Asian Journal of Research in Computer Science

Asian Journal of Research in Computer Science

7(3): 34-48, 2021; Article no.AJRCOS.66504
ISSN: 2581-8260

Automatic Generation of Scripts for Database Creation from Scenario Descriptions

L. W. Amarasinghe[1] and R. D. Nawarathna[1*]

[1]Department of Statistics and Computer Science, Faculty of Science, University of Peradeniya, Sri Lanka.

Authors' contributions

This work was carried out in collaboration between both authors. Author LWA performed the literature review, developed the methods and performed the implementations and experiments. Author RDN fine-tuned and verified the theoretical formalism of the proposed method and supervised the findings of this work and the writing of the manuscript. Both authors discussed the results, provided critical feedback, and contributed to the preparation of the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v7i330180
Editor(s):
(1) Dr. R. Gayathri, Anna University, India.
Reviewers:
(1) Matheel Emaduldeen Abdulmunim, University of technology, Iraq.
(2) Gajendra Sharma, Kathmandu University, Nepal.
Complete Peer review History: http://www.sdiarticle4.com/review-history/66504

Original Research Article

Received 12 January 2021
Accepted 20 March 2021
Published 27 March 2021

ABSTRACT

Aims: Database creation is the most critical component of the design and implementation of any software application. Generally, the process of creating the database from the requirement specification of a software application is believed to be extremely hard. This study presents a method to automatically generate database scripts from a given scenario description of the requirement specification.
Study Design: The method is developed based on a set of natural language processing (NLP) techniques and a few algorithms. Standard database scenario descriptions presented in popular textbooks on Database Design are used for the validation of the method.
Place and Duration of Study: Department of Statistics and Computer Science, Faculty of Science, University of Peradeniya, Sri Lanka, Between December 2019 to December 2020.
Methodology: The description of the problem scenario is processed using NLP operations such as tokenization, complex word handling, basic group handling, complex phrase handling, structure merging, and template construction to extract the necessary information required for the entity relational model. New algorithms are proposed to automatically convert the entity relational model

**Corresponding author: E-mail: ruwand@pdn.ac.lk;*

to the logical schema and finally to the database script. The system can generate scripts for relational databases (RDB), object relational databases (ORDB) and Not Only SQL (NoSQL) databases. The proposed method is integrated into a web application where the users can type the scenario in natural or free text. The user can select the type of database (i.e., one of RDB, ORDB, NoSQL) considered in their system and accordingly the application generates the SQL scripts.
Results: The proposed method was evaluated using 10 scenario descriptions connected to 10 different domains such as company, university, airport, etc. for all three types of databases. The method performed with impressive accuracies of 82.5%, 84.0% and 83.5% for RDB, ORDB and NoSQL scripts, respectively.
Conclusion: This study is mainly focused on the automatic generation of SQL scripts from scenario descriptions of the requirement specification of a software system. Overall, the developed method helps to speed up the database development process. Further, the developed web application provides a learning environment for people who are novices in database technology.

Keywords: Databases; requirement specification; scenario description; natural language processing; database script; logical schema.

1. INTRODUCTION

Database management systems (DBMS) is a collection of programs that enables users to create and maintain databases. DBMS is therefore a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. Database technology is making a major impact on the growing use of computers and it is fair to say that databases play a critical role in almost all fields in modern digital society. A database may be created and maintained manually or the whole process can be automated.

Generally, the requirements of a system are documented as a collection of scenarios. A scenario is a textual description of specific use cases of a system. End users can freely state their requirements through scenarios. When a system is being developed, a database that captures all the key aspects of the user scenarios should exist to store data effectively. Therefore, mapping of a user-written scenario to a database script directly helps to achieve the functional requirements of the system. It has always been a tedious task to follow the entire set of steps manually in creating a good database. Converting a scenario given in natural language into an SQL script would be extremely beneficial for the database designers. Further, it will help to minimize human errors involved with designing enterprise applications. Overall, the productivity of the system design can be increased by automating the database creation process.

As a solution to the above-mentioned issue, a method is developed in this study to convert a statement of a scenario of the expected system written in natural language to a database script that can be executed to create the database. Natural language processing (NLP) is one of the main areas of computer science and artificial intelligence. NLP includes read, decipher, understand, and make sense of the languages that humans use naturally to interface with computers in both written and spoken contexts. In the proposed solution, first, the user must enter the description of the scenario in text format as the input. The description must be in the English language and the user can describe the scenario according to their understanding of the system. The solution supports three main database types, specifically, relational databases (RDBs), object-relational databases (ORDBs) and NoSQL. Therefore, the output will be an SQL script that depends on the database type selected by the user. This solution is implemented as a web application. One of the advantages of this web application is that it can be used as a tool for teaching and learning of the database creation process.

Therefore, the objectives of this work are to propose a novel algorithm for the extraction of key elements and their relationships of a given scenario description to create the entity relational model, to develop algorithms to automate the conversion of entity relation model to a logical schema and then to SQL scripts and finally to develop a web application integrating all algorithms for automating the database script generation process.

According to the literature, studies in this area either aimed at developing methods to convert requirement specifications to entity-relationship (ER) diagrams or unified modeling language (UML) diagrams such as class diagrams. Some of the existing work falling under these two categories are summarized below.

