ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

Master's Thesis Report

Query Constructor Framework for web based search interface to relational databases

By

Anwar Indris Jemal

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

October, 2015

Addis Ababa

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

Query Constructor Framework for web based search interface to relational databases

By

Anwar Indris Jemal

ADVISOR:

Solomon Atnafu (PhD)

APPROVED BY

EXAMINING BOARD

1. **Solomon Atnafu (PhD), Advisor** _____
2. **Fekede Getahun (PhD)** _____
3. **Mesfin Kifle (PhD)** _____

DEDICATION

To My Wife and My Daughter: without your full attention and support, I wouldn’t have finished this thesis.

Acknowledgements

ALHAMDULILLAH. Above all thank you ALLAH s.w. (GOD Almighty) for giving me the ability and strength to finish this work which I wouldn't be without your help.

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

My next deepest gratitude goes to my wife Halimet Ahmed and my daughter Hibatullah Anwar. Halimet, thanks a lot for your understanding and unconditional support with patience from the beginning till the end. I would like also to greatly thank Meymuna Kemal for her great help.

Finally, I extend my heartfelt thanks to my staff members of Commercial Bank of Ethiopian Service desk department Ato Hailu Tegenaw, Ato Waleligh Adugnaw, Ato Tegaw Goshu, Ato Dagne Weldeyesus, and Ato Awel Abdo, and also my classmate friend Robel Arega. Thanks you all for your un-expressible support in helping me support to accomplish this research along with my work.

Table of Content

Contents page

List of Figures

List of Tables

Abbreviations

AAU	Addis Ababa University
CDS	Combined Databases Schema
DBM	Database Management
DBMS	Database Management System
DSC	Database Schema Constructor
GCT	Group Common Terms
HSP	Heuristic SPARQL planner
HTML	HyperText Markup Language
IE	Information Extraction
IR	Information Retrieval
IrO	Irrelevant Object
ORDMS	Object Oriented RDBMS
QBE	Query By Example
QCFSI	Query Constructor Framework Searching Interface
RDBMS	Relational DBMS
RIF	Rule Interchange Format
SQL	Structured Query language
SQM	SQL Query Constructor
W3C	World Wide Web
XML	eXtensible Markup language

ABSTRACT

In today's information world there are various mechanisms of electronically searching for information. Keyword based searching is one of the popular searching mechanism in web documents as well as database systems. A lot of research attempts and studies have been made in areas of keyword based searching over relational databases. Some of the researches focus on improving searching mechanisms integrated in the database systems, whereas others focused on designing frameworks for user interfaces over a database. But most studies focused on keyword based searching with different searching and ranking mechanism. These studies are applied on single database at a time, fixed on only user's input keyword, and with no error correction mechanisms used.

This work focuses on improving keyword based searching on relational databases using a query constructor framework over a set of databases. It has two major tasks. The first task is construction of Combined Database Schema for the databases connected to the system and the second task is enabling users for searching using keyword and their synonyms. The system passes through various stages during searching before returning the result. After the system accepts users searching keywords it tries to reduce noise, rearrange them to a certain order and include synonym words and word families, constructs related SQL Statements, Execute the SQL statements, manage the results returned, prepare display format, and finally display the result ranked by relevancy. Using the links in the displayed results the displayed results user can navigate through databases to get the contents of their interest.

The work has made contributions by introducing techniques of integration of synonym words and word families, by connecting more than one databases from one or more database systems, and by developing of Combined Databases Schema (CDS). An experimental evaluation is conducted to test the query constructor framework. The result of the experiment and the evaluation has indicated encouraging results in using of single interface over multiple heterogeneous databases with the help of combined databases schema.

Keywords: Combined Database Schema, Schema constructor, searching database by keywords, query constructor framework, query formulation considering synonyms.

CHAPTER ONE: INTRODUCTION

1.1 Relational Database and the Traditional Web

Databases provide the ability to store, search, or manipulate data that the user is interested in. Databases are used in almost all business applications and the web. A relational database is an object in a database system constituting a set of relations (tables) that are related to each other by some form of relationships. Traditional database systems store and process data one tuple at a time, i.e., one row of a table at a time, thereby known by the general term row-stores [5]. Each tuple is processed by every operator in a query, before the next tuple is examined.

In most cases Relational databases store huge amounts of data which are queried using a structured query language, SQL. A user is required to know the underlying schema and the related query language in order to formulate meaningful queries on the data. This is a substantial barrier for casual users, such as users of Web-based information systems to get the information they need. HTML forms can be provided for predefined queries. For example, a University Web Site may provide a form interface to search for faculty and students. Searching for departments would require yet another form, as would be searching for courses offered. Creating an interface for each such task is laborious, and is also confusing to users since they must first expend time and effort finding which form to use. Furthermore, forms are not suitable for ad hoc querying or exploratory browsing.

With the growth of the World Wide Web, there has been a rapid increase in the number of users who need to access online databases without having a detailed knowledge of the database schema or query languages. Even relatively simple query languages designed for non-experts are too complicated for Novice users. Query languages for semi-structured/XML data are even more complex, increasing the impedance mismatch further.

Search engines on the Web have popularized an alternative unstructured querying and browsing paradigm that is simple and user-friendly. In a user interface with one input box and one command button users type keywords and then follow hyperlinks to navigate from one document to the other. No knowledge of schema is needed but this is more often used in document processing not in databases. These all indicate the need for simpler searching mechanism that integrates the web with relational databases in a way to include novice users need.

1.2 Keyword based text searching and SQL

The relevant information that is to be retrieved from a relational database may require writing complicated and refined SQL statements. Since the size of data stored in relational databases increases with time, the number and complexity of SQL statements that need to be written increases proportionally. To make it easier to query such databases, a keyword-based approach is used, which alleviates the need to write complex query statements. The keyword based approach we used here is very similar in idea to that used in text databases. However, instead of searching through documents, the system searches records in inter-related relational tables where this study is interested in. Since in most cases keywords are of type text, this approach can be useful when the database has large number of fields of type text where the value in such a field can be considered as a small text document that can be used for keyword-based search.

Today, Keyword based search on relational databases is more useful and popular among many users without knowing and having technical background. Not only keyword based searching but also aggregate keyword search [11] and full text search on relational databases which was proposed recently has attracted interest of many users as well as researchers. In full text searching the search engines uses all the words together as searching criteria, and aggregate keyword search uses set of words while keyword based searching can include both. But still effectiveness and performance is an issue in keyword searching in relational databases [9]. Moreover most keyword based searching is performed on searching interface provided with the database schema where few users can access. Direct Integration of the databases with web pages is so important that text based searching is possible for more users.

1.3 Motivation

Accessing data stored in a relational database requires the use of SQL statements on SQL command line interface. This requires a direct access to the database and knowledge of SQL Commands, the type of database system, the structure and organisation of the database objects, and the type of data stored. This is so challenging for Novice and Native users, who are not familiar with SQL environment where the users face difficulty to formulate SQL Commands since by nature SQL commands have complex syntax structure for them. As a result these users might not get the information they need on time even if they got help from domain knowledge users and their work might be delayed till they get the required SQL statements [15].

In this information world information societies should access enough, better, and relevant data with less effort. Different mechanisms have to be provided in order to ease information access by any level of users from any type of data sources including databases over the Internet. Several researchers have been done studies recently in simplifying users query mechanisms: like keyword based free text searching in relational databases with user friendly interfaces. This work is to add contribution for a searching mechanisms of databases which interfaced by web technology. Since relational database requires the use of SQL at backend the approach in this work is designed in a way to mask (1) the Structured Query Language knowledge required by the user and (2) the complexity and structure of the heterogeneous nature of database systems.

In a commercial bank of Ethiopia different types of database application systems are used to support business operation. For example **Coupon web based database application** is developed to support the lottery system of saving account, **Mortgage desktop database application** is used to support loan service of Addis Ababa city housing project, **Smart Desktop Database application** is used to support offline banking operation, **Temenos web based database application** and **Branch Power Desktop database application** are used to support online major banking operation, etc. A customer in commercial bank of Ethiopia may have a number of account information in all these applications. In order to get the customers' account information from these applications, it requires to access each application one at a time which consumes more time and effort of bank personnel as well as the customer.

This work is focused to minimize such challenges by interconnecting all or most of database systems in one web based user interface. This requires the design and implementation of a query constructor framework that can facilitate a keyword base over a combined set of databases that may be of heterogeneous type of database systems. The combined set of databases is of kind virtual database where here after called Combined Database Schema CDS that facilitate query between these set of databases objects.

1.4 Problem Statement

Current familiar data storage and processing systems and data sources use relational database management system even if there is start and implementation of other forms of data processing.

Even though the major RDBMSs have provided full-text search capabilities, they still require users to have knowledge of the database schemas, use a structured query language to search for information [19].

Researches that considered full text or keyword search on relational data [3, 5, 8, 16, 17, 20, 21] focused on devising better performance, efficient, and improved way of searching based on set of keywords that only return documents and records that contain combination of the words in some ranked manner. In these cases there is a possibility to return documents and records describing irrelevant issues and to leave relevant ones. This is because, in a keyword – based search, the main ambiguity is that, a single word may have different meanings in different context, whereas different words may also refer to the same thing. Thus, we need to search by considering the context of the given query [2].

Most systems designed for specific purpose/ application which tried to integrate traditional relational database with web interfaces are specific for application domain, schemas, and platform or technology which cannot be applied directly to different schema and platform and also does not support database concurrency. In such cases, users may require additional information concerning the schema and data.

Consider organizations which use two or more web based database applications having a feature where users can search for related content. A user can search for a related content by typing word or phrase based query in his/her own word in search box provided by each system independently. In current situations only keyword based search is possible due to the nature of the databases. Moreover, even if the related reply to the user's search exists in the database it may return nothing because of no direct matching of the words obtained. In addition to that, most of keyword based searches are directly applied to a single table at a time.

Keyword search can provide a very simple and easy-to-use mechanism for casual users to get information from databases rather than SQL but still the user need to know the exact word or phrase he or she has to look for. Failure to use the exact word or phrase results that the user may not retrieve the required information even if using some synonym words. The user has to be assisted with multiple options of searching words or phrases while or after typing his/her own searching texts. Suppose a student registered in a University wants to look for total courses and credit hours he/she must take before graduation. If the student logged in the university website and types searching text that request total subject and contact hours he/she must take no record is returned in the sense of keyword based searching. But, if there is a mechanism of handling synonyms there is a high possibility that an equivalent record of total courses and credit hours returned.

Moreover if the student typed “sabject” instead of “subject” and “corses” instead of “courses” the keyword based algorithm may return no result unless there exist a mechanism to handle such minor errors and this is one of the issues this thesis will consider.

In addition, it is clear that today’s customized web applications as described above and traditional SQL applications require knowledge of the schema. Enabling keyword search in databases that does not require knowledge of the schema is a challenging task. It is obviously true that one cannot apply searching techniques from the documents world to relational databases in a straight forward manner. For example, due to database normalization, logical units of information may be fragmented and scattered across several physical tables. Given a set of keywords, a matching row may need to be obtained by joining several tables on the fly; and also the physical database design (e.g., the availability of indexes on various database columns) needs to be leveraged for building compact data structures critical for efficient keyword search over relational databases [52].

In General currently available keyword based searching mechanisms has one or more of the following drawbacks.

1. **Direct matching of words is required**: This forces the user to know some information about the database schema and its content
2. **Specific system dependency**: Most searching mechanisms are developed for specific task systems which cannot directly be applied to other system
3. **Single Table Access**: Most of the keyword based searching mechanisms access single table per query
4. **No AND/OR operation consideration**: AND/OR operations are not considered because in most cases the words are taken as stop words.
5. **No similar or synonym word usage**: No result relevant to users intension is returned if the user is failed to use the exact keyword
6. **Domain Dependency**: One cannot directly apply searching techniques from the documents world to databases in a straight forward manner due to the difference in nature of relational database systems and document retrieval.

Considering these drawbacks, this thesis is proposed to add contribution on current research of relational searching mechanisms on relational database data in such a way that simplifies users' effort and better relevancy of free text based data searching is applied on relational databases from a web page interface which can integrate set of keywords and their synonyms.

Specifically the proposed framework of keyword and synonym based searching mechanism will consider the above drawbacks of keyword based searching and is expected to have the following features.

- General layered framework and independent to specific system ,
- Can connect to one or more databases concurrently,
- construct **combined Database Schema** about the databases and their records without migrating or altering their contents,
- can perform query operation on multiple tables of multiple databases for single query text,
- Considering AND/OR and the Join operations, translate user query to equivalent structured query form so that the query can be applied to the database tables.
- Enables end users to apply keyword and synonym based searching without the knowledge of the schema as well as the data without use of indexing
- Users are free to write set of search texts in any order of combination,
- Suggests relevant words or phrases related to user's search text,
- Offers easy and user friendly web based user interface,
- handle some minor typing errors like spelling errors, use of special charactes and numeric, stop words,

From this point of view, the following three research questions will be addressed in this research

1. How is it possible to represent relational databases in a form of combined database Schema that contain data about the database objects with their content without migrating its data or records to new form?
2. How effective keyword and synonym based searching can be applied on relational databases for the user with no the knowledge of the databases and its content?
3. How can synonym and keyword based searching is effective than keyword only based searching?

A general framework that address these research questions will be constructed which generates some form of combined database schema of the associated database that can define certain information about the relational databases where user can apply keyword and synonym based searching to a relational database through web based interface and relevant result is returned though.

1.5 Objective

General Objective

The general objective of this thesis is to design a query constructor framework for an effective keyword based query with web based search interface to relational databases to help users without database systems knowledge.

Specific Objective

Specifically the objective of this thesis is to

- Review of related literatures and works
- Design and Develop a keyword searching algorithm that can connect to one or more databases concurrently and construct a learned database Schema about the databases and their objects without any altering their records.
- Design and develop the necessary algorithm; that tracks users text as keyword, synonyms, stop words, etc. based on learned database schema
- Design and develop the necessary algorithm that translates the users free text query to SQL statement using SELECT, FROM, WHERE, AND, JOIN, NOT, LIKE, IS EQUAL TO clauses in applicable way for one or more tables
- Develop easy and user friendly (interactive) interface to databases where users type search texts, submit query text, and result is displayed in tabular form.
- Design and implement prototype for testing, and analysis of the efficiency, effectiveness, and secureness of the framework.
- Test the implementation of the framework and conduct experiment to show the effectiveness and efficiency

1.6 Scope and Limitation

As stated in above sections of this document the proposed framework have the basic functionality that helps end user to search for keyword and synonym based free text searching. This means basically the framework will connect and access to one or more databases, construct separate database schema of each database, translate user query to industry standard, return relevant result and facilitate user in searching. The key terms of clauses in SQL statements to be considered are SELECT, FROM, and WHERE and operators IS EQUAL TO, LIKE, AND, NOT, and JOIN which are more common. Having this in mind there are some limitations that this work doesn't consider for the development and use of the tool. Indeed, this framework has the following limitations:-

- Once the combined database schema is constructed it doesn't track the changes in the databases online and automatically update the schema.
- As the number of databases connected and accessed increased the response time of the system may decrease.
- The database systems considered in this work are Microsoft SQL Server 2008, MySQL 5.5, and Oracle 11g. Other database systems are not implemented and tested.
- The system is dedicated to keywords of format text and other formats like numeric, currency, date and time, etc. are treated as text.

1.7 Methodologies

The methodologies used in this thesis include breaking down tasks and procedure, data collection and organization mechanisms, coding, experimentation and testing, result analysis.

1.7.1 Review of literature and related work

Review of literature and related work is among the major tasks planned to accomplish in the beginning of this work. Collection of different recent literatures in a related domain, review of the literatures for better exploration between works done and existing gaps, and organisation of the review are accomplished.

1.7.2 Data Collection and organization

Data collection and organization is one of the basic tasks of this thesis where the collected data is used for testing and experimentation purpose. Data is planned to be collected from one or more volunteer organizations and also from internet where the data is used only for this research purpose. For example, data of incident management system of Commercial bank of Ethiopia, built in data in Oracle and SQL server databases are planned to use. These data is used as in the original form without applying any data processing.

1.7.3 Design of the framework

Design of the framework has two core design tasks; design of Combined Databases Schema and design of keyword-Query translator. The design of Combined Databases Schema includes study, analysis, and organization of the objects in the databases connected to the system to make ready for later use by Keyword-Query Translator. The design of Keyword-Query Translator includes design of query text processors, construction and management of SQL statement in use of Combined Databases Schema and other resources.

1.7.4 Implementation of the designed framework

The output of this thesis is (1) a framework or tool developed using Java programing languages having a feature where the Combined Database Schema is used to construct, and (2) a user friendly web searching interface to databases which help end users to search for information using free text. So coding is part of the major implementation task of this research which brings the design at hand into tangible and usable application. For simplicity standard ways of coding like use of comment, use of expressive variable names, error handling and reporting method, etc. is used

1.7.5 Testing and Experimental Evaluation

Experimentation and testing phase helps to test whether the intended objective is met or not. In this phase full text search will be performed using built in searching mechanisms in databases and the proposed searching system, and the relevancy of the results will be evaluated. Search of texts with different combinations of words also applied to the search system and the result of these combinations is compared to each other. Here not only the relevancy of the result returned but also the performance with respect to time is tested. Users other than the researcher who does not know about the internal structure and schema of the system are also invited to perform free text search related to the subject.

1.7.6 Result Analysis

The result of experimentation and testing will be analyzed qualitatively as well as quantitatively by using different known analysis methods. Tabular, graphical, and mathematical means of analysis and interpretation of the result is the major task of this phase. The result of this analysis will greatly help to answer the research question and confirms the accomplishment of the target goals or objectives or not.

1.8 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 presents the review of literature, Chapter 3 deals with the review of related, Chapter 4 presents the design of the proposed framework while in chapter 5 discusses the implementation of the designed framework, and evaluates the implementation according to standard benchmarks. In chapter 6 we present the discussion of the contribution of the work, conclusion and the recommendations.

CHAPTER TWO: LITERATURE REVIEW

2.1 Keyword based searching

The Web contains a vast amount of content, most of it in the form of unstructured text [22] and some are structured [22, 26].Web content often has a rich, and implicit, structure that deserves a correspondingly-rich set of query tools. Web Search engines are the common tools for querying the web using keyword based text to make web information accessible, and generally to perform just one type of query [22, 23]. In response to a few keywords, search engine returns a relevance-ranked list of documents. Unfortunately, treating Web text as nothing but a collection of standalone documents ignores a substantial amount of embedded structured documents that are obvious to every human reader, even if current search engines are blind to it.

For example, by combining techniques from databases, researchers in information retrieval and artificial intelligence, investigated three different models for processing structured text queries which are quite different. But all involve some amount of information extraction and a novel query language, with a goal to examine the corpus of Web text and create a "structured overlay" that describes data contained in the text. The models differ in how much structure they attempt to extract from the text, and how much they gather from the query. But, it is still too early to determine a single best method for querying Web texts. Indeed, it may be that different applications will require different query models [22, 24]. The Schema Extraction Model is one that uses a fragment of SQL for structured overlay in a relational database consisting of values found using an information extraction (IE) system.

Searching in relational data introduced challenges which forced to reconsider granularity of search results [23]. By processing a text database with information extraction systems, it is possible to materialize a variety of structured "relations," over which regular SQL can be formulated [24, 26]. One of key challenges to process SQL queries in this text databases is efficiency: where query processing strategies minimizes the size of the relevant result. Another key challenge is result quality: in the traditional relational world, all correct execution strategies for a SQL query produce the same (correct) result; in contrast, a SQL query execution over a text database might produce answers that are not fully accurate or complete, for a number of reasons.

Moreover, generating a structured query correctly (e.g., a SQL query for relational databases) requires the users to have a full understanding of the database [22, 24, 28, 29] schema so that they can precisely specify both the locations of the entities and attributes they are searching for, and the relationships among those entities and attributes, which can be a discouraging task [28]. Due to this establishing effective database query mechanisms that help ordinary users for easy access to database resource is one of the most elusive goals of database research.

Various alternative query models have been proposed by some researchers to give users the ability to query databases without schema knowledge. Those models, including the simple keyword search and labeled keyword search model, aim to extract meaningful data fragments that match the structure-free query conditions based on various matching mechanisms (e.g. Keywords). Typically, most matching mechanisms are content-based: they are defined on data node inter-relationships and incur significant query evaluation cost. Following its merit empowering users to effectively access structured and relational data using keyword queries has become natural question [26, 29]. Ideally, the result of a keyword search over structured and relational data will automatically assemble relevant pieces of data that are in different locations but are inter-connected and collectively relevant to the query. There are several advantages of such an approach [27, 29]. First, it can relieve casual users from the steep learning curve of studying structured query languages and data schemas when accessing structured data. Second, it allows users to easily access heterogeneous databases. For instance, for websites with database back-ends, this approach provides a more flexible search method than the existing solution that uses a fixed set of pre-built template queries. Increasingly, approaches' of considering synonyms of user's keyword will have significant effect in flexibility of query as well as in result quality.

Furthermore, these approaches help to reveal interesting or unexpected relationships among entities. Making database searchable will substantially increase the information volume that a user can access, have potential to provide search results with better quality compared with keyword search on textual documents, and thus increase the database usability and make significant impact to people's lives.

2.2 Keyword searching in databases

An important means of allowing non-expert end-users to pose ad hoc queries whether over single databases or integrated database systems is through keyword search [20]. Given a set of keywords, the query processor finds matches across different tuples and tables. It computes

and executes a set of relational sub-queries whose results are combined to produce the k highest ranking answers. There are also several reasons for Keyword queries becoming a convenient alternative to traditional SQL in querying relational databases with large, often unknown, schemas and instances [21]. The more the relational data complexity is increasing and the user base is shifting towards the less technically skilled, the more the keyword searching is becoming an attractive alternative to traditional SQL queries, mainly due to its simplicity.

