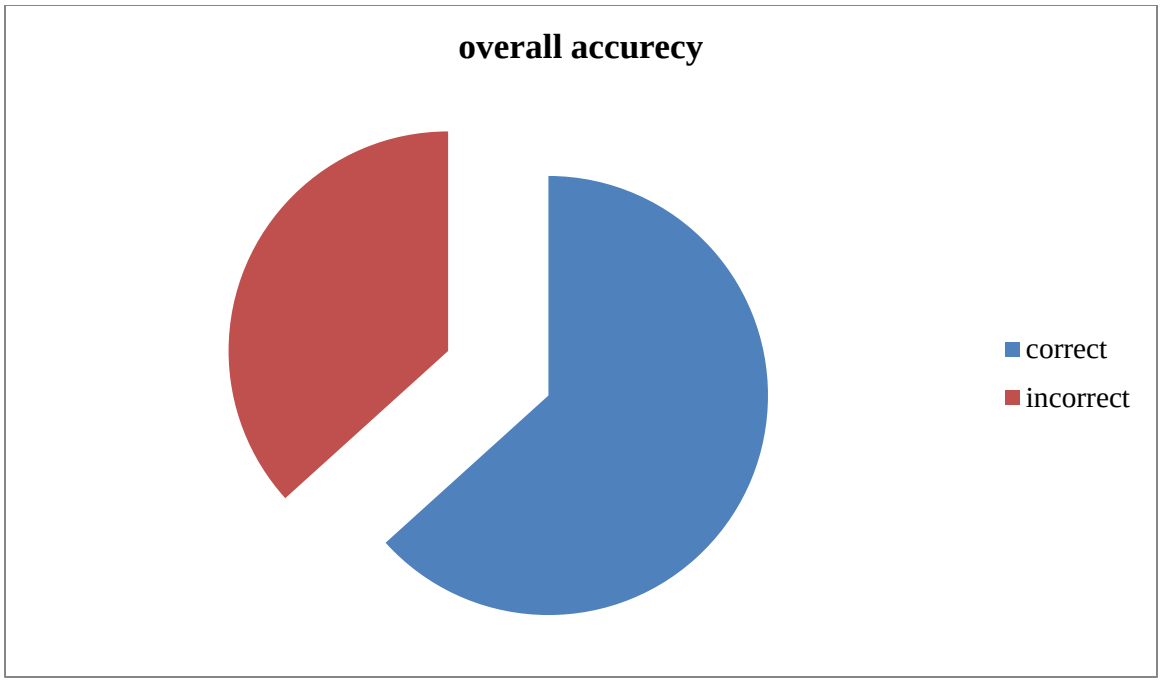

Figure 5.10 Allover system performances

5.4 SUMMARY

This chapter generally discussed the prototype implementation of ALIDB system within their question categories. It also shows accuracy and its measurement in terms of correct and incorrect queries for each query category. Finally, it discusses the user group wise accuracy and overall accuracy of the ALIDB system.

Chapter-6

Conclusions and Future Works

6.1 Conclusion

The thesis has been proposed Amharic Language Interface to Database. It demonstrated the integration of various techniques that can enhance the accuracy of the (Natural to Formal language) translation process or translation of unstructured query sentences to structure one. It also tries to improve user interface so that user can enter the query in free form without any restriction but it is also use Amharic language or enter only Amharic query sentences.

The basic objective of the study is to improve human computer interface while accessing the structured data from the database by using their native language or Amharic language. The user enter Amharic query sentences, the sentences tokenize with each word and mapped each token word from system dictionary and handle table_name, column_name, select key word, where key word , operator and values from the tokenized word and generate the structure query sentences.

However, if the user enters very few words or is not enter full sentences in query and if those words or sentences are difficult to analyze for words sense disambiguation due to handling of table_name, column_name, select key word, where key word, operator and values, then the system either may not be able to generate such queries or it may produce partial output.

