ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Modeling and Designing Amharic Query System to Bilingual (English-Amharic) Databases

By

Tihitina Petros Degsew

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

April, 2014

ADDIS ABABA UNIVERISTY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

Modeling and Designing Amharic Query System to Bilingual (English-Amharic) Databases

By

Tihitina Petros Degsew

ADVISOR:

Solomon Atnafu (PhD)

APPROVED BY

EXAMINING BOARD:

1. Dr. Solomon Atnafu, Advisor _____
2. _____ _____
3. _____ _____

DEDICATION

To My Family: Without your full attention and support, I wouldn’t have finished this thesis work.

Acknowledgements

Above all, thank you GOD for giving me the ability to be where I am. Without the Almighty's help I couldn't reach this point. Thanks LORD for giving me the strength to achieve whatever I have achieved so far and you have done so much for me.

I would like to express my deepest appreciation and thanks to my advisor Dr Solomon Atnafu for his motive, encouragement and advice throughout this work. Without his guidance and comments the completion of this research would have not been possible.

My deepest gratitude goes to my families. My dad and mom, thanks for understanding and supporting me to accomplish this work. My sister Mahider and my brothers Ermias and Natna'el, What can I say? Thanks a lot for your unconditional support and moral starting from the beginning till the end of the work. I couldn't be able to finish it without you guys. My special thanks go to Tsega and Mabrat for making my environment comfortable for the work. Tedy and Ayu thanks for motivating and helping me.

Esayas, thanks for supporting me on the title selection and providing me with articles, necessary tools and software and also for reading and commenting on my documents. Abreham, you have been with me all the time especially throughout the toughest part of my life. You have been guiding, supporting and providing me with the necessary information. You have also helped me to manage and use my time effectively when I was in need of help. I owe you.

Finally, I extend my heartfelt thanks and respect to my friends, classmates and colleagues Rekik, Zenawi, Temesgen, Yibeltal, Tekeste, Bekalu, Blen, Engidu, Derese, Tigabu, Eskiyas, Lisan, Chalachew, Yoseph, Getasew and to all those people who were not mentioned here but their contributions have been useful for the completion of this work.

Table of Contents

List of Figures

List of Tables

List of Appendices

Acronym and Abbreviations

AQS	Amharic Query System
NLIDB	Natural Language Interface to Databases
SQL	Standard Query Language
DB	Database
NL	Natural Language
NLP	Natural Language Processing
HMM	Hidden Markov Model
PCFG	Probabilistic Context Free Grammar
AQA	Amharic Question Answering
SNLP	Shallow Natural Language Processing
LIL	Logic Interface Language
SVM	Support Vector Machine
ALP	Amharic Language Processing
IR	Information Retrieval
XML	Extensible Markup Language

ABSTRACT

Seeking for information is a very important part of the day to day activities of human beings. Information can be acquired from different sources, one of the major sources being databases. Formulating a query in a way the computer system understands is necessary to retrieve data from a database. To accomplish that, one should write SQL statements. But everyone is not able to write SQL queries because of its requirement to be able to know the technical structures of SQL query formulation. Instead of formulating SQL statements, using natural language to retrieve data from database prompted the development of Natural Language Interface to Database Systems (NLIDB). NLIDB allows translating Natural language queries into SQL queries. Hence more friendly to naïve and non technical information seekers to use their own language to access database contents.

Research on NLIDB started in the late sixties. Since then, many research works have been conducted. A lot of researches have been done for different languages like English, Arabic, Spanish, Punjab, etc. However, little is done to design NLIDB in Amharic language. Only one work is available that attempted to model and design Amharic Structural Query Language (ASQL). The model was syntax based. It requires users to know the syntax of the ASQL to formulate their queries. In this work, we designed a NLIDB system for Amharic. With this Amharic Interface users can extract information from database systems using plain Amharic statements without being constrained to query language syntax.

The Amharic Interface to databases translates the plain Amharic statements into English SQL queries. We designed a general architecture for Amharic NLIDB; we designed different algorithms to extract database elements from databases, to manage Amharic statement pre-processing, to map the translated tokens to database elements, and to generate equivalent SQL statements. Furthermore, we have identified and used the necessary tools and resources such as tokenizer, normalizer, stop word remover, Amharic stemmer, bi-lingual (Amharic-English) translation dictionary, list of thesaurus, list of stop words, etc. Finally we have implemented and tested the Amharic Interface for Database query. The evaluation of the system shows encouraging results.

Keywords: Amharic Query Interface for Databases, Amharic Query System for Bi-lingual databases, Query Translator, Natural Language Query Mapper, Natural Language Interface for Databases (NLIDB), Database Element Extractor.

CHAPTER ONE

INTRODUCTION

1.1. Overview

Information is playing an important role in our day to day lives. One of the major sources of information is databases. However, accessing data from databases using SQL statements is too technical for novice users. Stating queries to databases in natural languages is a very convenient and easy way of accessing data from database systems, especially for casual users who do not know the syntax and technical details of database query languages such as SQL. Accessing data from database would have been popular if it uses Natural language query system, like querying the web, instead of using technical query statements.

Natural Language interface for Databases (NLIDB) is a system which allows users to access information stored in a database by typing their query in Natural Language like English, French, Chinese, etc. It interprets the natural language query into Structured Query Language (SQL) to perform action on target database. Currently, a lot of database applications are developed locally with local language content. To assist native users to perform query on the local database, the use of local language interface is required. Though, there are many efforts to develop NLIDB for different languages, there is no work done that enables users to query databases using Amharic statements.

1.2. Motivation

Novice and native users, who need to access data stored in a database, face difficulty to formulate SQL commands because SQL commands have complex syntax structure for them. Most of the time, these users ask for help from those who have SQL background. As a result, these users might not get the information they need on time or their work might be delayed till they get the required SQL statements. These problems can be solved by natural language interface to Databases (NLIDB).

NLIDB is a simplified query interface that doesn't need much technical knowledge. It is simple and easy to use. Users need to input a simple natural language statement in the system interface.

The anticipated system then converts this simple statement in to an equivalent formal SQL statement. In this research, a system which will accept users query in Amharic language and retrieve data from bilingual (English-Amharic) database will be developed.

The need for Amharic Query System is because Amharic is the working language of federal government of Ethiopia (a country of more than 85 million populations). The very big majorities of public and private organization workers are not skilled in formulating SQL statements and hence are unable to retrieve information from databases.

The need to retrieve data from bi-lingual (English - Amharic) database is becoming high as more and more organizations are automating their systems and a number of databases are having bi-lingual contents. There are organizations which use database that contains both English and Amharic content. For example, Woredas and Kebeles hold residential information and the like in Microsoft excel sheets and Microsoft access in Amharic fonts. Amharic keyboard is used to enter data in Amharic content. In some of vehicle license training center, student's information is stored in Amharic. Ethiopian federal court system uses a system called CourtStar which is developed by Cybersoft Plc. CourtStar is a database that work in a networked environment. It is already implemented in the regional courts such as the Addis Ababa Municipal courts, the Oromia Supreme Court, the Tigrai Supreme Court, the SNNPR Supreme Court, the Somalia Supreme Court and the Harari Supreme Court. It is designed to store – detail case information, adjournment information, daily court case hearing schedules, court case circulation management, etc. it is UNICODE based multi-lingual feature that enabled CCMS to work in the language of the various regions (i.e, Amharic, Oromifa, Tigirgna, Somaligna). Besides these, in the future a lot of Amharic content that might be stored in a database of different organization.

1.3. Problem Statement

Database administrators and users with SQL background do not need to use NLIDB. They can formulate the query by themselves. But users, who don't speak English language and don't know SQL commands, will not be able to access the database unless they know the syntax and semantics of executing a query to the database.

In addition, to retrieve information from a database, users should have to know different things which they are not expected to know like:

- The database design,
- Table relationships and
- Other information.

For example, let us consider there is a table called CUSOMER. To get information about all customer data, the user should write the following SQL statement.

```
Select * from CUSTOMER;
```

To get NAME and PHONE NUMBER of customers, the user should write:

```
Select NAME, PHONE from CUSTOMER;
```

Similarly, to get information about the person called Abebe, the user should write the following SQL command and execute it.

```
Select * from CUSTOMER where FIRSTNAME = 'abebe' ;
```

However, the challenge is that these SQL statements require users to follow defined syntax rules and a simple error terminates the execution of the query. Then the problem is how users interact with the database system. Users may find themselves handicapped to access information on the database. The use of NLIDB, tries to eliminate the restrictions of SQL syntax and creates a simplified natural language interface to access contents of databases.

For example, through the Amharic interface to databases, users can write the following Amharic statements and the Amharic NLIDB identifies equivalent SQL statements that can be processed by the SQL query executer.

- የሁሉንም ደንበኛ መረጃ አሳየኝ
- ከደንበኛ ላይ ስም እና ስልክ አሳየኝ

- ከደንበኛ ላይ መጀመሪያ ስም አበበ ጋር እኩል አሳየኝ

Users without SQL knowhow might need to access databases related to their task. For example, a local exporter doesn't need to know English to read the daily price of export items, from the database; users who have access to database and who need information stored in a database for specific task (for example, employee works in human resource department may need to see staff information) don't have to go to database administrator to get the information; users might want to access information from the database which do not have any other alternative to access information.

Even for local users who speak English and who don't have SQL knowhow, formulating SQL commands is also difficult. To support these users to fully access the database, a system that retrieves data from both English and Amharic database is needed.

In languages such as china, Punjab, Arab and Spanish, many natural language interfaces to databases have been developed, which provides flexible options for manipulating queries [6, 3]. In Ethiopia, with the interest of using natural language in computing environment, a number of researches are coming up that deal with localization of computer-based application. And currently more and more database applications are developed locally with local language content. But a little emphasis is given for making databases accessible in local languages. To the best of the researcher knowledge, a research by Mekuria Sinkie [17], is the only research which designed and developed an Amharic query language. It gave a good insight for users with Amharic Language.

However, it was restricted and syntax-based. For untrained users it is not feasible to use restricted and syntax based query language because they still have to know the syntax of Amharic Query Language. Instead of using Amharic Query Language, it is better for untrained users to use meaningful Amharic statement to access the content of a database. Moreover, the previous research work on Amharic Query Language was almost the exact replica of the English SQL command. It was syntax based, which English SQL Commands were translated into Amharic language.

Amharic is the second most-spoken Semitic language in the world next to Arabic and also the working language of the Federal Democratic Republic of Ethiopia. As a result different types of Amharic documents exist, which need to be stored in databases. Therefore, this research is intended to model and design Amharic Query System that is easy to formulate for accessing content of bilingual databases. That is, by using the proposed system one can simply access the database by typing a query as a plain Amharic statement. The advantage of designing Amharic query system for databases is that it is more understandable and easy to use for the native users as compared to the previous work.

From this point of view the following **Research Questions** will be addressed in the research:

- What relationship exists between Amharic statements and SQL commands?
- How can one generate query results of databases using Amharic statements?
- How can we make effective bilingual database query using Amharic queries?

1.4. Objective

General Objective

The general objective of this research is to design and develop Amharic query Interface to bilingual (Amharic-English) databases.

Specific Objectives

To meet the general objective, the following specific objectives are taken into consideration

- Review previous related works,
- Identify different techniques and algorithms that have been used in the area of NLIDB,
- Analyze structure of Amharic language statements,

- Develop a model, algorithm and techniques to map Amharic statements to its equivalent SQL commands,
- Design and develop Amharic Query System,
- Test the designed system accuracy and describe its performance.

1.5. Scope

Due to limitation of time and resources, the scope of the proposed system is limited to the following points:

- Amharic query statements associated to a limited set of database queries will be considered. i.e, data manipulation queries such as insert, delete, update commands and all data definition queries are not considered.
- It might not fully handle ambiguity issues,
- It will not incorporate Amharic spelling checker.

1.6. Methodologies

In order to accomplish the objective of the research the following methodologies are used.

1.6.1. Literature Review

Literatures on NLIDB or related fields are reviewed. From the literatures, techniques to analyze the structure of Amharic statements and SQL queries are studied and selected based on their applicability and performance. The structure of Amharic language statements is analyzed and the relationship between Amharic statement and SQL commands is studied.

1.6.2. The Development Environment

Our system is developed and tested using one laptop computer with specification of processor - Intel(R) Pentium(R) Dual CPU, CPU speed – 1.00 GHz for each processor, RAM – 1 GB, Hard disk – 150 GB and installed operating system is Windows7.

Development Tools: To develop the system, different tools from different sources are used. Some of them are open source software; software developed for research purposes and licensed software. Tools that are used to develop the prototype of the system are explained below.

Programming Environment: Java JDK1.6.0_14 with NetBeans IDE 6.9 is used as a programming environment. This version of java program is installed in Windows7 operating system. This software is chosen because; it is easy to design User Interface and it is platform independent.

Database Environment: Microsoft Excel and SQL Server 2005 are used as databases. These databases are chosen because they can store both English characters and utf8 representation of Amharic characters. MS excel is easy to design the database, create tables and insert Amharic data in the tables. Excel is also used to rearrange the data we used for evaluating the performance of the system, explained in chapter six. Later the data will be imported to SQL server. This is done to easily execute SQL commands in SQL server 2005 database.

Operating System: Windows7 is chosen as operating system. Because, it supports Unicode character and its graphical user interface is very attractive.

Experiment: is conducted:

- To analyze how well the system translates the given Amharic statement into the corresponding SQL command,
- To check how well users are satisfied compared to the SQL.

1.7. Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the literature reviewed in the area of NLIDB. It gives brief introduction, history and development of NLIDB. Different structures of NLIDB, structure of Amharic language and SQL are explained. Chapter 3 presents related works in the development of NLIDB systems. Chapter 4 presents the design as well as the implementation of the system. Chapter 5 presents Experiments done to evaluate the performance of the system developed. It also discussed the system and outlines the contribution of the work. Finally, in chapter 6 conclusion and future work is presented.

CHAPTER TWO

LITERATURE REVIEW

2.1. Natural Language Interface for Databases

Databases are used in most Information Technology applications of governmental and private companies and many other sectors. Retrieving information stored in a database requires knowledge of database languages such as SQL (Standard Query Language). However, only database experts or users who know SQL are able to write and execute SQL commands. In order to facilitate full use of the database systems, its accessibility to non-expert users is desirable [25]. This has led to the development of Natural Language Interface for Databases in many languages such as English, Spanish, Punjab, etc.

Natural Language Processing (NLP) is becoming one of the most active areas in Human-computer Interaction. Integration of NLP capabilities (e.g. semantic interpretation of a sentence) with database technologies has been an interesting research task for both the artificial intelligence (AI) and the database (DB) community since the late 60's and early 70's. Since then, several NLIDB systems have been developed. Presently, there is ongoing effort in materializing earlier research goals into practical applications [35].

Users considered the traditional information retrieval techniques insufficient in retrieving precise information [30]. The rising in number of native speakers to use their natural language to access information and the increasing availability of information in different languages is the basis for research on restricted natural language query interface [14]. With the high interest of using national languages in the computing environment, the number of researches that deal with localization of computer-based applications is also increasing in Ethiopia [9, 10, 29].

A Natural Language Interface for database (NLIDB) allows the user to access information stored in a database by typing requests expressed in some natural language [32].

2.2. History of NLIDB

NLIDB systems prototype had already appeared in the late sixties and early seventies. Accordingly, the best known NLIDB of that period is LUNAR, LADDER, RENDEZVOUS, PLANES, PHILIQA, CHAT-80, TEAM, ASK, JANUS, EUFID and DATALOG [32].

LUNAR was introduced in 1971 and it answers questions about samples of rocks brought back from the moon [32]. It uses two databases; one for the chemical analysis and the other for literature. The LADDER system was designed for US Navy ships. The system uses semantic grammars technique that interleaves syntactic and semantic processing. It has three layered architecture. The first is Informal Natural Language Access to Navy Data (INLAND), which accepts questions in a natural language and produces a query to the database. Then the queries are directed to the Intelligent Data Access (IDA), which is the second component of LADDER. The query is fragmented for each lower level syntactic unit in the English language input query and these fragments are then combined to higher level syntactic units to be recognized. File Access Manager (FAM) is the third component and its task is to find the location of the generic files and manage the access to them in the distributed database. It either be used in large databases or it configured to different underlying database management systems

In late seventies RENDEZVOUS System was developed [32]. It is dialogue-based for users to formulate their queries Users could access databases via relatively unrestricted natural language. On the same time PLANES (Programmed LANguage-based Enquiry System) was developed for at the University of Illinois Coordinated Science Laboratory. PLANES include an English language front end with the ability to understand and explicitly answer user requests.

PHILIQA was developed in 1977 and was known as Philips Question Answering System, uses a syntactic parser which runs as a separate pass from the semantic understanding passes [32]. It was implemented in Prolog and in Eighties it becomes one of the most referenced NLP systems. The database of CHAT-80 consists of facts (i. e. oceans, major seas, major rivers and major cities) about 150 of the countries world and a small set of English language vocabulary that are enough for querying the database.

In 1987 TEAM was developed. It was designed to be easily configurable by database administrators with no knowledge of NLIDBs [32]. A large part of the research of that time was devoted to portability issues. ASK was a system developed in 1983. It allowed end-users to teach the system new words and concepts at any point during the interaction. It was a complete information management system, providing its own built in database and the ability to interact with multiple external databases, electronic mail programs and other computer applications.