Works on keyword search primarily focuses on single-database, single-query settings: each query is answered in isolation, despite possible overlap between queries posed by different users or at different times; and the number of relevant tables is assumed to be small, meaning that sub-queries can be processed without using cost-based methods to combine work. As we apply keyword search to support ad hoc data integration queries over scientific or other databases on the Web, we must reuse and combine computation

While a substantial body of research has been conducted on keyword search over databases, such work has been directed at settings in which a single database is answering a single user's query in isolation and focus on scaling keyword search interfaces to data integration scenarios with commonly-used sources, by exploiting the overlap among the join sub queries that partially answer a single keyword search. Also, the overlap among different keyword searches performed over time and even across users.

For example, The systems in [32, 31] DISCOVER enumerate ordered conjunctive queries (called candidate networks) for a particular keyword search, and rank tuples based on the size of the query that produced the tuple, as well as standard IR scores that are derived from a particular DBMS. The model in the Q System [33, 34] is similar to that of DISCOVER, in that conjunctive queries are found over a schema graph. Each edge in the graph is annotated with a cost that represents how useful the edge is. Additionally, each relation may be annotated with a cost that denotes how authoritative it is. These edge and node costs are learned and may be different across user queries. Unlike the previous systems, query execution in BANKS [34] and BLINKS [35] involves traversing a data graph nodes as tuples and edges as key/foreign key relationships between tuples, in order to enumerate the top-k list. Nodes and edges in the graph are annotated with scores and weights, with the ranking function forming a monotonic combination of both [20].

The challenge for keyword search systems lies in (1) efficiently returning the top-k answers, without exhaustively computing every answer satisfying the query conditions [20], (2) to

discover their intended semantics, (3) to discover the database structures that contain the keywords, and (4) explore how these structures are inter-connected to form an answer [21]. If we know the potential range bounds of scores for the conjunctive queries, we may be able to eliminate some queries from consideration, because they will not return top-scoring answers. If we fetch tuples from the various relations in decreasing order of their score, we may likewise be able to stop once we have returned enough answers. These concepts have formed the corner stones of work on top-k query processing, ranging from keyword search systems. The work in this paper includes producing results for multiple pipelined, ranked queries in order of users input within a middleware layer for data integration. Queries results are being ranked in order to form top-k answers.

Existing approaches typically rely on indices built a-priori on the database content [21]. This seriously limits their applicability if a-priori access to the database content is not possible. Examples include the on-line databases accessed through web interface, or the sources in information integration systems that operate behind wrappers with specific query capabilities. Furthermore, existing literature has not studied to its full extend the inter-dependencies across the ways the different keywords are mapped into the database values and schema elements.

Generally, most works consider the database as a network of interconnected tuples, they detect those containing the keywords in the query, they generate connected components based on how these tuples are associated, and they return these connected tuples as an answer to the query. To do so, specialized structures that index the database content [20] are used. By using these indices, they may directly retrieve the tuples of interest, or they may instead construct the queries expressions that retrieve these tuples when evaluated. This is the basic idea followed by the modern commercial database management systems supporting full-text search over their relational database.

Unfortunately, existing techniques suffer from two main limitations. The first is that they require a-priori access to the data instance in order to build the indices that will locate the tuples related to the given keywords at run time. This seriously limits their applicability if such access is not possible. Examples of such situations include databases on the hidden web and sources located behind wrappers in data integration systems [37] that typically expose only their schema information and lack notification mechanisms for their data updates. The second limitation is that no considerable attention has been paid to the inter-dependencies among the query keywords.

In this work, we propose a novel technique for answering keyword queries over relational databases. The keywords are translated into a number of SQL queries that capture the possible keywords and their synonyms, if exist. The generated SQL queries can be evaluated on the database, and their results serve as the answer to the keyword query. One of the novelties of the technique is that it is not based on a-priori access to the database instances except once to construct the schema. Moreover, our approach exploits the relative positions of the keywords in the query in order to make a more relevant search of the synonym that most likely represent those of the keyword query, and then rank them accordingly. The strategy can not only be easily incorporated to many relational database management systems, but it can also be used as an enhancement of existing keyword searching techniques that utilize the database instance, offering them significant improvements over effectiveness and efficiency.

An advantage of this approach is that it can be used to assist users browsing databases with large unknown schemas. It is often the case that users formulate keyword queries without some specific structure in mind but in an exploratory manner, mainly when they are neither fully aware of the type of information that is stored in a database, nor of the way this information is stored. The possible formulation of a keyword query according to the schema and domain information of the database will be generated by our approach. In contrast to other keyword searching techniques on relational data that return sets of linked tuples, we can return the interpretations expressed in SQL. The study of these queries can reveal tables, attributes and join paths, providing the user with enough information to understand the kind of data that is stored in the database and the way this data is structured.

2.3 Synonym based searching

One way of increasing quantity and quality of searching results is to perform searching query expansion. A particular case of query expansion is when search terms are named entities (i.e., name of people, organizations, locations, etc.) which constitute a major fraction of queries [42]. In this case, query results can be increased by also searching for synonyms of the named entities. A problem of query expansion using synonyms is the effect of rapidly changing synonyms of named entities over time, e.g., changes of roles or alterations of names.

Though, recently Synonym discovery and searching has becoming an emerging topic in a variety of language processing tasks (Baroni and Bisi, 2004; Fellbaum, 1998; L in, 1998; Pereira et al., 1993; Sanchez and Moreno, 2005; Turney, 2001) [41] as part of extending keyword search. However, due to the difficulties of synonym judgment and the uncertainty of

applying synonyms to specific applications, it is still unclear how synonyms can help Web scale search tasks.

Recent works in Information Retrieval (IR) has been focusing mainly on related words (Baiet al., 2005; Wei and Croft, 2006; Riezler et al., 2008). But in Web scale data handling needs to be accurate and thus synonyms are assumed more appropriate than related words for introducing less noise and alleviating the efficiency concern of query expansion. [41] Explored both manually built thesaurus and automatic synonym discovery, and applied a three-stage evaluation by separating synonym accuracy from relevance judgment and user experience impact.

According to [41], the main difficulties of discovering synonyms for Web search are the following:

1. Synonym discovery is context sensitive that depends of the whole structure of search texts. There are quite a few manually built thesauri available to provide high quality synonyms (Fellbaum, 1998); most of these synonyms have the same or nearly the same meaning only in some senses. If we simply replace them in search queries in all occurrences, it is very easy to trigger search intent drifting. Thus, Web search needs to understand different senses encountered in different contexts. For example, “baby” and “infant” are treated as synonyms in many thesauri, but “Santa Baby” has nothing to do with “infant”. “Santa Baby” is a song title, and the meaning of “baby” in this entity is different than the usual meaning of “infant”.
2. Context can not only limit the use of synonyms, but also broaden the traditional definition of synonyms. For instance, “dress” and “attire” sometimes have nearly the same meaning, even though they are not associated with the same entry in many thesauri; “free” and “download” are far from synonyms in traditional definition, but “free cd rewriter” may carry the same query intent as “download cd rewriter”.

3. There are many new synonyms developed from the Web over time. “Manually editing synonym list is prohibitively expensive. Thus, there is a need of an automatic synonym discovery system that can learn from huge amount of data and update the dictionary frequently.

M Thangaraj et’al. [16] 2011, Proposed a technique named NCOSBS (New Context – Oriented Synonym Based Search), which considered to effectively retrieves the relevant document from the collection using both context and synonym. It has two major segments such as pre–processing and the query evaluation tried to analyze the major issues related to the operations

of the digital content. The results of this structure show the effectiveness of the contexts as well as the importance of the searching mechanism. According to their recommendation this work can be extended further by incorporating additional features like learning mechanism, automatic query completion.

A general framework for discovering entity synonyms is proposed by [17]. They studied novel similarity functions that overcome the limitations of previously proposed functions. They developed efficient and scalable algorithms to generate such synonyms, and robust techniques to handle long entity names. The experiments demonstrate superior quality of the synonyms and efficiency of their algorithms, as well as its impact in improving search. To handle long entity names with extraneous tokens, they propose techniques to effectively map long entity names to short queries in query log.

Other work by [18] focused on a synonym discovery approach based on co-clicked query data, and improved search relevance and user experience significantly based on the approach. This Synonym Handling in Web data Search uses automatic synonym discovery methods which generate synonym pairs for each query. The recommendations for future works are investigating more synonym handling methods to further improve the synonym discovery accuracy, and to handle the discovered synonyms in more ways than just the query side.

2.4 Searching interfaces to databases

The popularity of search engines on the Internet has led to a nearly uniform interface for searching: a single ubiquitous input box that accepts set of keywords [19], [23]. Data integration in autonomous web database scenarios has drawn much attention in recent years, as more and more data becomes accessible via web servers which are supported by back-end databases [27]. Web search engines are widely used for searching textual documents, images, and videos. There are also vast collections of structured and semi-structured data both on the Web and in enterprises, such as relational databases, XML, data extracted from text documents, workflows, etc. [29]. The web interfaces simplified users searching practices to backend databases with free keyword texts. Web interfaces are becoming preferable than that of built in searching mechanisms in the DBMS technologies. Doing so this work preferred use of web based interface for its keyword and synonym based querying approach.

2.5 Query construction and refinement

A well designed search query exhibits both good recall and precision; retrieving a large number of documents and records relevant to the user's query text, whilst minimizing the number of irrelevant search results presented to the user [39]. Structured query put in a structured query language specifically designed for a database is very expressive as it can describe the users' information need precisely [40]. Query refinement is an interactive process of query modification that can be used to narrow down the scope of search results. One of the most important classes of queries that must be answered are ambiguous queries - broad queries that have many results in possibly many different domains of knowledge, for instance, "research" or "java" [19].

As a single keyword can occur in nearly any textual attribute of a database, the number of possible query interpretations grows sharply with number of textual attributes and the size of the schema. As a consequence, database search applications can encounter difficulties in detecting the desired information quickly and accurately and may return irrelevant or incomplete search results [40]. In a case of a large-scale database such as Freebase [40] the number of query construction options based on the keyword occurrences can rapidly increase. At the same time, these options are not informative enough to enable efficient query construction as each option subsumes only a small portion of the dataset. That's why querying of such database opens additional challenges.

2.6 Resulting Ranking mechanisms

Automated ranking and returning the most relevant results of a query is a popular paradigm in Information Retrieval [25] and is an important component in collaborative filtering research [23]. The importance of ranking query results according to the user's original search query is to obtain a relevant list of high quality searching results that are similar and semantically related to the query, but should contain additional useful query terms. The resulted ranked list itself can then be used as subsequent queries. Ranking methods can take multiple factors into account, and can incorporate the factors that are relevant to producing good query refinements [19].

There are a plethora of different ranking methodologies for Web information retrieval available which can break them down into two categories: static and dynamic schemes. A static cost

represents a global quality assessment of a document based on some criteria independent of the search query, whereas a dynamic cost is calculated during query execution and takes the search query itself into consideration. There are hybrid approaches that combine static and dynamic ranks [19]. In fact, there is recent work [38] which shows that median rank aggregation is an efficient, flexible ranking algorithm for combining multiple factors which provides nice theoretical properties.

[25] In contrast to IR, database systems support only a Boolean query model. For example, a selection query on a SQL database returns all tuples that satisfy the conditions in the query. Therefore, the following two scenarios are not gracefully handled by a SQL system:

1. Empty answers: When the query is too selective, the answer may be empty. In that case, it is desirable to have the option of requesting a ranked list of approximately matching tuples without having to specify the ranking function that captures "proximity" to the query. An FBI agent or an analyst involved in data exploration will find such functionality appealing.

2. Many answers: When the query is not too selective, too many tuples may be in the answer. In such a case, it will be desirable to have the option of ordering the matches automatically that ranks more "globally important" answer tuples higher and returning only the best matches. A customer browsing a product catalog will find such functionality attractive. Conceptually, the automated ranking of query results problem is really that of taking a user query and mapping it to a Top-K query with a ranking function that depends on given conditions in the user query. These key questions are:
   - ? How to derive such ranking functions automatically?
   - ? How well do ranking functions from IR apply?
   - ? Are the ranking techniques for handling empty answers and many answers problems different?
   - ? How to execute such Top-K queries efficiently over large databases?

Full-text database systems require an index to allow fast access to documents based on their content. A work of [30] is an inverted file indexing scheme based on compression which allows users to retrieve documents using words occurring in the documents, sequences of adjacent words, and statistical ranking techniques.

For a full-text database system, indexes should efficiently support three kinds of activity. First, satisfying the query; in the context of text retrieval. Second, it must be possible to efficiently insert new records, deletion and change should still be supported. Last, it must be possible to statistically rank all of the records in the collection with respect to the query, so that the records most likely to be of interest to the user are retrieved first .

In General, Basic ranking techniques require statistics about the words occurring in the collection, such as their frequency within the collection, so that words' importance can be estimated. More sophisticated ranking techniques also use parameters such as the frequency of each word in each record and the length of each record. Additional functionality may also be considered desirable. One addition is indexing on adjacency of words: allowing queries on phrases or word sequences as well as individual words. More generally, we might seek records in which two words lie within some specified distance of each other. Another possible extension is indexing on stems, substrings, patterns, and other partially specified terms. Moreover it also depends on user's query text.

CHAPTER THREE: RELATED WORK

3.1 Keyword based querying

There are various approaches of keyword based searching over web databases. The following are some of the many.

V.Hristidis and Y. Papakonstantinou [32] proposed a general system architecture that most IR approaches follow. The search results are networks of tuples that collectively contain all search terms. The candidate network generator enumerates all networks of tuples that are potential results. Because the total number of candidate networks is limited only by the actual data, efficient enumeration requires that a maximum candidate network size be specified. Hristidise al. [32] refined DISCOVER by adding support for OR semantics and by including an IR scoring function (pivoted normalization scoring [43]) to rank results. Their monotonic score aggregation function enables efficient execution. F. Liu et al. [45] propose four additional normalizations for pivoted normalization scoring to adapt it to a relational context.

Y.Lou. et al [44] returns to a non-monotonic score aggregation function for ranking results; the non-monotonic function precludes the use of existing query processing algorithms. Qin et al. [46] investigate query processing to eliminate the middleware layer between search systems and the underlying relational database.

G.Bhalotia. et al [35] introduced the backward expanding search heuristic for enumerating results. Edges in the data graph denote a relationship between the vertices (a foreign key between the relational tuples). For every edge (u, v) induced by a foreign key, a backward edge (v, u) with a larger weight is also added. These backward edges enable edge directionality to be considered when identifying results. Bidirectional search [47] improves the performance of the original system. Ding et al. [48] use dynamic programming—the DPBF algorithm—to identify the minimal group Steiner tree in time exponential in the number of search terms.

H. He et al [36] use a bidirectional index to improve query performance. EASE [49] proposes a graph index to support efficient searches on unstructured, semi-structured, and relational data where query results are sub graphs whose radius does not exceed a maximum size. Golenberg et al. [50] guarantee a polynomial delay when enumerating results but enumerate results by height rather than weight. Dalvi et al. [51] investigate keyword search on external memory graphs using a multi-granular graph representation to reduce I/O costs as compared to data

structures managed by virtual memory. STAR [36] approximates the optimal result tree in pseudo-polynomial time, which is shown to outperform several other proximity search heuristics.

DBXplorer [52] is a system for Keyword-Based Search Over Relational Databases that returns all rows either from single tables, or by joining tables connected by foreign-key joins such that each row contains all keywords. Enabling such keyword search requires (a) a preprocessing step called Publish that enables databases for keyword search by building the symbol table and associated structures, and (b) a Search step that gets matching rows from the published databases". By considering lack of space as a reason, they studied only for the case of a single database. This work includes techniques of keyword and synonyms search extended over multiple databases.

Summary of experiments as reported in the literature by Joel Coffman [23] Empty cells indicate that the information is not relevant or were not provided and could not be determined from the details of the original evaluation.

	Experiment	Type		Queries				
System	Dataset	Perf.	Qual.	Selection	$\|Q\|$	$[\![q]\!]$	$\overline{[\![q]\!]}$	Rel.
BANKS [2]	DBLP / other	✓		Researchers	7			
DISCOVER [15]	TPC-H	✓		Random		2–5		
	TPC-H	✓		Random	200	2–5		
	TPC-H	✓		Random	100	2	2.0	
Efficient [14]	DBLP	✓		Random	100	2		
	DBLP	✓		Random		2–5		
Bidirectional [17]	DBLP	✓	✓	Random	200	2–7		○
Effective [21]	Lyrics		✓	Search log	50	2–20	6.7	●
DPBF [8]	DBLP[a]	✓		Random	500	2–6	4.0	
	MovieLens[a]	✓		Random	500	2–6	4.0	
	MovieLens	✓		Random	100	4	4.0	
BLINKS [13]	DBLP	✓		Researchers	60	2–4	3.0	
	IMDb[a]	✓		Researchers	40	2–8	5.0	
SPARK [22]	DBLP	✓	✓	Researchers	18	2–4	2.7	●
	IMDb	✓	✓	Researchers	22	2–3	2.4	●
	MONDIAL	✓	✓	Researchers	35	2–3	2.2	●
EASE [20]	DBLife	✓	✓	Researchers	5	4–5	4.6	○
	DBLP	✓	✓	Researchers	5	2–4	3.2	○
	MovieLens	✓	✓	Researchers	5	3–4	3.4	○
	previous 3	✓	✓	Researchers	5	3–4	3.8	○
Golenberg *et al.* [11]	MONDIAL[a]	✓		Random	36	2–10	6.0	
Dalvi *et al.* [7]	DBLP	✓	✓	Researchers	8	2–6	3.5	○
	IMDb	✓	✓	Researchers	4	2–3	2.5	○
STAR [18]	DBLP[a]	✓		Random	180	3,5,7	5.0	
	IMDb[a]	✓		Random	180	3,5,7	5.0	
	YAGO[a]	✓		Random	120	3,6	4.5	
Qin *et al.* [27]	DBLP	✓		Researchers	17	3–5	3.1	
	IMDb	✓		Researchers	20	3–5	3.3	

[a]The queries are equally partitioned among the number of query terms.

Legend	
Perf.	performance evaluation
Qual.	search quality evaluation
$\|Q\|$	total number of queries
$[\![q]\!]$	range in number of query terms
$\overline{[\![q]\!]}$	average number of terms per query
●	explicitly define relevance
○	relevance considered, but criteria unclear

Figure 3-1: Summary of different works [23]

3.2 Information Retrieval

The traditional searching mechanisms in relational databases require users to have knowledge of the database schema and to use a structured query language such as SQL or QBE-based interfaces. Even though most of the major RDBMSs have integrated full-text search capabilities using relevance-based ranking strategies developed in information retrieval (IR), they still have the above two requirements for users

The recent researched approaches make use of the database structure and provide users with a possibility to refine their information need and actively participate in the information retrieval process. Traditional information retrieval techniques for enabling keyword search in document collections use data structures such as inverted lists [25, 52] that efficiently identify documents containing a query keyword. A straight forward mapping of this idea to databases in a table that stores information at row level granularity requires to keep the list of rows that contains the keyword.

WordNet is a database of words for the English language which helps in finding matches between the flat lists of keywords. It groups English words into sets of synonyms, hypernyms, hyponyms, or other terms, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members [21, 53]. The applications appreciate its knowledge; it is a useful tool in diverse fields, for example in information retrieval. This tool or database can be integrated with this work in a way to get synonym words and word families of users' keywords.

As using classical Information Retrieval engines, users of Web portals are often swamped by a mass of answers. To solve this problem, commonly referred to as information overload, classical ranking or clustering schemes may be used in order to help users finding the most relevant documents.

Seid Muhie et'al. [54], presented Amharic language information retrieval system called Question Answering for Amharic (AQA). A novel technique is contributed where it is developed to determine the question types, possible question focuses, and expected answer types. It also generate proper information retrieval query based on Amharic language specific issue investigations. This research targeted in helping Amharic users in getting relevant document retrieval from large collection of free-text documents. Extending this idea to relational database will benefit users more.

3.3 Research Gap

Most of keyword based searching to relational database systems focuses on query optimization and improving of performance for relevant result in the following conditions

- Use of users input text as searching keyword
- User does not require the knowledge of schema and structure of the database
- Use of single database
- Consideration of table relationship constraints

There are some works that focus on use of keyword and synonym words searching that targeted only the document world where the technique cannot be directly applied to the relational database world due to difference in nature of store and access. There are also some databases that use the term synonym in referencing a database object for remote access which is completely different to the idea of this work. An integration of keywords with their synonyms and word families with analyzing and use of combined database schema are among the contributions of this work. Besides points mentioned in similar researches, this work will consider

- Minor errors correction in users search text: spelling errors up to two characters, inclusion of numbers and special characters in search texts, use of stop words considered as minor errors.
- Use of synonym words and word families. For example use, using, used, user, etc. considered as word families
- Virtually combining and use of multiple databases together
- Virtually combining and use of multiple heterogeneous database systems

To accomplish these, use of combined database schema and employing a schema-based Keyword and synonym matching of content with optimized query statement is required while maintaining or even improving result quality. However, understanding and constructing schema is a non-trivial task [27, 28].

The following table 3-1 clearly describes the gap the thesis considered from previous researches.