A tool called “Database designer for MySQL” that allows building effective and clear database structure visually is described in [1]. It is mentioned that this tool supports triggers, stored procedures, reverse engineering of MySQL databases. The tool is compatible with all types of MySQL databases such as MyISAM, ISAM, InnoDB, and BDB. R. Dedhia et al. proposed an algorithm to generate entity-relationship diagrams automatically in [2]. The proposed algorithm focuses on both syntactic analysis and semantic heuristic for extracting the major components such as entity, attribute, and the relation of an entity-relationship diagram.

Also, in [3], an XML-based ER-diagram drawing and translation tool was proposed by Y. Li, S. Lu and S. Xu. The tool automatically generates a relational database schema by converting the ER diagram. Further, in [4], E. S. Btoush and M. M. Hammad proposed a method to generate ER diagrams from requirement specification based on natural language processing techniques. This approach provides an opening of using natural language documents as a source of knowledge for generating ER data models. One of the structural approaches is used to parse specification syntactically based on a predefined set of heuristics rules. M. Uma at el. in [5] attempted to develop a method to convert a database query given in natural language to SQL format using NLP techniques.

When considering the work on automating object-oriented analysis and design process, L. Mich [6] proposed a system called NL-OOPS. It converts from natural language to object-oriented requirements using a system called LOLITA (Large-scale Object-based Language Interactor, translator and Analyzer). It analyses the user requirement provided in natural language and extracts only objects from the description. However, LOLITA cannot identify attributes and classes. A method for automating the object-oriented Analysis (OOA) process using natural language processing techniques and linguistic theories was proposed by K. Li, R. G. Dewar and R. J. Pooley in [7]. They have proposed a pragmatic intermediate solution for the identification of classes through the development of a dialogue algorithm. This method includes a more comprehensive set of heuristics to automate the OOA process.

P. More and R. Phalnikar designed a model to generate UML diagrams from natural language specifications [8]. This study facilitated the requirements analysis process and creates UML diagrams from textual specifications using natural language processing (NLP) and domain ontology techniques. In [9], a system is designed to convert user requirements to a UML class diagram by H. Herchi and W. B. Abdessalem. They investigated how NLP techniques and domain ontologies can be exploited to support the object-oriented analysis process. This system in [9], first, accepts user needs in natural language and then identifies the classes, their attributes and associations between them to include them in a structured XML file.

Structural annotations for a large-scale database were designed by Jordan B. L. Smith et al. described in [10]. This work describes the design and creation of an unprecedentedly large database of over 2400 structural annotations of nearly 1400 musical recordings. Here structure refers to the partitioning of a piece of music into sections and the grouping together of similar or repeated sections. F.A.C. Fokkema et al. proposed a method to create a variation database using the "LSDB-in-a-Box" approach in [11]. They have developed the Leiden Open Variation Database (LOVD) software using the "LSDB-in-a-Box" idea for the easy creation and maintenance of a fully web-based gene sequence variation database.

A tool called “EnviroTox” is developed to generate a curated aquatic toxicology database to support ecoTTC analysis and development described in [12]. For this creation of an ecoTTC tool, a large, diverse environmental data set was developed from multiple sources, with harmonization, characterization, and information quality assessment steps to ensure that the information could be effectively organized and mined.

Overall, from the existing work outlined above, it can be observed that there is no existing system that can automatically generate RDB, ORDB and NoSQL scripts directly from the scenario descriptions of the requirement specification covering each step of the database creation process. Therefore, the major contribution of this study is to propose a comprehensive solution to convert a given scenario description to a database script in one of RDB, ORDB or NoSQL database types.

The remainder of this article is organized as follows. Section 2 presents the methodology of the proposed system. Results of the method evaluation are discussed in Section 3. Finally, Section 4 provides some concluding remarks.

2. METHODOLOGY

The proposed script generation model is developed based on information extraction using natural language processing techniques and a few newly proposed algorithms. Information extraction is the process of identification of the occurrences of a particular class of objects/concepts and relationships among them of a given text [13,14]. This study attempts to extract the required information (i.e., entities, attributes and relationships) from scenario descriptions to create tables of the database. The main steps of the proposed method are shown in Fig. 1. The details of each step of the methodology are presented in the following subsections.

2.1 Processing of the Scenario Description Using Natural Language Processing Techniques

A database scenario description is a story or storyline of use cases of a software system usually written using a natural language. Examples for such scenarios can be found from the standard database textbooks such as [15]. The scenario descriptions available in textbooks are standard, well-written, and widely used for teaching and learning purposes. Therefore, the proposed method is developed based on standard scenario descriptions found in database textbooks. As shown in Fig. 1. first, the scenario description is extracted from the requirements specification of the system. The user can enter any type of scenario as shown in Fig. 2. and select the type of database among RDB, ORDB and NoSQL.