JANUS system had similar abilities to interface to multiple underlying systems (databases, expert systems, graphics devices, etc) [32]. All the underlying systems could participate in the evaluation of a natural language request, without the user ever becoming aware of the heterogeneity of the overall system. The EUFID system consists of three major modules, not counting the DBMS. First is analyzer module, second is mapper module and third is translator module. DATALOG is an English database query system based on Cascaded ATN grammar. By providing separate representation schemes for linguistic knowledge, general world knowledge, and application domain knowledge, DATALOG achieves a high degree of portability and extendibility.

2.3. Recent Developments in NLIDB

Three specific NLIDB systems are developed recently and they are PRCISE, WASP and NALIX [32].

PRECISE introduces the idea of semantically tractable sentences which are sentences that can be translated to a unique semantic interpretation by analyzing some lexicons and semantic constraints [32]. The target database is in the form of a relational database using SQL as the query language. The strength of PRECISE is based on the ability to match keywords in a sentence to the corresponding database structures.

NALIX (Natural Language Interface for an XML Database) was developed at university of Michigan. It uses XML as a database and schema free XQuery as query language [32]. It is syntax- based system and developed in three steps; generating a parse tree, validating the parse tree, and translating the parse tree to an XQuery expression.

Word Alignment-based Semantic Parsing (**WASP**) is designed to address the broader goal of constructing a complete, formal, symbolic, meaningful representation of a natural language sentence [32]. A predicate logic (Prolog) was used as the formal query language. WASP builds a semantic parser from a corpus annotated with their correct formal query languages. The whole learning process is done using statistical machine translation techniques. The system was also evaluated on different natural languages: English, Spanish, Japanese and Turkish. There were no significant differences observed between English and Spanish, but the Japanese corpus has the lowest precision and the Turkish corpus has the lowest recall.

2.4. Different Structures to Build NLIDB

2.4.1. Components of NLIDB

Computing scientists have divided the problem of natural language access to a database into two sub-components: Linguistic and Database [31].

Linguistic Component: is responsible for two purposes. One is for translating natural language input into a formal query and the other is for generating a natural language response based on the results from the database search.

Database Component: performs Database Management functions. The response of a database is input for a natural language generator. The natural language generator inspects the parse tree in order to generate adequate natural language response. Natural language database systems use syntactic knowledge in order to properly relate natural language input to the database elements. Users query is translated into a statement in a formal query language. to produce the required data, the query is processed by the database management system. These data then passed back to the natural language component. Then the Natural Language response is generated.

2.4.2. Various Approaches of NLIDB

Natural language is the topic of interest from computational viewpoint due to the implicit ambiguity that language possesses. Several researchers applied different techniques to deal with language [25].

Symbolic Approach (Rule Based Approach): Natural Language Processing is a symbolic activity. A sentence that obeys well specified grammar rules is constructed from group of words. Words are symbols that stand for objects and concepts in real worlds. Language is analyzed in various levels to obtain information. Certain rules are applied to this obtained information certain to achieve linguistic functionality. In symbolic processing rules are formed for every level of linguistic analysis the same as Human Language capabilities include rule-base reasoning; it is supported well by symbolic processing. Symbolic approach tries to capture the meaning of the language based on these rules.

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. Corpora are primarily used as a source of information about language. There are a number of techniques which enable the analysis of corpus data. Recent research indicates that empirical or corpus–based methods are currently the most promising approach to developing robust, efficient natural language processing (NLP) systems. Most of the empirical NLP methods employ statistical techniques such as n-gram models, hidden Markov models (HMMs), and probabilistic context free grammars (PCFGs).

2.4.3. Architectures of NLIDB

There are different architectures of NLDB [35]. We present below few of them.

Pattern-matching systems: Early NLIDBs uses pattern-matching techniques to answer the user's queries. Pattern-matching approach is simple: no elaborate parsing and interpretation modules are needed and it is easy to implement. But implementing pattern-matching in shallow would often lead to bad failures.

Syntax-based systems: In this architecture, users query is parsed (i.e. analyzed syntactically). Then the parsed tree is directly mapped to an expression in some database query language. LUNAR is an example of a system that is developed using this architecture.

Semantic grammar systems: Semantic grammar systems, follows the same procedure as syntax-based to parse the input and map the parse tree to a database query. But the grammar's categories do not necessarily correspond to syntactic concepts. It uses semantic, which allows

semantic knowledge to be easily included in the system. Since, semantic grammars contains knowledge about a specific knowledge domain, it is difficult to port to other knowledge domains.

Intermediate representation languages: In this architecture, the system is divided into two. The first part includes the process of generating a logical query from a sentence. The second one starts from the logical query to generation of database query. In the intermediate logical query, users' query is expressed in terms of high level world concepts. One can add reasoning capabilities to the system by embedding the reasoning part inside a logic statement. Because of independent nature of the logic query language, it can be adopted to different database query languages as well as to other domains, such as expert systems and operating systems.

2.5. Structure of Amharic Language statements

2.5.1. The Amharic Characters

Amharic is the working language of the Federal Government of Ethiopia, which is a country of around 85 million populations. The present writing system of Amharic is inherited from Ge'ez alphabets. Ge'ez, which belongs to the class of Semitic language family, was the language of literature in Ethiopia in earlier times [2]. The Amharic writing system consists of a core of 33 characters (called fidels) where each of which occurs in a basic form and in six other forms known as orders. The seven orders represent the different forms of a consonant. Each form is made in accordance with the sound that goes with the symbol.

2.5.2. Word Categories

Amharic words are categorized into the following eight classes (or parts of speech) [8]. These are the noun, Verb, Adjective, Adverb, Preposition, Pronoun, Conjunction and Interjection classes.

Noun Class: Amharic nouns are words used to name or identify of a class of things, people, places, ideas, etc. or any subset (member) of these classes. Amharic nouns can be either primitive or derived, where nouns or words in general are said to be in their primitive forms if they exist in their original form (e.g ጓደኛ 'a friend') whereas they are referred to as derived if they originated (derived) into their present state from a different, and possibly completely different categories (e.g. ጓደኛ**ም**ነት 'friendliness').

Verb Class: The Amharic Verbs are words that usually come at the end of a complete grammatical declarative sentence. They also take suffixes to agree with the subject of the sentence.

Adjective Class: These are words that usually come before nouns and serve as modifiers of the nouns. In the sentence ልጁ ንበዝ ነው 'He clever boy is' 'He is a very clever boy.' for instance, the adjective ንበዝ 'clever' is an adjective, and it modifies the noun ልጁ 'boy'. Like nouns Amharic adjectives can be either primitive or derived.

Prepositions in Amharic: Prepositions are words that will have meanings only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives. For example: affixes can be /ለ/ 'for', /ከ/ 'from', /በ/ 'with', /የ/'that' or 'of', /ስለ/ 'for' and so on. They are used to form adverbial phrases appearing before nouns. Amharic prepositions can take various forms. First, in the phrase ስለ ትምህርት 'About education', the preposition ስለ appears as a simple preposition that stands alone as a separate word. Secondly, a preposition can appear as one that is prefixed to other words. For example, in the phrase በመኪና 'By car', the preposition በ is prefixed to the noun መኪና 'car'. When they are prefixed to verbs (e.g.ስለ ወደቀ 'Since he fell down', የሰረቀ 'that stole'), they form different types of subordinate clauses that are typical components of complex sentences. Thirdly, prepositions can appear as compound prepositions consisting of two parts: prepositional prefixes and post positions with a noun at the middle. For example, in the phrase ከወንድሙ ጋር 'with his brother', ከ 'from' is used as a prepositional prefix whereas ጋር 'with' is an independent preposition that usually follows nouns.

Adverb Class: In Amharic adverbs include: ገና 'yet',ቶሎ 'quickly', ክፉኛ 'severely', and ሞኝነት 'foolish'. Thus, phrases that are combinations of prepositions and some other words like nouns (e.g.በ-ኃይል 'By force') or prepositions and adjectives (e.g በደህና 'well') perform the functions of adverbs in the language.

Conjunctions: Conjunctions are two types; coordinating and subordinating conjunctions. Conjunctions are used to synchronize words, phrases, clauses and sentences. However, one problem with this trend of classification is the fact that sometimes the same forms (e.g **ስለ,** በ and ከ) forms are used both as prepositions and conjunctions. This categorical ambiguity of such

words can easily be solved by making careful analysis of the context in which the words are used.

Numerals: This class of words represents numbers, i.e. both cardinal and ordinal numbers. In Amharic, ordinal numbers are formed from their cardinal number equivalents and the suffix /-ኛ/. For example: Cardinal Gloss Ordinal Gloss ሁለት two ሁለተኛ second Like English.

2.5.3. Phrasal Categories

A phrase is a structure in a language that is constructed from one or more words in the language. It is categorized into five categories, namely noun phrase (NP), verb phrase (VP), adjectival phrase (AdjP), adverbial phrase (AdvP) and prepositional phrase (PP) [7].

Noun Phrases: A noun phrase (NP) is a syntactic unit in which the head (H) is a noun or a pronoun. It can be simple or complex. The simplest NP consists of a single noun (e.g. Kassa) or pronoun such as እሱ (he), እሷ (she), እነሱ (they), etc. A complex NP can consists of a noun (called head) and other constituents (like complements, specifiers, adverbial and adjectival modifiers) that modify the head from different aspects.

Verb Phrases: A verb phrase (VP) is composed of a verb as a head and other constituents such as complements, modifiers and specifiers. For example, in the VP በ-መኪና ወደ ትምህርት ቤት ሄደ። (He) went to school by car', both በ-መኪና =" 'By car' and ወደ ትምህርት 'To school' are prepositional phrases (PPs) modifying the verb ሄደ 'went' from manner and place points of view, respectively.

Adjectival Phrases: The adjectival phrase (AdjP) is constructed in a similar manner to a NP and a VP. It can be composed of an adjective (head), and other constituents such as complements, modifiers and specifiers. For example, in the AdjP ያቺ በጣም ቆንጆ 'That very beautiful (girl)', ያ 'that' is a specifier, በጣም 'very' is a modifier modifying the head of the AdjP (ቆንጆ 'beautiful').

Prepositional Phrases: Amharic prepositional phrase (PP) is constructed from a preposition (P) head and other constituents such as: nouns, noun phrases, verbs, verb phrases, etc. In the PP እንደ ሰው በ-ምድር 'like a man on earth', for instance, እንደ 'like' and u 'on' are prepositions that are

combined with the nouns ሰው ‘a man’ and ምድር ‘earth’, respectively to form their PPs. The two PPs, in turn, combine to result in the bigger PP that is provided in the example.

Adverbial Phrases: An Adverbial phrase (AdvP) is constructed from one or more adverbs in the language. In the example, ክፉኛ ተጋጩ ‘(They) crashed severely’, ክፉኛ ‘severely’ (head) is the only adverb that formed the AdvP. However, in ቶሎ ቶሎ ተራመደ ’He walked briskly’, the phrase is composed of two adverbs ቶሎ ቶሎ ‘briskly’ that make up the AdvP having the later ቶሎ as its head.

2.5.4. Sentence Formalism

A sentence is a big phrase that expresses a complete idea [8]. For example, the sentence ሁለት ትላልቅ ልጆች በመኪና ወደ ጎጃም ትላንት ሄዱ።። ‘Yesterday two big children went to Gojam by car’, transmits a complete idea because it answers the questions ‘who?’, ‘What?’, ‘Where?’, ‘How?’ and ‘When?’. A sentence is formed by combining two phrases: an NP and a VP. An NP always precedes a VP. All the remaining phrase types are subcomponents of a sentence and are found within one of the above main components of a sentence. In the above example, one can understand that it is composed of the NP (ሁለት ትላልቅ ልጆች) [Two big children] and the VP (በመኪና ወደ ጎጃም ትላንት ሄዱ) [went to Gojam by a car yesterday] which, in its turn, consists of VP (ትላንት ሄዱ) and PP (በመኪና ወደ ጎጃም), and so on. Based on the number of verbs they contain sentences can be classified into two: simple and complex sentences [7].

Amharic Simple Sentences: A simple Amharic sentence consists of an NP, which is the subject, followed by a verb phrase that comprises the predicate. ስጋው ፈጭማ ነው:: the meat is white. Example: እስቴር አስተማሪ ሆነች, ‘Aster became a teacher’; ካሳ ምሳውን አልበላም:: ‘Kassa did not eat his lunch’.

Amharic Complex Sentences: Complex sentences are composed of complex phrases such as NP, VP, or AdjP. The pattern of combination could take the form of a complex NP and a simple VP, a simple NP and a complex VP, or both complex NP and VP. A complex NP is one that contains an embedded sentence with in it. The phrase, for instance, ካሳ የገዛው የሱፍ ኮት ‘The wool coat that Kassa bought’ is a complex NP whose head is ኮት ‘coat’. This head has been combined with the complement የሱፍ ‘wool’ in order to produce the simple NP የሱፍ ኮት ‘a wool coat’. This

simple NP, in turn, has combined with the dependent clause ካሳ የገዛው ‘that Kassa bought’ to produce the above complex NP. The presence of the relativizer, የ ‘that’ in it indicates that the clause is a subordinate clause and it cannot stand alone.

2.5.5. Punctuation Marks

When writing Amharic sentence, the individual words in a sentence are separated by two dots (:), but this practice is not exercised especially in type-written texts. The end of a sentence is marked by a square-formed four dots (::). The symbol (፣) represents a comma, while (፤) corresponded to a semicolon. The language has also borrowed some punctuation marks (e.g., ‘!’ and ‘?’) from foreign languages. Some of the Amharic punctuation marks are listed in Appendix I.

2.5.6. Issues to be considered in Amharic Writing System

There are many issues observed regarding the writing system of Amharic language [8]. Some of them are summarized below.

Consonants with Different Forms

As discussed above, Amharic borrowed most of its scripts from Geez. However, it did not select from Geez alphabet those symbols that are only necessary for its consonants. As a result, there are certain phonemes with different symbols, where they have meaning in Geez, but their meaning is not known in Amharic.

Table 3.1 Amharic Alphabets with Different Forms

Character	Characters with the same sound
ሀ	ሃ, ሐ, ኀ, ኃ, ኻ
ሠ	ሰ
ኦ	ዐ
ጸ	ፀ

It is not clear whether one should write “ኃይል” (power) as ሀይል, ሐይል or ኀይል, also “ፀሀይ” (sun) as ፀሐይ or ጸሀይ. As a result there arises some confusion and inconsistencies in Amharic spelling.

Certain Interchangeably Used Orders

There is confusion regarding the first order and the fourth order of some consonants. For example, it is not clear which one to choose ሀ (as ሀይሉ) or (as ሃይሉ) to spell 'Häylu' – name of a person. As a result, one can find the same word "Häylu", spelled differently in six forms, which are ሀይሉ, ሃይሉ, ሐይሉ, ሓይሉ, ኀይሉ, and ኃይሉ. Similarly, "eye" can have three forms, which are ዐይን, ዓይን, and አይን, with pronunciation "äyn".

In addition, the second order of the consonant w, which is ዉ, and its sixth order, which is W are interchangeably used and there is no consistency. For example, one can find the word 'dog' spelled as ውሻ and ዉሻ ("wshä" and "wushä" respectively).

Compound Words

Another issue to be considered when writing Amharic is the division of compound words. For example, it is not clear which one ወጥቤት "wät'bet" or ወጥ ቤት "wät' bet" is the correct spelling for 'kitchen'. Another issue is with regularizing spellings and regularizing punctuation. For example, the word "sämtoäl" ('he has heard') may be spelled as ሰምትዋል, ሰምቷል or ሰምትዋል. Translation from foreign words into Amharic is also another issue. For example, the word 'television' translated into different forms, which include: ቴሌቭዠን, ቴሌቭዥን, ቴሌቪዥን.

Abbreviations

There is also an issue when spelling abbreviations. For example, when abbreviating the phrase ዓመተ ምህረት (in the year AD), one can find ዓ.ም., ዓ.ም or ዓም as possible abbreviations. Similarly, the use of hyphen is not consistent.

2.6. Structure of Standard Query Language (SQL) Statements

Structured Query Language (SQL) is an international standard (and American National Standards Institute – ANSI) for definition and manipulation of databases [24]. Almost all database vendors support SQL while adding their own SQL extensions.

SQL is an abbreviation for Structured Query Language and it is a tool for organizing, managing, and retrieving data stored by a computer database. It works with one specific type of database, called a relational database. SQL is used to control all of the functions that a DBMS provides for its users including:

- **Data Definition**: lets a user to define the structure and organization of the stored data and relationships among the stored data items.
- **Data Retrieval**: allows a user or an application program to retrieve stored data from the database and use it.
- **Data Manipulation:** allows a user or an application program to update the database by adding new data, removing old data, and modifying previously stored data.
- **Access Control**: can be used to restrict a user's ability to retrieve, add, and modify data, protecting stored data against unauthorized access.
- **Data Sharing:** used to coordinate data sharing by concurrent users, ensuring that they do not interfere with one another.
- **Data Integrity:** defines integrity constraints in the database, protecting it from corruption due to inconsistent updates or system failures.

