A Scenario-based ER Diagram and Query Generation Engine

Sashini Hettiarachchi
Faculty of Information Technology
University of Moratuwa
Moratuwa, Sri Lanka
dhsashini@gmail.com

Chinthani Sugandhika
Faculty of Information Technology
University of Moratuwa
Moratuwa, Sri Lanka
chinsugandhika@gmail.com

Ashini Kathriarachchi
Faculty of Information Technology
University of Moratuwa
Moratuwa, Sri Lanka
ashiddk95@gmail.com

Supunmali Ahangama
Faculty of Information Technology
University of Moratuwa
Moratuwa, Sri Lanka
supunmali@uom.lk

G. T. Weerasuriya
Faculty of Information Technology
University of Moratuwa
Moratuwa, Sri Lanka
thiliniw@uom.lk

2019 4th International Conference on Information Technology Research (ICITR) | 978-1-7281-6199-0/20/$31.00 ©2019 IEEE | DOI: 10.1109/ICITR49409.2019.9407793

***Abstract*— Designing and developing a database is a crucial task in the System Development Life Cycle (SDLC). To design a database, it is essential to have proper knowledge about drawing Entity-Relationship (ER) Diagrams. Drawing ER diagrams is challenging for novices and people without a technical background. Furthermore, to retrieve data from a database requires expert domain knowledge about a database querying language like Structured Query Language (SQL). To address these issues, a system is proposed to identify and extract the necessary information from a given scenario to automatically generate the ER diagram. Based on that ER diagram, the system creates the database and is capable of generating SQL queries for any given type of natural language queries, in order to simplify accessing the data stored in the database.**

Keywords*— *ER diagram, SQL query generation, Natural Language Processing

I. Introduction

When developing software applications, requirement gathering, analysis and design phases play major roles in SDLC. Natural language scenarios are analysed and transformed into a form of design diagrams under these phases. During the past decade, a number of research has been conducted on automating the extraction of useful information from natural language text scenarios using Natural Language Processing (NLP) Techniques and converting them to design diagrams.

Out of the conceptual design diagrams such as UML diagrams, Entity Relationship (ER) diagram occupies a central role in designing the database of a software system. An ER diagram is a graphical representation that depicts the relationships among people, objects, places or events in a given scenario. It provides the basic foundation for relational databases. To extract entity-relationship models from a given scenario, it requires expert technical knowledge and needs to undergo a lengthy process which is time consuming. On the other hand, the user should have thorough expertise and domain knowledge regarding databases and query languages like SQL to interact with the databases. There are some tools that enable users to draw the design diagrams such as "Lucidchart", and "ERDPlus", but they do not support the identification of the entities, relationships, attributes, relationship types and cardinalities for a given scenario. A complete single system does not exist which could identify the necessary details and automatically generate an ER diagram and generate SQL queries for the natural language queries that follow the same scenario.

The aim of this research is to propose an approach to automate the process of identifying entities, attributes, attribute types, relationships and cardinalities of the relationships and generate an ER diagram for a given scenario and generate the database and generate possible SQL queries for the natural language queries that follow the same scenario. This approach minimizes the user involvement and reduces time consumption and extra burden on analyzing the requirements given in the scenario. Furthermore, non-technical people with or without database and querying knowledge can use this system to automatically generate the database and generate SQL queries.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 focuses on the proposed approach. Subsequently, implementation and evaluation are discussed under Section 4 and Section 5 mentions some of the limitations of the system and Section 6 includes discussion and future work.

II. Related Work

Related work can be reviewed under two sections as given below.

A. Related Work for ER Diagram Generation

ER converter [1] is a heuristic-based tool which transforms a natural language text to an ER model where syntactic heuristics are used for transformation purposes. ER converter system has been tested using 30 different database problems gathered from database books and exam papers. Each problem ranges between a word size of 50-100

978-1-7281-6199-0/18/$31.00 ©2018 IEEE

words and therefore, this system is suitable for small domain scenarios only.

In another model [2] it suggests a structural approach that syntactically parse the specifications based on a predefined set of heuristics. This is much more convenient because it requires minimum human intervention during the process. However, there are some limitations due to the ambiguity of the knowledge base and it is less accurate.

