Model-Based Architecture for Building Natural Language Interface to Oracle Database

S. Aquter Babu[1] , Dr. C. Lokanatha Reddy[2]
[1] *Assistant Professor, Dept. of Computer Science, Dravidian University, Kuppam, Andhra Pradesh, India*
[2] *Professor, Dept. of Computer Science, Dravidian University, Kuppam, Andhra Pradesh, India*

Abstract:

Structured Query Language (SQL) is a very standard interface since many years to retrieve information from a database. But, to use SQL to formulate query, one should know the syntax of SQL and Database Schema. However, not everybody is able to write such queries in SQL, especially those who lack computer background. To override the complexity of SQL, and to facilitate easy retrieval of information from database, Natural Language Interfaces to Databases (NLIDBs) have been developed. A Natural Language Interface to Database (NLIDB) allows the user to retrieve information from the database by submitting questions in a natural language such as English. Many Natural Language Interface to a Database (NLIDB) systems were developed since 1960's with different architectures such as Pattern-Matching architecture, Syntax-based architecture, Semantic Grammar architecture and Intermediate Representation Languages (IRL) architecture. Developing a NLIDB with one of these architectures requires more effort and time. This has motivated us to propose a novel architecture called Model-based Architecture that can be used to develop a Natural Language Interface to Oracle Database (NLIOD) with less effort and time. This paper presents the Model-based Architecture for NLIOD.

Keywords: *database, database management system (DBMS), Structured Query Language (SQL), natural language interface, architecture, models, extensibility, portability.*

I. INTRODUCTION

Since a long time ago, many organizations are storing their data in the databases to facilitate their users to easily retrieve and manipulate data. Databases contain a collection of related data, stored in a systematic way to model a part of the world. Now a days, databases have become one of the major sources of information. Recently, with the growth of technologies such as computers and laptops, personal digital assistants (PDAs), cellular phones, and the Internet, information can be accessed almost anywhere, at anytime, by anybody, including those who do not necessarily have computer backgrounds.

Access to the information stored in a database has traditionally been achieved by writing quires using formal query languages, such as SQL (Structured Query Language). This formal query language interface forces the users to have cognitive skills that the typical users do not possess. The cognitive load demanded by exiting database query languages (such as SQL) during data retrieval includes the following:

- A good understanding of database schema design.
- A more or less extensive knowledge of database systems, database query language syntax and structure.
- The ability to manually or mentally create message graphs to join tables and to express desired database requests.

It is very difficult for a common person to learn querying using SQL because he/ she may be unaware of the database structure namely tables, their corresponding attributes and data types, primary keys and foreign keys and more. NLIDBs have been developed as a solution to the problem of accessing information from a database in a simple way, allowing any type of users to retrieve information from a database by typing and submitting questions in a natural language such as English.

Many Natural Language Interface to a Database (NLIDB) systems were developed since 1960's such as LUNAR, LADDER etc. with different architectures such as the following.

- Pattern-Matching Architecture
- Syntax-based Architecture
- Semantic Grammar Architecture
- Intermediate Representation Languages (IRL) Architecture

We have developed a Natural Language Interface to Oracle Database (NLIOD) using Model-based architecture. The NLIOD system accepts a natural language sentence such as a sentence written in English language from the user, analyses it using rules and models and builds an SQL query for the Oracle database. The core functionality of the system is based on rules and models. The system administrator will define the rules and models by analyzing the database. This paper presents the Model-based Architecture for NLIOD.

II. BACKGROUND

This section presents Some History of NLIDBs.

A. Late 1960s and Early 1970s

Prototype NLIDBs had already appeared in the late sixties and early seventies. The best known NLIDB of that period is LUNAR[1], a natural language interface to a database containing chemical analyses of moon rocks. LUNAR and other early natural language interfaces were each built having a particular database in mind, and thus could not be easily modified to be used with different databases.

B. Late 1970s

By the late seventies several more NLIDBs had appeared. RENDEZVOUS[2] engaged the user in dialogues to help him/her formulate his/her queries. LADDER[5] could be used with large databases, and it could be configured to interface to different underlying database management systems (DBMSs). LADDER used semantic grammars, a technique that interleaves syntactic and semantic processing. Although semantic grammars helped to implement systems with impressive characteristics, the resulting systems proved difficult to port to different application domains. Indeed, a different grammar had to be developed whenever LADDER was configured for a new application. As researchers started to focus on portable NLIDBs, semantic grammars were gradually abandoned. PLANES[3] and PHILIQA[4] were some of the other NLIDBs that appeared in the late seventies.