Table 3-1: Research Gap Considered

No	Research Title	Key Features in the thesis							
		Keyword Based Searching	**Minor Error Correction**	**Use of Synonyms**	**User Does not require Knowledge of Database**	**Use of all tables in Single Database**	**Use of table Relation ships**	**Use of multiple database together**	**Use of multiple database systems together**
1	Discover: Keyword search in relational databases[32]	√			√	√	√		
2	Keyword searching and browsing in databases using BANKS[35]	√			√	√	√		
3	Effective Keyword Search in Relational Databases[45]	√			√	√	√		
4	DBXplorer: A system for Keyword-Based Search Over Relational Databases[52]	√			√	√	√		
5	**Query Constructor Framework for web based search interface to relational databases [This Thesis].**	√	√	√	√	√	√	√	√

CHAPTER FOUR: Design of the Query Constructor Framework

4.1 Introduction

The Query Constructor Framework for Searching Interface (QCFSI) is a web based system that enables novice users apply keyword based text searching to interfaced databases. The system is designed in a way that accepts one or more set of related or unrelated, ordered or random combination of texts as keywords and then performs text processing mechanisms like removing of stop words and then also consider for synonym words of every keyword, if exists. Users are not required to have the knowledge of the database schema and the architecture behind the system under consideration so that they can type any texts of interest in a text search box provided, and wait for result to display.

The combination of processed set of keywords and their synonyms together make up set of querying keywords. These set of querying keywords are directly searched from the database objects with the help of associated SQL statements. The SQL statements are constructed from querying keywords that are provided in the text search box. A prior developed database schema of connected databases is the key component of constructing SQL statements in providing variables and parameters for the statement. Afterwards, the SQL statements will be sent to the respective databases. The order of executing SQL statements may matter the order of result to be displayed.

The result of the query is displayed in relational table form in away stored in the database where the first record is displayed in plain text link. Before query result is displayed back to the interface, results are organised in ranked manner so that results are ordered in better relevancy to the users search texts. Text processing steps have the role in ranking of the results.

This chapter will briefly consider the design of all components of the system part by part where each component can perform a specific task. Basically each component is required accepting an input, processing a task as required, and returning an output which can be input for the next component. The sum of tasks of each component and their interaction will make up the whole.

The rest of this chapter is organized in describing the various stages of the system design. The resources required for the implementation and use of the system is also described. The design stages used are **System Architecture**: describing client/server architectural style, **Architectural Design**: describing the high-level architectural structure, **System Resource Design**: about resources used in all stages of searching, **Searching Flow Design**: is about series

of steps followed from entering the user keywords to result displayed, **Combined Database Schema Design**: the design of a virtual database used for SQL Query construction, **Query Keyword Preparation Design**: for preparation of noise reduced query text, and **Result Ranker Design:** which is about ordering results based on relevancy.

4.2 System Architecture

QCFSI has been experimented on databases from different systems built and being used in commercial bank of Ethiopia.

Architectural style

The architectural style that is chosen is **three-tier** architecture, shown below, where objects are organized into three layers: presentation/interface layer, application logic, and data services layer (see Figure 4.1 and 4.2). The presentation interface is accessed by number of users from client machine using browsers where each user requires a separate client machine. Since the application and the databases will not be deployed on every client machine, a separate machine with high performance like server is required for the database and application logic where all clients can concurrently access. It is also recommended to separate application server and database server since application server requires high processing performance whereas database server will require high storage. Even though the system is designed for three-tier architecture it also works in two-tier architecture and single machine.

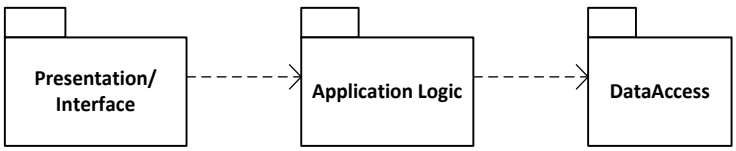


Figure 4-1: three-tier architecture style

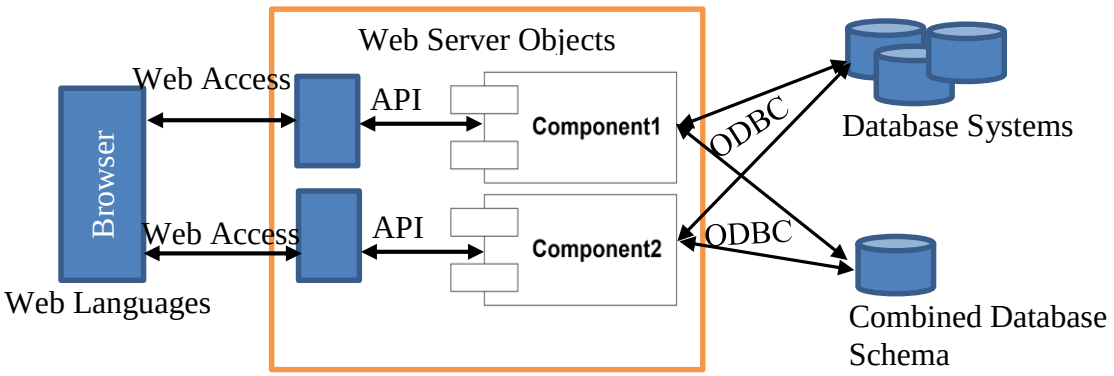


Figure 4-2: System Architectural Design

4.3 Architectural Design

The searching layer design is the hierarchy of layers from the searching interface to the database(s) which is designed for easy management of system components and reducing complexity of implementation and processing. It is a logical layer of system components where its structure and complexity is hidden from the users. The layers separated the system into its components, as described in Figure 4-3 with component description below. The designs of the components are presented in the rest sections of this chapter. Section 4.5 describes the subcomponents in detail. The components indicated with broken lines in the diagram describe the contribution of this work in modifying the existing trend.

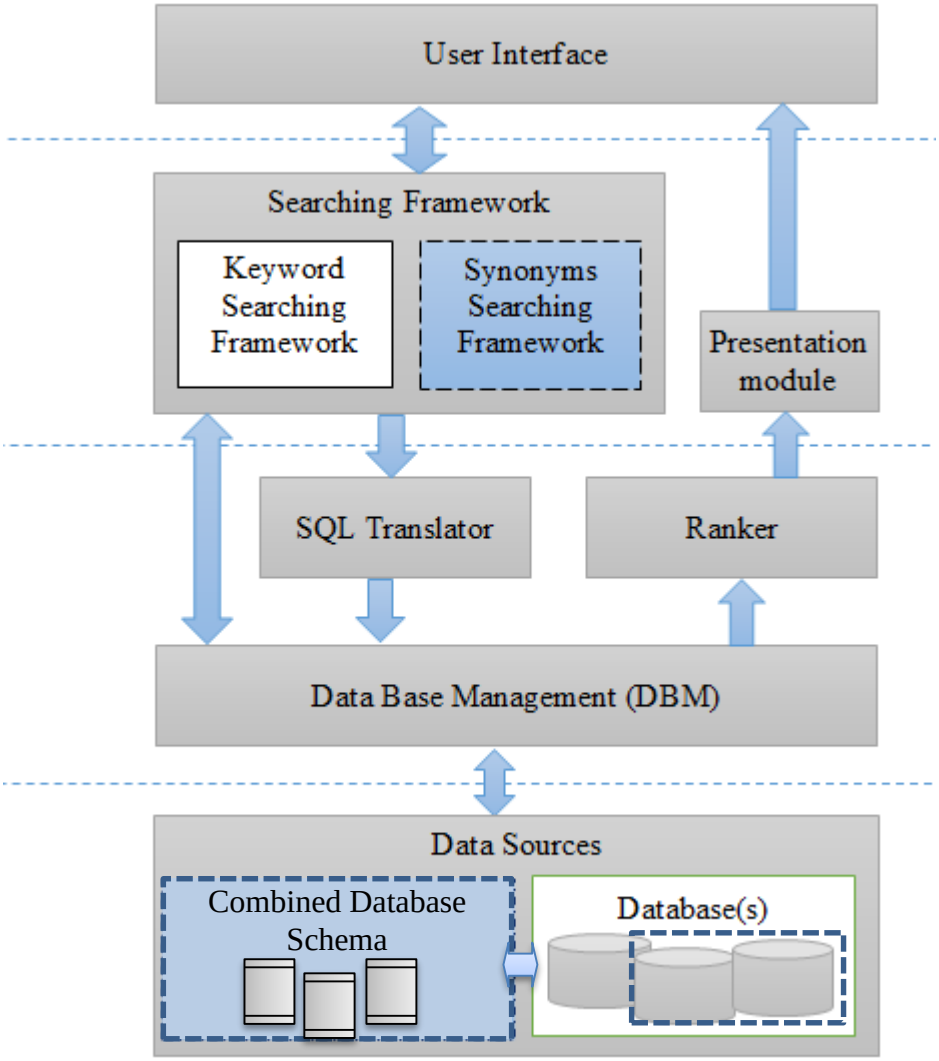


Figure 4-3: High-level System Architecture Design

Components of the Architecture

1. **User Interface:** a Graphical User Interface with input text field, request submitting Button, and result displaying area at front end and having input validation mechanism at back end.

2. **Searching Framework**: Includes Keyword Searching framework and Synonym Searching Framework: receives input from upper layer, perform text processing, and pass noise reduced query text to the lower layer. It also suggests the user for possible keywords.

3. **SQL Translator:** is a layer where translation of processed texts to SQL statements is applicable. The layer
   - Step_1: Receives processed query texts from upper component, and SQL statement parameters from lower component,
   - Step_2: Construct SQL statements and pass to the DBM for executions,

4. **Database Management (DBM):** is an interfacing medium to database systems where it receives SQL Statements, execute in to data sources, and passes the query result to the Ranker module.

5. **Ranker:** the ranker module ranks the SQL query execution result received from DBM and passes to the presentation module. But at first the SQL statements are also ranked based on order of processed query keyword.

6. **Presentation Module:** receive ranked result and present to the user interface in appropriate display format. It is obvious that results from database execution are in tabular form. The result to be displayed contains results in tabular form, links to other query keyword, suggestion keywords, etc. which is organised by this module.

7. **Databases**: is the lower layer where data is stored for retrieval of upper layers. No processing operation is processed in this layer except for data retrieval.

4.4 System Resources Design

During searching process the system uses three different types of resources of data, Dictionary Corpus, Combined Database Schema, and system databases, as illustrated in the Figure 4.4.

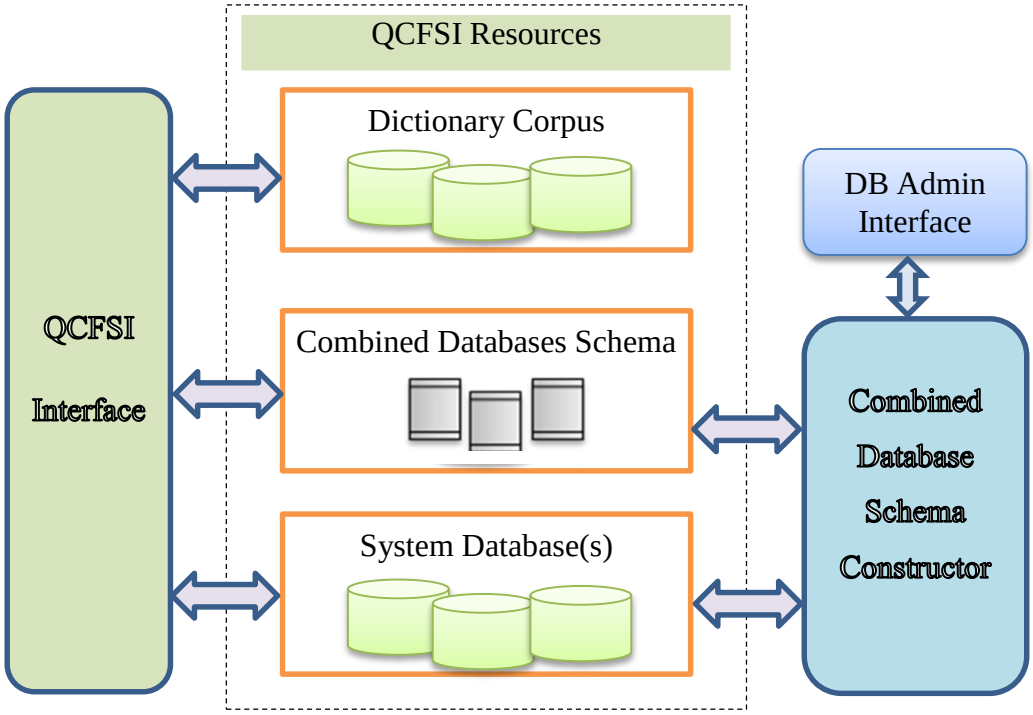


Figure 4-4: Query Construction Framework System

Dictionary Corpus

Dictionary corpus is a set of English words stored in database objects where the system uses this resource for the following purposes.

- To identify and remove most common stop words from the search texts provided by the user.
- To identify and make correction to minor errors of the search texts provided by the user like correction of misspelled words, and suggestion of words related to user's input.
- To identify synonym words and word families of user's search texts and incorporate in the query formulation.

Combined Databases Schema

The combined databases schema is just a database with a collection of relevant objects like tables that stores Meta data of other database(s) where the searching system is connected to. This database is one of the critical resource in which the searching system uses for construction of SQL query statements. From the basic SQL statements syntax we can see that it requires parameters like Database Names, Table Names, and sometimes table constraints. The combined databases schema stores data of these parameters collected from Meta data of databases connected to the system. The design of the combined databases schema is discussed in Section 4.6.

System Database(s)

The resource database here refers to all databases that are directly connected to the web application where the querying of SQL statements are applied for retrieval of user interested records. The design consideration of the system includes only database systems **Microsoft SQL Server 2008**, **MySQL Server 5.5**, and **Oracle Server 11g**. This is because these systems are the most familiar, and also techniques applied and tested on these are assumed to work on other similar database systems. The design and structure of each database is not the concern of this work since the study focuses on use of databases in use from some other system.

4.5 Searching Flow Design

In the text searching process the system is required to pass and undergo various stages and processes following accepting of search texts and before displaying the result. In order to return relevant and enough result these stages has to be designed inconsideration to various issues like reduction of noises in query text pre-processing, incorporation of the different available database systems, way of accepting input and returning results, reducing existing gaps in previous researches, etc. The components indicated with broken lines in the Figure 4-5 describe the contribution of this work in modifying the existing trend like inclusion of synonym words and word families in database searching rather than using only keywords, use of multiple databases from heterogeneous database systems rather than using only single database, use of virtually designed combined database schema.

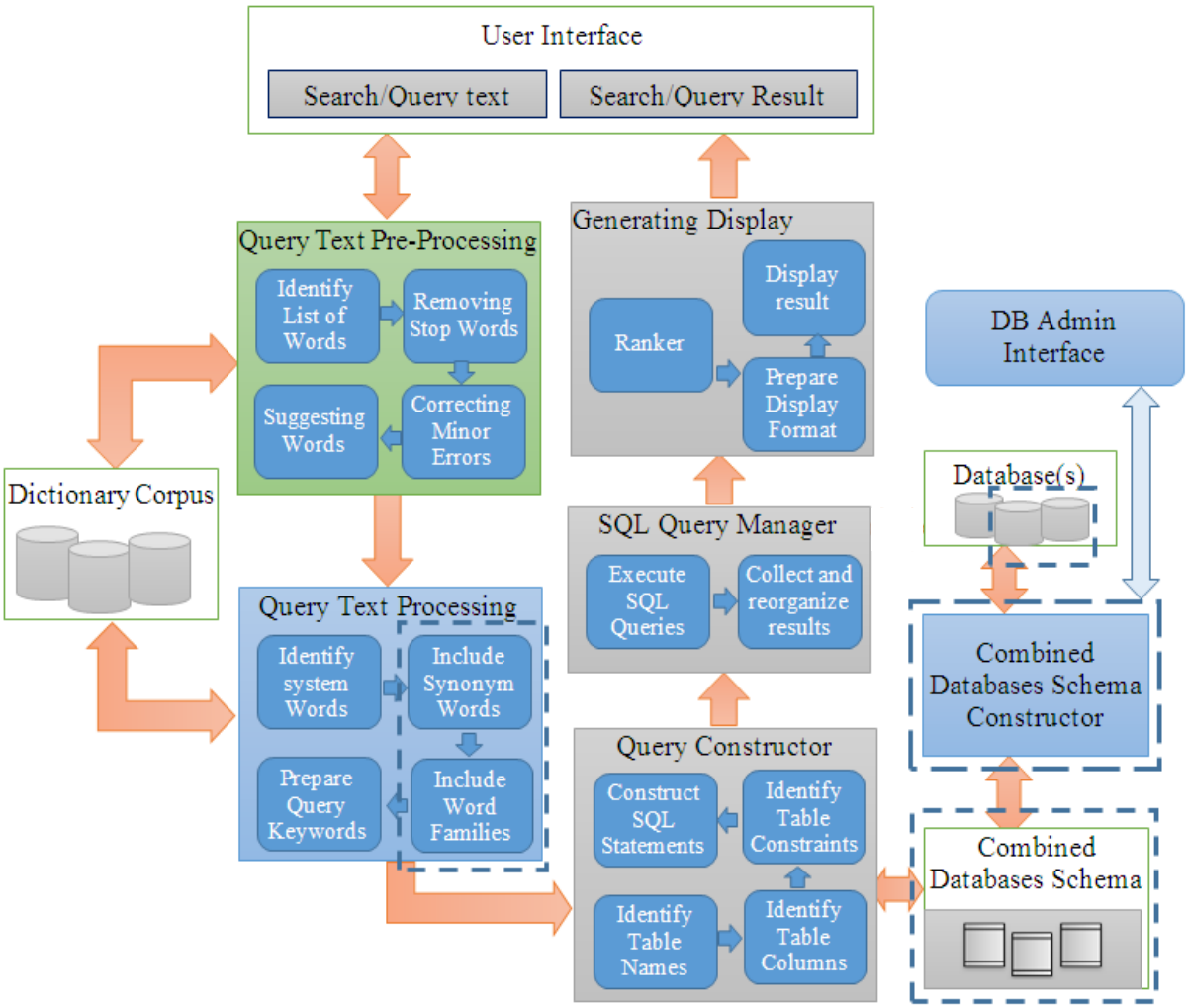


Figure 4-5: Architecture of the searching flow design

Components of the design

1. **User Interface**: the user interface passes input from the user and accepts output to display to the user. It used as the bridge between the system components and the user in hiding the process and complexity of the system
    I. **Search/Query Text**: is a text from the user that passes to the next system component
    II. **Search/Query Result**: is an organized output of the system displayed to the user

2. **Query Text Pre-Processing**: this is the first stage of the process where noise from users input is processed and reduced, if exists.
    I. **Identify List of words**: a user may enter more than one word and these words are identified assuming that words are separated with space(s) or tab.
    II. **Removing stop Words**: from identified set of words more common stop words like characters, prepositions, etc. are identified and removed as they are not useful for the query.
    III. **Correcting Minor errors**: minor spelling errors up to two characters, numbers included in words, special characters included in words, etc. be identified and corrected
    IV. **Suggesting Words**: this is applicable only when the system detect minor errors and corrected them, then it will display list of suggested words to the user.

3. **Query Text Processing**,
    I. **Identify System Words**: Identify, if there exist, database object names, attributes, or constraints from users searching text in consideration of the combined database schema where system words are stored in.
    II. **Include Synonym Words**: designed to identify one or more synonyms of keywords from the dictionary corpus, if exists.
    III. **Include Word Families**: some words contain set of word families. For example, the word write has families like writing, written, wrote, etc. Including the word families will increase likelihood of returning required result.
    IV. **Prepare Query Keywords**: all set of processed keyword will be prepared to use in SQL statements in ordered arrangement according to users search keywords.

4. **Construct SQL Statements**: the construction of SQL statement requires the type of database system(s) considered, the database name, the table name, table column names, and also table constraints. The combined databases schema is expected to provide these items so that these can be used in constructing the appropriate SQL statements.
    I. **Identify Databases**: the database system in use and database names are identified
    II. **Identify Table Names**, Columns: Table names in each of the databases and associated column names of each tables are identified.

III. **Identify Table Constraints**: the Primary-Foreign key constraint relations between tables are identified.

IV. **Construct SQL Statements**: used to construct SQL statement based on collected parameters and query keywords

5. **Managing SQL Queries**: this stage of the design is about retrieving the relevant records up on executing the SQL select statement
   I. **Executing SQL Queries**: is about connecting to the databases and executing the SQL statement and returning the result.
   II. **Collect and organize results**: a result to the query statement is organized in a way better for display in the order of their relevancy.

6. **Generating Display**:
   I. **Ranker:** the result obtained from the SQL Statement execution will be ranked in relevancy order before displayed to the user.
   II. **Preparing Display format**: this designs the display format of the result to the user. Tabular presentation of the record and re-navigable hyperlink of texts are basic part of the display format.
   III. **Displaying the result**: the organised result will be displayed to the user interface in the designed format.

Note that the broken lines in the Figure 4-5 indicates the contribution of this work

4.6 Combined Databases Schema Design and Algorithm

4.6.1 Combined Databases Schema Design

The combined databases schema is a database of databases Meta data that used for providing inputs for construction of SQL statements. It is just a database that stores the metadata of the databases connected to the system. The combined database schema is designed to have the structure as is shown in Figure 4-6.

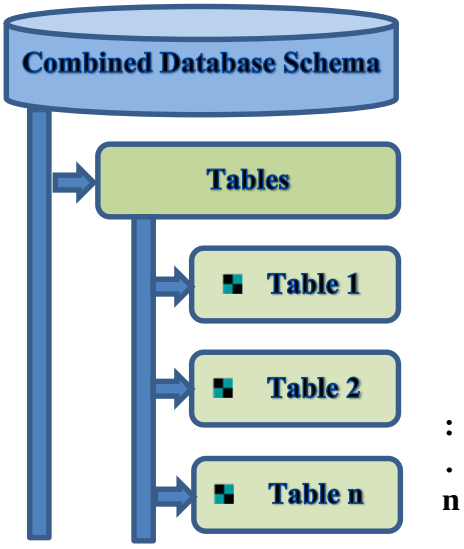


Figure 4-6: Combined Database Schema Structure

The combined database schema contains one or more tables depending on the number of directly connected databases to the system. Each table contains about five attributes or fields where these fields describe the overall structure of a given database. The fields are **TABLE_ID, DB TECHNO, TABLE NAME, COLUMN NAME,** and **KEY_FK_MAP.** Figure 4.7 describes the structure and design of the tables where:

- **TABLE_ID**: Unique identifier of table records
- **DB TECHNO**: is the database system like Microsoft SQL Server, MySQL, etc.
- **TABLE NAME**: table of a tables in the databases
- **COLUMN NAME**: Column names of a given table
- **KEY_FK_MAP**: Primary-Foreign Key constraints of table relations with in a given database. This field helps for construction of the SQL statement in use of JOIN operation for related tables.