2.6.1. Retrieving Data from Databases

To query data from database the SELECT statement of SQL is used. It contains six clauses called SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY. The SELECT and FROM clauses of the statement are always required. The remaining four clauses are optional. They can be included in a SELECT statement only when you want to use the functions they provide [24]. The formal syntax of SQL statement to query data from the database is:

```
SELECT [ DISTINCT | ALL ]
  column_expression1, column_expression2, ....
  [ FROM from_clause ]
  [ WHERE where_expression ]
  [ GROUP BY expression1, expression2, .... ]
  [ HAVING having_expression ]
  [ ORDER BY order_column_expr1, order_column_expr2, .... ]
```

The SELECT statement retrieves data from a database and returns it in the form of query results. Every SELECT statement produces a table of query results containing one or more columns and zero or more rows. For simple queries, the English language request and the SQL SELECT statement are very similar. When the requests become more complex, more features of the SELECT statement must be used to specify the query precisely.

The SELECT Clause

The SELECT contains SELECT statement, which specifies the data items to be retrieved by the query [16]. The items are usually specified by a select list, a list of select items separated by commas. Each select item in the list generates a single column of query results, in left-to-right order.

A select item can be:

- **A column name**: when list of column names appears as a select item, SQL simply takes the value of those columns from each row of the database table and places it in the corresponding row of query results.
- **A constant:** specifying that the same constant value is to appear in every row of the query results.
- **An SQL expression**: indicating that SQL must calculate the value to be placed into the query results, in the style specified by the expression.

The FROM Clause

The FROM clause specifies list of table names which contain the data to be retrieved by a query. It consists of the keyword FROM, followed by a list of table specifications separated by commas. Those lists of table names are the source tables of the query and of the SELECT statement.

The WHERE Clause

When using WHERE clause, only certain rows of data are included in the query results. It is used to specify the rows you want to retrieve. It consists of the keyword WHERE followed by a

search condition that specifies the rows to be retrieved. The desired rows are specified by the search condition.

A search condition selects rows using either of the following methods; by comparing values, by checking a value against a range or set of values, by matching a string pattern, and by checking for NULL values.

The simple search conditions return a value of TRUE, FALSE, or NULL when applied to a row of data. To form more complex search conditions, AND and OR keywords can be combined with simple search conditions. The keyword AND is used to combine two search conditions when both must be true. The keyword OR is used to combine two search conditions when one or the other or both must be true.

The five basic search conditions are:

Comparison Test: It compares the value of one expression to the value of another expression. It computes and compares the values of two SQL expressions for each row of data. The expressions can be as simple or complex. For example, simple expression can be s column name or a constant and complex expression can be arithmetic formulas. There are six different ways SQL offers to compare the two expressions. Those are: equal to (=), not equal (!>), less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=).

Range Test (BETWEEN): This is used to check whether the value of an expression falls within a specified range of values. It involves three SQL expressions. The first one defines the value to be tested. The second one defines the low end of the range and third one defines the high end of the range to be checked.

Set Membership Test (IN): This is used to check whether the value of an expression matches one of a list of target values.

Pattern Matching Test (LIKE): This is used to check whether the value of a column containing string data matches a specified pattern. There are different wildcards used in SQL expression for different purposes. Those are: The percent sign (%) which matches any sequence of zero or more characters; the underscore (_) which matches any single character. Wildcard characters can

appear anywhere in the pattern string, and also several wildcard characters can be within a single string.

Null Value Test: It checks whether a column has a NULL (unknown) value.

GROUP BY clause

This is used to specify a summary query. A summary query groups together similar rows instead of producing one row of query results for each row of data in the database. Then it produces one summary row of query results for each group.

HAVING clause

This works in a similar manner as WHERE clause. It is used to include only certain groups produced by the GROUPBY clause in the query results. It uses a search condition to specify the desired groups.

ORDER BY clause

This is used to sort the query results. It specifies that the query results should be sorted in ascending or descending order, based on the values of one or more columns. If it is omitted, the query results are not sorted.

2.6.2. Query Results

The result of SQL query is a table of data. The query results always have the same tabular, row/column format as the actual tables in the database. The difference is the structure of the tables, columns, rows and the data is not the same on the result of a query and the tables in the real database.

Simple Queries

The simplest SQL queries request all data or few columns of data from a single table in the database. Only SELECT clause and FROM clause is needed for simple queries. The SELECT clause specifies all data or the requested columns and the FROM clause specifies the table that contains the requested data.

Selecting All Columns (SELECT *)

The result of SELECT * clause displays the contents of all the columns of a table. Displaying all contents usually can be useful when you first encounter a new database and want to get a quick understanding of its structure and the data it contains. For this, SQL uses asterisk (*) in place of the select list as an abbreviation for "all columns".

Duplicate Rows (DISTINCT)

This is used to avoid displaying duplicate rows in the result of SQL statement. This is done by including in the query the primary key of a table in its select list. Then, every row of query results will be unique because the primary key has a different value in each row. i.e., if the primary key is not included in the query results, duplicate rows can occur.

Rules for Single-Table Query Processing

To generate the query results for a select statement follow these steps [24]:

i. Start with the table named in the FROM clause.
ii. If there is a WHERE clause, apply its search condition to each row of the table, retaining those rows for which the search condition is TRUE, and discarding those rows for which it is FALSE or NULL.
iii. For each remaining row, calculate the value of each item in the select list to produce a single row of query results. For each column reference, use the value of the column in the current row.
iv. If SELECTED DISTINCT is specified, eliminate any duplicate rows of query results that were produced.
v. If there is an ORDER BY clause, sort the query results as specified. The rows generated by this procedure comprise the query results.

Combining Query Results (UNION)

Sometimes, it is necessary to combine the results of two or more queries into a single table of query results. UNION feature of the SELECT statement has this capability. The UNION

operation can be used within a SELECT statement to combine two or more sets of query results into a single set.

Unions and Sorting

The ORDER BY clause cannot be used in either of the two SELECT statements combined by a UNION operation because they are fed directly into the UNION operation and is never visible to the user. However, one can specify the ORDER BY clause after the second SELECT statement. This sorts combined set of query results produced by the UNION operation. Since the columns produced by the UNION operation are not named, the ORDER BY clause must specify the columns by column number.

Multiple UNIONs

The UNION operation can be used repeatedly to combine three or more sets of query results.

CHAPTER THREE

RELATED WORKS

Native speakers and users, who want to use their own natural languages, are the basis for research on natural language query interface. The difficulty they face is not only about to know the language (English) but also they have to know the programming language/SQL - Syntax/ to get information from database.

3.1. Information Retrieval

Information retrieval can be defined as finding data which satisfies information needs from within large collections of storages. It deals with the representation, storage, organization of, and access to information items. Its aim is to find and retrieve documents relevant to a given query, usually where documents and query are in the same language. The following are some works done on IR.

Seid Muhie *et'al.* [30], presents a Question Answering System for Amharic (AQA). Novel technique were developed to determine the question types, possible question focuses, and expected answer types as well as to generate proper Information Retrieval query, based on Amharic language specific issue investigations. This research helped users in getting relevant documents from large collection of free-text documents.

Mequannint Munye [18] designed a generic model and developed a bilingual Web search engine for Amharic and English languages. The search engine comprises query preprocessing for both Amharic and English query preprocessing, query translation for both Amharic and English languages, Amharic search engine and English search engine. From this research one can find techniques on how to query and retrieve data in both languages. This research helps users in getting relevant documents from large collection of documents in both English and Amharic languages.

Omar *et'al.* [22] contributed a new technique for resolving lexical ambiguity in Natural Language Question posed to a Question Answering system. Their approach integrates context knowledge and concepts ontology of a domain, into a shallow natural language processing

(SNLP) technique for Lexical semantic ambiguity, which may occurs when a user's Natural Language Question contains words that have more than one meaning. In this paper, a novel approach called CKCO for resolving lexical semantic ambiguity in Natural Language Question posed to QA system is proposed. The proposed approach is obtained by combining two pieces of knowledge; context knowledge and ontology of concepts knowledge of interesting domain, into a shallow natural language processing (SNLP) technique. The result showed that the proposed approach is capable of resolving ambiguous words in the Natural Language Question which consists of multi words.

3.2. NLIDB Systems Using Different Approaches

Sujatha *et'al.* [6] have outlined different architectures to build Natural Language Interface for databases. They briefly discussed the components of NLIDB, its advantages and disadvantages. Various approaches to build the NLIDB are mentioned with their strength and limitation. This paper also gave good information on how to develop such kind of systems and explains various limitations.

Michael *et'al.* [20] presented a state-of-the-art authoring system for natural language interfaces to relational databases. They tried to study the structure of SQL commands and used semantic grammars to map user requests to SQL command. Their system was graphical user interface based which is very suitable for end users.

Frank *et'al.* [12] described a Natural Language Interface to database systems which is based on the query formation capabilities of a high-level formulator. The formulator relies on the Semantic Graph of the database, which is a model of the data stored in the database. The interface accepts user's input in natural language and extracts the necessary information needed by the formulator. This extraction process is performed using keywords obtained from the Semantic Graph and the database. Because keywords have several meanings within a given domain, keyword meaning disambiguation is done using a statistical approach which involves comparing vectors of n-grams. Processing natural language to extract query information is done using keywords and statistical keyword disambiguation using a vector similarity measure. They developed a method for automatically formulating a query (in SQL) based on incomplete

information from the user. The formulation is done utilizing a Semantic Graph of the database which describes the objects and relationships between objects modeled in the database. The semantic graph consists of three elements: nodes, attributes and links. Statistical methods of the keyword from the user's input are used by the high-level query formulator to compose a formal database query. The system uses no grammars or much outside knowledge and most of the knowledge utilized by the system can be obtained from the database itself. The system is the most suitable for applications where the user input is short and simple, and it is necessary to support domains with a limited ontology.

Natural Language Interface to database system developed by Nelu, *et'al*. [21] accepts queries in English language and attempts to understand them. It maintains its own dictionary. This dictionary contains words related to database and relationships. In addition to this, interface also maintains pre-defined data structures and in order to interpret the query. They present the architecture, of the Intelligent Natural Language Interface to Database system. The system is designed to accept any relational database schema. The system accepts users' natural language sentences as input, parses them semantically and builds an SQL query for the database. The core functionality is based on the semantics and rules, which can be modified by the system administrator. The system is composed of two modules: a pre-processor and a run time processor.

Saravjeet *et'al*. [33] uses NLP or interfacing with the Database using natural language. Their system is developed to be able to execute both DDL and DML quires, input by the user in his/her natural language (English). A limited Data Dictionary is used where all possible word related to a particular system are included. The Data Dictionary of the system must be regularly updated with words that are specific to the particular system. Ambiguity among the words will be taken care of while processing the natural language. The results show that their software is correct and handles the SQL Queries without any problem. The methodology adopted for preparing their system is classified through - Morphological Analysis: Individual words are analyzed into their components and non word tokens such as punctuation are separated from the words; Syntactic Analysis: Linear sequences of words are transformed into structures that show how the words relate to each other; Semantic Analysis: The structures created by the syntactic analyzer are assigned meanings; Discourse integration: The meaning of an individual sentence may depend

on the sentences that precede it and may influence the meanings of the sentences that follow it; Pragmatic Analysis: The structure representing what was said is reinterpreted to determine what was actually meant.

Porfírio *et'al.* [23] designed an architecture based on an Intermediate representation Language (LIL), where the natural language question is transformed into an intermediate logical query before the final translation into an SQL query. The system has two modules. The first module controls the linguistic component where user questions are transformed into LIL expression while the second component is in charge of database connection, translating LIL expression into SQL expression. This paper shows a method to completely separate linguistic component from database component which makes the translation process easy.

Ramasubramanian and Kannan [26] designed and implemented a temporal natural language interface to query from temporal databases. They implemented temporal event matching approach in the form of pattern recognition to capture events in evolution. Thus system recognizes the evolutionary nature of time, more naturally, compared to traditional languages.

Rajendra and Manish [25] also made rules to tackle linguistic phenomena using shallow parsing and discuss advantages of a novel Natural Language Interface comprising of shallow parsing based algorithms in conjunction with some intelligent techniques to train the system. Results showed that this approach can analyze a wide range of questions with high accuracy and produce reasonable textual responses. This paper uses a technique to answer certain language dependant phenomena such as semantic symmetry and ambiguous modification.

A research done by Alessandra and Alessandro [1] deals with structural kernels and their combinations for inducing the relational semantics between pairs of Natural Language (NL) questions and Standard Query Language (SQL). In this research development of NLIDB is based on syntactic analysis. The goal of the research was the development of a NLIDB that automatically maps NL questions into SQL queries based on a machine learning approach by using syntactical representation of machine training questions. The researchers assess the high effectiveness of product kernels and the feature pair spaces, which highly improve the traditional linear kernel. The researchers used a set of structural kernels, e.g. Sequence and Tree Kernels, and Support Vector Machines (SVMs), which is a well-known algorithm to map natural

language questions to SQL. They used BOW (Bag-Of-Words), which is information retrieval mechanism, because it has been shown to be effective to represent textual documents. For questions, they used Charniak's syntactic parser while for queries they implemented an ad-hoc SQL.

Michael Minock [19] introduces the STEP system for natural language access to relational databases. STEP adopts a phrasal approach; an administrator couples phrasal patterns to elementary expressions of tuple relational calculus. This 'phrasal lexicon' is used bi-directionally, enabling the generation of natural language from tuple relational calculus and the inverse parsing of natural language to tuple calculus. This ability to bi-information of user interest is housed within relational databases and interaction is purely textual; user requests are single sentences of natural language and answers are multiple sentences of natural language. The advantage of STEP's phrasal approach is that it avoids many of the difficulties associated with ambiguity in large scale domain independent grammars and maps directly to the underlying database relations. Additionally, via fudging operations, STEP finds acceptable parses for many non-grammatical inputs, a common occurrence in practice. Finally a phrasal approach allows for easier specification of idiomatic and idiosyncratic domain language. A disadvantage of STEP's approach is that a specific phrasal lexicon must be authored for each new database.

3.3. NLIDB System for Different Languages

Besides the English language, a lot of researches have been done for different languages. The work by Rashid *et'al.* [27] presents a good architecture of NLIDB system for Urdu language. This language is used in Pakistan, India and in South Asia. The researchers tokenize the input query, extracted patterns and used attribute value mapping algorithms to map input queries to SQL. They constructed domain specific dictionary and semantic knowledgebase with some rules. They used the dictionary and knowledgebase to efficiently transform the query into SQL statements. They have got quite satisfactory experimental result by applying their model in school management system and employee information system databases.

Spanish Natural Language Interface for a Relational Database Querying System was developed by Rodolfo *et'al* [28]. This system has two modules: database query and database customization.

The database query module contains lexical parsers, syntax checker, semantic analyzer and result module while database customization contains domain customization and logical data. Their system has a promising result to continue further research on the area.

Amandeep [3] developed a system for Punjabi Language Interface to Databases. Similar to Urdu, this language is used in India and Pakistan. The system is database specific. It is developed for Agriculture database since most Punjabi's people are farmers. Using agriculture database, accepts query in Punjabi language that is translated into SQL query, by mapping the Punjabi language words, with their corresponding English words with the help of database maintained. Major concepts in this system are: Tokenizing the input query and searching that token in the database whether it belongs to table name or column name; after analyzing query whether it is simple query or a query with condition, system will form appropriate SQL query; Execute SQL query on the database to retrieve required information. And provide output to the user. The system executes query for selection of where table, queries for selection of certain columns of tables and queries for selection of rows from certain columns.

Mekuria [17] designed Amharic Language query interface, AmharicQL, to interact with a database system in which the main objective of AmharicQL is to provide users a native language query interface. Moreover, it offers simplicity in the user's query formulation process. The research gave a good insight for users with Amharic Language, who knows the SQL – query syntax. However, it doesn't consider those who are non-expert for SQL-commands. It was syntax based, which the English SQL Commands were translated into Amharic.

From the works presented above, it is clear that information can be retrieved using Natural Languages. One of these systems is NLIDB. There are different architectures, design, techniques etc to develop NLIDB systems. Different NLIDB systems have been developed for different languages and for different databases such as XML, temporal and relational databases. NLIDB systems can be effective way of accessing information for users when different NLP techniques (morphological, syntactic, semantic and others) are used. These make the system more flexible.

When we come to Amharic Language, to the best of the researcher's knowledge, there is no research work done that enables users to query data from database using Amharic Language statements. In the proposed research, Amharic Language query system will be modeled. This is done by generating SQL command from Amharic.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION OF AMHARIC QUERY SYSTEM FOR BI-LINGUAL DATABASES

The Amharic query system accepts query statements in Amharic language only. SQL statements are generated from the Amharic statements and it will be executed by the query engine. The result of the query, which will be retrieved from the database, can have both English and Amharic contents. When designing the database, the database name, table/relation name and attribute name are expected to be in English language but contents of the database can be both in English and Amharic language.

In this chapter, different components of the system are explained and detail description of the modules and the sub-modules, their interaction and the flow of the query execution, the implementation of the whole system and the algorithms designed to perform each of the tasks in the stated modules is presented. The techniques used in each module, the algorithms developed to identify and map database elements and to generate SQL statements are also explained.