C. Mid 1980s

In the mid-eighties NLIDBs were a very popular area of research, and numerous prototype systems were being implemented. A large part of the research of that time was devoted to portability issues. Some Systems that appeared in the mid-eighties were TEAM[7], ASK[8], JANUS[9], DATALOG[10], EUFID[11], and many others.

D. Some other NLIDB systems

1) NALIX

NALIX[13] (Natural Language Interface for an XML Database) is an NLIDB system developed at the University of Michigan, Ann Arbor by Yunyao Li, Huahai Yang, and H. V. Jagadish in 2006. The database used for this system is extensible markup language (XML) database with Schema- Free XQuery as the database query language. Schema-Free XQuery is a query language designed mainly for retrieving information in XML. The idea is to use keyword search for databases. However, pure keyword search certainly cannot be applied. Therefore, some richer query mechanisms are added.

Given a collection of keywords, each keyword has several candidate XML elements to relate. All of these candidates are added to MQF (Meaningful Query Focus), which will automatically find all the relations between these elements. The main advantage of Schema-Free Xquery is that it is not necessary to map a query into the exact database schema, since it will automatically find all the relations given certain keywords.

NALIX can be classified as a syntax based system, since the transformation processes are done in three steps: generating a parse tree, validating the parse tree, and translating the parse tree to an XQuery expression. However, NALIX is different from the general syntax based approaches; in the way the system was built: NALIX implements a reversed-engineering technique by building the system from a query language toward the sentences.

2) PRECISE

PRECISE [12] is a system developed at the University of Washington by Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates in 2004. The target database is in the form of a relational database using SQL as the query language. It introduces the idea of semantically tractable sentences which are sentences that can be translated to a unique semantic interpretation by analyzing some lexicons and semantic constraints.

PRECISE was evaluated on two database domains. The first one is the ATIS domain, which consists of spoken questions about air travel, their written forms, and their correct translations in SQL query language. In ATIS domain, 95.8% of the questions were semantically tractable. Using these questions gives PRECISE 94% precision. The second domain is the GEOQUERY domain. This domain contains information about U.S. Geography. 77.5% of the questions in GEOQUERY are semantically tractable.

The strength of PRECISE is based on the ability to match keywords in a sentence to the corresponding database structures. This process is done in two stages, first by narrowing the possibilities using Maxflow algorithm and second by analyzing the syntactic structure of a sentence. Therefore PRECISE is able to perform impressively in semantically tractable questions.

As other NLIDB systems, PRECISE has its own weaknesses. While it is able to achieve high accuracy in semantically tractable questions, the system compensates for the gain in accuracy at the cost of recall. Another problem is as PRECISE adopts a heuristic based approach, the system suffers from the problem of handling nested structures.

III. NATURAL LANGUAGE INTERFACE TO ORACLE DATABASE USING MODEL-BASED ARCHITECTURE

This section discusses the proposed novel Model-based architecture that has been used to develop a Natural language Interface to Oracle Database (NLIOD).

The NLIOD system, which is developed using Model-based architecture, accepts a natural language sentence such as a sentence written in English language from the user, analyses it using rules and models and builds an SQL query for the Oracle database. The core functionality of the system is based on rules and models. The system administrator will define the rules and models by analyzing the database.

The following figure 1 shows the functions of the NLIOD system.

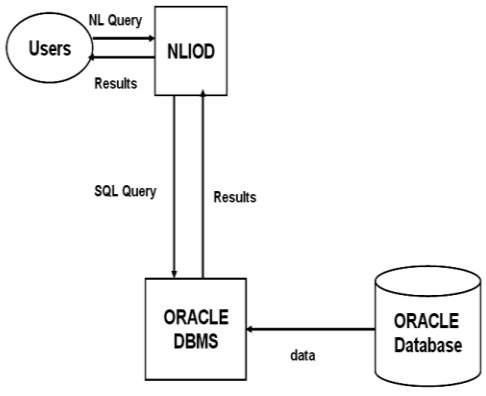


Figure 1: Functions of the NLIOD system

The NLIOD functions as follows.

a) NLIOD accepts Natural Language Query (NL Query) such as English query from the user.

b) NLIOD converts the NL Query to SQL Query and sends the SQL query to ORACLE DBMS Software.

c) ORACLE DBMS software executes the SQL query and retrieves data from the Oracle Database.

d) ORACLE DBMS sends the results of the SQL query to the NLIOD

e) The NLIOD then shows the results to the user.