In [3], a model is proposed to draw ER diagrams based on requirement specifications written in English. This proposed model contains three main modules as Pre-processing, Machine Learning (ML) and ER diagram modelling module. To pre-process the text in the requirement specification document, NLP techniques were used. In ML module, a supervised learning approach was used to identify the entities, attributes, relationships and relationship types. The system gives high accuracy for identifying entities, attributes and sub-entities. As further work, they have suggested to use a similar approach to determine the relationships and relationship types [3].

B. Related Work for SQL Query Generation

Kate et al. [4] proposed a system which enables users to enter queries using speech. This system is capable of converting the speech to text and then transforming them to SQL queries. The queries are then executed and output the result.

This research area has a considerable amount of history with a goal of producing a natural language interface for databases to reduce the gap between the human and the computer. There are three main approaches used for developing similar systems.

- Symbolic approach (rule-based approach)
- Empirical approach (corpus-based approach)
- Connectionist approach (neural network-based approach)

Kate et al. [4] have used four types of architectures for those systems as given below.

1) Pattern matching systems: These systems are easy to implement and the main advantage is the simplicity of the system. However, its shallowness may lead to the failures of the system.

2) Syntax based systems The user's input query will be parsed and will be analysed syntactically in this system. The syntactic structure of the question is used to form the relevant database query.

3) Semantic grammar systems: Semantic grammar systems are very like the syntax-based systems as both answer the questions by parsing the input query and then mapping that parsed user query to a database query. Nevertheless, semantic grammar systems require some previous knowledge about the database. Therefore, it is difficult to be used in other domain areas.

4) Intermediate representation languages: These systems convert the user's input query into an intermediate logical query and subsequently, it will be converted to the database query language. It is not only applicable to a specific database and can be used in other domains too.

'SAVVY' is a system which uses rules to convert the natural language query into SQL [5]. There are patterns which are written for several types of queries. When a user enters his/her natural language query to the system, these patterns will be executed. The implementation of this system is very simple. However, this will not be very applicable for every case and may lead to failures because if its shallowness [5].

'LADDER' is another system for providing a natural language interface for a database about US Navy ships. That was also a system which was designed considering a specific scenario in mind. It has been mentioned that 'LADDER' uses semantic grammar to parse the questions [5].

Amit et al. [6] have given a good insight on how to automatically generate SQL queries by understanding a natural language statement. Their main focus was on creating a complex select queries including nested queries with more than two depth levels, queries including aggregate functions and etc. Similarly, they have focused on natural joins and correlated queries. The natural language statement is subjected to various OpenNLP natural language processing techniques. This system is unable to process natural language statements having qualitative quantifiers (e.g. youngest) and also this approach is applicable only for MySQL databases [6].

III. Proposed Approach

The proposed approach consists of two main modules; namely, ER diagram generation module and SQL query generation module. Fig 1 depicts the block diagram of the proposed approach. Both modules perform text pre-processing using NLP techniques. Furthermore, heuristics were developed to identify the main components of the ER diagram as well as to identify the implied SQL query of a given natural language query.

The ER generating module uses existing heuristics from [1][2][7][8] as well as self-developed heuristics which were discovered by studying scenarios in standard database-related books [9][10]. The heuristics of the SQL query generation module has been discovered by studying different ways of asking the same question in natural language along with studying the semantics and structure of the English language. The heuristics findings are given below.

A. Heuristics used to identify the entities

Heuristic 1: A common noun may indicate an entity [9,10]. E.g. An **employee** is working on at least one **project**.

Heuristic 2: When retrieving entities, every proper noun should be ignored. [2] E.g. Location name, Person name

Heuristic 3: If consecutive nouns are present, check the last noun. If it is not one of the words in set S where S= {number, no, code, date, type, volume, birth, id, address, name}, most likely it is an entity. Else it may indicate an attribute [1]. E.g. Each department has a unique **name,** a unique **number**, and a particular employee who manages the department