4.1. Types of Queries

In users query, ambiguities can be created because of different reasons. Some of them are:

- different words exist which have the same meaning; for example, ማለዳ and ጠዋት, ምን ያህል and ስንት, አየ and ተመለከተ have the same meaning,
- based on their understanding, users can interpret things differently,
- Users can say unrelated things to refer something; semantic,
- For the same information need, users may state different query expressions,
- The ability to use language meaningfully with attention to grammatical structure, and
- Because of the language ambiguity, query statements may be interpreted differently.

In order to avoid these ambiguities, users must include table names in their query. If they want to query only few attributes, they must list them in their query as stated in the translated database

elements by the database element identifier. This will avoid ambiguities created and hence maximize the performance of the system.

Besides, if users want to query data with additional criteria, they should include connector and operator keywords like ሁሉንም, ስንት, እና, ወይም, እኩል, የበለጠ, ያነሰ, etc. These keywords are explained later in this chapter. The keyword ሁሉንም will retrieve all attributes of a table. When users include the word ስንት in their query, the system maps the keyword to COUNT(*).

For example, consider a database which has the following tables.

Table 4.1 – List of Tables for a Sample Database

Tables	**Amharic Equivalent**
Subscriber	ተመዝጋቢ
Product	ምርት
Customer	ደንበኛ
Balance	ሂሳብ
DependantProduct	ጥገኛምርት

Take only the Customer table and see its attributes.

Table 4.2 – List of Attributes for Sample Customer Table

Attributes	**Amharic Equivalent**	**Data type**
CustID	መለያ	Nvarchar
FirstName	መጀመሪያስም	Nvarchar
LastName	መጨረሻስም	Nvarchar
Email	ኢሜይል	Nvarchar
Address	አድራሻ	Nvarchar
Phone	ስልክ	Nvarchar

Using this information, users might formulate the following queries.

- የሁሉንም ደንበኛ መረጃ አሳየኝ
- ከደንበኛ ላይ መጀመሪያስም እና ስልክ አሳየኝ

- ከደንበኛ ላይ መጀመሪያስም ከአበበ ጋር እኩል የሆነ አሳየኝ
- አጠቃላይ ደንበኛ ብዛት ስንት ነው?

Detail explanation about each query

- የሁሉንም ደንበኛ መረጃ አሳየኝ

  The word ደንበኛ is identified as a Table name. In the query, there is no word identified as an attribute name as well as operator and value. So the SQL statement will be to select all elements of the table.

  SQL Satement - Select * from CUSTOMER;

- ከደንበኛ ላይ መጀመሪያስም እና ስልክ አሳየኝ

  Here the word ደንበኛ is identified as Table name and words መጀመሪያስም and ስልክ are identified as attribute names. There is no operator or value identified. So the SQL statement will be generated to select first name and phone from the customer table.

  SQL Statement - Select FIRSTNAME, PHONE from CUSTOMER;

- ከደንበኛ ላይ ሁሉንም መረጃ መጀመሪያስም ከአበበ ጋር እኩል የሆነ አሳየኝ

  Here we have conditional statement. The word ደንበኛ is identified as table name. The word መጀመሪያስም is identified as column name. The word እኩል is identified as operator. It will be translated to equal. The word ሁሉንም will be mapped to *. The word አበበ will be a value. Customer table, FirstName attribute, equal operator and Abebe value is used to generate the SQL statement.

  SQL Statement - Select * from CUSTOMER where FIRSTNAME = 'Abebe';.

- አጠቃላይ ደንበኛ ብዛት ስንት ነው?

  Here the word ስንት changes the SQL statement. ስንት is changed to count and instead of using all in the query, count is used.

The SQL statement - Select COUNT(*) from CUSTOMER;

In the above examples, in all queries the table name, attribute name, operator and value are identified. Thus, the system can transform the queries to their equivalent SQL statements without much ambiguity.

4.2. Model of the Amharic Query System

Linguistic and database components are very common components in NLIDB systems. Our system has four main modules with sub-modules. The four main modules are: the NL query statement preprocessing module, which is the linguistic component of the system; the query mapper module and the SQL generator module, which are the database components of the system and finally the query executer module.

Users can enter their query through user interface. The natural language preprocessor is responsible to make the query suitable for the next phases of preprocessing steps. The stemmed tokens, which are the output of the preprocessor, will be the input for Query mapper module. The query mapper then identifies tables, attributes, operators and values and stores them for further processing. SQL Generator module will use the stored table names, attribute names, operators and values to generate the final output of the system which is an SQL statement. The final module of the system, query executer, will process data from the database after the generated SQL is executed. The architecture of the system is shown in Figure 4.1.

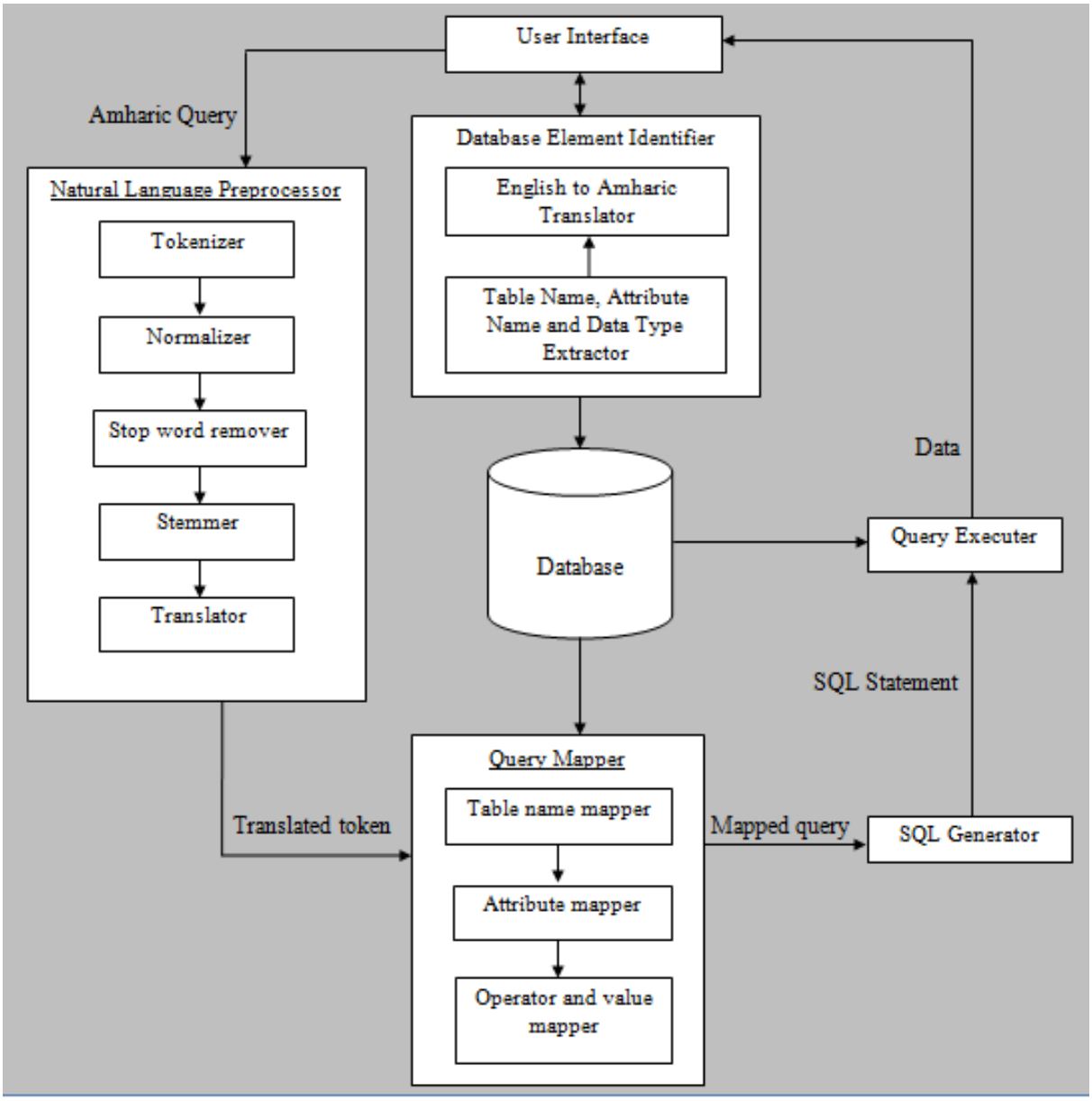


Figure 4.1 – Architecture of Amharic Query System

4.3. User Interface Implementation

User can formulate their query based on the information they get from the interface. They should include table name and attribute names of that specific table related to their requirement in their query. This will make the interface more suitable for both native and novice users.

The user interface is developed using NetBeans's forms and tools. It is responsible to accept the query in the form of Amharic statement and display the generated SQL statement. The interface allows users to choose the database they want to query from before stating their query. Besides these, users can view information about the database they chose on a popup window. After the database is chosen the interface is able to display list of tables for that specific database with their list of attributes of each table translated in Amharic language along with their data type. The translation is needed to make it easy for users to formulate their queries after viewing database elements in Amharic. This is done by translating elements of the database to Amharic language after it is retrieved from the database.

4.4. Database Element Identifier

The database element identifier extracts table names and attributes names with their data types of the selected database and displays it to the user in the user interface. Since the Amharic query system accepts user query in Amharic statement, the displayed database elements have to be in Amharic too. The database elements are translated to Amharic before displayed to the user. For this purpose, in addition to the dictionary we used in translation module, a separate bi-lingual dictionary is used. This dictionary is a text file managed by the system's administrator. It includes list of table names and attribute names in both English and Amharic language. In addition to that, it also includes list of operators of SQL statement. These operators are presented under section 4.6. The system's administrator can modify the dictionary based on the database elements.

4.5. Natural Language Query Preprocessor

NLIDBs are designed to extract information from the database using a query in natural language. So, before the input Amharic query statement is translated to SQL statement, it should be preprocessed for further stages. Different techniques are used to analyze the input query. This module is responsible to analyze the Amharic query statement and prepare input for the query mapper module. Input for this module is Amharic query and translated English tokens are its output. In order to do that, different NLP techniques are used. The NL preprocessor module has

Tokenizer, Normalizer, Stop Words Remover, Stemmer and Translator sub-modules. The processes in the Natural Language preprocessor module are shown in Figure 4.2.

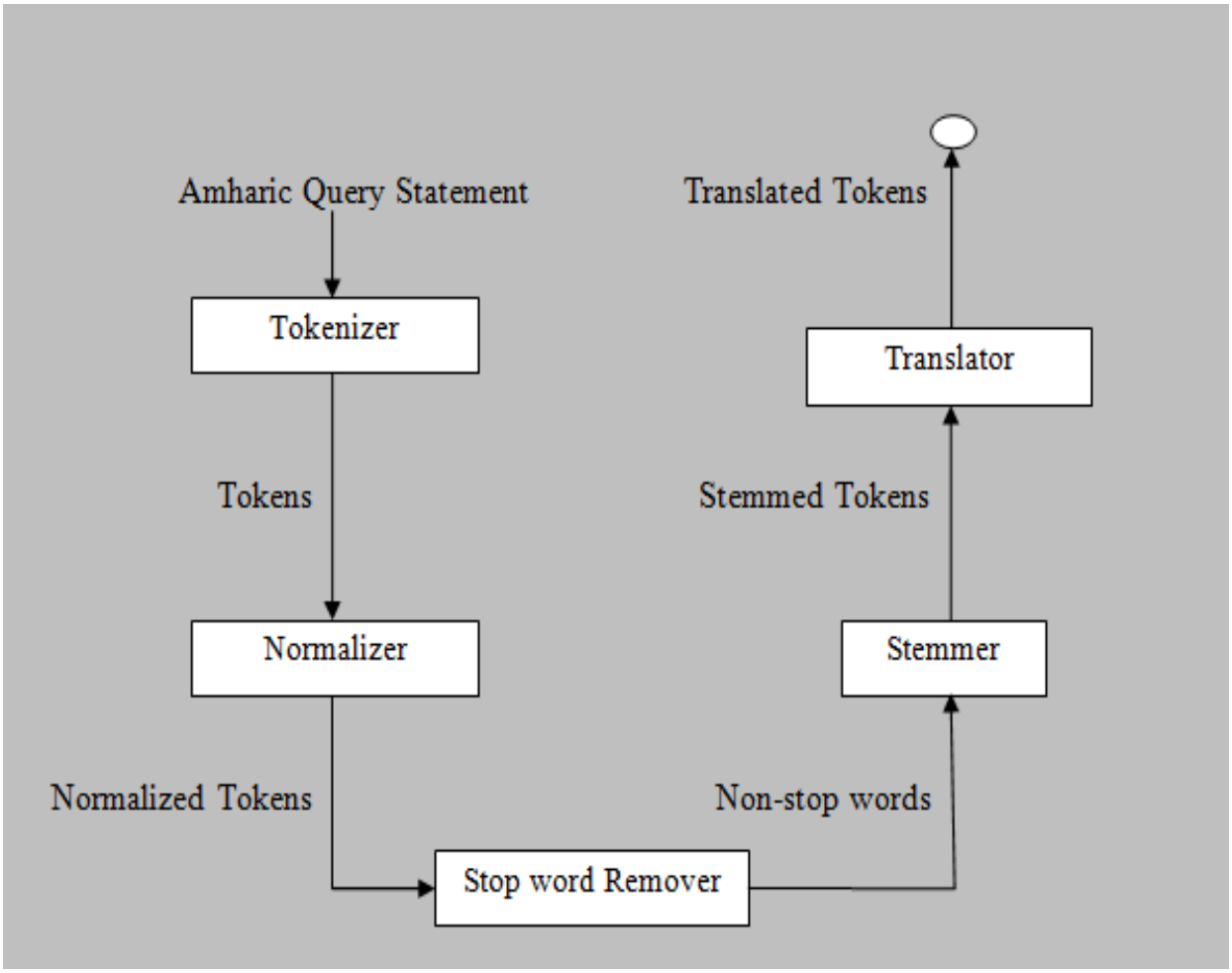


Figure 4.2 – The Processes of Natural Language Preprocessor

4.5.1. Tokenizer

Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. In our case, the output token is a set of words that matches a database element. Each word in a query statement is tokenized and will be ready to be input for the next processing. Our algorithm tokenizes the Amharic query statement by using the white space and list of Amharic punctuation marks as delimiters.

The white space and Amharic punctuation marks will be removed from further processing. For example, the Amharic query: አጠቃላይ ሰራተኛ ብዛት ስንት ነው? tokenized in to [አጠቃላይ] [ሰራተኛ]

[ብዛት] [ስንት] [ነው]. The list of Amharic punctuation marks used as delimiters to tokenize the Amharic query is attached in Appendix I.

4.5.2. Normalizer

Normalization is the process of transforming character, word, text into a single canonical form. Amharic language has characters which have the same sound but different representation. Those characters are replaced by the same character.

In this part Amharic characters which have the same sound but different representation are replaced by the same character. Those Amharic characters are: - ሀ , ሃ , ሐ , ሓ , ኀ , ኃ , ኻ ፤ አ , ኣ , ዐ , ዓ ፤ ሰ , ሠ ፤ ጸ , ፀ. For example, ሃ , ሐ , ሓ , ኀ , ኃ and ኻ are replaced by ሀ and ሑ , ኁ , ኹ are replaced by ሁ.

Besides the characters in Amharic language; words, which have the same meaning, can have the same orthographic forms. Similarly multiple forms of writing and pronouncing the same word are replaced by a single word. These words are collected from different sources. But we haven't used the formal collection of these words because we didn't found. List of these words used in word normalize module are attached in Appendix III. Some examples are:

ጡዋት፣ ጧት - to say morning; are replaced by - ጠዋት

ኢ-ሜይል፣ ኢሜል - to say email; are replaced by - ኢሜይል

ቴሌብሾን - to say television; is replaced by - ቴሌቪሾን

ትላንት - to say yesterday; is replaced by - ትናንት

ደሞዝ፣ ደምዎዝ - to say salary; are replaced by - ደመወዝ

4.5.3. Stop Word Remover

In computing, stop words are words which are filtered out prior to, or after, processing of natural language text. Any group of words can be chosen as the stop words for a given purpose. So, a group of words that are identified as stop words for Amharic language are used for this work. Users might enter words which are irrelevant in SQL generation. So, these stop words, which are

not necessary for further processing, are eliminated. This is done to maximize the relevance of words for the query processing. Only words useful to extract the data from the database are selected. We have used stop words list which are used by the work of Mekuanent Muniye [18] on bi-lingual search engine. From those lists we have carefully identified and excluded words which are useful in the SQL generation module like እና ፣ ወይም ፣ ሁሉንም ፣ እኩል ፣ መሀል ፣ ያነሰ ፤ የመሰለ ፣ ውስጥ because these words are mapped as operator and in select clause in SQL generator module.

We have added some words, which are not useful for further processing, in the stop words list (like አጠቃላይ፣ ብዛት፣ በላይ፣ አሳየኝ፣ ያላቸውን፣ አውጣልኝ). These words minimize the performance of the system. List of Amharic stop words selected specifically for our system is attached in Appendix II.

4.5.4. Stemmer

In Amharic language, words can be inflected with prefix (adding affix before the stem word), suffix (adding affix after the stem word) and infix (adding affix inside the stem word). This module is responsible for reducing inflected or derived word into its stem form. As indicated in most Information Retrieval research documents [18, 30, 32], this will help in finding proper dictionary translation.