The following figure 2 shows the Model-based architecture used by the NLIOD system.

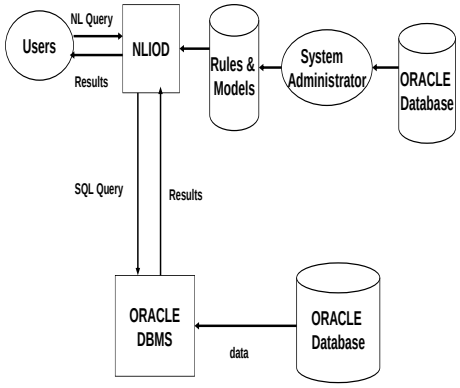


Figure 2: Model-based architecture used by the NLIOD system

In Model-based architecture, the system administrator will define the following by analyzing the data and relationships among the tables in the ORACLE Database.

- Rules related to Database Schema
- Rules related to the action verbs
- Semantic sets
- Default attributes and
- Models

A. Oracle Database

Oracle Database is a database created using Oracle DBMS software. For the purpose of demonstration and implementation of Model-based architecture, We have considered the Oracle Database called Company database consisting of the following tables.

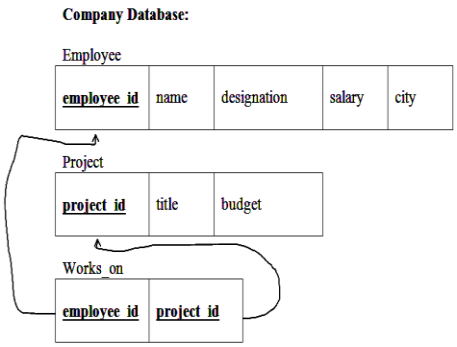


Figure 3: Company Database Schema

We have assumed the following rules in the company.

- One employee can work on several projects.
- One project can have several employees.

This shows that there exists M:N (many-to-many) relationship between employee and project entity types. The relationship type works_on shows which employees are related to which projects.

The primary and foreign keys of the above tables are follows.

Table	Primary Key	Foreign Key(s)
employee	employee_id	--- NIL ---
project	project_id	--- NIL ---
works_on	{employee_id, project_id}	employee_id, project_id

B. Rules Related to Database Schema

These rules show the relationships among the tables in the database. For example, employee table and project table are related in the Company database. These rules are developed by the system administrator by using Primary Key and Foreign Key information in the database schema.

C. Rules Related to action verbs

These rules show how tables in the database are related with action verbs such as "work". For example, employee table and project table are related with the action verb "work".

D. Semantic Sets

The semantic set contains synonyms of a word. For each table name and attribute name, a semantic set is defined by the system administrator.

The following table 1 shows the semantic sets for each table name and attribute name of company database.

Semantic set name	Elements
employee	employee, employees, worker, staff member
employee_id	employee_id, number
name	name, first name, last name, full name
designation	designation, job, job title, job description
salary	salary, pay, income, wage, earnings, money, remuneration, payment
project	project, projects, venture
project_id	project_id, number
title	title, name, description
budget	budget, funds, amount

Table 1 : Semantic sets for each table name and attribute name of company database

E. Default Attributes

The system administrator will define default attributes for each of the tables in the database.

The following Table 2 shows the default attributes for each of the tables in the company database.

Table name	Default attribute
Employee	name
Project	title

Table 2: Default attributes for each of the tables in the company database

The default attribute is used by the system when it is not able to identify the attribute of a table in the given input sentence. This is the case in the question Who works on project 'bank'? NLIOD identifies that the words "works on" are related to the tables employee and project, but it does not identify their attributes. In this situation, the default attribute of employee table i.e., name is used.