6.2 Future Works

The following areas of ALIDB can be considered for future work:

- To execute queries that involves or executes insert, update and delete data from database
- To execute queries that includes aggregate functions (i.e. sum (), max (), average) and DDL (like alter, drop, create types of queries) statements.
- To add the future to execute queries that include group by, Order by and having clauses.
- To add the future to work on independent queries and different domains
- To design the approach of ALIDB in such a way that is not dependent on database
- To design the approach of ALIDB in such a way that is in linguistic part

Reference

1. Shish Kumar and Kunwar Singh Vaisla, Hindi Language Interface to Database using Semantic Matching, 2014, India

2. Rajender Kumar, MohitDua, Shivani Jindal, "Domain-Independent Hindi Language Interface to Relational Database", international conference on computation of power, energy, information and communication (ICCPEIC), 2014,India

3. K.Murugan and T. Ravichandran, Intelligent Query Interface for Temporal Database with Natural Language Processing using Efficient Context Free Grammar, Karpagam University,2012.

4. UmairShafique and Haseeb Qaiser, "A Comprehensive Study on Natural Language Processing and Natural Language Interface to Databases," International Journal of Innovation and Scientific Research ISSN 2351-8014 Vol. 9 No. 2 Sep. 2014, pp. 297-306,University of Gujrat

5. Ashish Kumar and Kunwar Singh Vaisla, ARTICLE · MAY 2013, Natural Language Interface to Databases: Development Techniques, India

6. NeeluNihalani, Mahesh Motwani and Sanjay Silakari, An Intelligent Interface for relational databases, 2014, Indian.

7. Rohini B. Kokare,Kirti H. Wanjale,2014, A Survey of Natural Language Query Builder Interface for Structured Databases using Dependency Parsing, SavitribaiPhule Pune University

8. Himani Jain,Hindi language interface in database ,M.S thesis paper, Indian,2011

9. Chong Wang, Miao Xiong, Qi Zhou and Yong Yu, A Portable Natural Language Interfaceto Ontologies,2014,Shanghai JiaoTong University

10. Androutsopoulos, G.D. Ritchieand P. Thanisch,1995,Natural Language Interfaces to Databases – An Introduction, University of Edinburgh

11. Abhijeet Gupta, December, 2013, Complex Aggregates In Natural Language Interface To Databases. Indian

12. Mahesh Singh1, Nikita Bhati, 2014, A Noval Hindi language interface for databases. India

13. Arati K. Deshpande1 and Prakash. R. Devale, " natural language query processing using probabilistic context free grammar", International Journal of Advances in Engineering & Technology, May 2012. India.

14. B.Sujatha, S.ViswanadhaRaju and Humera Shaziya,2012,A Survey of Natural Language Interface to Database Management System, J.N.T.University

15. Ann Copestake and Karen Sparc k Jones,1989,Natural Language Interfaces to Databases, University of Cambridge

16. Himani Jain, Parteek Bhatia, Hindi language interface to databases ,Journal of Global Research in Computer Science ,Volume 2, No. 4, April 2011,,Indian

17. AksharBharati,Y. Krishna Bhargava, Rajeev Sanga,2014, Reference and Ellipsis in an Indian Languages Interface to Databases, Indian

18. MohitDua, Shivani Jindal, and Rajender Kumar, An Architectural Overview of Natural Language Interface to Knowledge Base,2014 international conference on computation of power, energy, information and communication, Indian

19. BentamarHemerelain, Hafida Belbachir,2010,Semantic Analysis of Natural Language Queries for an Object Oriented Database ,University of Science and Technology of Oran

20. GauriRao, ChanchalAgarwal, SnehalChaudhry, Nikita Kulkarni, and Dr. S.H. Patil, natural language query processing using semantic grammar International Journal on Computer Science and Engineering Vol. 02, No. 02, 2010, 219-223,Indian

21. A. Kaur, and P. Bhatiya, "Punjabi Language Interface to Database", M.tech thesis, Department of CSED, Thapar University, 2010

22. Manju Mony, Jyothi M. Rao and Manish M. Potey, “An Overview of NLIDB Approaches and Implementation for Airline Reservation System” International Journal of Computer Applications (0975 – 8887) Volume 107 – No 5, December 2014, Dept. of Computer Engineering K. J. Somaiya CoE Vidyavihar, Mumbai

23. Neelu Nihalani , Sanjay Silakari , Mahesh Motwani, “Natural language Interface for Database: A Brief review”, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011, Bhopal, MP India

24. Abhijeet R. Sontakke1,and Amit Pimpalkar, “A Review Paper on Hindi Language Graphical User Interface to Relational Database using NLP” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 10, October 2014

25. Yohan Chandra, “natural language interfaces to databases”, December 2006, University of north texas ,Ms thesis.

26. Neelu Nihalani, Mahesh Motwani and Sanjay Silakari, “Natural Language Interface to Database using Semantic Matching”, International Journal of Computer Applications (0975 – 8887) Volume 31– No.11, October 2011.