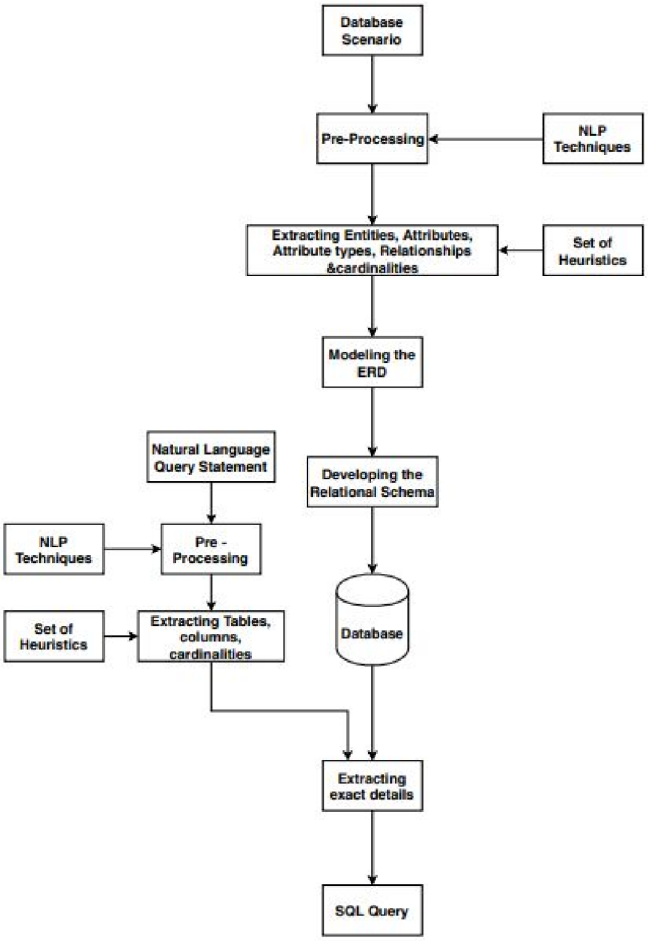

Fig. 1. Proposed Model

Heuristic 4: A noun such as "record", "system", "information" and "organization" may not be a suitable candidate for an entity [2]. E.g. Description of the 'mini world' is given below for an example **database** application called company.

Heuristic 5: Common nouns that are mostly used in text scenario can be an entity [Self-developed]. E.g. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the **department**. A **department** may have several locations. Application requires to get the total no of employees per **department**.

Heuristic 6: The common noun immediately followed by "each" is an entity [Self-developed]. E.g. **Each** song recorded at Notown has a title and an author.

B. Heuristics used to identify the attributes

Heuristic 7: The nouns that are followed by noun phrases such as "has" or "have" represent the attribute types [2]. E.g. Employee **has** an employeeId.

Heuristic 8: The nouns that are followed by the noun phrases "has" or "have" which are preceded with a pronoun, belong to the previously mentioned proper noun/ entity [Self-developed]. E.g. **It** has an id, name and address.

Heuristic 9: In a given text, if a sequence of the word is separated by a comma (",") those words become attributes [Self-developed]. E.g. You need to store the **name, SSN, address, phone number, and salary** of each technician.

Heuristic 10: The immediate two words that follow the same sequence of words be "of the" then the noun that comes after the above mentioned "of the" becomes the parent entity of the given sequence of attributes [Self-developed]. E.g. you have to store a **start date, an end date, and the text of the** contract.

C. Heuristics used to identify the primary key attribute

Heuristic 11: Primary Key of an entity can be identified by the adverb "unique" [2]. E.g. Each department has a **unique name**, a **unique number,** and a particular employee who manages the department.

D. Heuristics used to identify the multi-valued attribute

Heuristic 12: The noun that follows the keyword "several" is not an entity, then it becomes a multi-valued attribute [Self-developed]. E.g. A department may have **several locations**.

E. Heuristics used to identify the composite attribute

Heuristic 13: Assume that there is a noun, and if that noun follows a sequence of nouns given within brackets, then that noun becomes a composite parent attribute and the nouns within the brackets become its children [Self-developed]. E.g. We store employee's **name (First Name, Middle Initial, Last Name),** ...

F. Heuristics used to identify the relationship

Heuristic 14: Relationship may be a verb. The relationship is in the middle of two entities [Self-developed]. E.g. Instructors **teach** from two to four courses.

G. Heuristics used to identify 'one to one' cardinality of the relationships

Heuristic 15: Entities as singular nouns [Self-developed]. E.g. We keep track of the start date when that **employee** began managing the department.

Heuristic 16: The verb is in the singular form [Self-developed]. E.g. Employee who **manages** the department.

Heuristic 17: The word "uniquely" represents the "one" cardinality [Self-developed]. E.g. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug **uniquely** from among the products of that company.

H. Heuristics used to identify 'one to many' cardinality of the relationships

Heuristic 18: Cardinality "one" represent the singular noun and "many" represent the plural nouns [Self-developed]. E.g. The EST **university** has three **faculties**

Heuristic 19: "only one", "one", "exactly one", "uniquely" represent the cardinality one [Self-developed].