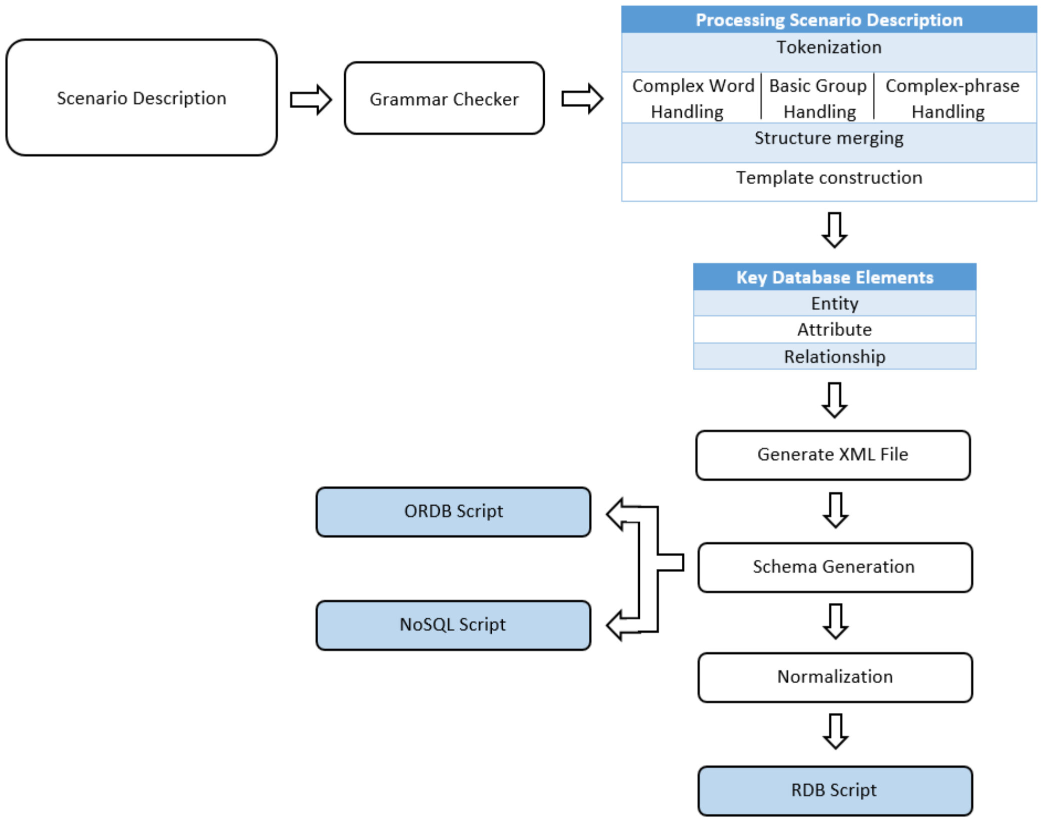
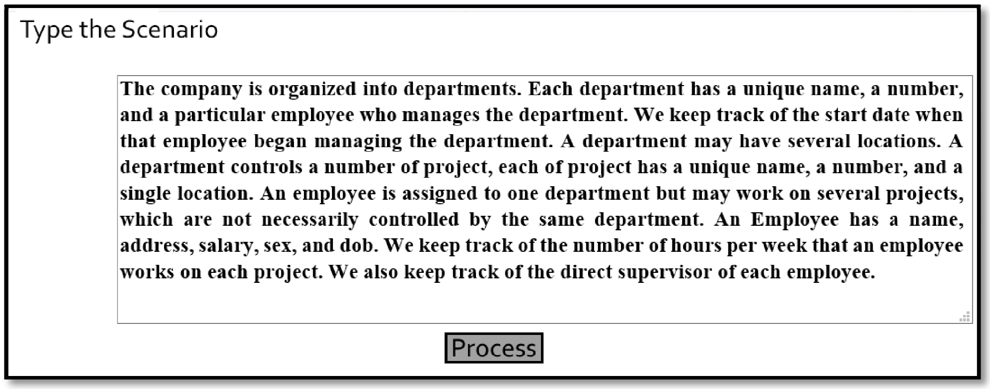


Fig. 1. Overview of the Proposed Model



The company is organized into departments. Each department has a unique name, a 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. A department controls a number of project, each of project has a unique name, a number, and a single location. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. An Employee has a name, address, salary, sex, and dob. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.

Fig. 2. A sample scenario description

As the first step, the grammar checking tool identifies the grammar errors and if there are no grammar errors, then the scenario descriptions go through several NLP operations as indicated in Fig. 1.

During tokenization, sentences of the scenario are segmented into tokens, i.e., words, numbers, punctuations. Then, the complex word handling module handles complex words and they are recognized with a combination of lexical entries and finite-state grammar rules. For example, this step identifies “Co” or “Inc” or “Ltd” as “company”. In the basic group handling module, the system handles basic groups, meaning noun groups and verb groups. Prepositions and conjunctions are also considered.

The main aim of the complex-phrase handling module is to combine the basic groups into complex phrases. Again, the aim of this step is to have finite-state rules and thus can be processed quickly and that results in unambiguous output phrases. Structure merging model merges structures that are obtained in the previous step. Template construction is based on eight general syntactic templates that cover most of the relationships among the words. The eight templates include a verb, noun preposition, verb preposition, infinitive, modifier, noun coordinate, verb coordinate and appositive.

2.2 Key Database Elements Extraction

The system can identify which words can be attributes, entities and relationships from the processed scenario description is described in Section 2.1. To build a fully automated database script generation system, entities, attributes and relationships are identified for the given scenario by using a rule checker. The rule checker includes the following rules. Attributes are nouns mentioned along with their entity. If there are two nouns together in the scenario, the second noun is considered as an attribute. If there are any personal pronouns and nouns appearing together, then the noun is considered as an attribute. If there are any adjective or determiner and noun are occurring together, then again a noun can be considered as an attribute. Adverbs uniquely indicate the primary key of an entity. Multivalued attributes can be identified using plural nouns. Derived attributes can be extracted using a popular attribute list. According to the user-given scenario, the system identifies entities using nouns and verbs.