In this module, Amharic words which are inflected with prefix and suffix are reduced to their stem form. For example, the word ቤቶች will be stemmed to ቤት. We used stemmer algorithm adopted by the work of Mekuanent Muniye's [18]. This stemmer stems words which are derivates or inflected by adding prefix, infix and suffix to them. But, for our work we excluded infix part because the result of it was totally different. In addition, after the some words are stemmed their format is changed. For example, የአበበ is changed into አበብ after the prefix is removed. So we have edited the stemmer not to change the last character of the word after stemming the word.

The stemmer also changes the format of proper names. For example the name ራህመቶ is changed to ራህመት by changing its last character to Sadis form and ጎጃም to ጎጃ by considering the last character as suffix and removing it when passing through the stemmer. To avoid this first proper

names with their equivalent English name are identified and they are used directly in translation module. We have adopted Mekuanenet Muniye's [18] list of proper names and added around 100 additional proper names.

4.5.5. Translator

Translation is the process of changing or converting word, sentence, text etc. to another language without changing the underlying meaning or message. On the next main modules (Query mapper and SQL generator), the preprocessed tokens are used in searching table names, attribute names, operators and values from the database. The comparison will be performed in English language. So, it is necessary to translate the tokens for further processing. Hence, before generating an SQL statement, the preprocessed query has to be translated from Amharic to English which is the language of SQL statement. Our translation module is responsible to translate Amharic words into their English equivalent. This translation module has 2 components. Those are dictionary look up and transliteration. The process of the translator module is shown below in Figure 4.3.

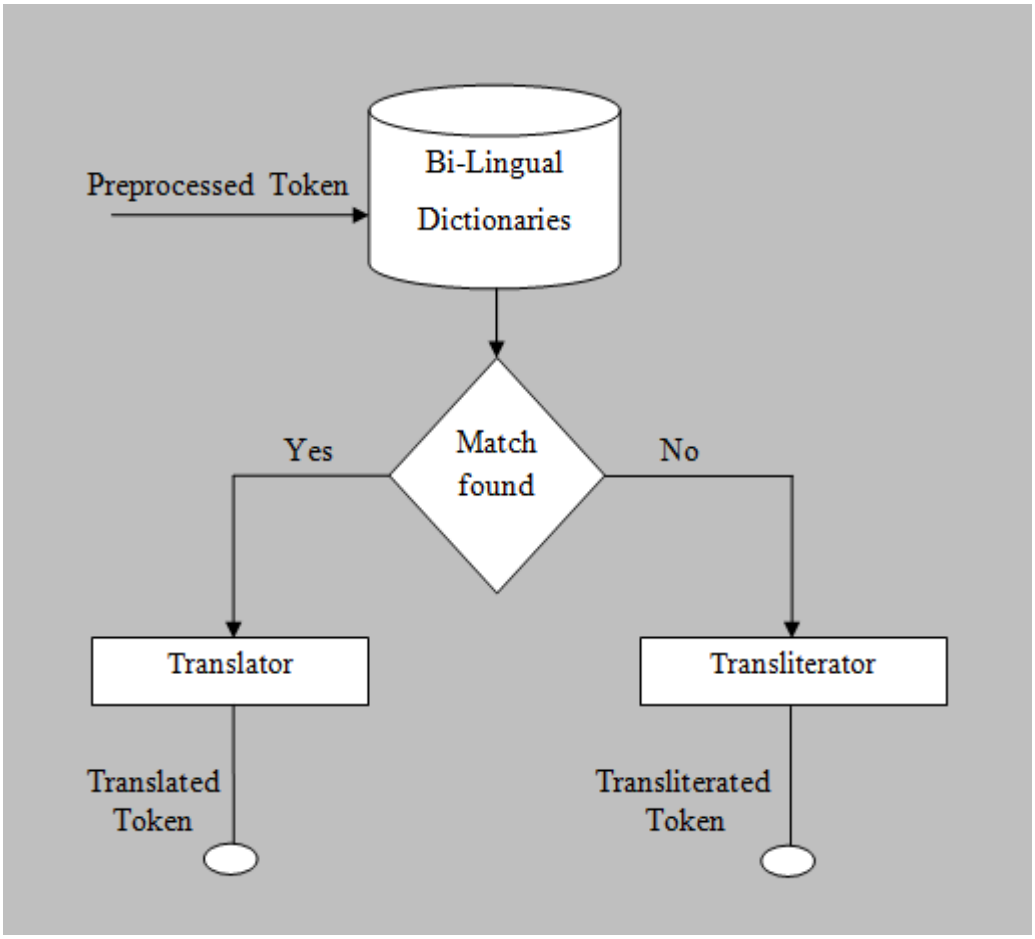


Figure 4.3 – The processes of the translator module

Dictionary Used

The aim of a natural language interface is to facilitate the user to formulate queries in a natural way. This will enable users to write their query in different natural ways. Users query entered in Amharic language should be translated into English before further processing.

For this purpose, we used the bi-lingual dictionary adopted by Mekuanenet Muniye [18]. His dictionary was up to letter S. It was necessary to include the remaining words (from Letter S to Letter Z). So we have carefully selected the remaining words from Aklilu Amsalu (English to Amharic [5] and Amharic to English [4] dictionaries). Dictionary words are selected based on the previously constructed dictionary; which means by excluding stop words and derived words. Finally, we manually constructed dictionary for the remaining words and we merged it with the first one. Around 1000 additional words are added in the bi-lingual dictionary.

In addition to the bilingual dictionary, we have used another two dictionaries to translate proper names and Amharic operators. As explained in section 4.5.4, identified proper names shouldn't pass through stemmer. The translation module first finds a match for tokens in proper names. If match is found, the token will be automatically translated. If match is not found, match of tokens are further searched in Amharic operators dictionary. As it is explained in database element identifier part, the bi-lingual dictionary (in this case Amharic operator dictionary) is updated by administrator of the system and it includes list of table names attribute names and operators of SQL statement in both English and Amharic language. If match is not found in Amharic operator dictionary, then it is further searched in the manually constructed bi-lingual dictionary. Finally, if match is not found in all the three dictionaries then, transliteration is applied. The algorithm used to translate Amharic token to English word is shown on the Figure 4.4 below.

Transliteration

Transliteration is the process of representing letters and words of one language in to the corresponding characters and words of another language. Although language changes, pronunciation as well as alphabetical arrangements more or less remains unchanged.

Transliteration gives the word from a different language in letters that you can understand so as to be able to pronounce it.

Our bi-lingual dictionary contains around 5000 words. Which means all words cannot be found in the bi-lingual dictionary. For words which can't be found in the bi-lingual dictionary or in the database elements list or in the list of operators, transliteration is implemented. In which case, the Amharic token is changed into English letter by letter. For this work, System for Ethiopic Representation in ASCII (SERA) convention is used as a reference to develop the transliteration module. SERA is a convention used for finding the closest match between the Latin and Ethiopic phonetic system. However, minor convention modifications have been done in some order of the Amharic characters to fit for our system. For example, each fifth order (*hamis*) of each Amharic character is represented by its consonant representation plus "E" in SERA, such as: H=hE, M=mE, etc, and the sixth order (*sadis*) is represented by is consonants plus "i" in SERA, such as: H=hi, M=mi, etc,. To fit our system used H=hie, M=mie and L=hi, M=mi etc to pronounce these characters in English language. Amharic characters and equivalent English characters used in this module are attached in Appendix III.

```
Input: Amharic bag of words
Output: Translated English bag of words
For each Amharic token in a question
    Search token in proper names dictionary and translate
    If match not found in proper name list
    Search token in Amharic operator dictionary and translate
    End If
    If match not found in Amharic operator list
    Search token in bi-lingual dictionary and translate
    End If
    If match not found in Bi-lingual dictionary
    Apply Transliteration
    End If
End For loop
```

Figure 4.4 – Algorithm for Translation Module

4.6. Query Mapper

In order to map the Natural language preprocessing result to appropriate SQL statements we need to have the list of table names, the list of attribute names, the list of operators and values for the condition in the where component of query statement. The Query Mapper module is responsible in transforming the preprocessed user query into proper SQL statement. In our case, it means it maps set of tokens to set of database elements. In this module Amharic stemmed tokens, which are the output of preprocessing module, will be mapped to table names, attribute names, operators and values. The Query Mapper module has three sub-modules. These are table name mapper, attribute name mapper and operator and value mapper. Each algorithm of this module is explained below.

4.6.1. Table Name Mapper

In the table name mappped algorithm, each token is compared with list of table names of the selected database, once the database is selected and the user stated the query. Users are expected to enter table names and attribute names in to the system. If a token match one of the table names, then that token will be stored as table name for the SQL generator module. Which means the system will map the token with the table to which that query is associated.

For example, if user's query statement is የሁሉንም ደንበኛ መረጃ አሳየኝ, then after translated to English, each word is compared to the table names of the database. Since there is CUSTOMER table, the word ደንበኛ will be mapped to it. So the word 'CUSTOMER' will be stored as table name. But if none of the tokens match the table names the user will be requested to enter the query by including table name of the table that is required to get information from.

4.6.2. Attribute Name Mapper

Once the table is identified, the next step is looking for attribute names for that specific table identified from the remaining tokens. Attribute name identification algorithm compare the remaining tokens with attributes of the specific table. The system will use this algorithm if and only if table name identification algorithm found match for table name. The same as table names, when attributes are found from the remaining token, it will be stored. This is an iterative process since more than one attribute might be chosen as a result.

If any of the remaining tokens does not have a map in the attribute list of the specific table, there will be no attribute name stored. Only table names are stored. The algorithm for identifying tables and identifying attributes are shown on Figure 4.5 below.

```
Input: English bag of words
Output: Table names and Attribute names
For each translated token in a query
        Check tokens match Table names
        If match exists
                Store token as Table name
                For each translated token in a query
                        Check tokens match Attribute name of the Table name
                        If match exists
                        Store token as Attribute name
                        End If
                End for loop
        End If
End For loop
```

Figure 4.5 – Algorithm for Table Name and Attribute Name Identification

4.6.3. Operator and Value Mapper

In this module, two different algorithms are implemented. These algorithms will be applied if and only if there exits table name and attribute name of the selected database in the user query phrase. The structure of the WHERE clause in SQL statement uses operators and values. So the first algorithm maps the tokens with the SQL operators and the second algorithm maps tokens to values if operators exist in the query. As it is presented in the translation part, Amharic

conditional words (operators) are identified and they are translated to English operators. In this list, some of the words are excluded from stop words list to exclude confusion with operator names and some are translated from English to Amharic.

For example the Amharic word እኩል will be translated to 'Equal' and mapped with =, the word like ያነሰ will be translated to 'Less than' and mapped to <, the word የበለጠ will be translated to 'Greater than' and mapped to >, the word መሀል- will be translated and mapped to 'BETWEEEN', the word like የመሰለ will be translated and mapped to 'LIKE'. Identified Amharic conditional words and their equivalent where operators are shown in the Table 4.3 below.

Table 4.3 – List of Amharic Operators

Amharic Conditional words	Operator	Description
እኩል	=	Equal
እኩል ያልሆነ	!=	Not equal
የበለጠ	>	Greater than
ያነሰ	<	Less than
መሀል	BETWEEN	Between an inclusive range
የመሰለ	LIKE	Search for a pattern
ውስጥ	IN	To specify multiple possible values for a column

Once the Amharic operators are identified and translated to English, the next step is to convert English operator words into SQL operator symbols. As the above table 4.3 shows, after the English conditional word is converted to SQL operator symbols, then the symbol will be stored as operator for further processing.

In value identification algorithm the remaining tokens which are not considered as stop words, table names, and attribute names and operators are considered as values. It works in 3 ways. Case 1: if the token is a number, then it will be considered as a value; Case 2: if the remaining token is found in proper names list, then it will be taken as a value; Case 3: if case 1 and case 2 are not met, then simply the rest of the token will be taken as a value to construct the SQL based on the where operator selected.

```
Input: English bag of words
Output: Operators and Values
For each translated token in a query
        Check tokens match Operators
        If match exists
                Translate token into operator symbol
                Store the operator symbol
                For each translated token in a query
                        Search for values
                        If value found
                        Store token as a Value
                        End If
                End for loop
        End If
End For loop
```

Figure 4.6 – Algorithm to map WHERE clause

4.7. SQL Generator

After the query is processed by the Query Mapping algorithm, it is equipped to be transformed to SQL. In this module, the output of Query Mapper module (the mapped table name, attribute name, operators and values) are used to construct the equivalent SQL statement. Which means the output of this module is SQL statement. The SQL statement can have different structure based on the query typed by the user on the interface. The generated SQL statement will be displayed for the user. Users can then execute it to get the query result.

Classification of SQL statements

The output of SQL generator module is classified into three different structures of SQL statements. The first classification is SQL statement that retrieves all data. It is in the form of "Select * from tablename;". This kind of SQL statement will be generated if any of the remaining tokens does not have a map in the columns list of the specific table or if the system finds ሁሉንም keyword in the tokens. But the keyword ስንት exists instead of the word ሁሉንም, then the format of the SQL statement will be "Select count (*) from tablename;".

Another classification is SQL statement includes table name and one or more of attributes of the specific table based on the users' requirement. It is in the form of "select column1, column2, column3, . . . from table_name;".

The third classification of SQL statement includes table name and the where condition or it might include table name, column name and where condition operators. It might be in the form of "select * from table_name where column_name operator 'value';". SQL statement in the form of "Select column1, column2, column3 from table_name where colum_name = 'value';" will be generated if the output of Query Mapper module identifies all tablename, attribute names, operators and values. But if the keyword ስንት exists in the query, the format of SQL statement will be changed to "Select count(*) from table_name where colum_name operator 'value';". The algorithm used in generating SQL statements is shown below.

```
Input: Table names, Attribute names, Operators and Values
Output: SQL statement
If all Table name, Attribute name, Operator and Value exists
        Generate SQL – select attributes from table_name where attribute_name operator 'value';
Else if only Table name, Operator and Value exists
        If count keyword is found
        Generate SQL – select count(*) from table_name where attribute_name operator 'value';
        Else
        Generate SQL – select * from table_name where attribute_name operator 'value';
        End If
Else if only Table name and Attribute name exists
        Generate SQL – select attributes from table_name;
Else if only table name exists
        If count keyword is found
                Generate SQL – select count(*) from table_name;
        Else
                Generate SQL – select * from table_name;
        End If
End If
```

Figure 4.7 – Algorithm to generate SQL Query Statement

4.8. Query Executer

This is the module of the system that is responsible to collect the data from the database and display it to the user. It is developed by using java ODBC database connectors. Once the SQL statement is generated it will be visible to the user. The user can then run the script and see the result data on the interface. So that users can collect the information they need from it. Since this is the function of the query engine component of the database in use, no special algorithm is developed.

CHPTER FIVE

EXPERIMENT

Experiment is done to evaluate the effectiveness of the Amharic Query System and to assess the satisfaction of users. In this chapter we will discusses the methods used in the experimental evaluation of Amharic Query system to bi-lingual databases. In the literature, there is no standard evaluation method for comparing and assessing NLIDBs. The most practiced approach is to evaluate the query statement translation success; That is, to evaluate how much the Amharic Query Interface is resilient to possible variations of Amharic query expressions in the process of query formulation [11]. To do this, on one hand, we have listed the possible various ways of expression of statements for the same information retrieval need and evaluated its success and failure rate. On the other hand, we have selected different users and requested them to use the Amharic Query Interface to get the information they need from the database in different levels of complexity.

5.1. Database Design

For evaluation purpose Ethio Telecom's customer information database is used. The original database is designed in Oracle database. It contains more than 4000 tables. The design of the database is complex. To fit our system, we have selected only 10 critical tables and designed a smaller database with another relationship. We designed the new database in SQL Server 2005 database because SQL Server supports utf8 format. We are able to store Amharic data in some of the tables. Hence, the design of the database we used for the test purpose is shown on Figure 5.1 below.