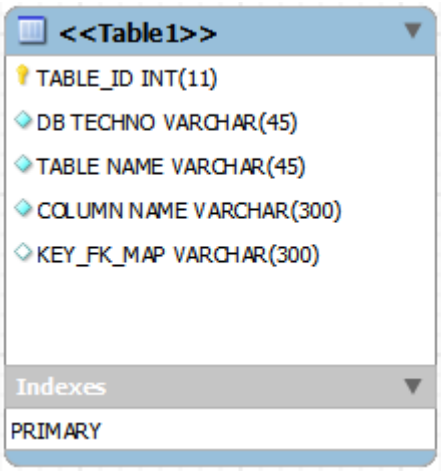


Figure 4-7: Structure and design of the tables of Combined Databases Schema

4.6.2 Combined Databases Schema Construction Algorithm

The combined database schema constructor algorithm collects all relevant data about databases connected to the system and constructs the combined databases schema. Figure 4.8 describes how database schema constructor algorithm constructs the database schema.

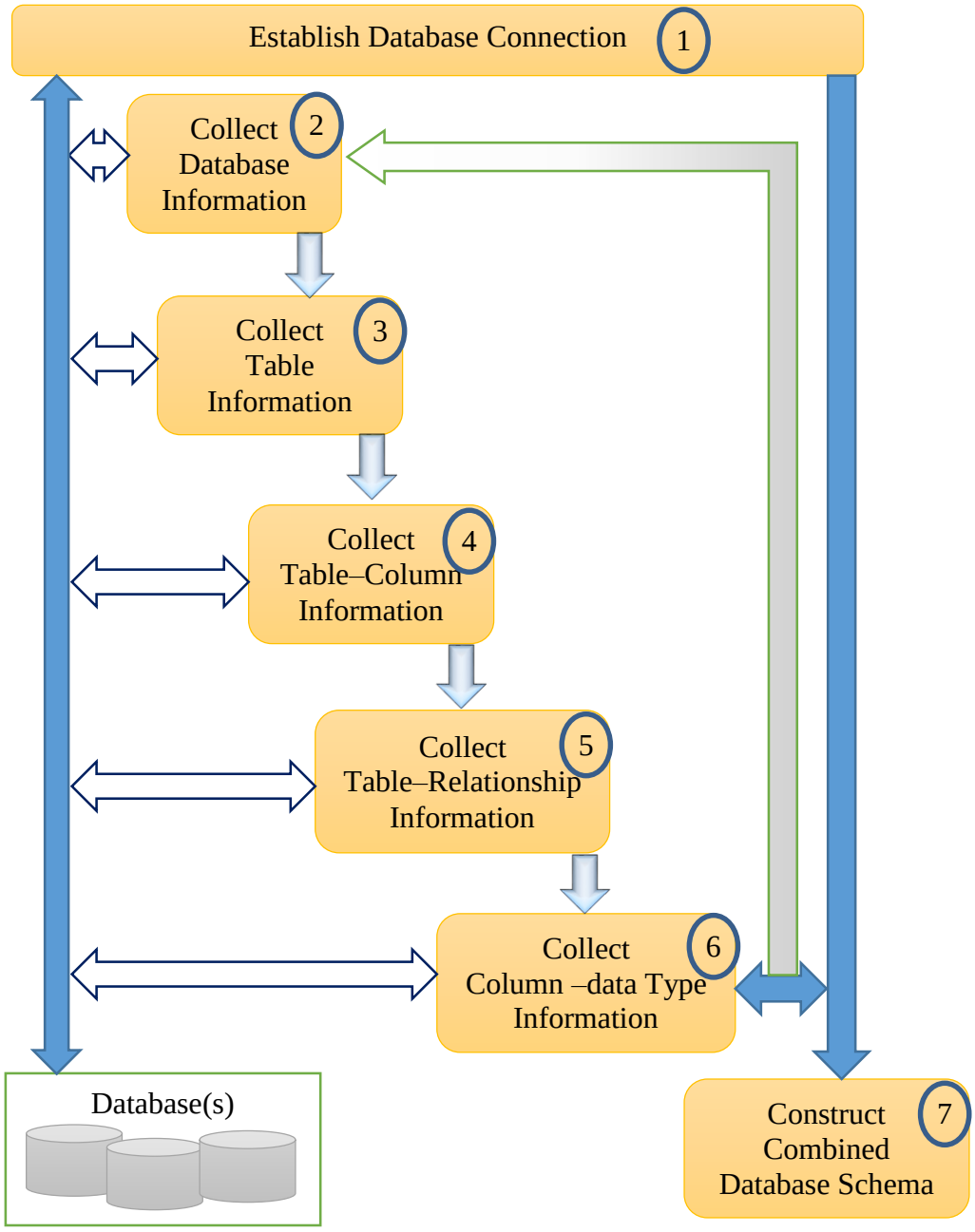


Figure 4-8: Combined Database Schema Construction Algorithm

4.7 Query Keyword Preparation Design

Query Keyword refers to set of keywords that is value for attribute parameter of SQL query statement followed from the where clause. The Query Keyword is a noise reduced and modified searching keyword of the user which is directly searched from the connected databases. In this work user's searching keyword passes various stages before being transformed to Query keyword set. The query keyword set may contain texts, numbers, amount, date and time, etc but all these are treated as text formats since the system scope is limited to text format keywords. The following Figure 4-9 describes the design how searching keyword is transformed to query keyword.

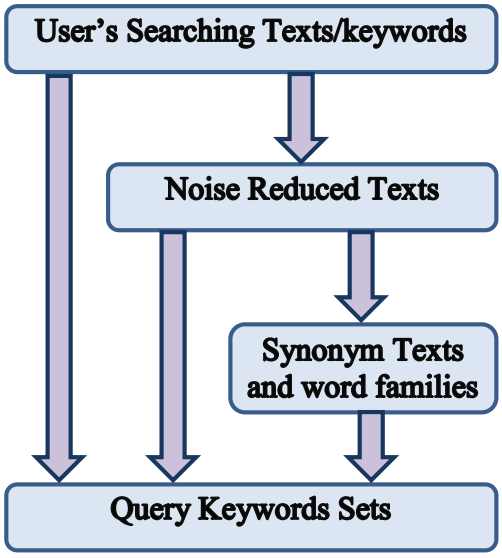


Figure 4-9: Query Keyword Preparation

Components of Query Keyword Preparation Design

I. **User's Searching Texts/keywords:** set of texts the user typed to searching box of the user interface and submitted to the system

II. **Noise Reduced Text:** set of texts driven from User's Searching Texts and where stop words and minor error related noises like spelling errors up to two characters, numbers and special characters included in words are reduced in use of dictionary corpus.

III. **Synonym Texts and word families:** set of synonym texts and word families of noise reduced user searching texts are included from the dictionary corpus.

IV. **Query Keywords Sets:** collection of set of processed user searching texts that is directly translated to the equivalent SQL query statement.

4.8 Result ranking

The importance of ranking query results according to the user's original search query is to obtain a relevant list of searching results that are similar and related to the query, and also should contain additional useful query terms.

The order of displaying the query result depends on the relevancy of the information it provides. In this work the relevancy of the result describes a result containing most keywords of the user's search text. A retrieved record containing all or most of the user's search text is assumed to be more relevant and record containing few number of search text is assumed as less relevant. A record with no users search text and no synonym words of user's search text are considered as irrelevant result, and hence the system should not return irrelevant result.

The point to be noted here is, the ranking process in this work starts from the process of refining and re-ordering of the query keyword in consideration of its nearest to the users search keyword to the process of displaying the result to the user. The overall process and criteria of ranking the result is discussed in later sub sections.

4.8.1 Factors in Ranking

Ranking methods can take multiple factors into account, and can incorporate the factors that are relevant to producing good query refinements. These factors affect the searching algorithm from accepting search text up to displaying the result to the user. According to this work the factors of ranking starts from query text pre-processing stage. The following are ordered list of factors considered in this work for better and relevant output of the result.

Factor-1. There is an option where user is assumed to enter one or more set of keywords with no errors and these keywords are assumed to exist in all or one of the databases. These keywords will directly be searched with exact matching without any modification and a result returned will considered as the more relevant result. If this condition is not satisfied the system will consider the next factor.

Factor-2. Considering if the user makes some minor errors while typing; a noise reduced query texts will be sent to the database for searching exact matching.

Factor-3. By reconsidering what is described in Factor-1 above a search text will be sent to the databases not for direct matching rather for records containing the search texts

only. This factor is applied to the users search text before processed for noise reduction.

Factor-4. By reconsidering what is described in Factor-2 above a search text will be sent to the databases not for direct matching rather for records containing the search texts only. This factor is applied to the users search text after processed for noise reduction.

Factor-5. With consideration of not for users search text as whole but rather exact matching for different combination of query text parts will get this preference of ranking.

Factor-6. By considering what is described in Factor-5 not exact matching but records containing the query texts will get this preference of ranking.

4.8.2 Ranking Mechanism

There are a plethora of different ranking methodologies available in searching world which has been grouped into two categories: static and dynamic schemes. A static cost represents a global quality assessment of a document based on some criteria independent of the search query, whereas a dynamic cost is calculated during query execution and takes the search query itself into consideration. There are also hybrid approaches that combine static and dynamic ranks [19].

The ranking criteria considered in this thesis depends on set of query texts and SQL query statement constructions based on the factors described in Sub Section 4.8.1. Before execution of SQL statement the system will order the SQL statements to be executed in consideration nearest of query keyword they contain to user's search text. The first executed from ordered SQL query statement's result gets the first rank and the last executed SQL query statement gets the last rank in the result displayed, and so on. But also for more than one query result for a given query keyword, the result with larger frequency of query keyword will get preference of being displayed first.

CHAPTER FIVE: IMPLEMENTATION and EVALUATION

5.1 IMPLEMENTATION

The design, architecture, and interaction of the system components are discussed in previous chapter, Chapter four. This section describes the implementation of all modules and components of the designed system for query constructor framework to relational databases. The development Environment, the Implementation of combined databases schema and its constructor, implementation of the searching interface, and implementation of searching flow and the algorithm are presented in sub sections of this chapter.

5.1.1 The development Environment

Development Tools Used

The following lists of tools are used in implementation of the whole system components.

- A Laptop computer with Processor: Intel® Core ™ 2 Duo T5300 @ 1.73GHZ 1.73 GHZ, RAM: 2GB, 111GB HDD, and one desktop Computer with Intel ® Core ™ 2 duo CPU E8400 3.00 GHz 3.00 GHz, 2 GB installed RAM, 32 bit windows 7 operating system, 250GB HDD . A 32 bit windows 7 operating system installed in both systems
- NetBeans IDE 7.3 application software with feature of programming language of HTML, Java, and JSP are used.
- MySQL Server 5.5, Microsoft SQL Server 2008, and Oracle Database 11g Express Edition database systems are considered in the implementation
- Apache tomcat web server, Maxthon 3.4, Mozilla Firefox 36.0.4, Google Chrome 41, and Internet Explorer 8 web browser application

Programming Environment

Java JDK 1.6.0_11 and HTML 5 with NetBeans IDE 7.3 are used as programming languages. This software is chosen because it is easy to design user interface and it is platform independent.

Database Environment

MySQL Server 5.5 database system is used to construct and use for the database schema where attributes, constraints and other parameter that require for the construction of SQL query statement is retrieved from this database. For this system not to depend on a unique database system MySQL Server 5.5, Microsoft SQL Server 2008, and Oracle Database 11g Express Edition database systems are connected to the system for applying searching SQL queries and retrieving query results.

5.1.2 Implementation of Combined Databases Schema

The combined databases schema of the system is first created and continually updated by an advanced system user who has enough knowledge and control of the connected databases. Since data of the database schema is collected from the connected databases, having knowledge and credential of the databases is a must. The database schema is constructed by a framework tool called **Combined Databases Schema Constructor (CDSC)** in away suitable to store data of other databases connected to the system for searching purpose.

Implementation of Database Schema Constructor Algorithm

The CDSC is the tool developed using java programming language and make connection to MySQL Server 5.5 database system. The constructor uses the algorithm as the design is described in Figures 4-6, 4-7, and 4-8 in order to construct the combined database schema. Figure 5-1 is the interface of the constructor as it runs to construct the schema. The construction mainly has the following functions in creating the database schema with its own default system database. These functions are iterative until all connected databases are included in the schema. In every function the administrative user is required to select, connect and instruct the constructor until all the needed databases are included in the schema.

1. The constructor will create connectivity to the database systems
2. Collect all available databases in the database systems
3. Collect all available tables in a database iteratively
4. Collect all available table columns and the data type in a table iteratively
5. Collect table constraints in a database
6. Construct the databases schema.

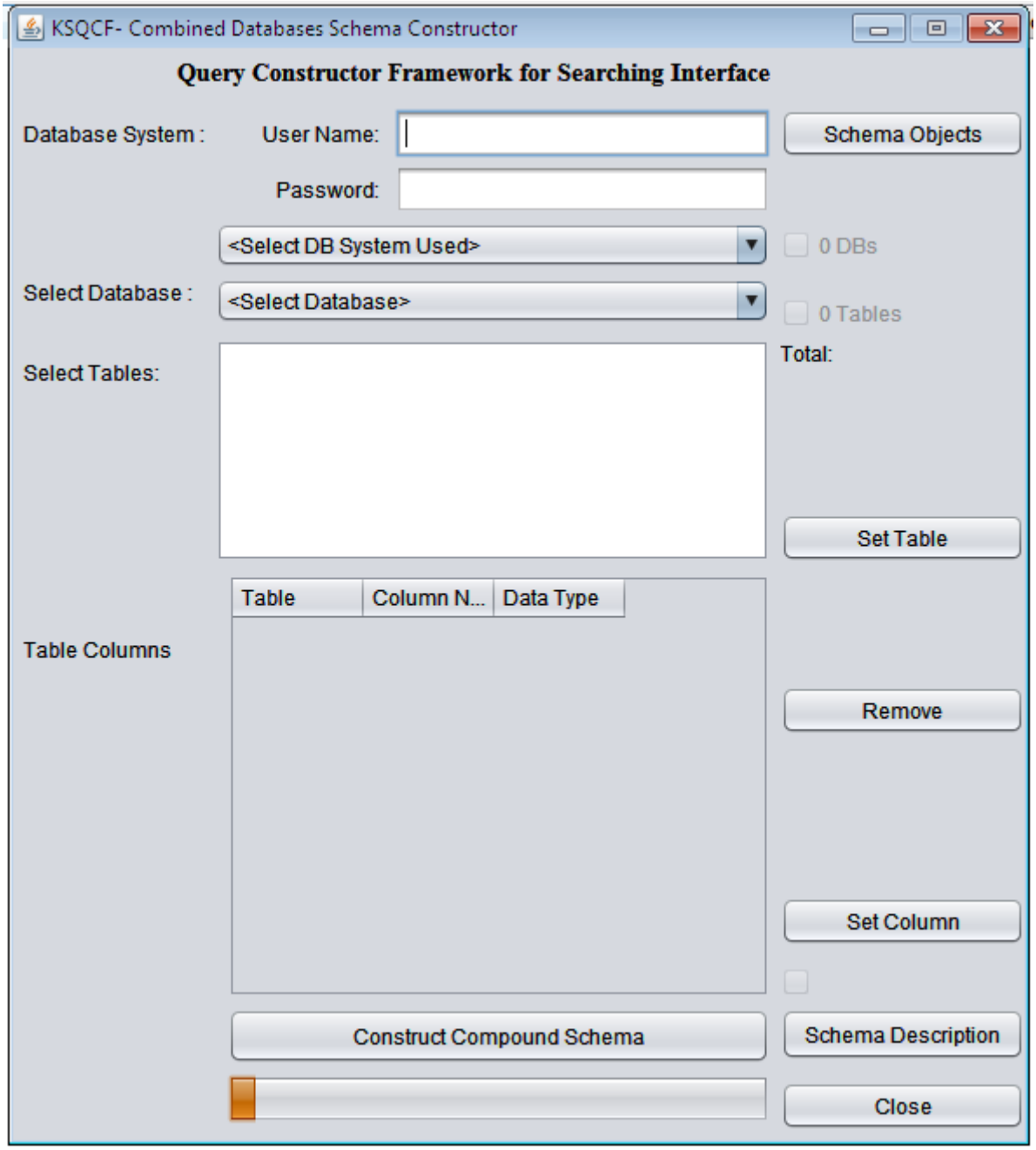


Figure 5-1: Query Constructor Framework User interface

The structure of implemented Combined database Schema

The combined database schema contains basic data about the databases the system access that used for the construction of SQL query statements. The information about which database system contains what databases, which database contains what tables, which table contains what columns, and how does database tables are related each other are accessed. This information is stored as shown in Figure 5-2

TABLE_ID	DB TECHNO	TABLE NAME	COLUMN NAME	KEY_FK_MAP
1	MySQL 5.5	account status	Status_ID:Status_Type	user account:Status_ID:account status:Status_ID
2	MySQL 5.5	account type	Account_ID:Account_Type	user account:Account_id:account type:Account_ID
3	MySQL 5.5	authoriz	UA_ID	authoriz:UA_ID:user account:UA_ID
4	MySQL 5.5	job position	J_Code:Description	privileged user:J_Code:job position:J_Code
5	MySQL 5.5	mission	Mission Code:Mission name:Status	mission round:Mission Code:mission:Mission Code
6	MySQL 5.5	mission roles	Role_Code:Role Type	physician mission roles:Role_Code:mission roles:Role_Code;u...
7	MySQL 5.5	mission round	Round ID:Mission Code:Round Name	mission round:Mission Code:mission:Mission Code;schedule:R...
8	MySQL 5.5	notification	NCode:Message:Recipients Mobile:Recipients E-M...	notification:NCode:schedule:Sch_ID
9	MySQL 5.5	notification receipant	Table_ID:NCode:Patient Code:Phy_ID	
10	MySQL 5.5	patient	Patient Code:RD_ID:RH_Code:Treatment ID:Nam...	patient:RD_ID:referal director:RD_ID;patient:Treatment ID:tre...
11	MySQL 5.5	patient history	Heastory_ID:Treatmemt History	
12	MySQL 5.5	physician	Phy_ID:Group_ID:Spc_Code:Name:GRole:Gender...	physician:Group_ID:physician group:Group_ID;physician:Spc...
13	MySQL 5.5	physician group	Group_ID:Treatment ID:Description	physician:Group_ID:physician group:Group_ID;physician grou...
14	MySQL 5.5	physician mission r...	PMR_Code:Phy_ID:Role_Code	physician mission roles:Role_Code:mission roles:Role_Code;p...
15	MySQL 5.5	privileged user	UID:J_Code:Name:Gender:Address	privileged user:J_Code:job position:J_Code;user account:User...
16	MySQL 5.5	referal director	RD_ID:RH_Code:Name	patient:RD_ID:referal director:RD_ID;referal director:RH_Cod...
17	MySQL 5.5	referral hospital	RH_Code:Name:Address	patient:RH_Code:referral hospital:RH_Code;referal director:R...
18	MySQL 5.5	room	Room No:Room Label	
19	MySQL 5.5	schedule	Sch_ID:Room No:Round ID:Patient_Code:Group_I...	notification:NCode:schedule:Sch_ID;schedule:Group_ID:phys...

Figure 5-2: Sample structure of the table in the combined database schema

CDS Construction Pseudo Code

The following pseudo code in Listing 5-1 presents the general algorithm of construction of combined database schema (CDS). Processes of accessing connected system databases, gathering database objects information and construction of the combined database schema is the major task that the algorithm accomplishes. Every process steps of construction of the combined database schema is controlled by the constructing user. From the listing the numbering is to refer line of code, ( ) is to parameters and ‘ ‘ is to comments.

1:	**Input**: Database connection Parameters,	
2:	Receive Get_Database_systems Command from User	
3:	Get Database systems	'user select DB System'
4:	For each database system	
5:	Connect to Database system(username, password, default database)	'to switch between DB systems'
6:	Get databases in the DB system	'list of database names'
7:	Populate Databases to the User	
8:	Receive Get_Tables Command from User	
9:	Get tables in the database	
10:	Receive Get_Columns Command from User	
11:	For each selected tables	
12:	Get columns for each table (table name)	
13:	Get Data Types of each Column (table name, column name)	
14:	Populate Table_Column_DataType Matrix	
15:	End for loop	
16:	Receive PK-FK_constraints Command from User	
17:	Get PK-FK constraints of selected tables	
18:	Close database connection	
19:	Connect to the schema (username, password, default database)	
20:	If data schema object contains the object	
21:	Remove the object (object name)	'to replace with updated object'
22:	End if	

```
23:     Construct the schema object (object name)
24:     Feed data to the schema object                    'Data about databases connected'
25:     Close the schema connectivity
26:
        End for loop
27:     Output: New or updated Combined Database schema object
```

Listing 5-1: CDS Construction Pseudo Code

5.1.3 Implementation of Searching Interface

The searching interface is the Graphical User interface where the user directly access to enter search texts and get related result back. The interface is developed using JSP programming Language with NetBeans IDE 7.3 Web Application development as shown in Figure 5-3. The interface is used to hide the complexity with in the rest of system components and pass user's search text to the searching algorithm. The interface returns result back in text hyperlinks and relational tabular form where the text hyperlink is about the first row of the relational table. Clicking on the hyperlink will hide or view the whole table, as shown in Appendix C.