E.g. A student can enrol in **only one** degree program at a time.

Heuristic 20: Singular nouns represent the cardinality one and "one or more", "several" represent the cardinality many. E.g. faculty can offer **one or more** degree programs.

Heuristic 21: Even if the verb is in singular form, if phrases like 'number of' come after the verb, the object noun of that sentence become "many" while subject noun becomes "one" [Self-developed]. E.g. A department controls a **number of** projects

Heuristic 22: "no more than" represent the cardinality one [Self-developed]. E.g. Each album has a number of songs on it, but **no** song may appear on **more than** one album.

Heuristic 23: Plural nouns represent the cardinality "many", "two type", "more than one", "one or More", "several", "number of", "at least one", "Multiple" like phrases also represent the cardinality Many [Self-developed].

I. Heuristics used to identify the many to many cardinality of the relationships

Heuristic 24: The first sentence explains the cardinality of one entity and second sentence explain the cardinality of other entity [Self-developed].

Heuristic 25: Plural nouns represent the cardinality Many and "Two types of"," More than one", "One or More" "Several", "Number of", "At least one", "Multiple" etc. phrases also represent the cardinality Many [Self-developed]. E.g. A lecturer can be assigned to **one or more** course units. A course unit can be assigned to **one or more** lecturers. A degree program has **two types of** course units. A course unit can be available in **more than one**-degree program.

J. Heuristics used to extract the required details out of the natural language query and transform them into SQL queries.

Heuristic 1: Table name comes after "of", "from" prepositions [Self-developed]. E.g. Give me the name **of** Employees whose age is 40. Select name and age **from** Employees table whose age greater than 24.

Heuristic 2: Attribute values always be proper nouns or cardinal digits [6] E.g. Give me the name of Employees whose salary greater than **45000** or last_name equals **Perera**.

Heuristic 3: Condition concatenating operator lies between 2 attributes (or attribute values) [Self-developed]. E.g. Give me the name of Employees whose age is 40 and a salary greater than 45000 **or** last_name equals 'Perera'.

Heuristic 4: Following keywords appear in retrieval queries "find", "show", "determine", "give", "choose", "list", "what" [Self-developed].

Heuristic 5: Nouns that come after the above keywords become the attributes that you need to select [Self-developed]. E.g. "**List name, age** and **salary** from Employees. **Give** me the name of Employees whose **age** is 40 and **salary** greater than 45000 or last name equals 'Perera'.

Heuristic 6: If there is a table name and the immediate previous word is "of", "from", the nouns that come before "of", "from" become the attributes to be retrieved [Self-developed]. E.g. List **name, age** and **salary** from Employees

Heuristic 7: Attribute values can be given as adjectives of the table names [Self-developed]. E.g. List name and age of **female** employees who work in **HR** department.

Heuristic 8: When there is an adjective attribute value, the condition associated with that attribute value is concatenated to the other conditions list with 'and' operator [Self-developed]. E.g. List name and age of female employees who work in HR department SQL query: E.g. *SELECT* name*, age FROM employee WHERE gender='female' **and** department='HR';*

Heuristic 9: Following synonyms and phrases can be used to identify the aggregate functions [6].

Sum- total
Max- maximum/highest
Min- minimum/lowest
Count- number of

Heuristic 10: Noun that follows the above keywords becomes the attribute which is subjected to aggregation [Self-developed]. E.g. Give me the name of Employee who has the **minimum salary** in department 5.

IV. Implementation & Evaluation

The pre-processing of the system uses several natural language techniques such as tokenization, POS tagging, lemmatization, etc. The heuristics mentioned in Section 3, were implemented using regular expressions.

The ER diagram drawing module gives an overall accuracy of more than 70% for more than 40 tested scenarios taken from well-known database books [9] [10], past papers, tutorials and other online sources. While [1] has been tested with scenarios containing only 50 to 150 word size, the proposed system can manage a larger number of words than that.

Entity and attributes identification accuracy has increased up to 90% after adding the newly found heuristics. [1][2] have not given any heuristics to find composite and multi-valued attributes, but those are very important components of an ER diagram.

Although the research papers [1] and [2] mentioned some heuristics to find the relationships, this system was implemented by self-developed heuristics and previously available heuristics gave more than 80% accuracy.