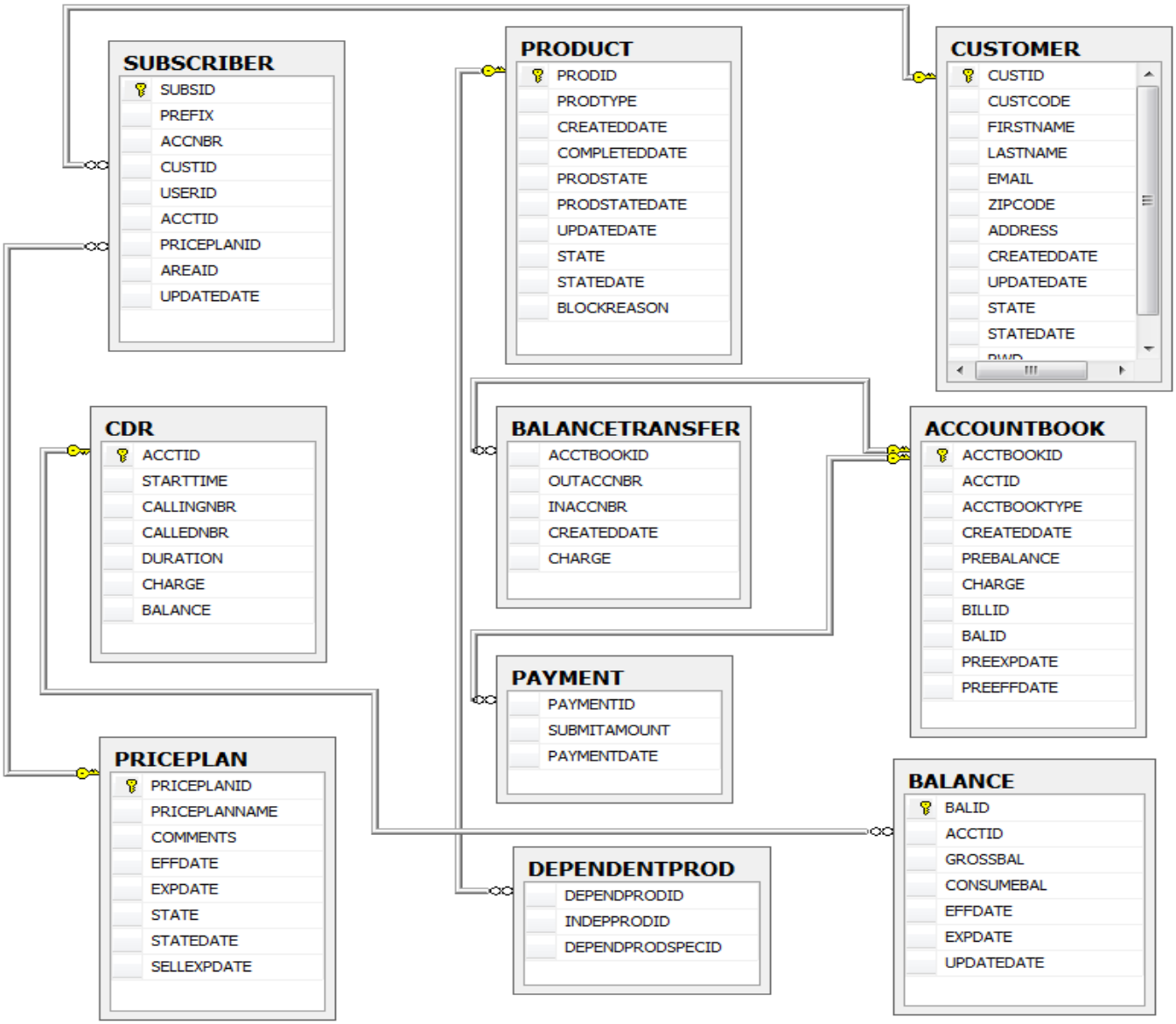


Figure 5.1- Design of Database for Evaluation

As it can be seen from Figure 6.1 above, the database contains 10 tables with relationships. Each table contains from 1000 to 1100 rows. Only CUSTOMER table contains data in both Amharic and English characters because most of the data inserted in other tables is number format. Amharic and English data are inserted in FIRSTNAME, LASTNAME and ADDRESS attributes of the CUSTOMER table.

5.2. User Selection

We have selected 20 users from different level of expertise. The targeted users are staffs of Ethio telecom because they know the database environment very well. Each of them formulated 10 queries of different types. In total we have selected 200 different queries. As indicated in chapter four, to formulate their queries users are allowed to see information of the database on the interface.

5.3. User Queries

We have briefed the users about the system. The queries were classified into 6 according to their complexity. Users are requested to formulate queries to match each of the following query formats.

Type 1: To query all data in a table:

Example: የሁሉንም ተመዝጋቢ መረጃ አሳየኝ?

Type 2: To know the total number of all records in a table:

Example: ስንት ደንበኛ አለ?

Type 3: To query few attributes of a table:

Example: ከምርት ላይ ምርትሁኔታ እና ምርትአይነት አውጣልኝ?

Type 4: To query all data which fulfill certain condition:

Example: ከተላለፈሂሳብ ላይ ሁሉንም ቆጣሪ ከ 50 የበለጠ አሳየኝ?

Type 5: To know the number of data which fulfill certain condition:

Example: በ 30 እና 40 መሀል ስንት ደውል መጠን አለ?

Type 6: To query few attributes of a table which fulfill certain condition:

Example: ከደንበኛ ላይ አ የመሰለ መጀመሪያስም እና አድራሻ አዳማ እኩል የሆነ አሳየኝ?

As it is discussed in Chapter 2, the select clause can have different formats. In Query - Type 1 and Type 4; the word ሁሉንም will match the symbol * of select clause. Similarly, in Type 2 and 5, the word ስንት will match count(*) of select clause and it counts the records of a table based on the criteria of the query and displays the number. Type 3 displays only the selected columns of the table. This will divide the whole table vertically. Finally, Type 6 statements display records in the same manner as Type 3 but it divides the original table both vertically and horizontally.

Different operators are considered when formulating SQL statement with where clause. The common ones are EQUAL (=), NOT EQUAL (!=), GREATER THAN (>), LESS THAN (<). In addition to them the operators IN, BETWEEN, LIKE are also considered.

Some examples of the tested queries are shown in Table 6.1 below.

Table 5.1 – List of Sample Amharic Query Types

Query Type	Amharic Query Phrases	Equivalent SQL Statements
Type 1	የሁሉንም ተመዝጋቢ መረጃ አሳየኝ?	Select * from SUBSCRIBER;
Type 2	ስንት ደንበኛ አለ?	Select count(*) from CUSTOMER;
Type 3	ከምርት ላይ ምርትሁኔታ እና ምርትአይነት አውጣልኝ?	Select PRODSTATE, PRODTYPE from PRODUCT;
Type 4	ከተላለፈሂሳብ ላይ ሁሉንም ቆጣሪ ከ 50 የበለጠ አሳየኝ?	Select * from BALANCETEANSFER where CHARGE > '50';
Type 5	በ 30 እና 40 መሀል ስንት ደውል መጠን አለ?	Select count(*) from CDR where DURATION BETWEEN '30' and '40';
Type 6	ከደንበኛ ላይ አ የመሰለ መጀመሪያስም እና አድራሻ አዳማ እኩል የሆነ አሳየኝ?	Select FIRSTNAME, ADDRESS from CUSTOMER where FIRSTNAME LIKE 'a%' and ADDRESS = 'Adama';

5.4. Evaluation Result

This section presents the results that we obtained when evaluating the system. The results are divided into two. The first section is evaluation of the system based on its accuracy. The second one is evaluation done on user satisfaction and naturalness.

5.4.1. Accuracy

Accuracy measures are performed by testing the collected Amharic queries. The result is classified as correctly answered queries and incorrectly answered queries for all query types. Then the percentage successes were calculated for each type of answers. Evaluation results for each type of queries are as shown on Table 6.2 where 'No.' stands for the number of queries stated.

Table 5.2 – Query Success Rate for Different Query Types

Query Types	Total Query	Correctly answered		Incorrectly answered		Invalid queries	
	No.	No.	%	No.	%	No.	%
Type 1	40	38	95	2	5	0	0
Type 2	35	30	85.7	5	14.2	0	0
Type 3	35	27	77.1	3	8.5	5	14.2
Type 4	30	22	73.3	6	20	2	6.6
Type 5	30	20	66.6	7	23.3	3	10
Type 6	30	16	53.3	9	30	5	16.6

The table below is compilation of the above result. It shows results obtained for Ethio Telecom Customer profile database. The overall success rate was around 76.5%.

Table 5.3 – Overall Query Success Rate of the Amharic Query System

Total Query	Correctly answered		Incorrectly answered		Invalid queries	
N	No.	%	No.	%	No.	%
200	153	76.5	32	16	15	7.5

5.4.2. User Satisfaction

To evaluate the user satisfaction a template that is used to qualitatively grade the user satisfaction on the usability of the system. The template uses the scales: 'NOT SATISFIED', 'SATIFSIED', and 'HIGHLY SATISFIED' on which users were asked to grade the Amharic query system. These values were summed up and percentage was calculated with respect to total queries. From the results obtained in accuracy measures and users satisfaction, naturalness of the system is computed. The result for overall user satisfaction of the system is shown in the below table.

Table 5.4 – User Satisfaction Result

Total Users	Qualitative Measure			Naturalness	
	Not satisfied	Satisfied	Highly satisfied	Natural	Un natural
20	25%	55%	10%	92%	8%

From the table above 65% of the users are satisfied by the Amharic query system and about 92% of the respondents feel that the system is a natural interface (i.e. they get the information from the database by stating statements in the way they naturally state queries in Amharic language).

5.5. Discussion

The effectiveness of any NLIDB system is highly dependent on the performance of its' preprocessing component. In this section, issues and constraints that may affect the performance of the Amharic Query Interface to bi-lingual database are presented.

5.5.1. Issues Related to the Stemmer

In the performance evaluation experiment conducted, the main reason that the system failed to generate the right SQL query from the Amharic Query statements is because the stemmer failed to extract the right stem. This occurs because of generation of wrong stems when removing original characters of a token as a result of removing prefixes and suffixes. For example, the stemmer result of the word ነጃም is ነጃ. The stemmer considers ም as a suffix but the result should be the same as the original word. Another example is, the word እንደሰው is transformed into እንደሰ and የሸመቤት results in ሸመቤት. This problem makes the system not to identify the correct values. So it will fail to generate a valid SQL query.

5.5.2. Constraints in the Translation

The other reason, which degrades the performance of the system, is the problem in the translation process. The problem occurs because of spelling differences when writing certain words. The word የሸመቤት can be written as የሺኣመቤት and it can have different forms in English such as: yeshemebet, yeshimebet, yeshiemebet. Thus, this results in translation inaccuracies.

5.5.3. Constraints related to the SQL generator

When the system generates SQL statements based on the operator and values, it counts the stored parameters. If the count is not as expected the system ignores some operators or values. Sometimes the system identifies operators but ignores values. And other times, the system identifies incorrect values from users query. These dramatically degrade the system's performance. This shows further improvements are required in the SQL generator.

5.5.4. Users Understanding of the System

During the experimental evaluation, it is observed that users sometimes do not include all the necessary keywords in their queries. They miss to include all what is expected to formulate a query. They also create error when typing, like spelling errors and having space on the table names ant attribute names. For example ጥገኛምርት is one word which maps the table DEPENDENTPROD. Similarly መጀመሪያስም is one word which maps FIRSTNAME attribute. Users include white space and write it as ጥገኛ ምርት and መጀመሪያ ስም to refer the table and the attribute respectively. For this purpose we have prepared a short users guideline indicating the key issues that users should know in using the Amharic interface. See Appendix- V.

Besides all these factors, the system performs very well executing Amharic queries. The system efficiently handles the type of SQL queries to be generated. The evaluation we have performed indicates a promising result. This implementation indicates that plain Amharic statements can be used to query data from databases. Naïve and average users are satisfied by the query interface. The evaluation indicates that queries can be written in Amharic language using restricted but not syntax-based grammar. The system indeed is in natural from users' perspective.

5.6. Contributions of the Work

We have reviewed related work in NLIDB and identified that no previous work is done in designing Amharic language query interface to query databases. We thus studied the specific characteristics of Amharic language and identified the different components that need to be considered to pre-process Amharic query statements. The specific contributions of the work are: Design of architecture for Amharic Query Interface for databases, Design of Amharic query pre-processing Engine, Design of Algorithm for Translation Module, Design of Algorithm for Table

name and Attribute name identification, Enhance the list of words in the bi-lingual dictionary used by earlier researchers by about 1000 words, modify the Stemmer designed by earlier researchers to fit the requirements of our system, modify list of stop words by earlier researchers so that useful stop words may not mismatch with operators in this system, design of a Tokenizer and Normilizer algorithms for words in Amharic query statement, Design and Implementation the SQL Generator, and Design and Implementation the Query Mapper Module.

In general, the system designed and implemented:

- Enables Amharic language users to retrieve data from a database using plain texts without requiring users to learn the standard query language, SQL,
- The system can be adopted for different local language query interface to databases,
- It can also be extended to Amharic query interface to Web database.

CHPTER SIX

CONCLUSION AND RECOMMENDATIONS

The research work presented in this work is summarized briefly and conclusions are given in this chapter. Furthermore, future works which would enhance the performance and usage of the system are presented as recommendations.

6.1. Conclusion

A lot of database applications are developed because of the recent enormous increase in use of information and communication technology in Ethiopia. But accessing database using natural language is not in practice. This work proposed a system for Amharic language query interface, referred as Amharic Query System to bilingual databases.

To accomplish this task, different methods and techniques are used. Initially, the structures of Amharic language and SQL queries have been studied. Detail literature reviews have been conducted. The architecture of the system is designed in a way convenient for Amharic language and users need. The designed system is implemented using different tools and techniques. The system allows users to access database contents using plane Amharic language statements that are not syntax-based. Users can formulate queries using Amharic grammars which include keywords to access and retrieve data from the database. The system transforms Amharic query statements into formal SQL statements before it executes it to get the desired information from database. To implement the system, different algorithms are designed and different language specific tools and components are used. The system is designed as database independent and can be used on any of the known databases. The performance evaluation conducted shows encouraging results with an overall query success rate of 76.5%. Usability test shows that 65% of the users are satisfied by the Amharic query system and about 95% of the respondents feel that queries are formulated as natural as they usually write Amharic to seek for information.

Amharic Query System to Bi-lingual database will be an interesting point for query interfaces in future Amharic database systems. If fully implemented, the system can be an effective querying mechanism for Ethiopian natural languages to database system in a similar manner to standard query languages.

6.2. Recommendations

Natural Language Interface to Bi-Lingual database is a very complex system. It needs proper identification of parameters, attributes and values. As a result, to accomplish that we used a simple approach in which users have to identify proper keywords. The following are some of the recommendations we suggest as future work to increase the performance of the system. The system only performs only single queries on a table. In the future the system can be extended in such a way that it can accommodate more complex query formulations that include the table join operations. In this case, users have to get database information and include keywords to access the required information. A system can also be improved to allow users to use free Amharic grammars so that the system itself can extract the required information using some semantic mechanisms. In SQL queries which includes where clause, the value can be either a number or in English content. So this work can also be extended for such queries in such a way that the value can be in Amharic text. Using Natural Language Processing techniques like morphological, syntactic, semantic and other language analysis tools can improve the performance of the system by allowing users to not restrict their queries. Furthermore, the system can be adopted for other local languages

References

1. Alessandra Giordani and Alessandro Moschitti, "Syntactic Structural Kernels for Natura", *Master's Thesis*, Department of Computer Science and Engineering, University of Trento, Italy, 2007.

2. Amanda Wimsatt and Rachel Wynn, "Amharic Language and Culture Manual", *Master's Thesis*, Texas State University, 2011.

3. Amandeep Kaur, "Punjab Language Interface to Database", *Master's Thesis,* Computer Science and Engineering department, Thapar University, Patiala – 147004, June 2010.

4. Amsalu Aklilu, "*Amharic-English Dictionary*", http://www.amharicdictionary.com/, Visited on June2/2013.

5. Amsalu Aklilu, "*English-Amharic Dictioary*", Oxford University Press, 1976

6. B.Sujatha, Dr.S.Viswanadha Raju, Humera Shaziya, "A Study of the Various Architectures for Natural Language Interface to DBs", *International Journal of Computer Science and Network (IJCSN) Volume 1*, Issue 4, August 2012.

7. Baye Yimam, "*Ye-Amarigna sewasew*", T.M.M.M.D. 1987.

8. Daniel Gochel Agonafer, "An Integrated Approach to Automatic Comlex Sentence Parsing For Amharic Text", *Master's Thesis*, School of Graduate Studies of Addis Ababa, 2003.

9. Daniel Yacob, "*Localize or be localized*", http://www.ictes2004-gstit.edu.et/session%-20III_fullpapers/Localize_Daniel.pdf, visited on October 3/2012.

10. Dawit Bekele, "*The Development and Dissemination of Ethiopic Standards and SoftwareLocalizationhhforccEthiopia*", http://portal.unesco.org/ci/en/files/16572/109057953212003_08_14_Final_Report_Ethiopic_Standards_updatph2003-11-21.doc/.

11. ELF Software Co.: Results from the Head to Head Competition. 2001. https://dl.acm.org/citation.cfm?id=2143429.2143523

12. Frank Meng and Wesley W. Chu, "Database Query Formation from Natural Langauge using Semantic Modeling and Statistical Keyword Meaning Disambiguation", *PhD Thesis*, Comupter Science Department, University of California, Los Angeled, CA, 90095, USA, 2006.

13. Getahun Amare, "*Ye-Amarigna sewasew beqelal aqerareb*", T.M.M.M.D, 1987.

14. Grefenstette, "*Multilingual Information Retrieval*", Xerox Research Europe, http://www.isi.edu/naturalmlanguage/conferences-/amta98-/program.html, last visited on October, 15/2012.

15. *The Amharic Language*, http://en.wikipedia.org/ wiki/Amharic_language, last accessed on October 1, 2012.

16. James R. Groff and Paul N. Weinberg, "*SQL: The Complete Reference*", Book, Amazon, 1999

17. Mekuria Sinkie, "Modeling and Developing Amharic Structured Query Language Interface to Databases"; *MSc thesis*, Addis Ababa University, Faculty of Informatics Department of Computer Science; July, 2007.

18. Mequannint Munye and Solomon Atinafu, "Amharic-English Bilingual Web Search Engine", *MSc thesis*, Addis Ababa University, College of Natural Science, Department of Computer Science, October 2012.