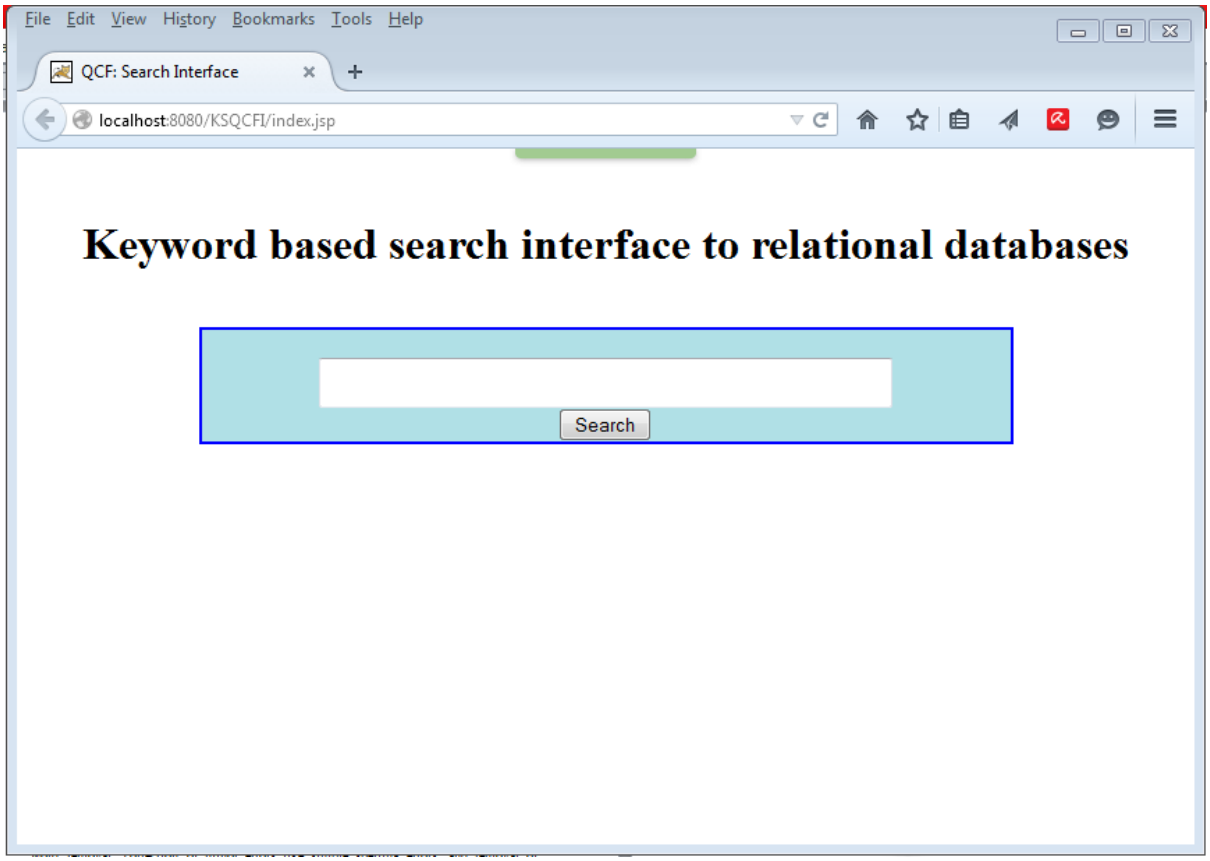


Figure 5-3: The User Interface

5.1.4 Implementation of Searching flow Design

The implementation of searching algorithm is the sum of implementation of each components described in section 4.5.

Query keyword preparation

As the design in section 4.5 and 4.8 describes the preparation of query keyword has two basic operations: Query text pre- processing, and Query text processing. These operations are the first part of the searching flow design which both are dedicated to prepare noise reduced set of query keywords which are part of inputs for SQL Query Statement construction.

Query Text Pre-Processing

The query text Pre-Processing algorithm is set of codes where noise, if exist, in the searching text is removed or reduced from user's input words and suggestion of possible noise reduces words to the user before the system processes further. Noise reduction process includes stop word removal, correction of spelling errors up to two characters, and removal of unsupported characters like special characters and numbers included in words. Figure 5.4 shows the flow of the query text pre-processing.

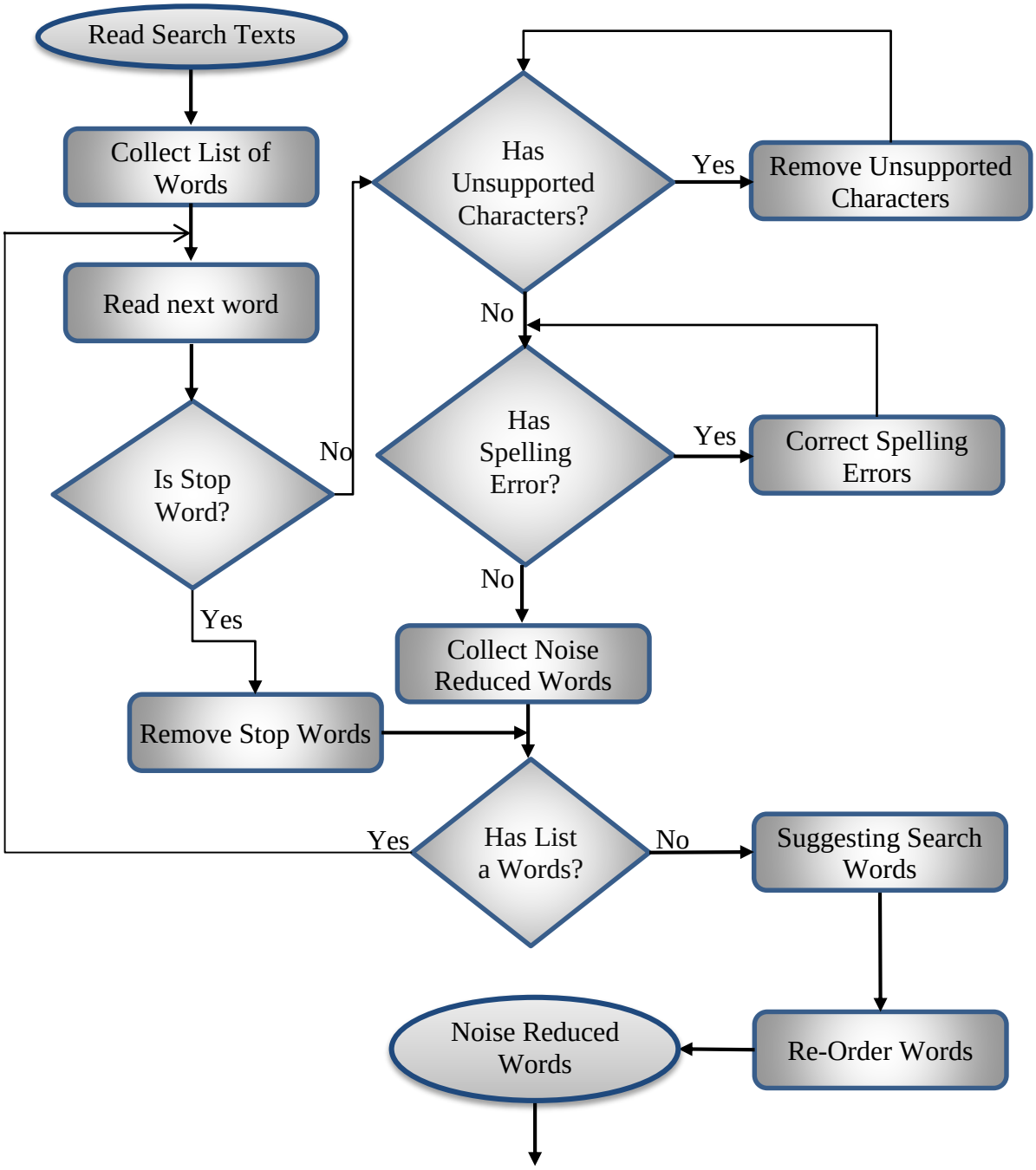


Figure 5-4: Flow of Text pre-Processing

Query Text Processing Algorithm

Identification of system words, identification of query keywords, and identification and inclusion of synonym words to the noise reduced search text are the main tasks of the query text processing algorithm. This algorithm takes noise reduced search texts and transforms to query keywords in a way suitable to include in SQL statement attribute values.

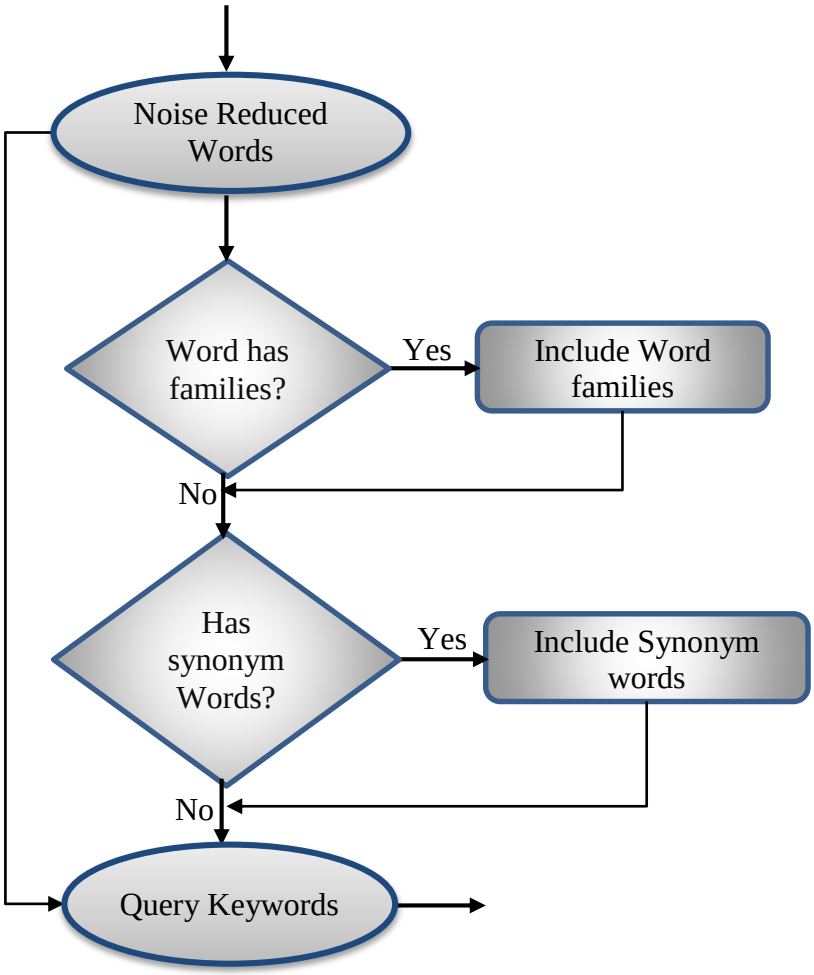


Figure 5-5: Algorithm flow of Text Processing

SQL Statement Constructor Algorithm

Construction of SQL Query statement requires parameters like database name, Table name, Attribute name, table constraints, and also values to attributes in the where clause. This algorithm is responsible to collect these parameters from the combined databases schema and construct the SQL statement that is equivalent to the text query statement. The algorithm takes query keywords, collects SQL statement parameters from the Combined Database Schema, and constructs syntax error free SQL statement. The SQL statements are constructed using basic clauses like SELECT, FROM WHERE, AND, NOT, IS EQUAL TO, LIKE, and JOIN parameters and operations. The query keywords are used as values for attributes in the where clause of the statement. Every query keyword is searched in every table. See Figure 5-6 for the process flow of the SQL statement constructor.

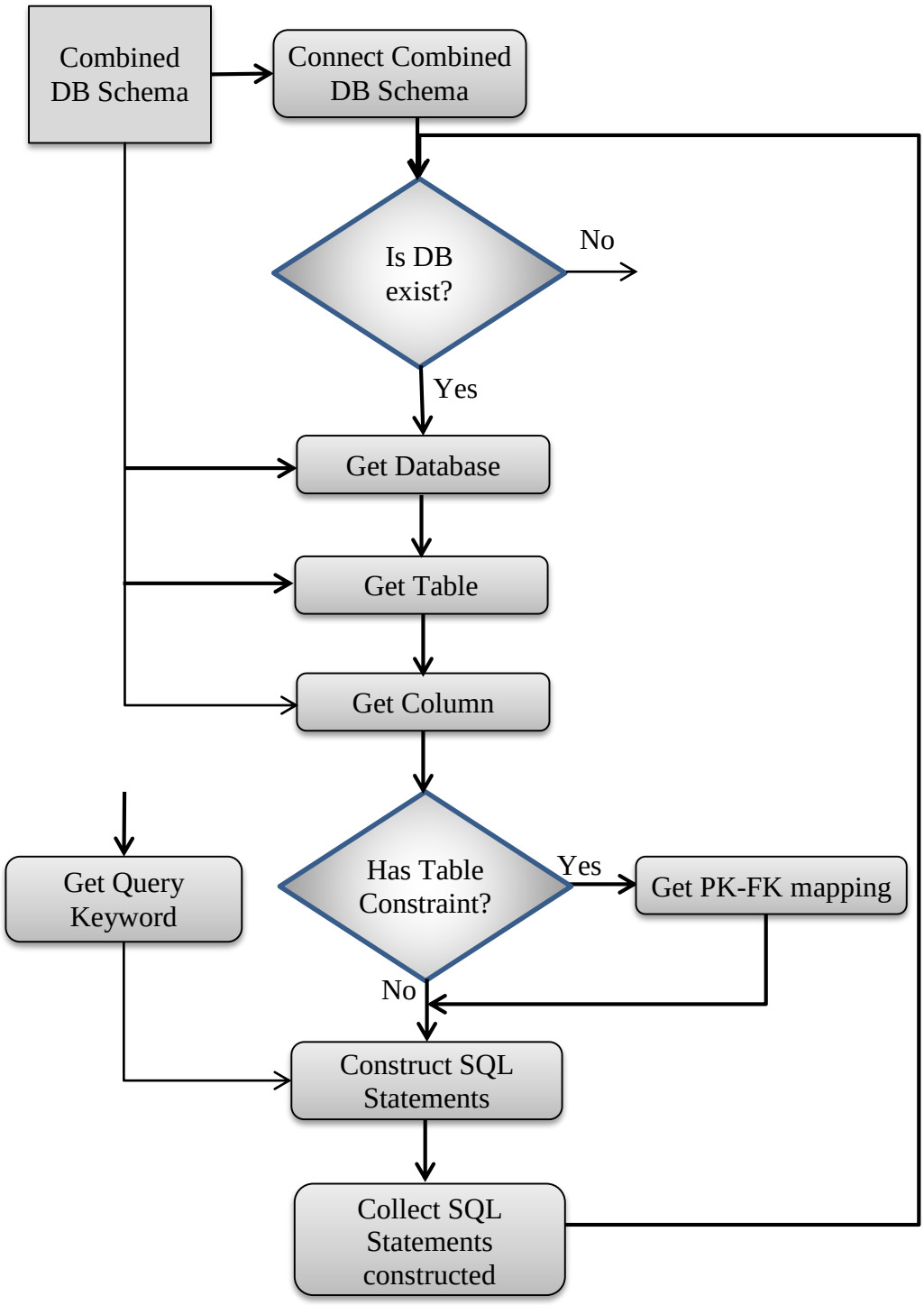


Figure 5-6: SQL Statement Constructor

SQL Query Manager

The SQL Query Manager (SQM) uses an Algorithm that has the following order of implementation activities. From step_2 to step_5 is performed iteratively until all dedicated databases are accessed. From Step_3 to Step_4 is performed iteratively until all dedicated SQL statements under a given database.

Step_1: receive SQL statements from SQL statement constructor,
Step_2: create connection to dedicated databases,
Step_3: Execute the SQL Query Statements,
Step_4: Collect and organize results of executed statements, and
Step_5: Kill established database connections.

Ranker

The ranker module will rank or sort the query result before displayed in the order of its relevance to user's query. The following Listing 5.2 pseudo code presents the algorithm of the ranker module. In the listing the following variables are used:

- **QUERY_RESULT**: a variable that stores the result returned from SQL statement Execution
- **USER_KEYWORDS**: a variable that stores users search text and noise reduced texts
- **WORD_FAMILY**: a variable that stores word families to users search texts
- **SYNONYM**: a variable that stores synonym words to users search texts
- **USER_KEYWORDS_QUERY_RESULT**: a variable that stores SQL query result of **USER_KEYWORDS.** So on.

```
1:  INPUT QUERY_RESULT
2:  FOR every QUERY_RESULT
3:  IF QUERY_RESULT Contains only USER_KEYWORDS
4:          Collect USER_KEYWORDS_QUERY_RESULT
5:  ELSE IF QUERY_RESULT Contains USER_KEYWORDS and WORD_FAMILY
6:          Collect USER_WORDFAMILY_KEYWORDS_QUERY_RESULT
7:  ELSE IF QUERY_RESULT Contains USER_KEYWORD and SYNONYM
8:          Collect USER_SYNONYM_KEYWORDS_QUERY_RESULT
9:  ELSE IF QUERY_RESULT Contains WORDFAMILY
```

```
10:                 Collect WORDFAMILY_KEYWORDS_QUERY_RESULT
11:         ELSE IF QUERY_RESULT Contains WORDFAMILY and SYNONYM
12:                 Collect WORDFAMILY_SYNONYM_KEYWORDS_QUERY_RESULT
13: ELSE IF QUERY_RESULT Contains SYNONYM
14:                 Collect SYNONYM_KEYWORDS_QUERY_RESULT
15:         END IF
16: END FOR
17: FOR Every QUERY_RESULT
18: FOR every USER_KEYWORDS_QUERY_RESULT
19:         Read USER_KEYWORDS_QUERY_RESULT
20:         Collect TO SORTED_QUERY_RESULT
21: END FOR
22: FOR every USER_WORDFAMILY_KEYWORDS_QUERY_RESULT
23:         Read USER_WORDFAMILY_KEYWORDS_QUERY_RESULT
24: Collect TO SORTED_QUERY_RESULT
25: END FOR
26: FOR every USER_SYNONYM_KEYWORDS_QUERY_RESULT
27:         Read USER_SYNONYM_KEYWORDS_QUERY_RESULT
28: Collect TO SORTED_QUERY_RESULT
29: END FOR
30: FOR every WORDFAMILY_KEYWORDS_QUERY_RESULT
31:         Read WORDFAMILY_KEYWORDS_QUERY_RESULT
32: Collect TO SORTED_QUERY_RESULT
33: END FOR
34: FOR every WORDFAMILY_SYNONYM_KEYWORDS_QUERY_RESULT
35:         Read WORDFAMILY_SYNONYM_KEYWORDS_QUERY_RESULT
36: Collect TO SORTED_QUERY_RESULT
37: END FOR
38: FOR every SYNONYM_KEYWORDS_QUERY_RESULT
39:         Read SYNONYM_KEYWORDS_QUERY_RESULT
40: Collect TO SORTED_QUERY_RESULT
41: END FOR
42: END FOR
```

Listing 5-2: Ranking Algorithm Pseudo Code

Display Generator Algorithm

There are two basic tasks in Display Generator Algorithm implementation**.** Firstly**, Preparing Display format** that designs the display format of the result to the user in Tabular presentation as well as hyperlink of texts. Secondly, **displaying the result** where the organised result will be displayed to the user interface in the designed format. See Appendix C what the format of the result displayed looks like. Most tasks of this algorithm are performed from client side in receiving organised data from the server. See Listing 5.3 for pseudo code of the display generator algorithm.

Pseudo Code

```
1:  Input: SORTED_QUERY_RESULT
2:  3: For each result in SORTED_QUERY_RESULT
4:          Get first row tuple in SORTED_QUERY_RESULT
5:          Prepare hyperlink of first Row
6:          Start tabular view
7:          For each tuple in SORTED_QUERY_RESULT
8:                  For each tuple element
9:                          Display a tuple element in SORTED_QUERY_RESULT
10:                 End For Loop
11:                 End If
12: End For Loop
13: End tabular view
14: End For Loop
```

Listing 5-3: Display Generator Algorithm Pseudo Cod

5.2 EVALUATION

The evaluation process includes evaluation of Combined Databases Schema Construction and the Searching framework with respect to the expected output and research questions.

The evaluation process will give answer to the following evaluation points;

- How correct is the combined Databases Schema created to support the query?
- How successful is the translation of the keyword based query text to SQL query statements?
- How much the keyword based searching is successful with the use of synonyms and word families enhanced keywords compared to without?
- Users rating for their Level of satisfaction in the relevancy of the search result with checklist provided to the user.

The evaluation is made based on sample experimental databases connected to the system and sample set of keywords provided by the researcher and the end users selected for the experiment.

5.2.1 Evaluation Parameters

Evaluation Environment setup

For the evaluation purpose a three-tier architecture is used such that separate machines for database, application, and for browsing. All the three machines where their specification is described below are connected in local area network accessible in commercial bank of Ethiopia service desk department.

Specifications of experimentation machines.