2.3 An XML Script to Represent the Key Elements

An XML script is considered because XML is the intermediate language that can easily convert extracted key elements to entity-relational model through a web environment. To develop this, the python language and element tree API which has a tree structure are used. There are two sections in the XML file, one for entities and their attributes, and the other one for entities and their relationship. Basically, this XML file contains the entity-relational (ER) model of the system. Next, it will be parsed to create the logical schema.

2.4 Generating the Logical Schema

In this step, the method maps the entity-relational (ER) model to the logical schema using the common ER to Relational mapping algorithm. As shown in Fig. 3. first, the cardinality details should be extracted from the preprocessed scenario. If there is a one-to-one relationship, the primary key of “one-side” goes to the other “one-side” as a foreign key as well as if there are one-to-many relationships between two entities, “one-side” primary key goes to the “many-side” as a foreign key. Further, if the cardinality is many-to-many then, a new table is created from two tables including the primary keys of both tables.

2.5 Generating Database Scripts

To create three types of scripts, the schema file should contain all the necessary details in a well-defined order. The syntaxes for each script are defined first and then relevant entities, attributes and relationships are extracted from the schema file to put them in the correct place as tables.

2.5.1 Relational databases (RDB) scripts

In the schema refinement phase, the logical schema is converted to the normalized schema using 1NF, 2NF, and 3NF. INF and 2NF can be automated according to the given requirement specification. But 3NF depends on the functional dependencies between each attribute of the entity. For the RDB scripts, the output script is based on the normalized schema. Fig. 4. illustrates the system algorithm used to generate RDB scripts step by step.

2.5.1.1 Normalization algorithm for RDB script

We denote the set of functional dependencies by F that are specified on relation schema R. It is considered to identify candidate keys by calculating the closure of attribute sets. Even it is not likely to find all the candidate keys, it is possible to identify a fine set of candidate keys that is needed to do the normalization up to a better extent. The normalization algorithm and the 3NF algorithm is shown in figure 5. In the normalization algorithm, it goes to the 3NF algorithm if there is only one attribute for the primary key. If not, then it checks the dependencies and creates tables for each dependency. In the 3NF algorithm, determined non-prime attributes and remove the depending attributes from the original table are added to a new table.

2.5.2 Object-Relational Database (ORDB) scripts

To create ORDB scripts, entities are identified as objects and all the relevant attributes for each object are also determined. Figure 6 demonstrates the algorithm for ORDB script generation. After creating a type of object, a new table is created. Primary keys and foreign keys are added as constraints for this script. Both key constraints are split from the list in the logical schema by identifying them as “P” and “F” respectively.

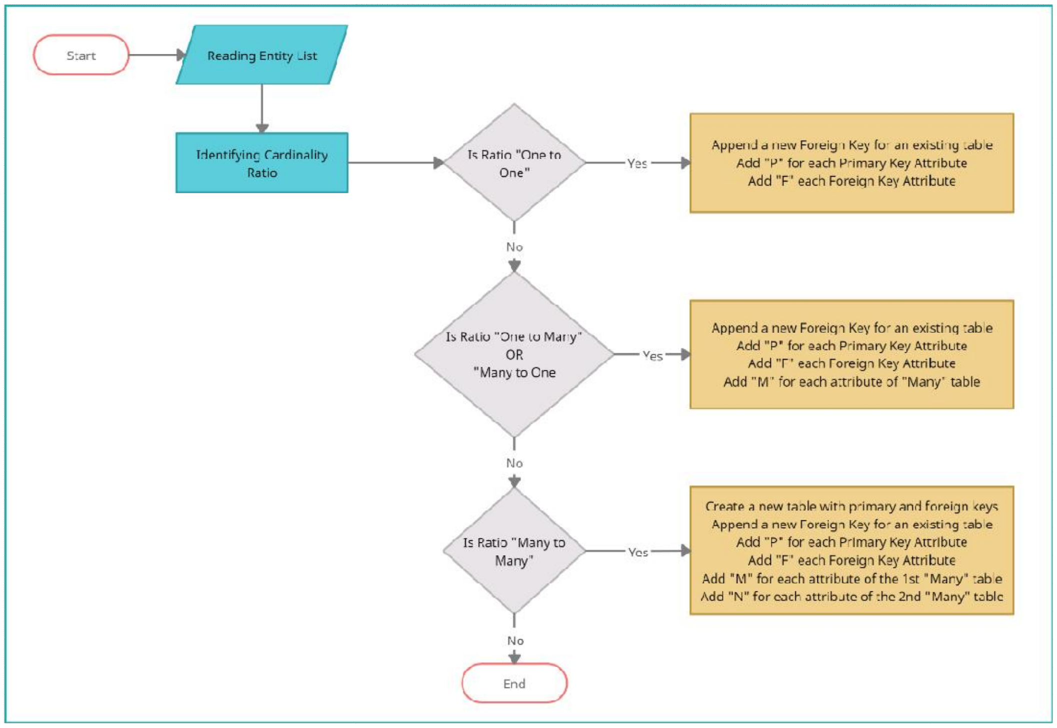

Start
Reading Entity List
Identifying Cardinality Ratio
Is Ratio "One to One"
Yes
Append a new Foreign Key for an existing table
Add "P" for each Primary Key Attribute
Add "F" each Foreign Key Attribute
No
Is Ratio "One to Many" OR "Many to One
Yes
Append a new Foreign Key for an existing table
Add "P" for each Primary Key Attribute
Add "F" each Foreign Key Attribute
Add "M" for each attribute of "Many" table
No
Is Ratio "Many to Many"
Yes
Create a new table with primary and foreign keys
Append a new Foreign Key for an existing table
Add "P" for each Primary Key Attribute
Add "F" each Foreign Key Attribute
Add "M" for each attribute of the 1st "Many" table
Add "N" for each attribute of the 2nd "Many" table
No
End