F. Models

The NLIOD uses four models to covert the English sentence to SQL query. They are as follows.

- **Model-I:** The {table} model
- **Model-II:** The {attribute}-of-{table} model
- **Model-III:** The {attribute}-of-{table1}-of-{table2} model
- **Model-IV:** The action model [table1]{action_verb}{table2}

For each of the above models, a matching SQL template is defined by the system administrator and is stored in the system configuration.

The following table 3 shows four examples for user sentences that match with each of the above four models.

Model	Example user sentence that matches with the model
Model-I	List all employees in the company
Model-II	What is the salary of employee 'raju'?
Model-III	What is the city of the employee of project 'hotel'?
Model-IV	Which employee works on project 'bank'?

Table 3: Examples for user sentences that match with each of the four models

The NLIOD uses the above rules and models defined by the system administrator to translate the user input sentence into database query language such as SQL. The SQL query is sent to the RDBMS such as Oracle for execution and display of the results.

IV. THE NLIOD IMPLEMENTATION AND EXPERIMENTAL RESULTS

The Natural language Interface to Oracle Database (NLIOD) using the Model-based architecture has been developed for experimental purpose using Visual Basic 6.0 as front-end and Oracle 11g as back-end.

The system administrator will create several tables in the database called system configuration tables to store the information related to rules and models.

For the purpose of demonstration, Model-I is described below.

Model-I i.e., the {table} model has the following two forms of SQL templates.

Form1 : SQL template

```
SELECT * FROM table
```

Form2 : SQL template

```
SELECT * FROM table WHERE table.default_attribute = <value of default attribute>
```

Figure 4: SQL Templates of Model-I : The {table} model

The following are some examples for natural language sentences that are related to form1 of this model.

- List all employee details
- List employees of the company
- Show all employee details
- List project details
- Employee details please
- Employee
- Employees
- Show me the details of employees

following are some examples for natural language sentences that are related to form2 of this model.

- Show employee 'raju'
- List employee 'suresh'
- Employee 'raju'
- Show the details of employee 'raju'

Algorithm for Model-I:

1. **if** only one table name and no attribute names, action verbs and value of default attribute found **then**

begin

1.1) retrieves the related SQL template i.e., form1 SQL template from the system configuration file

1.2) populates the template with table name

end

else

if only one table name and value of default attribute and no attribute names and no action verbs found **then**

begin

1.3) retrieves the related SQL template i.e., form2 SQL template from the system configuration file

1.4) populates the template with table name, default attribute and value of default attribute

end

Algorithm for NLIOD

The following is the algorithm for NLIOD.

1. accepts the natural language sentence such as English sentence from the user
2. divides the sentence into tokens
3. scans the token list and identifies the table names, attribute names, action verbs and values of default attributes using rules
4. **if** no table names or no action verbs or more than two table names found **then**

   begin

   4.1) Write "invalid sentence"
   4.2) Exit

   End

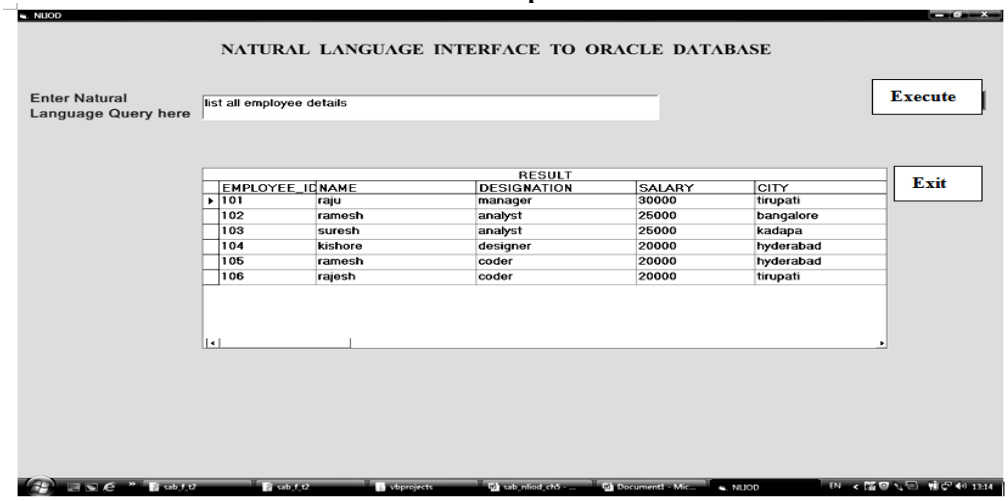
5. if Model-I is satisfied then
   5.1) Execute algorithm for Model-I
   else
   if Model-II is satisfied then
   5.2) Execute algorithm for Model-II
   else
   if model-III is satisfied then
   5.3) Execute algorithm for Model-III
   else
   if Model-IV is satisfied then
   5.4) Execute algorithm for Model-IV
   else
   begin
   5.5) write "Invalid sentence"
   5.6) Exit
   End
6. Sends the SQL query to Oracle RDBMS
7. receives the result set from the Oracle RDBMS
8. displays the result set on the screen