- The computer on which the databases are hosted is with: Processor- Intel ® Core ™ 2 duo CPU E8400 3.00 GHz 3.00 GHz, 2 GB installed RAM, 32 bit windows 7 operating system, 250GB HDD out of which 20GB Free,
- The computer on which the web application is hosted is with: Processor- Intel ® Core ™ 2 duo CPU T5300 1.73 GHz 1.73 GHz, 2 GB installed RAM, 32 bit windows 7 operating system, 111GB HDD out of which 5GB Free,

User selection

Five individuals of mixed level of expertise and job have been randomly and with quota selected from commercial bank of Ethiopia service desk department were used for the experiment. In the department there are two groups of users, the infrastructure group has given a quota of two users, and the application group users has given a quota of three users based on number of users in the group. Users from each group are selected randomly. Since part of the databases connected to the system is from the same department of the users selected, they know about the data stored in the database. It is obvious that for better searching users should have some idea about the data stored in the databases not about the structure of the databases which will help them in selecting searching text better. But for any novice user the system also responds based the users input keyword.

Data Source Selection

For the evaluation purpose the database management systems (DBMSs) used are MySQL 5.5 Server, Oracle 11g Enterprise, and Microsoft SQL Server 2008 R2. Four databases are selected to connect to the system from these DBMSs.

Databases Used

The commercial bank of Ethiopia uses many in-house developed database applications to support its main business operations. Among all, this work has considered only four of them for experimentation and testing purpose. The summarized statistics of the databases chosen are described in table 5.1.

Table 5-1: Database systems used

No	Database Application	Total Table	Total Columns	Total Rows of Data
1	Bond Management System: MS SQL Server	74	916	12,649
2	Coupon Management System: Oracle 11g	49	515	143,005
3	LMTS (Local Money Transfer System): MySQL	93	913	175,953
4	Data Archival System: MS SQL Server	39	282	4,267,169
Total	**4**	**255**	**2626**	**4598776**

Sample User's Searching Text

The sample user's searching text is meant the use of selected set of texts as keywords only for the evaluation of the systems performance in achieving the desired goal. The search texts are selected by both the researcher and the experimentation users. The researchers has chosen by considering the data in the sample databases to cross check among users response with the result whereas the experimentation users are based on the task or event they remember. Users are free to choose any type of set of text(s) since there is no special rule of keyword selection.

Searching Text provided by the researcher

The following are sample list of words or phrases or set of texts that are selected by the researcher as searching text only for this evaluation purpose. These query terms are selected by considering the types of data stored and their existence in multiple databases. Search texts with three words are complex than that of with two words and one word. The system will require more resources for complex search texts than simple search texts. See Appendix B for sample SQL statement constructed for one of the search texts.

RT-1. Cash Deposit
RT-2. Customer Liability case
RT-3. 0912 or 0911
RT-4. 2008-05-07
RT-5. Eth Shipping Lines
RT-6. Anwar ATM Cash

Searching text provided by the experimentation users.

The following are sample list of words or phrases or set of texts that are selected by the experimentation users as searching text only for this evaluation purpose. Five users have selected their own ten (each user two) search texts.

UT-1. Nefas Silk Lafto
UT-2. 10000105386323
UT-3. Negele Borena related
UT-4. 011200004
UT-5. Ahmed Mohammed
UT-6. 0008342bb
UT-7. Bezawit Getachew Assefa

UT-8. Nazeret or Modjo

UT-9. 2014-02

UT-10. 017200854

Evaluation Formulas

Number of Constructed SQL Query Statement

It is obvious that a single SQL Query statement is required to access data from single table. In the SQL query statement construction algorithm of this work the number of SQL statement constructed depends on total number of table objects and total number of query keywords. For a given query keyword three SQL query statement will be constructed per table. One statement for exact matching of searching keyword, the second for the result contain the query keyword separately as a word or phrase, and the third for result contain the query keyword is found as part of the string of records. This is to increase the possibility of getting relevant result for three different parameters. In mathematical term the total number of SQL statement to be constructed for a given set of query keywords is given by

$$\mathbf{SQL_t = 3*T*W_s}$$

Equation 5-1: Number of SQL Statements

Where

- $\mathbf{SQL_t}$ is the total Number of SQL query Statements to be constructed
- **T** is total number of tables to access
- $\mathbf{W_s}$ is Number of keyword sets in Query keyword set.

Rate or percentage

The rate of percentage of execution describes the accuracy of the amount obtained and is given by the formula

$$Rate\ or\ Percentage\ \% = \frac{Actual\ Amount}{Expected\ Amount}\ x\ 100\%$$

Equation 5-2: Rate or Percentage

5.2.2 Evaluation result

Evaluation of Database Schema Construction

In the combined database schema construction every database object will be considered to be constructed into a table object in the combined database schema. The ability of the framework to construct the appropriate database schema is evaluated. But the evaluation of the constructor cannot be described quantitatively since it has to perform all of its intended operation described in table 5.2 (100%) without error. If the constructor fails to perform one of its intended operation the schema will not be constructed. The constructor should also perform similar combined database schema construction for similar database at any time. This means that the constructor never produce different result for a given database at different times. Table 5-2 discusses the status of all operations the combined database schema constructor framework has performed during test.

Table 5-2: Evaluation of Database Schema Construction

No.	Constructor Operations	Operation Status
1	Create Connection to MS SQL Server	Success
2	Create Connection to MySQL Server	Success
3	Create Connection to Oracle 11 g r2 Server	Success
4	Read Databases MS SQL Server	Success
5	Read Databases MySQL Server	Success
6	Read Databases Oracle 11 g r2 Server	Success
7	Read Table objects and respective columns from databases of these database systems	Success
8	Read Table object relationships	Success
9	Create Combined Databases Schema	Success
10	Update Combined Databases Schema	Success
11	Delete or remove Combined Database Schema	Success

From the evaluation table 5-2 it is observed that the combined database schema constructor framework is fully performed the construction of the schema in a way to provide data for SQL Query construction. The framework has collected all required information about available database management systems, all databases connected, all tables in the databases, all columns in all tables, and object relationships in the database. The framework also constructed the

combined database schema after the advanced user has chosen set of relevant tables and columns from all available. Here for some reasons, like security, irrelevancy of data stored to users, blank or few data, etc. the advanced user may not consider all tables in the schema construction. Though for this testing purpose all columns in only tables containing records of the four databases accessed in the three database systems are considered for the schema construction.

Table 5-3 below shows the statistics of objects constructed for combined databases schema out of available objects as described in table 5.1. The total and percentage Figures in the table do not describe the quantitative and qualitative measure of the constructor framework or the schema constructed. It only describes the statistics of database objected constructed to the schema for the purpose of testing the system. The difference is due to use of few database objects for reasons discussed above. These objects are chosen by considering amount of data stored and relevancy to users need.

Table 5-3: Statistics of Combined Databases Schema

No	Database Application	Objects Constructed to Combined Database Schema out of available		
		Total Table	**Total Columns**	**Total Rows of Data**
1	Bond Management System: MS SQL Server	3	76	10,627
2	Coupon Management System: Oracle 11g	5	66	136,484
3	LMTS (Local Money Transfer System): MySQL	5	138	83,114
4	Data Archival System: MS SQL Server	5	50	4,263,039
Total	**Constructed to Schema**	**18**	**330**	**4,493,264**
	Available (see table 6-1)	**255**	**2626**	**4598776**
Percentage of Constructed to Schema		**7.06**	**12.57**	**97.71**

Removing of stop words and inclusion of Synonym

There are prior defined set of stop words where the system accesses during stop word removal. The system has detected and removed prior system defined stop words like Single character, prepositions, etc. that exist in the sample experimentation search keywords, for example ‘or’ in RT-3 and ‘case’ in RT-2 of searching texts provided by the researchers. There is also prior defined set of words with their synonyms and word families in the dictionary corpus and the system has successfully included synonym words and word families for search texts that have synonym and word families in the corpus. For example, the system used ‘Note’ as synonym for word ‘cash’ described in RT-1 and RT-6 of searching texts provided by the researchers.

Evaluation of the Query Keyword query translation to SQL Query

In this evaluation the number of expected SQL Query Statement as calculated from Equation 5-1 is compared with the number of SQL Query Statements returned by the system. It is must that the system should construct three SQL Query statements per a given keyword and this is so for return of more relevant result. Though From table 5-4 it can easy be seen that the system has returned an amount equal to the expected number indicating the system is fully performed the construction of SQL Query Statement that includes users Search keywords. But this does not means that the system will return result for all these SQL statements. The expected number of SQL Statements Constructed is obtained from the calculation result using the formula in section 6.1.5. For example, the system has prepared 5 query keyword sets for users search text annotated with RT-1 (i.e. cash deposit, note deposit, deposit cash, deposit note, deposit), the total number of expected SQL statements constructed for a total of 18 tables constructed in the schema (see table 5.3) are 3*5*18=270 as indicated in the table 5-4.

Table 5-4: Expected versus Obtained SQL Query Statements

Query	No of Query Keyword Sets	No of SQL Statements Constructed for Total of 18 tables considered		Remark
		Expected	Constructed	
RT-1.	5	270	270	
RT-2.	6	324	324	
RT-3.	18	972	972	
RT-4.	1	54	54	
RT-5.	15	810	810	
RT-6.	5	270	270	
UT-1.	5	270	270	
UT-2.	15	810	810	
UT-3.	5	270	270	
UT-4.	5	270	270	
UT-5.	5	270	270	
UT-6.	1	54	54	
UT-7.	5	270	270	
UT-8.	5	270	270	
UT-9.	1	54	54	
UT-10.	1	54	54	

Query result with synonym and without synonym

One of the contributions of this work is to include synonym words as part of users searching keyword for better return of results. Table 5-5 compares (1) number of SQL Query Statement construction with including and without including synonym words and word families, and (2) number of SQL Query result returned with including and without including synonym words. For example, for users search text RT-1 the system has returned a query result of 6 with considering the synonym words, and 4 query results without considering the synonyms where this indicates that the system has increased the number of query result returned by 2 (50%) because of including synonym words. From the experimentation result we can conclude that inclusion of synonym words and word families as part of query keyword will increase the number of SQL Query Statement constructed and the number of relevant result obtained in a significant amount.

Table 5-5: Query result with and without synonym words

keyword Type	Non Null Records returned by the translated SQL Statements			Change observed in percentage For including Synonym words	Remark
	Keyword With synonym	Keyword Without Synonym	Change		
RT-1	6	4	+2	+50%	
RT-2	11	9	+2	+22.22%	
RT-6	10	6	+4	+66.66%	

Evaluation of Users Level of Satisfaction

It is difficult to measure the level of satisfaction of system users due to the difference in nature of satisfaction and other factors. The most common methods of getting users satisfaction or opinion about an event is using interview and questionnaire. The level of satisfaction of the experimentation users is assessed or estimated with help of questionnaire prepared as shown in Appendix A. The experimentation users have given access to the system through local network from their own workstation. Six search keywords selected by the researcher is given to five experimentation users and requested each user to include their own two search queries related to their work area. This means each user can test for eight search keywords where the total of search keywords of the experiment became forty (40). By average it is observed from table 5-6 that 65% responses indicate that the system has fair response time, 67.5% response indicates as the system responds relevant result to provided search keywords, 77.5% response shows good ordering mechanism has used, 55% responses indicates acceptable quantity of result is returned, and 72.5% responses has indicated for less repetitiveness of the result.

Table 5-6: Experimentation Users level of satisfaction assessment

Satisfaction Parameters	Level of User's Satisfaction									
	1		2		3		4		5	
	no	%	no	%	no	%	no	%	no	%
Response Time	-	-	9	22.50	26	65.00	5	12.50	-	-
Relevancy of result	-	-	2	5.00	8	20.00	27	67.50	3	7.50
Order of relevancy from high to low	-	-	-	-	9	22.50	31	77.50	-	-
Quantity of Result	-	-	6	15.00	22	55.00	12	30.0	-	-
Repetitiveness of Result	-	-	-	-	7	17.5	29	72.5	4	10.0

The graphical representation of table 5-6 is described below in Figure 6-1.

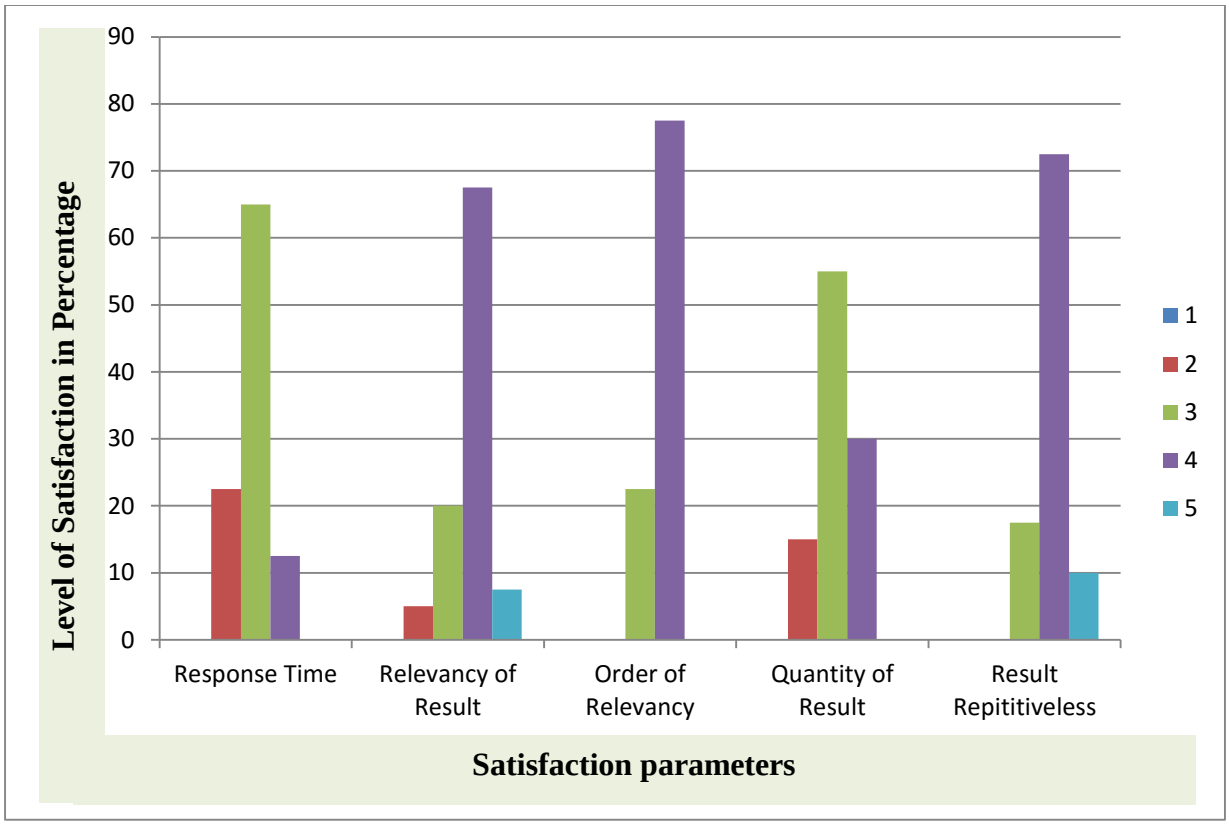


Figure 5-7: Experimentation Users level of satisfaction assessment

CHAPTER SIX: CONCLUSION

The query constructor framework for web-based search of relational database passes through various stages and uses different algorithms to perform keyword and synonym based searching. The quality and usability of the system depends on the summative quality and effectiveness of algorithms and processes in each stage. The research work presented in this report has made scientific contributions in areas of simplification of searching to relational databases with web-based interfaced databases using keywords.

In this chapter the contribution of the work, some of constraints that may affect the system with their possible means of handling, conclusion, and recommendations concerning the whole work is discussed briefly. Furthermore, future works that can enhance the performance of this work are also discussed briefly.

6.1 Contribution of the Work

A number of related researches in area of keyword and full text base searching in relational databases are reviewed where some of them are integrated in relational database system and others are interfaced as graphical user interfaces, like web interfaces. This thesis reviewed what is done previously and identified which gap still exists.

In general the contributions of this work are

- Design of a general architecture for keyword based search using a compound database schema,
- Design and development of a Compound database constructor,
- Removing of noise (minor errors) from the user's query text,
- Inclusion of synonym words and word families to the users' keywords so as to improve the results with better relevancy,
- Introducing use of single user interface to formulate query text over more than one database systems.
- Introducing use of more than one database systems connected to single system interface.

- The design of the framework for integration of more than one database systems, more than one database, and many tables having relational constraints in to combined databases schema.

The implemented system has the following features.

- It can be accessed using any web browser from any platform of machines that has connectivity to the application server.
- The system uses three tier architecture;
- The system can connect to one or more of the databases considered and tables of the databases can be selected;
- The system integrates three types of database systems MySQL 5.5 Server, SQL Server 2008 R2, and Oracle 11g Express Edition.
- The system accepts users query texts as searching keywords and Performs text pre-processing tasks such as noise reduction
- Prepares query keywords with adding their synonym words and categorising set of keywords,
- Extract database and table information and translate the pre-processed query text to SQL Query Statements,
- Executes the SQL Query Statements and organizes returned results in a ranked order based on their closeness to the query text,
- Generate links to the returned results to enable users navigate to the details of results.

In general the work will add contribution on studies focused on improving and facilitating keyword based searching in relational database.

6.2 Constraints

It is obvious that there might be constraints that affects or degrades the quality and effectiveness of the whole system as well as its components. These constraints are presented below.

Constraints in Combined Database schema

The combined database schema has to contain a full information or data about all databases connected to the system. The combined database schema constructor has to collect necessary information about the database systems connected to the system and with their objects. The challenge here is the difference in technological nature and configuration of the database

systems and their objects. Due to these differences a method or code used for a given database system requires some modification or change before applying to the other. For example, the way Meta data is stored in Microsoft SQL Server 2008 is different from that of MySQL, and Oracle. These challenges are handled by using a separate code and configuration methods.

For database schema construction the constructor user (or the database administrator) has to contain enough knowledge of the connected databases like for example the knowledge of databases, tables, columns, and their data. This knowledge is important in selecting which table and column to consider so that the novice system users are not confused by dealing with unrelated data of the system and that their access in seconds. The other challenge is the combined database schema constructor will not track the changes in the structure of connected databases unless the user constructed it.

Constraints in Noise Removal

The users that uses the system could have different skill of using the system including typing of search texts. The users are not required to necessarily know about the databases but it is advisable to know about the data in the databases and the keyword they use to search for getting information from the database. During keyword typing users may make some errors like, spelling error, too much keywords, unnecessary stop words, or combination of these. Minor errors are designed to be handled but complex errors cannot be handled by system and so the user might not get adequate result for query texts that are too complex or too far from the content of the databases.

Constraints in Including Synonyms

In fact the user is not expected to know every detail about the database and the exact data stored in it and also the users are not required to know the exact keyword he/she has to use. Having this all a user might type a keyword(s) that do not exist in the database and no relevant result is returned. The importance of including synonym words of the user's keyword comes here to overcome these problems. But, since sometimes synonym words depends on contextual meaning, collection of two or more words can be synonym for a single keyword, and collecting synonym words provides another burden. For the sake of this work more than 100 pair of synonyms are collected where some of them are designed to help the experimental evaluation of the work.

Constraints in SQL Statement Construction and Execution

The system designed to use three different database systems as discussed in the above sections. The Syntax of SQL query statement in these DB systems differ as for example MySQL database system may use

```
SELECT * FROM `HRMgt`.`EMPLOYEE`
```

Whereas the equivalent of this in SQL Server 2008 R2 is

```
USE [HRMgt] SELECT * FROM [EMPLOYEE],
```

or

```
SELECT * FROM [HRMgt].[dbo].[EMPLOYEE] .
```

This challenge is solved by developing separate algorithm for each database system.

If the number of databases and their object to be constructed to database schema is increased the length and number of SQL Query Statement constructed will also increase. As large number and length of SQL statements executed the performance of the system in returning the result timely will decrease. Execution of SQL Query Statement in one database system results better performance than executing by switching between different DB systems.

Users Understanding of the System

During the experimental evaluation it is observed that users sometimes use large collection of keywords, keywords with too much stop words, collection of abbreviations, numeric or date formats, and also incomplete keyword which is difficult to complete by the system. Such and other errors will force the system to spend more of its time in pre-processing these errors which at last degrades the system performance, and even the system may not return result or irrelevant result might be returned.

6.3 Conclusions

Recently keyword based searching is becoming more popular in areas of relational database in two ways: (1) integrated with database systems and (2) interfaces with separate GUI. A lot of researches attempts and studies have been made in areas of keyword based searching over relational databases. Some of the researches focus on improving searching mechanisms integrated in the databases systems where as others focused on designing frameworks and small

systems interfaced above a database. This work is categorised with the second group of researches focusing on simplifying searching trend and enhancing performance of returning better relevant result.

The system as a whole has two major tasks where each task is accomplished by two different groups of users with different level of domain knowledge. The first task is construction of combined databases schema for databases connected to the system. The GUI interfaced tool or framework developed using java programming language dedicated to constructing the combined databases schema. It connects to the database systems, collects about all databases tables and column information, and present to the user. The advanced user who has enough knowledge about the databases and their data will select the appropriate databases, tables and columns and orders the tool to construct the schema. Unless the database schema is constructed prior to searching to the system, it will not function.

The second task is searching for keyword. The system that enables searching is developed using JSP programing language with web interface. The system passes through various stages and algorithms during searching before returning the result. After the system accepted user's searching keywords it tries to reduce noise, include synonym words and word families, translate keyword based query to SQL Statements, Execute the SQL statements, manage the result returned in ranked order, prepare display format, and finally display the result in ordered relevancy. During all these process the system will communicate with the dictionary corpus, Combined Database Schema, and connected database. The displayed results will help user to navigate through databases using links in the result.

This work has made scientific contribution by introducing the use of synonym words, connecting to more than one heterogeneous database systems and more than one database, developing of combined database schema. The successful implementation of the overall system is tested with experimentation in consideration of sample experimentation data obtained from Commercial Bank of Ethiopia. The experimentation and evaluation process of the system has indicated that inclusion of synonym and word families will return better result than pure user's keywords.