Fig. 3. The algorithm used for the generation of the logical schema

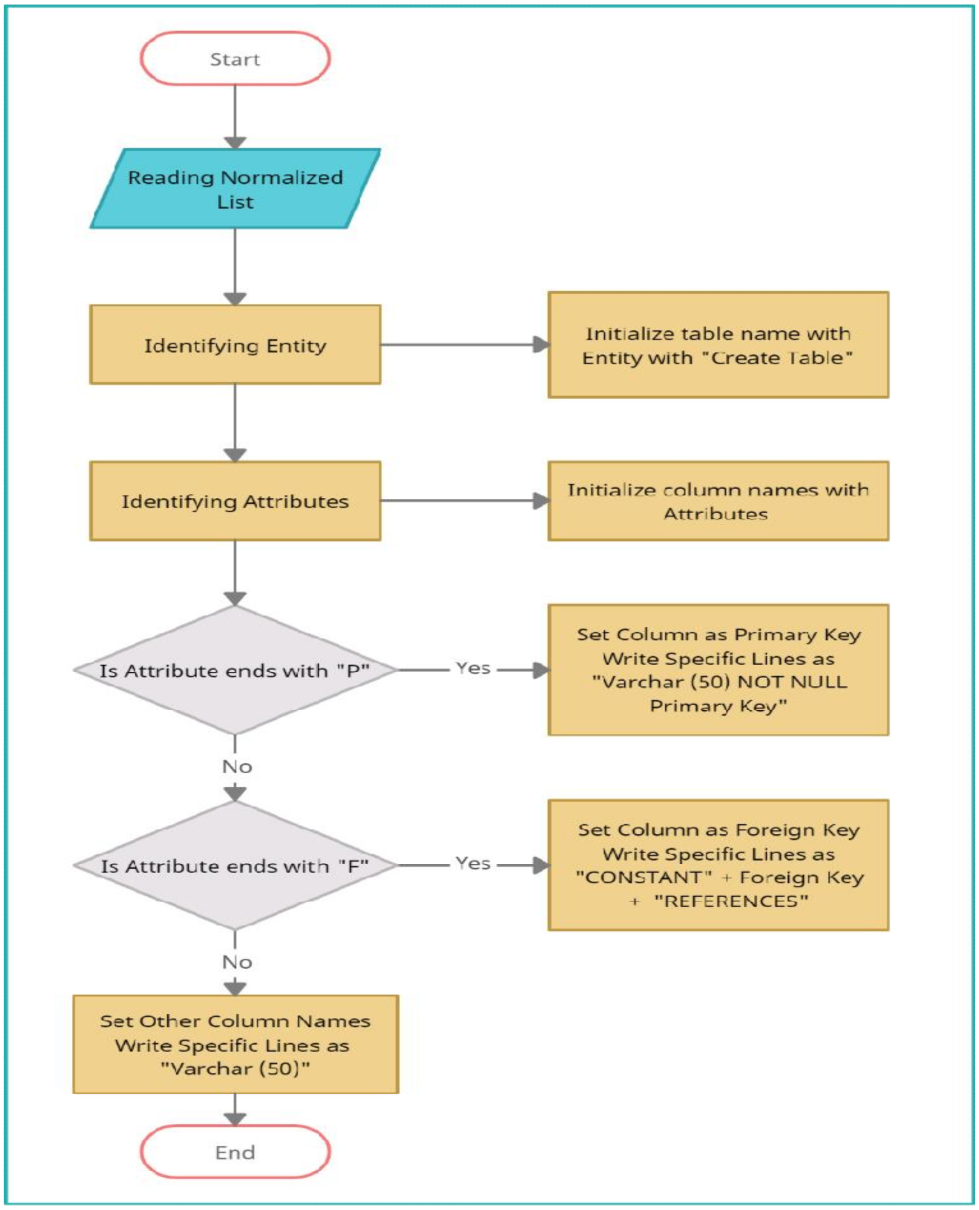

Start
Reading Normalized List
Identifying Entity
Initialize table name with Entity with "Create Table"
Identifying Attributes
Initialize column names with Attributes
Is Attribute ends with "P"
Yes
Set Column as Primary Key Write Specific Lines as "Varchar (50) NOT NULL Primary Key"
No
Is Attribute ends with "F"
Yes
Set Column as Foreign Key Write Specific Lines as "CONSTANT" + Foreign Key + "REFERENCES"
No
Set Other Column Names Write Specific Lines as "Varchar (50)"
End

Fig. 4. The algorithm used for the generation of RDB Scripts

2.5.3 NoSQL Script

The algorithm for NoSQL script generation considers the required NoSQL syntaxes and apply them to the logical schema. Fig. 7. shows the algorithm for the NoSQL script. Keywords “db” and “insert” are used here for new table creation. Entities are used as table names and the attributes are used as column names of the NoSQL table. Normalization algorithm is not used for NoSQL scripts as well as ORDB scripts.