27. Amandeep Kaur, “punjabi language interface to database”, Computer science and engineering department thapar university, Ms thesis ,June 2010

28. Neelu Nihalani, Mahesh Motwani and Sanjay Silakari, “N NIHALANI *et al:* an intelligent interface for relational databases” IJSSST, Vol. 11, No. 1, Bhopal, MP, India

29. Nikita Bhati and, Mahesh Singh, “Research on Analysis of Hindi language Graphical user Interface”, international journal of engineering sciences & research technology Bhati et al. , 3(8): August, 2014, Student, AITM., Palwal, India

30. http://zrftech.blogspot.com/2012/12/java-se-development-kit-7-documentation.html / 27-Jan- 2015

31. Gregg Petri,A comparison of Oracle and MYSQL,SELECT Journal 1st Qtr.2005

32. Rashid Ahmad, Mohammad Abid Khan and Rahman Ali, “Efficient Transformation of a Natural Language Query to SQL for Urdu”, Proceedings of the Conference on Language and technology, 2009.

33. Jadhav Sneha, Raut Shubhangi and A.S.Zore, “Natural Language to Database Interface”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 2, February 2014, India.

34. Rajender Kumar and Manish Kumar, “A Comprehensive Study of Natural Language Interface to Database”, International Journal of Advance Foundation and Research in Science and Engineering (IJAFRSE) Volume 1, Issue 1, June 2014, NIT Kurukshetra.

35. Pooja A.Dhomne, Sheetal R.Gajbhiye, Tejaswini S.Warambhe and Vaishali B.Bhagat, “Accessing database using NLP”, IJRET: International Journal of Research in Engineering and Technology Volume: 02 Issue: 12 | Dec-2013 page 589-594.

36. K. Shabaz, Jim D. O'Shea, Member, IEEE, Keeley A. Crockett, Senior Member, IEEE, A. Latham, Member, IEEE, “Aneesah: A Conversational Natural Language Interface to Databases”, Proceedings of the World Congress on Engineering 2015 Vol I WCE 2015, July 1 - 3, 2015, London, U.K.

37. Aanchal Kataria and Rajender Nath, “Natural Language Interface for Databases in Hindi Based on Karaka Theory”, International Journal of Computer Applications (0975 – 8887) Volume 122 – No.7, July 2015, Kurukshetra, Haryana, India.

38. Fei Li and H. V. Jagadish “Constructing an Interactive Natural Language Interface for Relational Databases” Proceedings of the VLDB Endowment, Vol. 8, No. 1 Copyright 2014 VLDB Endowment 2150-8097/14/09.

39. Jasmeen Kaur , Bhawna chauhan and Jatinder Kaur Korepal, “Implementation of Query Processor Using Automata and Natural Language Processing”, International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013

40. Yılmaz, Ekrem Çağlar, “a Turkish natural language interface for the semantic web:” a case study on Turkish universities, MSc Thesis paper August 2008, 42 pages, Atilim university.

41. Noam Chomsky (1965), Aspects of the Theory of Syntax, M.I.T. Press, Cambridge, Massachusetts.

42. http://www.google.images.com , 20-Feb- 2015.

43. Madeleine Bates (1995), Model of Natural language understanding, in proceeding of national academic science, USA, vol.92, Oct 1995.

44. Steven Bird, Ewan Klein, Edward Loper, (2008), Book on Natural language Processing with Python, O’Reilly publication.