The proper use of the system will return result relevant to the users request with better processing performance. A better understanding of the system will help on the proper use of the system. The following points present a summary.

- Keyword based searching over databases is a recently emerging field of work that helps novice users in searching through relational databases using keywords without the user having the knowledge of database structure and construction of SQL Statements.
- Processing user's keyword for noise reduction will increase the possibility of getting better result the user intends.
- Including synonym words and word families in user's searching keyword will increase the relevancy of the result returned and possibility of getting more results.
- Using of more than one database systems and more than one databases will increase the flexibility and usability of system that integrates these databases and systems.
- Developing a combined database schema will help in managing information of all databases connected to the system that enables easy retrieval and construction of SQL Statement.
- Use of small number of keyword sets and small number of database objects will decrease the length and number of constructed SQL query Statements where this intern increases the performance of executing the SQL statements and managing the returned results, and also the quality of result.

6.4 Future Work

The analysis of experimentation result of the system indicates that the system has shown better result of keyword based searching when synonym and word families are included. But this does not mean that the system is perfect and does not need further improvement. In the future the system can be improved for better performance of returning better relevant result of keyword and synonym based searching. Improving the mechanism of keyword and synonym based searching by enhancing noise reduction algorithm and addition of other features, consideration of contextual meaning of the keywords, and improving the combined databases schema constructor will be part of future work.

Reference

[1]. Stijn Vansummeren, Alejandro Vaisman, **Translating Relational Databases into Linked Open Data Master thesis for obtaining the Diploma Master in Computer Science and Engineering**, Europe University, 2012

[2]. Varun Kacholia, Shashan kPandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, HrishikeshKarambelkar, **Bidirectional Expansion For Keyword Search on Graph Databases,** Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

[3]. Erhard Rahm, **Evaluation of object-relational database systems for full text retrieval**, Univ. Leipzig, Germany, May 1998.

[4]. Rose Catherine K., S. Sudarshan, **Graph Clustering for Keyword Search**, Conference Paper, Proceedings of the 15th International Conference on Management of Data, International School of Information Management, Mysore, India Indian Institute of Technology Bombay, December 2009

[5]. Eyas El-Qawasmeh, OssamaAbu-Eid, AbdallahAlashqur, **A Framework For Processing Keyword-Based Queries In Relational Databases,** Journal of Theoretical And Applied Information Technology, Jordan University Of Science and Technology, Jordan, Applied Science University, Jordan, 2010.

[6]. Gaurav Bhalotia, ArvindHulgeri, CharutaNakhe, SoumenChakrabarti, S. Sudarshan, **Keyword Searching and Browsing in Databases using BANKS**, Computer Science and Engineering. Dept., I.I.T. Bombay, 2003.

[7]. H. Bast, A.Chitea, F.M.Suchanek, and I. Weber. **Ester: efficient search on text, entities, and relations**. In SIGIR, pages 671-678, 2007.

[8]. Luping Li, Stephen Petschulat, Guanting Tang, Jian Pei, Wo-Shun Luk**. Efficient and Effective Aggregate Keyword Search on Relational Databases**. Baidu Inc., Beijing, China, SAP Research, Vancouver, BC, Canada, Simon Fraser University, Burnaby, BC, Canada, 2012.

[9]. Hannah Bast, BjörnBuchhold. **An Index for Efficient Semantic Full-Text Search**. Department of Computer Science University of Freiburg79110Freiburg, Germany, 2013.

[10]. Holger Bast, Alexandru Chitea, Fabian Suchanek, Ingmar Weber. **ESTER: Efficient Search on Text, Entities, and Relations**. Max Planck-Institute for Informatik Saarbr¨ucken, Germany, Jun 2007.

[11]. Chavdar Botev, Sihem Amer-Yahia, Jayavel Shanmugasundaram. **Expressiveness and Performance of Full-Text Search Languages**, Cornell University, AT&T Labs–Research, Cornell University, 2006.

[12]. B. Aditya, Gaurav Bhalotia, Soumen Chakrabarti, Arvind Hulgeri, Charuta Nakhe, Parag S. Sudarshan. **BANKS: Browsing and Keyword Searching in Relational Databases**. Computer Science and Engineering. Dept., I.I.T. Bombay, Aug 2012.

[13]. Fang Liu, Clement Yu, Weiyi Meng, Abdur Chowdhury. **Effective Keyword Search in Relational Databases.** Computer Science Department University of Illinois at Chicago, Computer Science Department Binghamton University, Search & Navigation Group America Online, Inc.2006.

[14]. Richard Wheeldon, Mark Levene and Kevin Keenoy. **DbSurfer: A Search and Navigation Tool for Relational Databases.** School of Computer Science and Information Systems Birkbeck University of London Malet St, London WC1E 7HX, United Kingdom, 2004.

[15]. Tihitina Petros Degsew. **Modeling and Designing Amharic Query System To Bilingual (English-Amharic) Databases.** Thesis submitted to the school of graduate studies of the Addis Ababa University, February 2014.

[16]. M Thangaraj, Member, IACSIT,V Gayathri, **A New Context Oriented Synonym Based Searching Technique for Digital Collection**, International Journal of Machine Learning and Computing, Vol.1, No. 1, April 2011.

[17]. Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, Dong Xin, **A Framework for Robust Discovery of Entity Synonyms**, Microsoft Research, One Microsoft Way, Redmond, WA 98052, April 2012

[18]. Xing Wei, Fuchun Peng, Huishin Tseng, Yumao Lu, Xuerui Wang, Benoit Dumoulin, **Search with Synonyms: Problems and Solutions**, 701 First Avenue, Sunnyvale, California, USA, 94089, 2010

[19]. Reiner Kraft and Jason Zien, **Mining Anchor Text for Query Refinement**, Proceedings of the 13th international conference on World Wide Web , IBM Almaden Research Center, 2004

[20]. Marie Jacob and Zachary Ives, **Sharing Work in Keyword Search over Databases**, University of Pennsylvania Philadelphia, PA, USA. 2011.

[21]. Sonia Bergamaschi et. al. **Keyword Search over Relational Databases: A Metadata Approach**, University of Modena and Reggio Emilia, Italy; University of Zaragoza, Spain; University of Trento, Italy, 2011

[22]. Michael J. Cafarella, Oren Etzioni, Dan Suciu, **Structured Queries Over Web Text**, University of Washington Seattle, WA 98195.

[23]. Joel Coffman, Alfred C. Weaver, "**A Framework for Evaluating Database Keyword Search Strategies**". University of Virginia Char lottesville, VA.

[24]. Alpa Jain, AnHai Doan, Luis Gravano, **Optimizing SQL Queries over Text Databases**, Data Engineering, ICDE. IEEE 24th International Conference on. IEEE 2008.

[25]. Sanjay Agrawal et. al., **Automated Ranking of Database Query Results**, Microsoft Research and Computer Science Dept Stanford University, 2003.

[26]. Yi Chen , Wei Wang, Ziyang Liu, **Keyword-based Search and Exploration on Databases**, Data Engineering (ICDE), 2011 IEEE 27th International Conference on, pages 1380 – 1383, 11-16 April 2011.

[27]. Garrett Wolf Hemal Khatri, et al. **Query Processing over Incomplete Autonomous Databases**, Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, pages 1430 – 1432, 15-20 April 2007.

[28]. Cong Yu and H. V. Jagadish, **Querying Complex Structured Databases**, VLDB '07 Proceedings of the 33rd international conference on Very large data bases, Pages 1010-1021, 2007.

[29]. Yi Chen et. al. **Keyword Search on Structured and Semi-Structured Data**, Arizona State University, University of New South Wales and NICTA,2009

[30]. Justin Zobel et. al. **An Efficient Indexing Technique for Full-Text Database Systems**, VLDB '92 Proceedings of the 18th International Conference on Very Large Data bases, Pages 352 – 362, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©1992.

[31]. V. Hristidis, L. Gravano, and Y. Papakonstantinou. **Efficient IR-style keyword search over relational databases**. VLDB '03 Proceedings of the 29th international conference on Very large data bases - Volume 29, Pages 850-861, 2003.

[32]. V. Hristidis and Y. Papakonstantinou. **Discover: Keyword search in relational databases**. VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases, Pages 670-681, 2002.

[33]. P. P. Talukdar, Z. G. Ives, and F. Pereira. **Automatically incorporating new sources in keyword search-based data integration**. SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, Pages 387-398, New York, NY, USA, 2010.

[34]. P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. Pereira, and S. Guha. **Learning to create data-integrating queries**. Proceedings of the VLDB Endowment, Volume 1 Issue 1, Pages 785-796, August 2008.

[35]. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. **Keyword searching and browsing in databases using BANKS**. Data Engineering, Proceedings. 18th International Conference on, pages 431 – 440, IEEE, San Jose, CA, 2002.

[36]. H. He, H. Wang, J. Yang, and P. S. Yu. **Blinks: ranked keyword searches on graphs**. SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Pages 305-316, New York, NY, USA, 2007.

[37]. M. Lenzerini. **Data integration: A theoretical perspective**. PODS '02 Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 233–246. New York, NY, USA, 2002.

[38]. R. Fagin, R. Kumar, and D. Sivakumar. **Efficient similarity search and classification via rank aggregation**. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pages 301–312. ACM Press, 2003.

[39]. David Johnson, Vishv Malhotra, Peter Vamplew and Sunanda Patro, **Refining Search Queries from Examples Using Boolean Expressions and Latent Semantic Analysis**, International Conference on Artificial Intelligence in Science and Technology, Hobart, Tasmania, Australia, Hobart Tasmania, Australia, Nov 2014.

[40]. Von der Fakultat et. Al. **Integration Of Yago Ontology In The Iqp Query Construction System To Support Efficient Query Construction Over A Large-Scale Relational Database**, thesis of master of science in computer science, Iryna Oelze, ICDB, 2008.

[41]. Xi ng Wei , Fuchun Peng, H ui shi n Tseng, Yumao Lu, Xuerui Wang, Benoit Dumoulin, **Search with Synonyms : Problems and Solutions**, Yahoo ! Labs and Sunnyvale, Poster Volume, pages 1318–1326, Beijing, August 2010

[42]. Nattiya Kanhabua and Kjetil Norvag, **Exploiting Time-based Synonyms in Searching Document Archives**, JCDL '10 Proceedings of the 10th annual joint conference on Digital libraries, Pages 79-88, New York, NY, USA 2010.

[43]. A. Singhal. **Modern Information Retrieval: A Brief Overview**. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol. 24, No. 4. pp. 35-42, 2001.

[44]. Y. Luo, X. Lin, W. Wang, and X. Zhou. **SPARK: Top-k Keyword Query in Relational Databases**. SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Pages 115-126, New York, NY, USA, June 2007.

[45]. F. Liu, C. Yu, W. Meng, and A. Chowdhury. **Effective Keyword Search in Relational Databases.** In SIGMOD '06, pages 563-574, June 2006.

[46]. L. Qin, J. X. Yu, and L. Chang. **Keyword Search in Databases**: The Power of RDBMS. In SIGMOD '09, pages 681-694, June 2009.

[47]. V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. **Bidirectional Expansion For Keyword Search on Graph Databases**. In VLDB '05, pages 505-516, August 2005.

[48]. B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. **Finding Top-k Min-Cost Connected Trees in Databases.** In ICDE '07, pages 836-845, April 2007.

[49]. G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. EASE: **An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data**. In SIGMOD '08, pages 903-914, June 2008.

[50]. K. Golenberg, B. Kimelfeld, and Y. Sagiv. **Keyword Proximity Search in Complex Data Graphs**. In SIGMOD '08, pages 927-940, June 2008.

[51]. B. B. Dalvi, M. Kshirsagar, and S. Sudarshan. **Keyword Search on External Memory Data Graphs**. PVLDB, 1(1):1189{1204, 2008.}

[52]. Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das, "**DBXplorer: A system for Keyword-Based Search Over Relational Databases**," In Proceedings of International Conference on Data Engineering (ICDE), pp. 5–16, 2002.

[53]. Brigitte Safar, Hassen Kefi, Chantal Reynaud, **Onto Refiner, a user query refinement interface usable for Semantic Web Portals,** Application of Semantic Web Technologies to Web Communities Workshop, 16th European Conference on Artificial Intelligence, Valencia, Spain, August 22-27, 2004.

[54]. Seid Muhe Yimam, and Mulugeta Libisie, **Amharic Question Answering (AQA);** 10th Dutch-Belgian Information Retrieval Workshop, 2010.

[55]. Mequannint Munye, and Solomon Atnafu, "**Amharic-English Bilingual Web Search Engine**", Msc Thesis, Addis Ababa University, College of Natural Science, Department of Computeer Science, October 2012.

Appendix A: Assessment Check list

Query Constructor Framework for web based search interface to relational databases

Users Level of Satisfaction Assessment Check list

This system used for Keyword and synonym based searching over databases. The system has a web interface that accepts user input of keyword(s) and display result. The system is developed in partial fulfilment for the degree of Master of Science in computer science in Addis Ababa University.

This checklist is prepared to assess users' level of satisfaction in getting relevant result of a given set of keywords. I, the researcher, request your good willingness and cooperation in assessing the effectiveness of this system. Six sample searching keywords are provided below in this checklist and you are requested to provide two your own searching keywords. I thanks you in advance.

For this experimentation evaluation purpose three databases of (1) Incident Management System, (2) Network Devices Management System, and (3) Credit Management System of Commercial Bank of Ethiopia are used.

1. **User Information**

Department: _____

Job Title: _____

Year of Experience: _____

2. **Searching keywords provided by the researcher**

RT-1. Network Installing

RT-2. Branch extension case

RT-3. Hardware and Software repair

RT-4. Airport

RT-5. Power Star UPS

RT-6. Switch or router

3. **Please provide two your own Searching keywords**

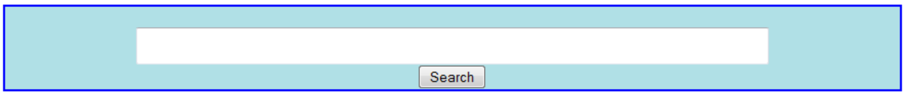
UT-1. _____

UT-2. _____

Instruction:

step_1: From your machine in the network access the system using a browser with address '**http://CBE-HO:8080/KSQCFI/index.jsp**' and you get the interface

Query Constructor Framework for web based search interface to relational databases



step_2: Use searching keywords provided by the researcher and your own keywords described in the cover page, insert to the search text box, and click the Search button after few seconds a result will be displayed. Based on the result you obtained please fill the following checklist. One separate checklist is provided for one searching keyword.

step_3: In the checklist circle your response.

= =

Type ID of Searching Keyword Used: ____________________________

Evaluation Parameters	**Level of User's Satisfaction**				
	1	**2**	**3**	**4**	**5**
Response Time	Very Slow	Slow	Fair	Fast	Very Fast
Relevancy of result	Unrelated	Irrelevant	Medium	Relevant	More Relevant
Order of relevancy from high to low	Unordered	Poor order	Acceptable	Good Order	Perfect Order
Quantity of Result	Very Less	Less	enough	More than enough	Too much
Repetitiveness of Result	-	Very repetitive	Acceptable	Less Repetitiveness	No repetitive

Thanks again!

E N D

Appendix B: Sample SQL Query Statement Constructed

The following are some of sample SQL Query Statement constructed for searching text '**Anwar ATM Cash'**.

_ _

1. Statement 0>> USE [bonddbtest] SELECT * FROM [tblCustomer] WHERE (([tblCustomer].[FullName] LIKE '%anwar%' AND [tblCustomer].[FullName] <> 'anwar' AND [tblCustomer].[FullName] NOT LIKE '% anwar %') OR ([tblCustomer].[BusinessName] LIKE '%anwar%' AND [tblCustomer].[BusinessName] <> 'anwar' AND [tblCustomer].[BusinessName] NOT LIKE '% anwar %') OR ([tblCustomer].[City] LIKE '%anwar%' AND [tblCustomer].[City] <> 'anwar' AND [tblCustomer].[City] NOT LIKE '% anwar %') OR ([tblCustomer].[SubCity] LIKE '%anwar%' AND [tblCustomer].[SubCity] <> 'anwar' AND [tblCustomer].[SubCity] NOT LIKE '% anwar %') OR ([tblCustomer].[State] LIKE '%anwar%' AND [tblCustomer].[State] <> 'anwar' AND [tblCustomer].[State] NOT LIKE '% anwar %') OR ([tblCustomer].[Woreda] LIKE '%anwar%' AND [tblCustomer].[Woreda] <> 'anwar' AND [tblCustomer].[Woreda] NOT LIKE '% anwar %') OR ([tblCustomer].[ZipCode] LIKE '%anwar%' AND [tblCustomer].[ZipCode] <> 'anwar' AND [tblCustomer].[ZipCode] NOT LIKE '% anwar %') OR ([tblCustomer].[HouseNumber] LIKE '%anwar%' AND [tblCustomer].[HouseNumber] <> 'anwar' AND [tblCustomer].[HouseNumber] NOT LIKE '% anwar %') OR ([tblCustomer].[PhoneNo] LIKE '%anwar%' AND [tblCustomer].[PhoneNo] <> 'anwar' AND [tblCustomer].[PhoneNo] NOT LIKE '% anwar %') OR ([tblCustomer].[Nationality] LIKE '%anwar%' AND [tblCustomer].[Nationality] <> 'anwar' AND [tblCustomer].[Nationality] NOT LIKE '% anwar %') OR ([tblCustomer].[Occupation] LIKE '%anwar%' AND [tblCustomer].[Occupation] <> 'anwar' AND [tblCustomer].[Occupation] NOT LIKE '% anwar %') OR ([tblCustomer].[IdCardNumber] LIKE '%anwar%' AND [tblCustomer].[IdCardNumber] <> 'anwar' AND [tblCustomer].[IdCardNumber] NOT LIKE '% anwar %') OR ([tblCustomer].[PlaceIdIssued] LIKE '%anwar%' AND [tblCustomer].[PlaceIdIssued] <> 'anwar' AND [tblCustomer].[PlaceIdIssued] NOT LIKE '% anwar %') OR ([tblCustomer].[InsertUserId] LIKE '%anwar%' AND [tblCustomer].[InsertUserId] <> 'anwar' AND [tblCustomer].[InsertUserId] NOT LIKE '% anwar %') OR ([tblCustomer].[UpdateUserId] LIKE '%anwar%' AND [tblCustomer].[UpdateUserId] <> 'anwar' AND [tblCustomer].[UpdateUserId] NOT LIKE '% anwar %'))

1: <u>anwar</u> : is the query keyword for this query result.

<u>Bond Management System</u> : is the system where this query result retrieved from
The link describes the first record of the result. The result contains about 2 rows with 23 columns. Click the link for more detail.
<u>ANWAR NASIR MELKE | HANA | HANA |</u>

----

2. Statement 1>> SELECT * FROM `baslmtdb`.`incoming` WHERE ((`incoming`.`BusClassID` LIKE '% anwar %' AND `incoming`.`BusClassID` <> 'anwar') OR (`incoming`.`MTID` LIKE '% anwar %' AND `incoming`.`MTID` <> 'anwar') OR (`incoming`.`TestNo` LIKE '% anwar %' AND `incoming`.`TestNo` <> 'anwar') OR

(`incoming`.`CodeNo` LIKE '% anwar %' AND `incoming`.`CodeNo` <> 'anwar') OR (`incoming`.`AccountNo` LIKE '% anwar %' AND `incoming`.`AccountNo` <> 'anwar') OR (`incoming`.`Ben_Name` LIKE '% anwar %' AND `incoming`.`Ben_Name` <> 'anwar') OR (`incoming`.`Remark` LIKE '% anwar %' AND `incoming`.`Remark` <> 'anwar') OR (`incoming`.`Rem_Name` LIKE '% anwar %' AND `incoming`.`Rem_Name` <> 'anwar') OR (`incoming`.`Dec_Name` LIKE '% anwar %' AND `incoming`.`Dec_Name` <> 'anwar') OR (`incoming`.`Details` LIKE '% anwar %' AND `incoming`.`Details` <> 'anwar') OR (`incoming`.`RecBy` LIKE '% anwar %' AND `incoming`.`RecBy` <> 'anwar') OR (`incoming`.`RelayedBy` LIKE '% anwar %' AND `incoming`.`RelayedBy` <> 'anwar') OR (`incoming`.`Phone` LIKE '% anwar %' AND `incoming`.`Phone` <> 'anwar') OR (`incoming`.`UserID1` LIKE '% anwar %' AND `incoming`.`UserID1` <> 'anwar') OR (`incoming`.`UserID2` LIKE '% anwar %' AND `incoming`.`UserID2` <> 'anwar') OR (`incoming`.`UserID3` LIKE '% anwar %' AND `incoming`.`UserID3` <> 'anwar') OR (`incoming`.`UserID4` LIKE '% anwar %' AND `incoming`.`UserID4` <> 'anwar') OR (`incoming`.`UserID5` LIKE '% anwar %' AND `incoming`.`UserID5` <> 'anwar') OR (`incoming`.`SmallName` LIKE '% anwar %' AND `incoming`.`SmallName` <> 'anwar') OR (`incoming`.`UserID6` LIKE '% anwar %' AND `incoming`.`UserID6` <> 'anwar'))

2: anwar : is the query keyword for this query result.

LMTS (Local Money Transfer System) : is the system where this query result retrieved from

The link describes the first record of the result. The result contains about 10 rows with 47 columns. Click the link for more detail.