2.6 Primary Key Generation

In the proposed method, the primary key is generated by using functional dependencies. We need to define matrices to identify the primary key. A primary key is a set of attributes that is completely dependent on all other attributes. Dependency matrix, dependency graph matrix, path matrix and circular dependency matrix are the matrices used to create the primary key automatically. The circular dependency matrix shows the primary key. For example, relation Employee {number, name, dob, enroll, date, address, works, supervisor} with functional dependencies ({number, name - dob, date, address, works, supervisor}, {number - enroll}, {address - works}, {name, address - supervisor}, {name, dob, supervisor – number, enroll, date, address, works}, {name, dob, address–number, enroll, date} gives circular dependency matrix as shown in Chart 1. For this example, {Number, Name} is identified as a primary key in which all simple attributes are directly dependent on the attribute set {Number, Name} or depend on the determinant key which not a candidate key. For {Name, DOB, Supervisor} and {Name, DOB, Address} which are potential candidate keys, have some dependencies of simple keys via {Number, Name} which is a potential primary key. For the attribute {DOB}, it depends on {Name, DOB, Supervisor} via attribute set {Number, Name}. For potential candidate keys that have dependencies with another potential candidate keys are ignored as primary keys.

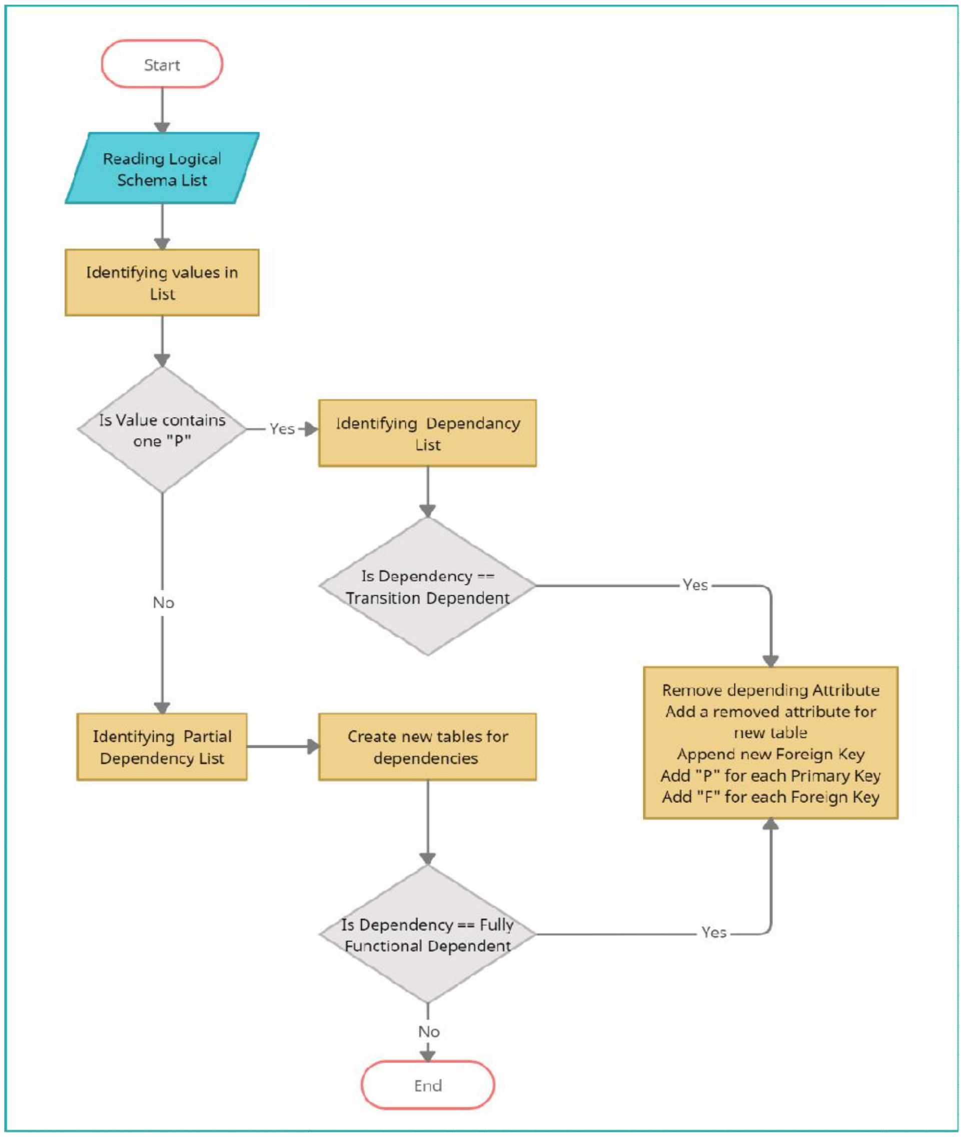

Start
Reading Logical Schema List
Identifying values in List
Is Value contains one "P"
Yes
Identifying Dependancy List
Is Dependency == Transition Dependent
Yes
No
Remove depending Attribute
Add a removed attribute for new table
Append new Foreign Key
Add "P" for each Primary Key
Add "F" for each Foreign Key
Identifying Partial Dependency List
Create new tables for dependencies
Is Dependency == Fully Functional Dependent
Yes
No
End