Appendices

Appendices 1: Mapping tables for column handler

Token words	Mapped words
ሃላፊዎች	Position
ታላፊዎች	Position
ሀላፊዎች	Position
ሐላፊዎች	Position
የስራ ድርሻ	Position
የሥራ ድርሻ	Position
ወርሃዊ ክፍያ	Salary
ታላፊነታቸውን	Position
ሀላፊነታቸውን	Position
ሃላፊነታቸውን	Position
ሐላፊነታቸውን	Position
ስማቸውን	Emp_name
ሥማቸውን	Emp_name
ወርሃዊ ክፍያቸውን	Salary
የትምህርት ደረጃውን	Levell
ትምህርት ክፍላቸውን	Dep_name
ታላፊነት	Position
ሀላፊነት	Position
ሐላፊነት	Position
ሃላፊነት	Position
ጸታ	Sex
ጸታቸው	Sex
የመታወቂያ ቁጥራቸውን	Emp_id
የተቀጠሩ	Hire_date
ት/ክፍል	Dep_name
መታወቂያ ቁጥር	Emp_id
መታወቂያ ቁጥራቸው	Emp_id
መለያ ቁጥር	Emp_id
መለያ ቁጥራቸው	Emp_id
ስም	Emp_name
ሥም	Emp_name
ስማቸው	Emp_name
ሥማቸው	Emp_name
ፆታ	Sex
ፆታቸው	Sex
ትምህርት ደረጃ	Levell
የትምህርት ደረጃ	Levell
የትምህርት ደረጃቸው	Levell
የተቀጠረበት ቀን	Hire_date
የቅጥር ቀን	Hire_date

የተቀጠሩበት ቀን	Hire_date
ትምህርት ክፍል	Dep_name
የትምህርት ክፍል	Dep_name
ትምህርት ክፍላቸው	Dep_name
የትምህርት ክፍላቸው	Dep_name
ያጠናው የትምህርት አይነት	Field_study
የተማረው የትምህርትአያነት	Field_study
ያጠኑትን የትምህረት አይነት	Field_study
የተማሩትን የትምህርት አይነት	Field_study
ሃላፊ	Position
ሀላፊ	Position
ሐላፊ	Position
ኀላፊ	Position
ሀላፊዋች	Position
ሃላፊዋች	Position
ሐላፊዋች	Position
ኀላፊዋች	Position
ሃላፊነት	Position
ሀላፊነት	Position
ሐላፊነት	Position
ኀላፊነት	Position
ሃላፊነታቸው	Position
ሀላፊነታቸው	Position
ሐላፊነታቸው	Position
ኀላፊነታቸው	Position