19. Michael Minock, "A Phrasal Approach to Natural Language Interfaces over Databases", *Master's Thesis,* Department of Computing Science, Ume°a University, Sweden, 1987.

20. Michael Minock, Peter Olofsson, Alexander N¨aslund, "Towards building Robust natural Language Interfaces to Databases", *13th International Conference on Applications of Natural Language to Information Systems*, 2008.

21. Neelu, Mahesh and Sanjay, "Natural Language Interface to Database using Semantic Matching", *International Journal of Computer Applications (0975 – 8887)*, Volume 31– No.11, October 2011

22. Omar Al-Harbi, Shaidah Jusoh and Norita Norwawi, "Handling Ambiguity Problems of Natural Language Interface for Question Answering", *International Journal of Computer Science Issues*, Vol. 9, Issue 3, No 3, May 2012

23. Porfírio P. Filipe and Nuno J. Mamede, "*Databases and Natural Language Interfaces*", 2000.

24. Prabhudev Konana, "Structured Query Language (SQL): A Primer on Data Definition Language (DDL)", *PhD Thesis*, The Graduate School of Business, The University of Texas at Austin, 2000.

25. Rajendra Akerkar And Manish Joshi, "Natural Language Interface Using Shallow Parsing"; *International Journal of Computer Science and Applications*, Vol. 5, No. 3, pp 70 – 90, 2005.

26. Ramasubramanian P and Kannan, "Intelligent Natural Language Query Interface for Temporal Databases*" Proceedings of the 13th international conference on Natural Language and Information Systems:* Applications of Natural Language to Information Systems, 2008, pp 187-198.

27. Rashid Ahmad, Mohammad Abid Khan and Rahman Ali, "Efficient Transformation of a Natural Language Query to SQL for Urdu", *Master's Thesis*, Department Of Computer Science,University of Peshawar Pakistan, Proceedings of the Conference on Language & Technology, 2009

28. Rodolfo, Alexander, Javier et'al "Spanish Natural Language Interface for a Relational Database Querying System", P. Sojka, I. Kopeˇcek, and K. Pala (Eds.): *TSD 2002, LNAI 2448,* pp. 123–130, 2002., Springer-Verlag Berlin Heidelberg, 2002

29. Samuel Kinde and ET.AL, "*Need for Revision of XML 1.0 to account for Localization Issues with Particular Emphasis on Ethiopic Script and Writing localized Systems*", http://www.digitaladdis.com/sk/ETXMLFinal.pdf, January 2002.

30. Seid Muhie Yimam and Mulugeta Libsie "Amharic Question Answering (AQA)"; *10th Dutch-Belgian Information Retrieval Workshop*, 2010.

31. Seymour Knowles, "A Natural Language Database Interface For SQL-Tutor", *5th International Conference*, November 1999.

32. Sujatha, Viswanadha and Humera, "A Survey of Natural Language Interface to Database Management System", *International Journal of Science and Advanced Technology (ISSN 2221-8386)*, Volume 2 No 6 June 2012.

33. Saravjeet Kaur, Rashmeet Singh Bali, "SQL Generation and Execution from Natural Language Processing", *International Journal of Computing & Business Research*, Proceedings of 'I-Society 2012' at GKU, Talwandi Sabo Bathinda (Punjab).

34. Harjit Singh, "Providing Inferential Capability to Natural Language Database Interface", *International Journal of Electronics and Computer Science Engineering*, pp 1634-1639, 2010.

35. Sujatha, Viswanadha and Humera, "A Study of the Various Architectures for Natural LanguageInterface to DBs", *International Journal of Computer Science and Network (IJCSN)*, Volume 1, Issue 4, August 2012.

APPENDICES

Appendix I – List of Amharic Punctuation Marks

Punctuation Marks	Symbol	Purpose
Double Colons (The four dots)	::	Marks end of a word and at the same time the end of a sentence
Colon	:	Separate individual words in a sentence
White Space		Separate individual words in a sentence, (the current practice)
Question mark	?	Marks the end of an interrogative sentence
Exclamation mark	!	Used at the end of such sentences or interjections that express such emotions as …
Semi colon	፤	Serves roughly the same function as comma, It separates related meanings.
Three dots	…	Marks deliberate omission of words, phrases or sentence
Quotation marks	“ ” ‘ ’	Used at the beginning and end of words that are being quoted
Parenthesis	( )	Encloses elaboration of Amharic meanings
Stroke	|	Separates date, month, year on official letters of an organization
Vertical stroke	‘	Indicates stress or germination in Amharic

Appendix II – List of Amharic Stop Words

ሁሉ	በሰሞኑ	አሉ
ኋላ	በኋላ	አስታወቀ
ሁኔታ	በኩል	አስታውቀዋል
ሆነ	በውስጥ	አስታውሰዋል
ሆኑ	በጣም	እስካሁን
ሆኖም	በተለይ	አሳሰበ
ሁል	በተመለከተ	አሳስበዋል
ላይ	በተመሳሳይ	አስፈላጊ
ሌላ	የተለያየ	አስገነዘቡ
ሌሎች	የተለያዩ	አስገንዝበዋል
ልዩ	ተባለ	አበራርተው
መሆኑ	ተገለጸ	አስረድተዋል
ማለት	ተገልጿል	እስከ
ማለቱ	ተጨማሪ	እባክህ
የሚገኝ	ተከናወነ	እባክሽ
ማድረግ	ችግር	አንጻር
ማን	ትናንት	እስኪደርስ
ማንም	ነበረች	እንኳ
ሰሞኑን	ነበሩ	እንኳን
ሲሆን	ነበር	እዚሁ
ሲል	ነው	እንደገለጹት
ሲሉ	ነይ	እንደተገለጸው
አለ	ና	እንደተናገሩት
ቢሆን	ነገር	እንደአስረዱት
ብለዋል	ነገሮች	እንደገና
ብዙ	ናት	ወቅት
ቦታ	ናቸው	እንዲሁም
በርካታ	አሁን	እንጂ

እዚህ
እዚያ
እያንዳንዱ
እያንዳንዳቸው
እያንዳንድ
ከኋላ
ከላይ
ከመካከል
ከሰሞኑ
ከውስጥ
ከጋራ
ከፊት
ወዘተ
ወደ
ዋና
ይናገራሉ
አመልክተዋል
መግለጹን
ሆኖም
አቶ
ስለሆነ
ጋር
ዛር
ያግሞ
ፊት
ጥቂት
ጊዜ
ግን
ጋራ
ድረስ
ይህ
ይታወሳል
ያ
የጋራ
የውስጥ
የሰሞኑ
የኋላ
ይገባል
ያሉ
ያለ
ውጪ
ወደፊት
ማለቱ
የሚገኙ
የሚገኝ
ማድረግ
ቢቢሲ
ቢሆን
በኩል
እሱ
እስዋ
ይሻላል
አጠቃላይ
ብዛት
በላይ
አሳየኝ
ያላቸውን
አውጣልኝ
መረጃ
አሳይ
አውጣ
እንዲሁም
ጋራ
ከ
ለ
የ
በ
አምጣ
አምጣልኝ

Appendix III – Amharic - English Character Mapping for Transliteration

	ግዕዝ	ካዕብ	ሳልስ	ራብዕ	ሃምስ	ሳድስ	ሳብዕ	ዲቃላ
ሀ	ሀ = ha	ሁ = hu	ሂ = hi	ሃ = ha	ሄ = hie	ህ = h	ሆ = ho	
ለ	ለ = le	ሉ = lu	ሊ = li	ላ = la	ሌ = lie	ል = l	ሎ = lo	ሏ= Lua
ሐ	ሐ = ha	ሑ = hu	ሒ = hi	ሓ = ha	ሔ = hie	ሕ = h	ሖ = ho	ሗ= hua
መ	መ = me	ሙ = mu	ሚ = mi	ማ = ma	ሜ = mie	ም = m	ሞ = mo	ሟ= mua
ሰ	ሠ = se	ሡ = su	ሢ = si	ሣ = sa	ሤ = sie	ሥ = s	ሦ = so	ሧ= sua
ረ	ረ = re	ሩ = ru	ሪ = ri	ራ = ra	ሬ = rie	ር = r	ሮ = ro	ሯ= Rua
ሠ	ሰ = se	ሱ = su	ሲ = si	ሳ = sa	ሴ = sie	ስ = s	ሶ = so	ሷ= sua
ሸ	ሸ = she	ሹ = shu	ሺ = shi	ሻ = sha	ሼ = shie	ሽ = sh	ሾ = sho	ሿ= shua
ቀ	ቀ = qe	ቁ = qu	ቂ = qi	ቃ = qa	ቄ = qie	ቅ = q	ቆ = qo	ቋ= qua
በ	በ = be	ቡ = bu	ቢ = bi	ባ = ba	ቤ = bie	ብ = b	ቦ = bo	ቧ= bua
ቨ	ቨ = ve	ቩ = vu	ቪ = vi	ቫ = va	ቬ = vie	ቭ = v	ቮ = vo	ቯ= vua
ተ	ተ = te	ቱ = tu	ቲ = ti	ታ = ta	ቴ = tie	ት = t	ቶ = to	ቷ= tua
ቸ	ቸ = che	ቹ = chu	ቺ = chi	ቻ = cha	ቼ = chie	ች = ch	ቾ = cho	ቿ= chua
ኀ	ኀ = ha	ኁ = hu	ኂ = hi	ኃ = ha	ኄ = hie	ኅ = h	ኆ = ho	ኋ= hua
ነ	ነ = ne	ኑ = nu	ኒ = ni	ና = na	ኔ = nie	ን = n	ኖ = no	ኗ= nua
ኘ	ኘ = gne	ኙ = gnu	ኚ = gni	ኛ = gna	ኜ = gnie	ኝ = gn	ኞ = gno	ኟ= gnua
አ	አ = a	ኡ = u	ኢ = i	ኣ = a	ኤ = ei	እ = e	ኦ = o	ኧ = e
ከ	ከ = ke	ኩ = ku	ኪ = ki	ካ = ka	ኬ = kie	ክ = k	ኮ = ko	ኳ= kua
ኸ	ኸ = khe	ኹ = khu	ኺ = khi	ኻ = kha	ኼ =khie	ኽ = kh	ኾ = kho	ዃ= khua
ወ	ወ = we	ዉ = wu	ዊ = wi	ዋ = wa	ዌ = wie	ው = w	ዎ = wo	
ዐ	ዐ = aa	ዑ = uu	ዒ = ui	ዓ = aa	ዔ = eeie	ዕ = ee	ዖ = oo	
ዘ	ዘ = ze	ዙ = zu	ዚ = zi	ዛ = za	ዜ = zie	ዝ = z	ዞ = zo	ዟ= zua

𞄇	𞄇 = zhe	𞄈 = zhu	𞄉 = zhi	𞄊 = zha	𞄋 = zhie	𞄌 = zh	𞄍 = zho	𞄎= zhua
𞄏	𞄏 = ye	𞄐 = yu	𞄑 = yi	𞄒 = ya	𞄓 = yie	𞄔 = y	𞄕 = yo	
𞄖	𞄖 = de	𞄗 = du	𞄘 = di	𞄙 = da	𞄚 = die	𞄛 = d	𞄜 = do	𞄝= dua
𞄞	𞄞 = pe	𞄟 = pu	𞄠 = pi	𞄡 = pa	𞄢 = pie	𞄣 = p	𞄤 = po	
𞄥	𞄥 = je	𞄦 = ju	𞄧 = ji	𞄨 = ja	𞄩 = jie	𞄪 = j	𞄫 = jo	𞄬= jua
𞄭	𞄭 = ge	𞄮 = Gu	𞄯 = gi	𞄰 = ga	𞄱 = gie	𞄲 = g	𞄳 = go	𞄴= gua
𞄵	𞄵 = te	𞄶 = tu	𞄷 = ti	𞄸 = ta	𞄹 = tie	𞄺 = t	𞄻 = to	𞄼= tua
𞄽	𞄽 = che	𞄾 = chu	𞄿 = chi	𞅀 = cha	𞅁 =chie	𞅂 = ch	𞅃 = cho	𞅄=chua
𞅅	𞅅 = tse	𞅆 = tsu	𞅇 = tsi	𞅈 = tsa	𞅉 = tsie	𞅊 = ts	𞅋 = tso	𞅌= tsua
𞅍	𞅍 = fe	𞅎 = fu	𞅏 = fi	𞅐 = fa	𞅑 = fie	𞅒 = f	𞅓 = fo	𞅔= fua
𞅕	𞅕 = pe	𞅖 = pu	𞅗 = pi	𞅘 = pa	𞅙 = pie	𞅚 = p	𞅛 = po	

Appendix IV - List of Normalized Words.

ጠዋት፣ ጡዋት፣ ጧት → ጠዋት

ኢሜይል፣ ኢ-ሜይል፣ ኢሜል → ኢሜይል

ቴሌቪዥን፣ ቴሌቭዥን → ቴሌቪዥን

ትናንት፣ ትላንት → ትናንት

ደመወዝ፣ ደሞዝ፣ ደምዎዝ → ደመወዝ

ውሻ፣ ዉሻ → ውሻ

ስድሳ፣ ስልሳ → ስድሳ

ረቡዕ፣ እሮብ → ረቡዕ

መካከል፣ መሀከል → መካከል

ኢትዮጲያ፣ ኢትዮጵያ → ኢትዮጲያ

ጊዜ፣ ግዜ → ጊዜ

ይህ፣ ይሄ → ይህ

ውጪ፣ ውጭ → ውጪ

Appendix V – Users Guideline

Rules in Amharic Query Formulation

There are certain rules that query formulators or users of the system should respect when formulating their query in Amharic. Those rules are presented in this part.

Selection of the Database to query

When using the systems, users must first choose the database they want to query from the combo box on the user interface. See figure A.

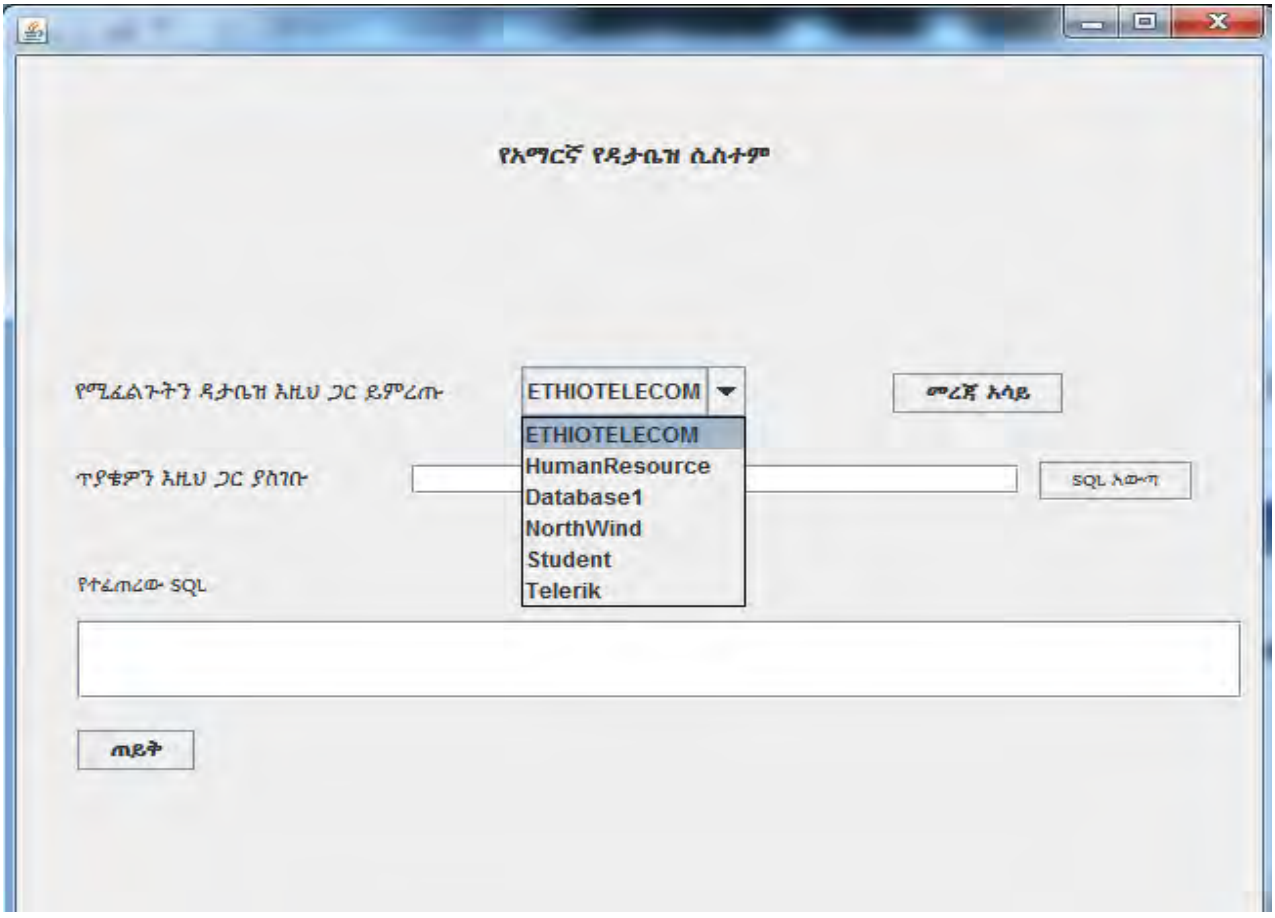


Figure A – Screenshot of the User Interface of the Amharic Query System

Displaying Database Elements (Table name, Attribute names and data type)

To view the Database elements, click on መረጃ አሳይ button to see this information. The system will display list of table names, corresponding attributes names and data types in a popup window. See figure B.