A. Examples

Example 1: *List all employee details*

Result : Pass
Model used : model_1_a
Query generated: select * from employee

Screen Output:



Example 2: *worker details please*
Result : Pass
Model used : model_1_a
Query generated: select * from employee

Example 3: *show all doctor details*
Result : Fail
Note : no table name is identified by the NLIOD system

Example 4: *show employee 'raju'*
Result : Pass

Model used : model_1_b
Query generated: select * from employee where employee.name = 'raju'

Example 5: *list employee 'ramesh'*
Result : Pass
Model used : model_1_b
Query generated: select * from employee where employee.name = 'ramesh'

V. CONCLUSIONS

The Model-based architecture used in the development of NLIOD has the following characteristics.

- The system administrator can quickly develop system configuration tables such as schema rules table, action verb tables, semantic set table, default attribute table etc. by analyzing any database for which natural language interface is required.
- The system administrator can quickly develop a table that contains information about models.
- The Model-based architecture uses the system configuration tables and models table to convert the natural language query such as query in English into SQL query and generates the results.
- The Model-based architecture requires little effort and time to develop any NLIDB such as NLIOD compared to other architectures such as Pattern-matching architecture, Syntax-based architecture etc.
- The Model-based architecture supports knowledge-domain portability. It can easily be configured for use in a wide variety of knowledge domains such as bank database, hotel database etc.
- The Model-based architecture supports DBMS portability. It can be easily modified to be used with different underlying database management systems (DBMSs) such as SQL Server, Sybase etc.
- The Model-based architecture supports extensibility. It can easily be extended by adding more models to support many different types of English sentences.

A. Performance of NLIOD

The NLIOD system performance was quite impressive; it managed to handle 90% of requests without any errors if the input request matches with any one of the four models proposed.

If the input request does not match with any one of the four models proposed, then the system will display message "Invalid Sentence".

It has been observed that Model-based architecture used in the development of NLIOD requires comparatively little effort and time to develop because of the following reasons.

- The system administrator has to create only system configuration tables by analyzing the database for which a natural language interface is required. It requires less effort and time.
- In Pattern-matching architecture, one has to develop a pattern for each attribute and combination of attributes of each table. In Model-based architecture, no need to develop patterns. Many requests in natural language match with one model.
- In Syntax-based architecture, one has to develop complex syntax. In Model-based architecture, no need to develop syntax.
- In Semantic Grammar architecture, one has to develop detailed semantic grammar. In Model-based architecture, no need to develop semantic grammar.
- In Intermediate Representation Language architecture, one has to develop an intermediate language to convert natural language query into SQL query. In Model-based architecture, no need to develop an intermediate language and natural language query is directly converted to SQL query.

B. Limitations of NLIOD

The only limitation of NLIOD is that it can not process the requests that do not match with any of the four models proposed. This limitation can be overcome by adding more models to the NLIOD system.

REFERENCES

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

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

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

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

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

[6] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch, "Natural Language Interfaces to Databases - An Introduction". Natural Language Engineering, 1(1): 29-81, Cambridge University Press, 1995.

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

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

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

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

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

[12] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, Modern Natural Language Interfaces to Databases: Composing Statistical Parsin with Semantic Tractability, COLING (2004)

[13] Yunyao Li, Huahai Yang, and H.V. Jagadish, NALIX: An Interactive Natural Language Interface for querying XML , SIGMOD, 2005