Appendices 2: Sample codes

```
public class SelectStatement implements Serializable {
    private static final long serialVersionUID = 1L;
    private Long id;
    public static String sqlValue = "";
    public ResultSet selectTableName(String tokenaized) {
        try {
            DBConnection connection = new DBConnection();
            ResultSet _rs = null;
            OraclePreparedStatement ops;
```

```
        String _select = "SELECT UNIQUE MAPPED_WORD from TABLE_HANDLER
WHERE  TOKEN_WORD LIKE ? ";
        ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
        ops.setString(1, "%" + tokenaized);
        _rs = ops.executeQuery();
        return _rs;
      } catch (Exception e) {
        return null;
      }
   }
   public ResultSet selectWhereColumnHandler(String tokenaized) {
      try {
        DBConnection connection = new DBConnection();
        ResultSet _rs = null;
        OraclePreparedStatement ops;
        String _select = "SELECT UNIQUE MAPPED_WORD from Wherecolumn_handler
WHERE  TOKEN_WORD LIKE ? ";
        ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
        ops.setString(1, "%" + tokenaized);
        _rs = ops.executeQuery();
        return _rs;
      } catch (Exception e) {
```

```
            return null;
        }
    }
    public ResultSet selectColumnName(String tokenaized) {
        try {
            DBConnection connection = new DBConnection();
            ResultSet _rs = null;
            OraclePreparedStatement ops;
            String _select = "SELECT UNIQUE MAPPED_WORD from COLUMN_HANDLER
WHERE  TOKEN_WORD LIKE ? ";
            ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
                    ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            ops.setString(1, "%" + tokenaized);
            _rs = ops.executeQuery();
            return _rs;
        } catch (Exception e) {
            return null;
        }
    }
    public ResultSet selectWhereName(String tokenaized) {
        try {
            DBConnection connection = new DBConnection();
            ResultSet _rs = null;
            OraclePreparedStatement ops;
```

```
        String _select = "SELECT UNIQUE MAPPED_WORD from WHERE_CONDITION
WHERE  TOKEN_WORD LIKE ? ";
        ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
        ops.setString(1, "%" + tokenaized);
        _rs = ops.executeQuery();
        return _rs;
      } catch (Exception e) {
        return null;
      }
   }
   public ResultSet selectSelectName(String tokenaized) {
      try {
        DBConnection connection = new DBConnection();
        ResultSet _rs = null;
        OraclePreparedStatement ops;
        String _select = "SELECT UNIQUE MAPPED_WORD from SQL_WORD WHERE
TOKEN_WORD LIKE ? ";
        ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
        ops.setString(1, "%" + tokenaized + "%");
        _rs = ops.executeQuery();
        return _rs;
      } catch (Exception e) {
```

```
            return null;
        }
    }
    public ResultSet selectConditionName(String tokenaized) {
        try {
            DBConnection connection = new DBConnection();
            ResultSet _rs = null;
            OraclePreparedStatement ops;
 String _select = "SELECT UNIQUE MAPPED_WORD from
CONDITIONAL_WORDHANDLER WHERE  TOKEN_WORD LIKE ? ";
            ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
                    ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            ops.setString(1, "%" + tokenaized);
            _rs = ops.executeQuery();
            return _rs;
        } catch (Exception e) {
            return null;
        }
    }
public ResultSet selectValuesFromAll(List<String> lstbl, List<String> lsCol, List<String>
lswhere, List<String> lswherecolumn, List<String> lssele,List<String> lsnumber,String
operator, List<String> lstokonized) throws SQLException {
        String COLUMN = "";
        String TABLES = "";
        String WHERE = "";
```

```
String OPRATORS = "=";
String SELECT = "";
String WHERECOL = "";
String tree = "";
String department = "", emp_education = "", employee = "";
int k = 1;
if (lsCol.isEmpty()) {
   COLUMN = "*";
} else {
   for (int i = 0; i < lsCol.size(); i++) {
      if (k == lsCol.size()) {
         COLUMN += lsCol.get(i) + " ";
      } else {
         COLUMN += lsCol.get(i) + ",";
      }
      k++;
   }
}
 k = 1;
if (lstbl.size() == 1) {
   for (int i = 0; i < lstbl.size(); i++) {
      if (k == lstbl.size()) {
         TABLES += lstbl.get(i) + " ";
      }
      k++;
```

```
        }
    } else if (lstbl.size() == 2) {
        for (int i = 0; i < lstbl.size(); i++) {
            if (lstbl.get(i).equalsIgnoreCase("department")) {
                department = lstbl.get(i) + " ";
            } else if (lstbl.get(i).equalsIgnoreCase("employee")) {
                employee = lstbl.get(i) + " ";
            } else if (lstbl.get(i).equalsIgnoreCase("emp_education")) {
                emp_education = lstbl.get(i) + " ";
            }
        }
        if (!employee.equals("") && !department.equals(""))
TABLES = " Department INNER JOIN Employee ON Employee.DEP_ID =
Department.DEP_ID ";
        } else if (!employee.equals("") && !emp_education.equals("")) {