NA | 0003359 | 8837 | 8837 | NA | MOHAMMED ANWAR AWOL | Kebele = 01/02/03@@@@@@@@:Tel. No = 0911139280:Kifle Ketema= ADK:@@@@@:House No = 18/088:ID No = 01/02/03-18/088:@@@@@:Date Iss. = 21.06.98 | MOHAMMED KEMAL | NA | MD = 04/06/09 by JIMMA MAIN BRN. Using Local TTNo = 0003359 DD 05/06/09 | GETAHUN ASSEFA | HOWOT G/EGZIABHER | NA | MULUG | GETAH | GETAH | NA | ESUEN | MAA | EJIGM |

---

3. Statement 2>> SELECT * FROM `baslmtdb`.`incoming` WHERE ((`incoming`.`BusClassID` LIKE '%anwar%' AND `incoming`.`BusClassID` <> 'anwar' AND `incoming`.`BusClassID` NOT LIKE '% anwar %') OR (`incoming`.`MTID` LIKE '%anwar%' AND `incoming`.`MTID` <> 'anwar' AND `incoming`.`MTID` NOT LIKE '% anwar %') OR (`incoming`.`TestNo` LIKE '%anwar%' AND `incoming`.`TestNo` <> 'anwar' AND `incoming`.`TestNo` NOT LIKE '% anwar %') OR (`incoming`.`CodeNo` LIKE '%anwar%' AND `incoming`.`CodeNo` <> 'anwar' AND `incoming`.`CodeNo` NOT LIKE '% anwar %') OR (`incoming`.`AccountNo` LIKE '%anwar%' AND `incoming`.`AccountNo` <> 'anwar' AND `incoming`.`AccountNo` NOT LIKE '% anwar %') OR (`incoming`.`Ben_Name` LIKE '%anwar%' AND `incoming`.`Ben_Name` <> 'anwar' AND `incoming`.`Ben_Name` NOT LIKE '% anwar %') OR (`incoming`.`Remark` LIKE '%anwar%' AND `incoming`.`Remark` <> 'anwar' AND `incoming`.`Remark` NOT LIKE '% anwar %') OR (`incoming`.`Rem_Name` LIKE '%anwar%' AND `incoming`.`Rem_Name` <> 'anwar' AND `incoming`.`Rem_Name` NOT LIKE '% anwar %') OR (`incoming`.`Dec_Name` LIKE '%anwar%' AND `incoming`.`Dec_Name` <> 'anwar' AND `incoming`.`Dec_Name` NOT LIKE '% anwar %') OR (`incoming`.`Details` LIKE '%anwar%' AND `incoming`.`Details` <> 'anwar' AND `incoming`.`Details` NOT LIKE '% anwar %') OR (`incoming`.`RecBy` LIKE '%anwar%' AND `incoming`.`RecBy` <> 'anwar' AND `incoming`.`RecBy` NOT LIKE '% anwar %') OR

(`incoming`.`RelayedBy` LIKE '%anwar%' AND `incoming`.`RelayedBy` <> 'anwar' AND `incoming`.`RelayedBy` NOT LIKE '% anwar %') OR (`incoming`.`Phone` LIKE '%anwar%' AND `incoming`.`Phone` <> 'anwar' AND `incoming`.`Phone` NOT LIKE '% anwar %') OR (`incoming`.`UserID1` LIKE '%anwar%' AND `incoming`.`UserID1` <> 'anwar' AND `incoming`.`UserID1` NOT LIKE '% anwar %') OR (`incoming`.`UserID2` LIKE '%anwar%' AND `incoming`.`UserID2` <> 'anwar' AND `incoming`.`UserID2` NOT LIKE '% anwar %') OR (`incoming`.`UserID3` LIKE '%anwar%' AND `incoming`.`UserID3` <> 'anwar' AND `incoming`.`UserID3` NOT LIKE '% anwar %') OR (`incoming`.`UserID4` LIKE '%anwar%' AND `incoming`.`UserID4` <> 'anwar' AND `incoming`.`UserID4` NOT LIKE '% anwar %') OR (`incoming`.`UserID5` LIKE '%anwar%' AND `incoming`.`UserID5` <> 'anwar' AND `incoming`.`UserID5` NOT LIKE '% anwar %') OR (`incoming`.`SmallName` LIKE '%anwar%' AND `incoming`.`SmallName` <> 'anwar' AND `incoming`.`SmallName` NOT LIKE '% anwar %') OR (`incoming`.`UserID6` LIKE '%anwar%' AND `incoming`.`UserID6` <> 'anwar' AND `incoming`.`UserID6` NOT LIKE '% anwar %'))

3: <u>anwar</u> : is the query keyword for this query result.

<u>LMTS (Local Money Transfer System)</u> : is the system where this query result retrieved from

The link describes the first record of the result. The result contains about 154 rows with 47 columns. Click the link for more detail.

<u>NA | 0000425 | 6848 | 6848 | NA | BEDIRU OMER INDRIS | @@@@:Tel. No = 0911425631:Kifle Ketema= REG.CUS:@@@@:@@@@:@@@@:@@@@:@@@@ | ANWAR HUSSEIN | NA | MD = 18/04/09 by HIRMATTA BRN. Using Local TTNo = 0000425 DD 20/04/09 | NEBIYAT | DESALEGN GUTEMA | NA | YITAG | NEBI | NEBI | NA | NEIMA | BOI | HEYSU |</u>

---

4. Statement 3>> SELECT * FROM CUSTOMER WHERE ((CUSTOMER.ACCOUNTNUMBER LIKE '% anwar %' AND CUSTOMER.ACCOUNTNUMBER <> 'anwar') OR (CUSTOMER.ACCOUNTNUMBER LIKE '% ANWAR %' AND CUSTOMER.ACCOUNTNUMBER <> 'ANWAR') OR (CUSTOMER.CUSTOMERNAME LIKE '% anwar %' AND CUSTOMER.CUSTOMERNAME <> 'anwar') OR (CUSTOMER.CUSTOMERNAME LIKE '% ANWAR %' AND CUSTOMER.CUSTOMERNAME <> 'ANWAR') OR (CUSTOMER.ACCOUNTTYPE LIKE '% anwar %' AND CUSTOMER.ACCOUNTTYPE <> 'anwar') OR (CUSTOMER.ACCOUNTTYPE LIKE '% ANWAR %' AND CUSTOMER.ACCOUNTTYPE <> 'ANWAR') OR (CUSTOMER.TELEPHONE1 LIKE '% anwar %' AND CUSTOMER.TELEPHONE1 <> 'anwar') OR (CUSTOMER.TELEPHONE1 LIKE '% ANWAR %' AND CUSTOMER.TELEPHONE1 <> 'ANWAR') OR (CUSTOMER.TELEPHONE2 LIKE '% anwar %' AND CUSTOMER.TELEPHONE2 <> 'anwar') OR (CUSTOMER.TELEPHONE2 LIKE '% ANWAR %' AND CUSTOMER.TELEPHONE2 <> 'ANWAR') OR (CUSTOMER.BRANCHCODE LIKE '% anwar %' AND CUSTOMER.BRANCHCODE <> 'anwar') OR (CUSTOMER.BRANCHCODE LIKE '% ANWAR %' AND CUSTOMER.BRANCHCODE <> 'ANWAR') OR (CUSTOMER.INSERTUSERID LIKE '% anwar %' AND CUSTOMER.INSERTUSERID <> 'anwar') OR (CUSTOMER.INSERTUSERID LIKE '% ANWAR %' AND CUSTOMER.INSERTUSERID <> 'ANWAR') OR (CUSTOMER.UPDATEUSERID LIKE '% anwar %' AND CUSTOMER.UPDATEUSERID <> 'anwar') OR (CUSTOMER.UPDATEUSERID LIKE '% ANWAR %' AND

CUSTOMER.UPDATEUSERID <> 'ANWAR') OR (CUSTOMER.UPDATEDATE LIKE '% anwar %' AND CUSTOMER.UPDATEDATE <> 'anwar') OR (CUSTOMER.UPDATEDATE LIKE '% ANWAR %' AND CUSTOMER.UPDATEDATE <> 'ANWAR') OR (CUSTOMER.ENTRYDATE LIKE '% anwar %' AND CUSTOMER.ENTRYDATE <> 'anwar') OR (CUSTOMER.ENTRYDATE LIKE '% ANWAR %' AND CUSTOMER.ENTRYDATE <> 'ANWAR') OR (CUSTOMER.VERSION LIKE '% anwar %' AND CUSTOMER.VERSION <> 'anwar') OR (CUSTOMER.VERSION LIKE '% ANWAR %' AND CUSTOMER.VERSION <> 'ANWAR'))

4: anwar : is the query keyword for this query result.

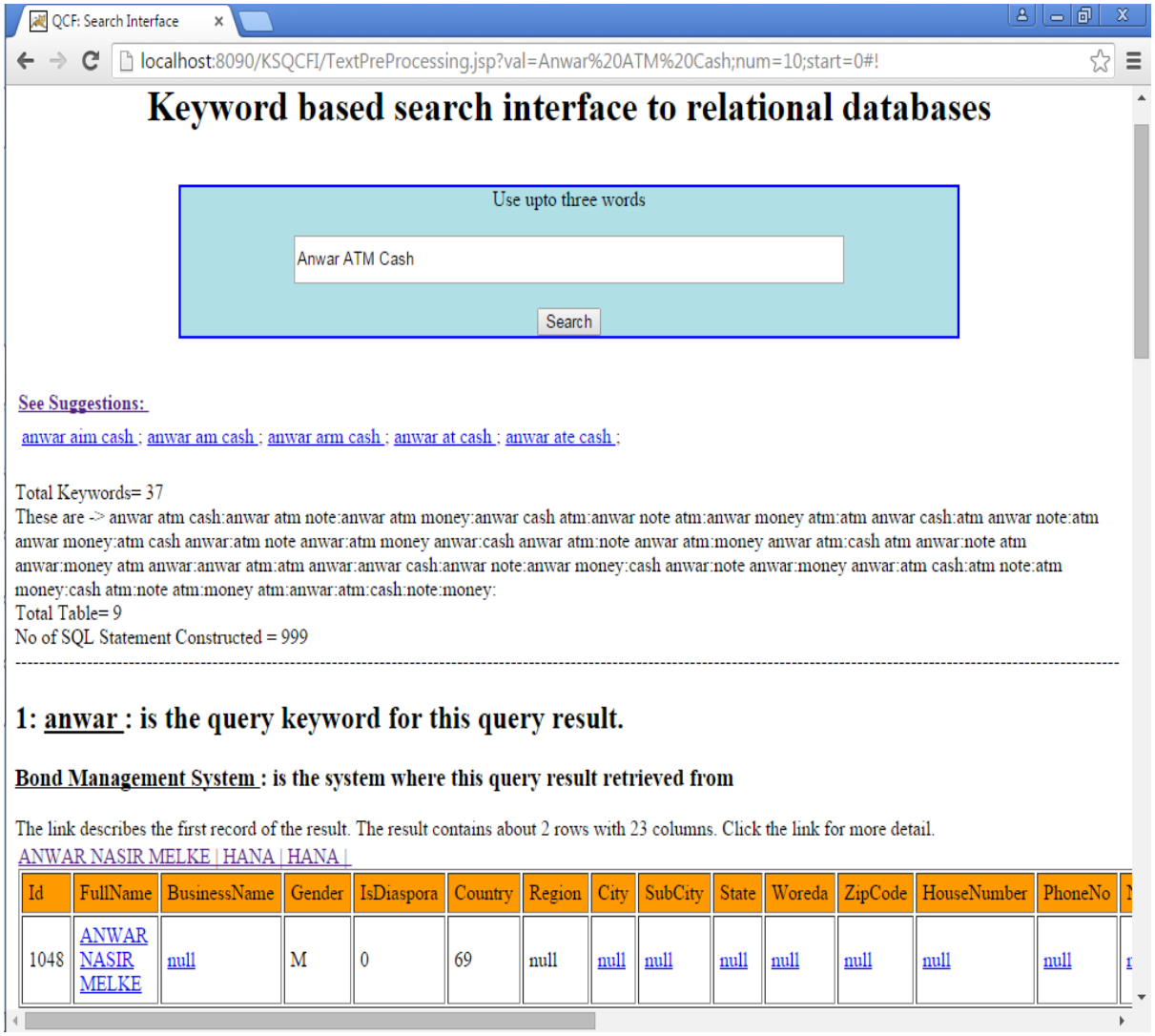
Coupon Management System : is the system where this query result retrieved from

The link describes the first record of the result. The result contains about 17 rows with 15 columns. Click the link for more detail.

1000005379664 | MUKTAR ANWAR ESMAIEL | 6501 | 0202 | admin | admin | 01/06/2015 | 01/06/2015 |

END

Appendix C: Sample User Searching result



Appendix D: User Guidelines for Combined Database Schema Construction

Database Schema constructor in a tool developed using Java programming language that enables to extract information about objects in the databases technologies. The constructor tool has user friendly GUI interface.

Combined Database Schema Construction

In order to construct the database schema the user has to follow the following series of steps.

step_1. Open the Schema constructor. The user interface will be opened as shown in Figure 0-1 below.

step_2. Enter user name and password, Select database technology, and if no username or password is provided a confirmation request dialogue box appears as shown in Figure 0-2. If you agreed confirm and this will connect to the database technology and collects all available databases.

step_3. Select the database you desired; this will connect to specific database, collects all available tables and populate to the interface.

step_4. select the tables to be considered and click on set table button and this will collect available columns in all tables and populate the result to the interface

step_5. If there are columns not to be considered select the column and click on remove button. After removing un-necessary columns click on set column. This will accept the columns, collects table relationship constraints, and make consolidates the total data collected

step_6. Finally, to construct the schema click on construct compound schema button

step_7. For the next database continue from step_3.

step_8. For the next database technology continue from step_2.

Note:

- Database Schema construction steps have to be followed in order described above.
- If operation is successful each checkbox will be selected at every stage. Unless the previous checkbox selected the system do not move to the next step.
- If there is a similar database Schema object previously constructed, the new object will overwrite it.

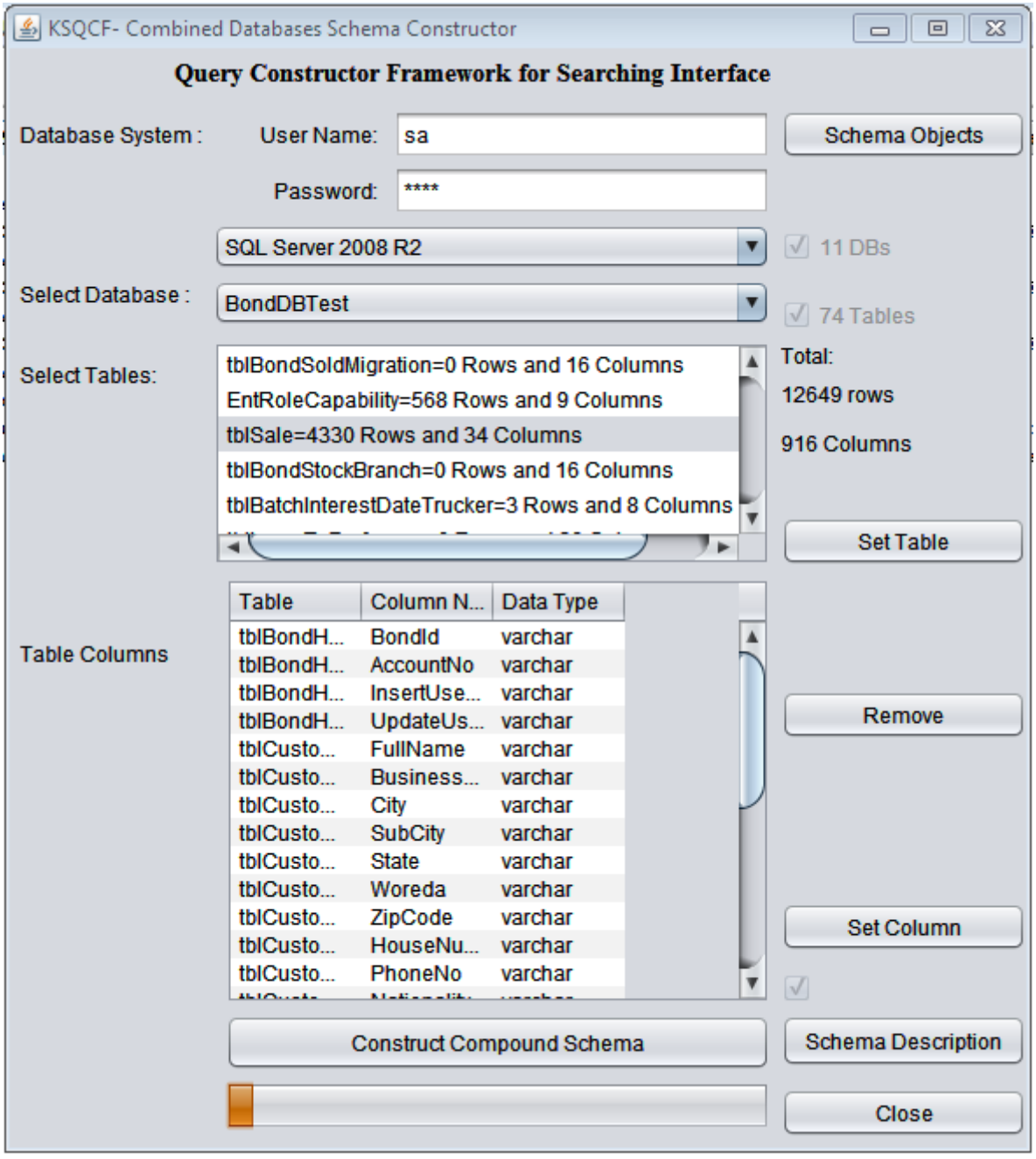


Figure A: Database Shema Constructor interface

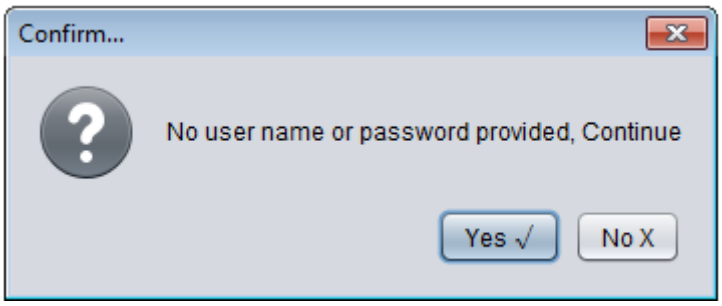


Figure B: Empty Username/password confirmation

Database Schema object Removal

To remove a constructed data model object for no longer use the following steps

step_1. On constructor interface click on model objects; an interface as shown in Figure C will be opened

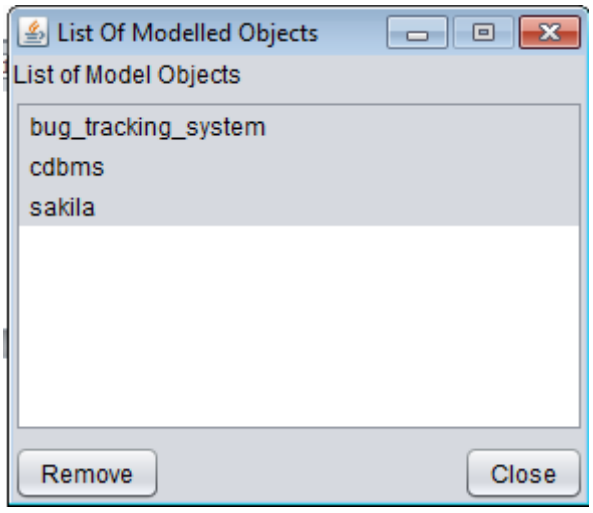


Figure C: Schema object removal interface

step_2. Select the object and click on remove button; a confirmation dialog box will appear.

step_3. Once you confirmed, the model object will no longer exists.

Schema Object Description

Schema object description is used to give explanatory detail of the object in the schema so that the user can consider the result of his search. To set schema object description

Step_1: On constructor interface click on Schema Description; an interface as shown in Figure D will be opened

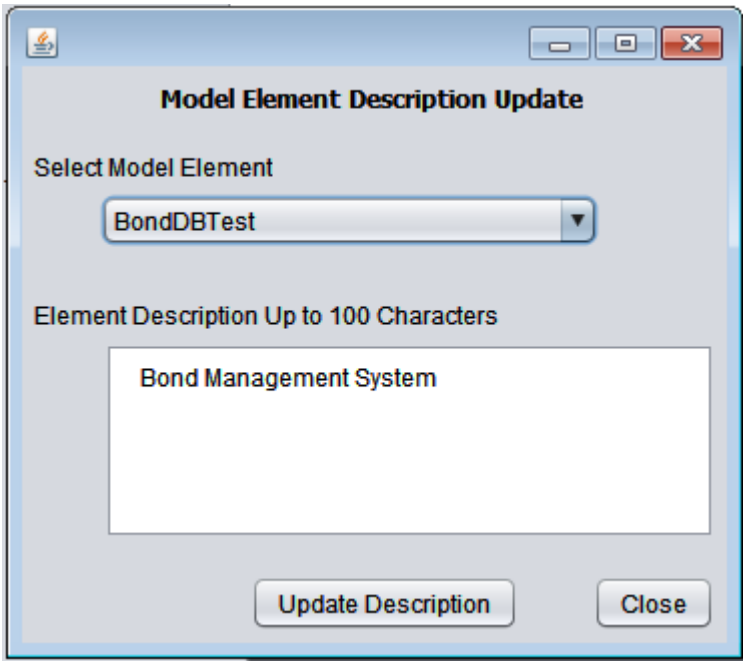


Figure D: Schema object removal interface

Step_2: Select the schema model element interested in, enter your description with less than 100 characters in the description box, and click on Update Description button.

* E N D *

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: **Anwar Indris Jemal**

Signature: _____

Date: _____

Confirmed By Advisor:

Name: **Solomon Atnafu (PhD)**

Signature: _____

Date: _____

Addis Ababa

October, 2015.