The paper [2] presented some heuristics to find cardinalities but not proposed any method to categorize the relationships according to cardinalities (one to one, one to many, many to many). However, the proposed system identifies and categorizes the cardinality of the binary relationships. Cardinality identification and categorizing module gave more than 70% accuracy for the tested data sets.

Furthermore, SQL query module was tested with more than 100 different natural language queries for the following 4 SQL query templates.

1 SELECT <all> FROM <table_name>
E.g. SELECT * FROM employee
2. SELECT <attribute_names> FROM <table_name>
E.g. SELECT first_name, salary FROM employee

3. SELECT <all> FROM <table_name>
WHERE<condition_1><condition_2>...<condition_n>
E.g. SELECT * FROM employee WHERE
salary>25000 and first_name='Kamal'
4. SELECT<attribute_names>FROM<table_name>
WHERE<condition_1><condition_2>...<condition_n>
E.g. SELECT first_name, last_name FROM employee WHERE salary>25000 and first_name='Kamal'

Through this approach, along with the newly found set of heuristics, template type 1 gave 100% accuracy for any given natural language query and template type 2 gave an accuracy of more than 80% and template types 3 and 4 gave an accuracy around 70%.

V. Discussion & Future Work

This paper discusses a heuristic based approach for identifying entities, attributes, attribute types, relationships and relationship types and automatically generate an ER diagram for a given scenario and mapping the natural language queries into SQL queries based on the discovered heuristics.

It is important to note that the scenario should be given in grammatically correct English language. The heuristics are applicable only for grammatically accurate English and other languages are not supported by the system. Further, in the scenarios, the entities should be clearly defined without using pronouns to describe them in the latter sections. Moreover, when identifying attributes, the data types of the attributes were not identified by the system. The user should manually enter the data types of the relevant attributes when developing the database. This can be performed as future work of the system. In addition, the context of the user or the scenario is not identified by the system. The information is extracted based only on the given text.

As explained in this paper, the heuristic based approach has shown considerable accuracy and by finding more heuristics, the accuracy of this approach can be further increased. As further work, this research can be extended to identify the weak entities, degree of the relationship and to generate Enhanced ER diagrams. In the SQL query module, table joins and nested queries can also be implemented.

Through this system an ER diagram can be drawn automatically as per a given scenario, as such, a novice user without technical knowhow can use the system to generate proper ER diagrams. Moreover, non-technical persons can do data retrieval easily by using questions written in natural language rather than learning SQL. Thus, the proposed system will be useful when developing and manipulating databases.

References

[1] N. Omar, J.R.P. Hanna, and P. McKevitt, "Heuristics-based entity-relationship modeling through natural language processing," *Proc. of the 15th Artificial Intelligence and Cognitive Science Conference (AICS-04)*. 2004.

[2] E. S. Btoush and M. M. Hammad, *"Generating ER Diagrams from Requirement Specifications Based on Natural Language Processing,"* International Journal of Database Theory and Application, vol. 8, pp. 61-70, 2015.

[3] P. G. T. H. Kashmira and S. Sumathipala, "*Generating Entity Relationship Diagram from Requirement Specification based on NLP*," 2018 3rd International Conference on Information Technology Research (ICITR), 2018.

[4] A. Kate, S. Kamble, A. Bodkhe, and M. Joshi, "*Conversion of Natural Language Query to SQL Query,*" 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018.

[5] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "*Natural language interfaces to databases – an introduction*," Natural Language Engineering, vol. 1, no. 1, pp. 29–81, 1995.

[6] A. Pagrut, I. Pakmode, S. Kariya, V. Kamble, and Y. Haribhakta, "*Automated SQL Query Generator by Understanding a Natural Language Statemen*t," International Journal on Natural Language Computing, vol. 7, no. 3, pp. 01–11, 2018.

[7] Chen, P.P.: "English Sentence Structure and Entity-Relationship Diagram, Information Sciences", Vol.1, No. 1, Elsevier, 1983, pp. 127-149

[8] Tjoa, A.M and Berger, L.: *"Transformations of Requirements Specifications Expressed in Natural Language into an EER Model*". Proceeding of the 12th International Conference on Approach, Airlington, Texas, USA, 1999, pp. 206-217.

[9] R. Elmasri and S. .. B. Navathe, *"Fundamentals of Database Systems (5th Edition),"* Boston, MA,USA, Addison-Wesley Longman Publishing Co., Inc., 2006, pp. 307-308.

[10] R. Ramakrishnan and J. Gehrke, *"Database Management Systems"*, McGraw-Hill College, 2000.