Fig. 5. The algorithm for normalization

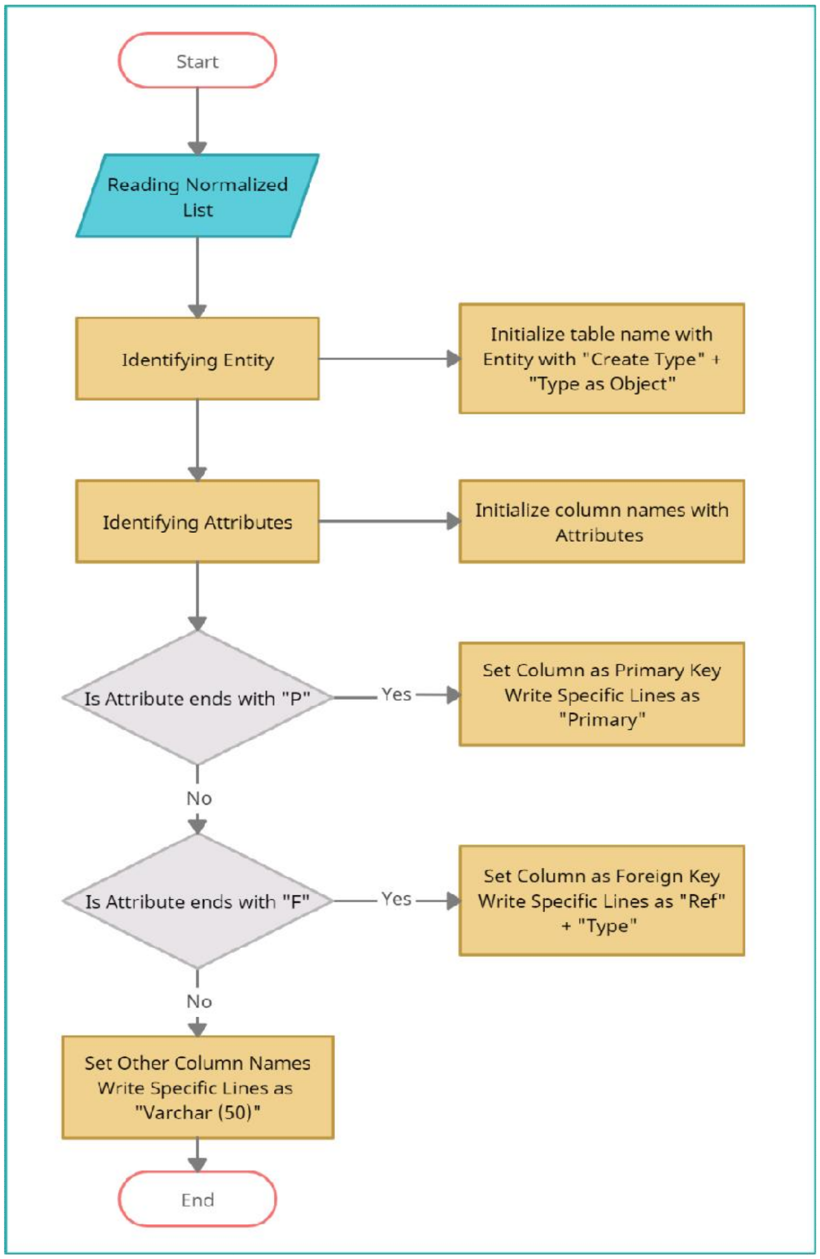

Start
Reading Normalized List
Identifying Entity
Initialize table name with Entity with "Create Type" + "Type as Object"
Identifying Attributes
Initialize column names with Attributes
Is Attribute ends with "P"
Yes
Set Column as Primary Key Write Specific Lines as "Primary"
No
Is Attribute ends with "F"
Yes
Set Column as Foreign Key Write Specific Lines as "Ref" + "Type"
No
Set Other Column Names Write Specific Lines as "Varchar (50)"
End

Fig. 6. The algorithm used for the generation of ORDB scripts

2.7 Web Application

The proposed method was integrated into a web application (see Fig. 8). where the user can enter the scenario for the database and select the type of script to be generated.

All the steps described in Subsections 2.1 to 2.8 are implemented in the web application. It provides three functions to generate RDB, ORDB and NoSQL scripts. According to the literature, still, there is no system that has these functionalities proposed in our method integrated into a single system. Python scripts are combined with PHP scripts to generate web application. It provides extremely simple user interfaces for the users to interact with the system.