            TABLES = " EMP_EDUCATION INNER JOIN Employee ON
EMP_EDUCATION.EMP_ID = Employee.EMP_ID ";
        } else {
            TABLES += department + " " + employee + " " + emp_education + " ";
        }
    } else if (lstbl.size() == 3) {TABLES = "EMP_EDUCATION INNER JOIN Employee ON
EMP_EDUCATION.EMP_ID = " + "Employee.EMP_ID  INNER JOIN Department ON
Employee.DEP_ID = Department.DEP_ID ";
    }
    for (int i = 0; i < lswhere.size(); i++) {
        WHERE += lswhere.get(i) + " ";
    }
```

```
        for (int i = 0; i < lswherecolumn.size(); i++) {
            WHERECOL += lswherecolumn.get(i) + " ";
        }
        for (int i = 0; i < lssele.size(); i++) {
            SELECT += lssele.get(i) + " ";
        }
        String _select = " ";
        String condition = "";
        List<String> lstcharactor = new ArrayList<>();
        List<Double> lstnumber = new ArrayList<>();
        for (int i = 0; i < lswherecolumn.size(); i++) {
            WHERECOL += lswherecolumn.get(i) + " ";
            condition = lswherecolumn.get(i) + " " + operator + " " + "' ? '
if (lswherecolumn.get(i).contains("salary") || lswherecolumn.get(i).contains("age")) { ||
lstwherecolumn.get(i).contains("Hire_date")
                condition = lswherecolumn.get(i) + " " + operator;
            } else {
                condition = lswherecolumn.get(i) + " " + operator;
            }
        }
        for (int i = 0; i < lstokonized.size(); i++) {
            if (lstokonized.get(i).matches(".*\\d.*")) {
                lstnumber.add(Double.parseDouble(lstokonized.get(i)));
            } else {
```

```
            lstcharactor.add(lstokonized.get(i));
        }       }
    try {
        DBConnection connection = new DBConnection();
        ResultSet _rs = null;
        OraclePreparedStatement ops;
        if (!WHERE.equals("")) {
            if (condition.contains("salary") || condition.contains("age")) {
                for (int j = 0; j < lstnumber.size(); j++) {
                    _select = "";
_select = SELECT + "" + COLUMN + "" + "FROM " + TABLES + "" + WHERE + "" +
condition + "" + lstnumber.get(j);
sqlValue = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " " + WHERE + " " +
condition + "" + lstnumber.get(j);
                    System.out.println(_select);
                    ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
                            ResultSet.TYPE_SCROLL_SENSITIVE,
                            ResultSet.CONCUR_UPDATABLE);

                    _rs = ops.executeQuery();
                    if (_rs.next()) {
                        return _rs;
                    }
                }
            } else {
                for (int j = 0; j < lstcharactor.size(); j++) {
```

```
                _select = "";


                _select = SELECT + "" + COLUMN + "" + "FROM " + TABLES + "" + WHERE
+ "" + condition + "'" + lstcharactor.get(j) + "'";

sqlValue = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " " + WHERE + " " +
condition + "'" + lstcharactor.get(j) + "'";

                System.out.println(_select);

                ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,

                    ResultSet.TYPE_SCROLL_SENSITIVE,

                    ResultSet.CONCUR_UPDATABLE);


                _rs = ops.executeQuery();

                if (_rs.next()) {

                   return _rs;

                }

             }

          }

 _select = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " " + WHERE + " " +
condition;

          // System.out.println(condition);

          ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,

                ResultSet.TYPE_SCROLL_SENSITIVE,

                ResultSet.CONCUR_UPDATABLE);

          if (condition.contains("salary") || condition.contains("age")) {

             for (int i = 1; i <= lstnumber.size() * lsCol.size();) {

                for (int j = 0; j < lstnumber.size(); j++) {
```

```
                ops.setDouble(1, lstnumber.get(j));
            }
        }
    } else {
        for (int i = 1; i <= lstcharactor.size() * lsCol.size();) {
            for (int j = 0; j < lstcharactor.size(); j++) {
                ops.setString(1, lstcharactor.get(j));
                i++;
            }
        }
    }
    _rs = ops.executeQuery();
    if (_rs.next()) {
        return _rs;
    } else {
        _select = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " ";
        System.out.println(_select);
        ops = (OraclePreparedStatement) connection.con.prepareStatement(_select,
                ResultSet.TYPE_SCROLL_SENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
        _rs = ops.executeQuery();
        return _rs;
    }
} else {
    _select = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " ";
```

```
        sqlValue = SELECT + " " + COLUMN + " " + "FROM " + TABLES + " ";
        System.out.println(_select);
        ops = (OraclePreparedStatement) DBConnection.con.prepareStatement(_select,
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
        _rs = ops.executeQuery();
        return _rs;
    }               }
 catch (SQLException e) {
    System.out.println(e.getSQLState());
    System.err.println(e.getMessage());
    return null;
}  }}
```