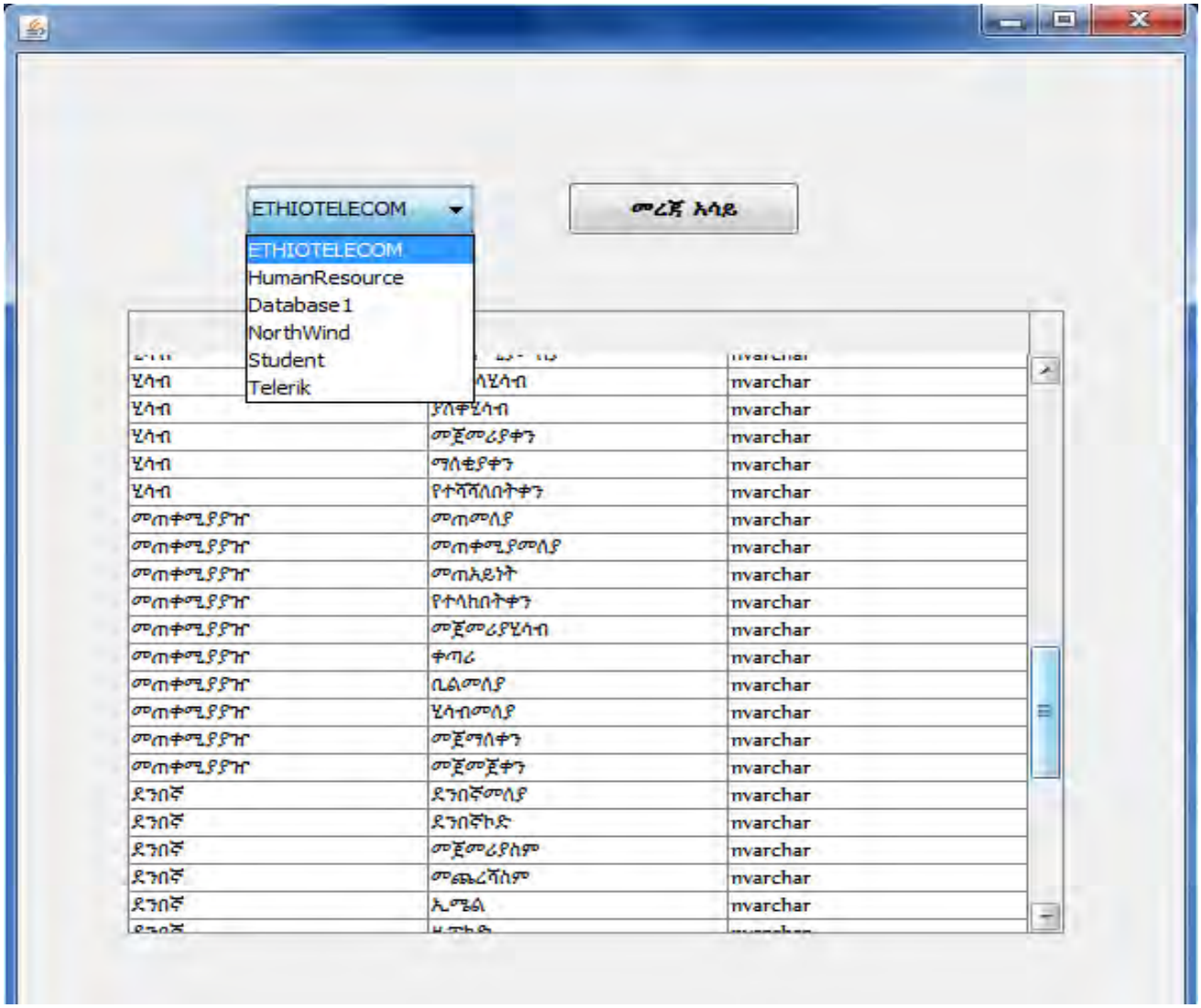


Figure B – Displaying Database Elements

Formulating Query

Users should formulate their query using the information in the database element list. Users must write the correct table name and attribute names they want from the information provided by the database element identifier.

Usage of Operators

To formulate a query data that includes certain conditions, include operators inside the query statements. Those operators are እኩል (=) ፣ እኩልያልሆነ (╞) ፣ የበለጠ (>)፣ ያነሰ (<)፣ መሀል (BETWEEN)፣ የመሰለ (LIKE).

Usage of connectors

When there exist more than one condition to be fulfilled in the query, there is a need to use connectors. Amharic connectors are እና and ወይም. For example: ከደንበኛ ላይ አ የመሰለ መጀመሪያስም እና አድራሻ አዳማ እኩል የሆነ አሳየኝ? in this query the connector እና is used to connect the two conditions.

Usage of Common Terms

Some common terms are identified to be included in the users query. These common terms give meaning for the user when formulating the query. i.e. users feel the naturalness of the system. Those are of them are: አሳይ፣ አሳየኝ፣ አውጣ፣ አውጣልኝ

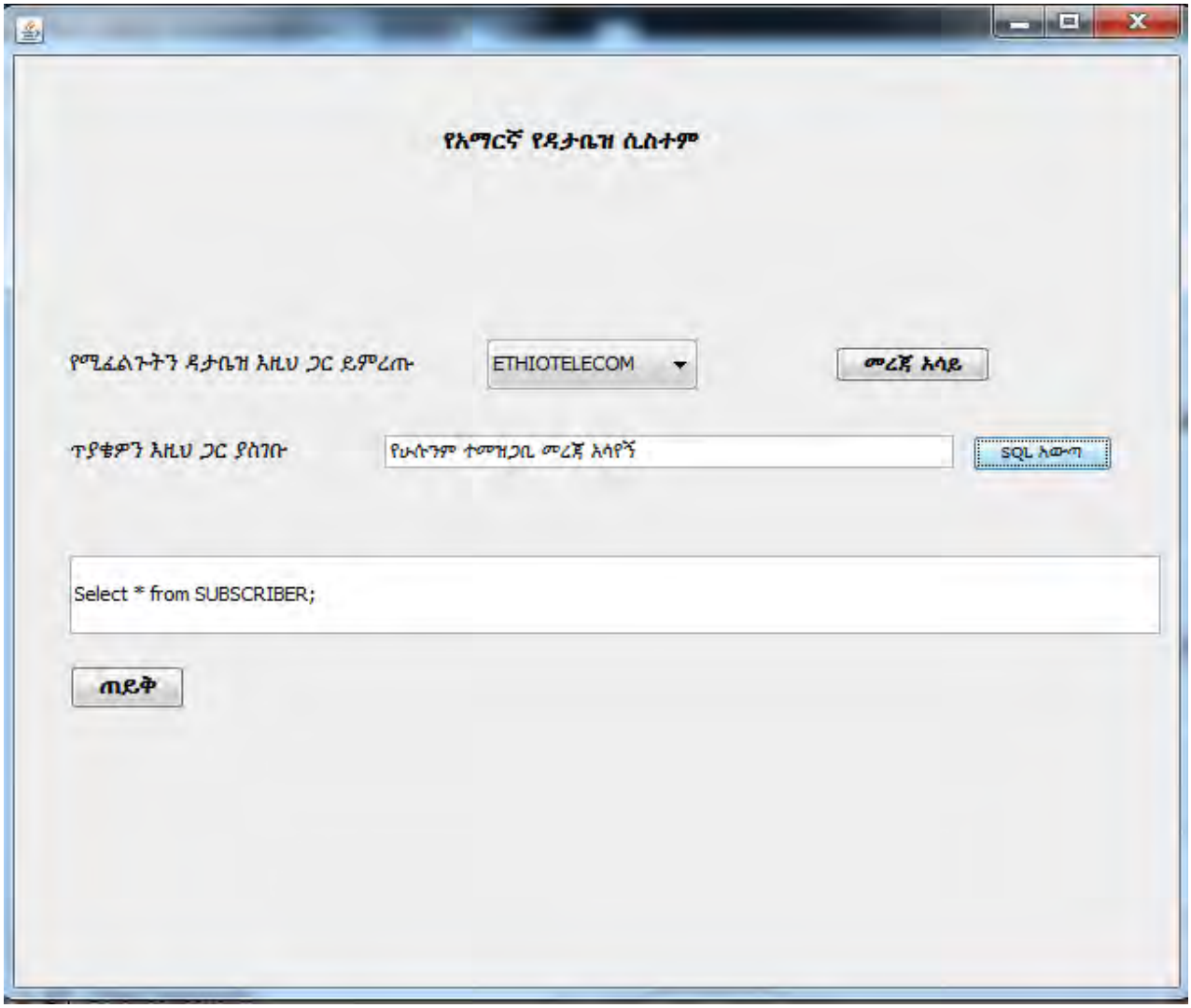
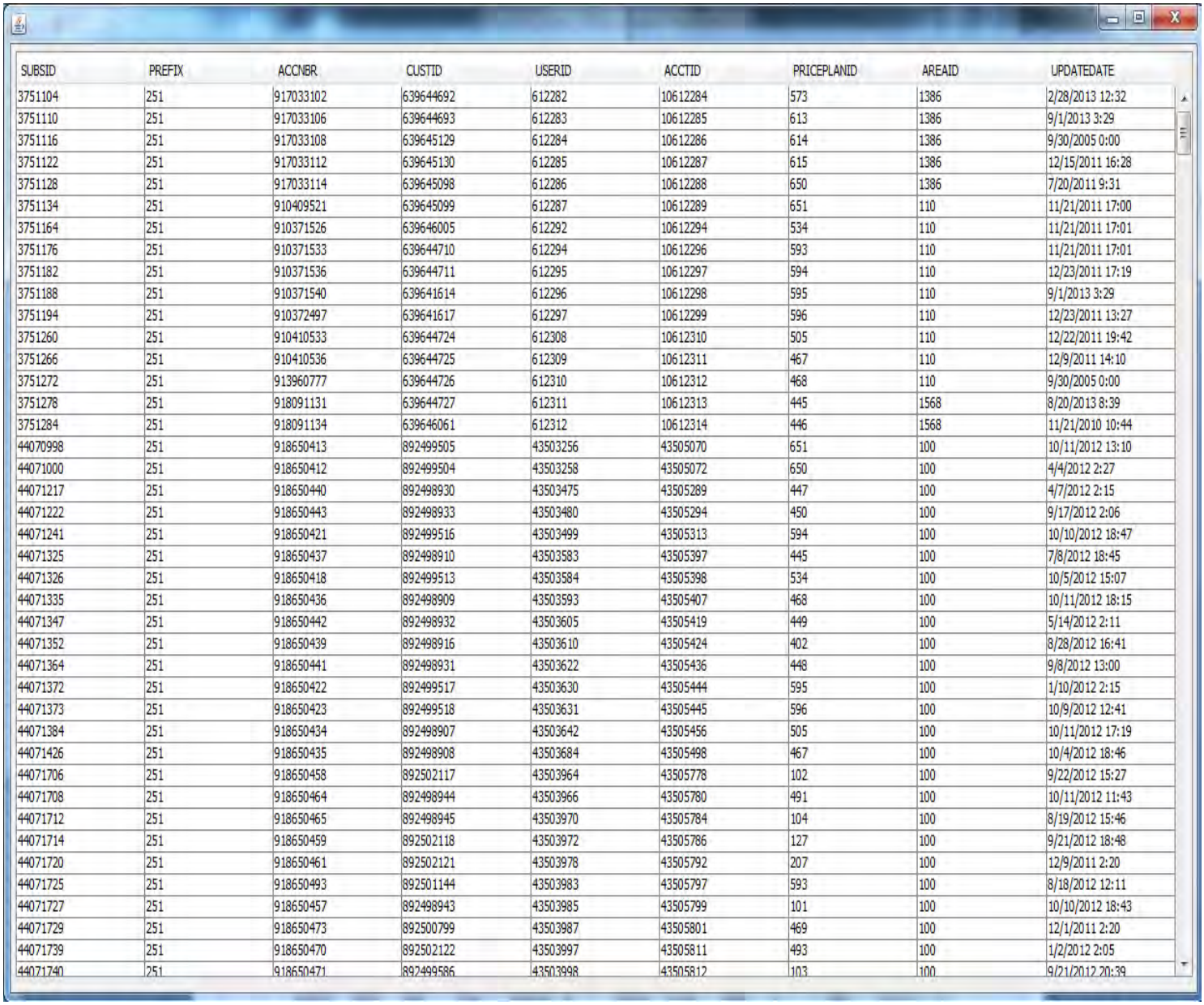


Figure C – The Query Interface with Query Statements

After formulating the query in Amharic, click on SQL አውጣ button to see the equivalent SQL statement generated by the system. See figure C.

The user, can check the equivalence of their query statement and its translated SQL statement, if necessary. This is particularly important for more skilled users. The user, then can click the 'ጠይቅ' button to see the result of the query. The result of the query is displayed on a separate popup window.



SUBSID	PREFIX	ACCNBR	CUSTID	USERID	ACCTID	PRICEPLANID	AREAID	UPDATEDATE
3751104	251	917033102	639644692	612282	10612284	573	1386	2/28/2013 12:32
3751110	251	917033106	639644693	612283	10612285	613	1386	9/1/2013 3:29
3751116	251	917033108	639645129	612284	10612286	614	1386	9/30/2005 0:00
3751122	251	917033112	639645130	612285	10612287	615	1386	12/15/2011 16:28
3751128	251	917033114	639645098	612286	10612288	650	1386	7/20/2011 9:31
3751134	251	910409521	639645099	612287	10612289	651	110	11/21/2011 17:00
3751164	251	910371526	639646005	612292	10612294	534	110	11/21/2011 17:01
3751176	251	910371533	639644710	612294	10612296	593	110	11/21/2011 17:01
3751182	251	910371536	639644711	612295	10612297	594	110	12/23/2011 17:19
3751188	251	910371540	639641614	612296	10612298	595	110	9/1/2013 3:29
3751194	251	910372497	639641617	612297	10612299	596	110	12/23/2011 13:27
3751260	251	910410533	639644724	612308	10612310	505	110	12/22/2011 19:42
3751266	251	910410536	639644725	612309	10612311	467	110	12/9/2011 14:10
3751272	251	913960777	639644726	612310	10612312	468	110	9/30/2005 0:00
3751278	251	918091131	639644727	612311	10612313	445	1568	8/20/2013 8:39
3751284	251	918091134	639646061	612312	10612314	446	1568	11/21/2010 10:44
44070998	251	918650413	892499505	43503256	43505070	651	100	10/11/2012 13:10
44071000	251	918650412	892499504	43503258	43505072	650	100	4/4/2012 2:27
44071217	251	918650440	892498930	43503475	43505289	447	100	4/7/2012 2:15
44071222	251	918650443	892498933	43503480	43505294	450	100	9/17/2012 2:06
44071241	251	918650421	892499516	43503499	43505313	594	100	10/10/2012 18:47
44071325	251	918650437	892498910	43503583	43505397	445	100	7/8/2012 18:45
44071326	251	918650418	892499513	43503584	43505398	534	100	10/5/2012 15:07
44071335	251	918650436	892498909	43503593	43505407	468	100	10/11/2012 18:15
44071347	251	918650442	892498932	43503605	43505419	449	100	5/14/2012 2:11
44071352	251	918650439	892498916	43503610	43505424	402	100	8/28/2012 16:41
44071364	251	918650441	892498931	43503622	43505436	448	100	9/8/2012 13:00
44071372	251	918650422	892499517	43503630	43505444	595	100	1/10/2012 2:15
44071373	251	918650423	892499518	43503631	43505445	596	100	10/9/2012 12:41
44071384	251	918650434	892498907	43503642	43505456	505	100	10/11/2012 17:19
44071426	251	918650435	892498908	43503684	43505498	467	100	10/4/2012 18:46
44071706	251	918650458	892502117	43503964	43505778	102	100	9/22/2012 15:27
44071708	251	918650464	892498944	43503966	43505780	491	100	10/11/2012 11:43
44071712	251	918650465	892498945	43503970	43505784	104	100	8/19/2012 15:46
44071714	251	918650459	892502118	43503972	43505786	127	100	9/21/2012 18:48
44071720	251	918650461	892502121	43503978	43505792	207	100	12/9/2011 2:20
44071725	251	918650493	892501144	43503983	43505797	593	100	8/18/2012 12:11
44071727	251	918650457	892498943	43503985	43505799	101	100	10/10/2012 18:43
44071729	251	918650473	892500799	43503987	43505801	469	100	12/1/2011 2:20
44071739	251	918650470	892502122	43503997	43505811	493	100	1/2/2012 2:05
44071740	251	918650471	892499586	43503998	43505812	103	100	9/21/2012 20:39

Figure D – Query Result.

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Tihitina Petros

Signature: _____

Date: _____

Confirmed by advisor:

Name: Solomon Atnafu (PhD)

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, April, 2014.