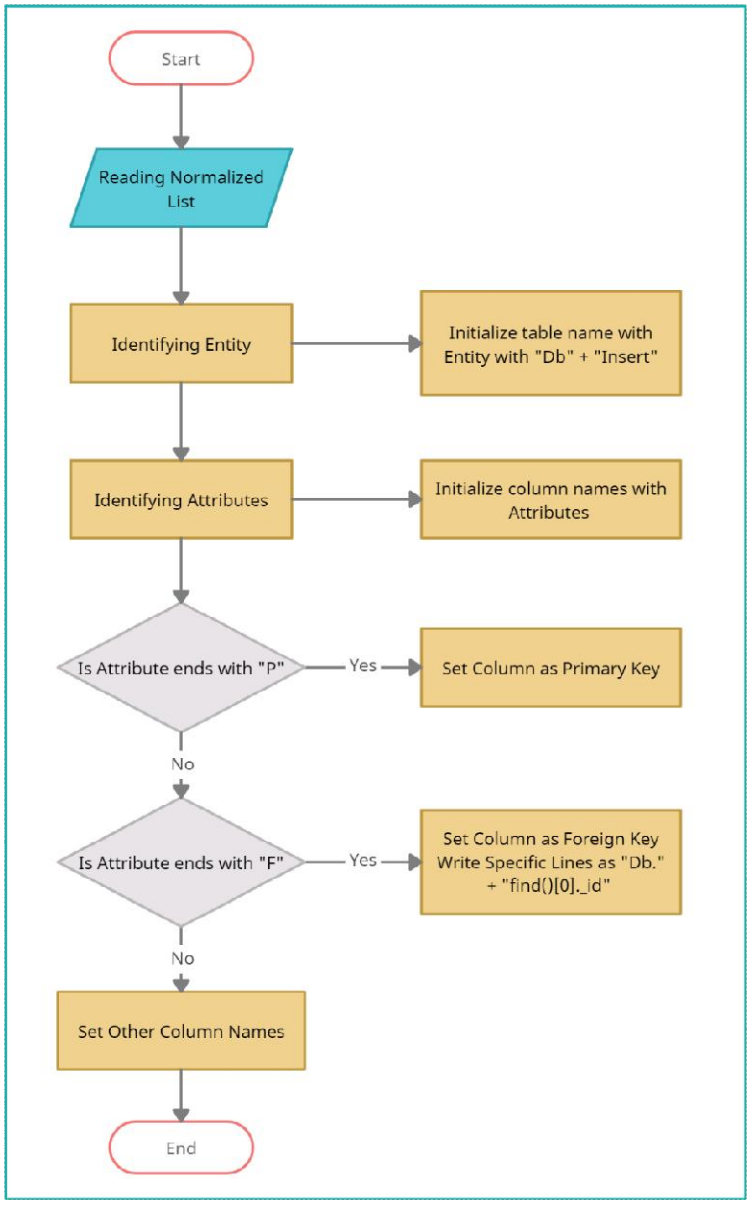
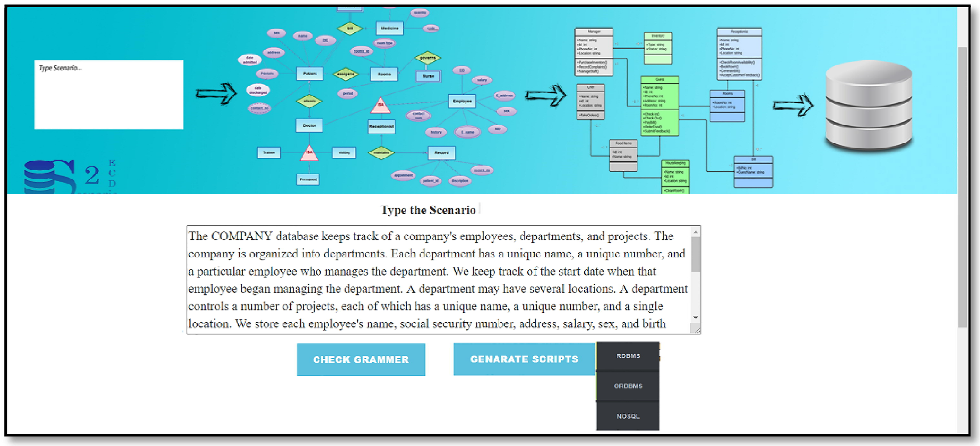

Start
Reading Normalized List
Identifying Entity
Initialize table name with Entity with "Db" + "Insert"
Identifying Attributes
Initialize column names with Attributes
Is Attribute ends with "P"
Yes
Set Column as Primary Key
No
Is Attribute ends with "F"
Yes
Set Column as Foreign Key Write Specific Lines as "Db." + "find()[0]._id"
No
Set Other Column Names
End

Fig. 7. The algorithm used for the generation of NoSQL scripts

	Number	Name	DOB	Enroll	Date	Address	Works	Supervisor
Number, Name	2	2	1	Number	1	1	Address	Name, Address
Number	2	0	0	1	0	0	0	0
Address	0	0	0	0	0	2	1	0
Name, Address	0	2	0	0	0	2	0	1
Name, DOB, Supervisor	1	2	2	**Number, Name**	**Number, Name**	**Number, Name**	**Number, Name**	2
Name, DOB, Address	1	2	2	**Number, Name**	**Number, Name**	2	**Number, Name**	**Number, Name**

Chart. 1. Sample circular matrix for relation employee



The COMPANY database keeps track of a company's employees, departments, and projects. The company is organized into departments. 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. A department controls a number of projects, each of which has a unique name, a unique number, and a single location. We store each employee's name, social security number, address, salary, sex, and birth

Fig. 8. Web Interface for the proposed model

3. RESULTS AND DISCUSSION

3.1 Experiments

To evaluate the proposed model, 10 different scenarios from a database textbook [15] were used which are standard, well-written and widely referred to by the experts in database technology. Table 1. gives the purpose, number of entities, attributes and relationships of each database. Each of these scenarios is provided with an ER diagram as well. The ER diagrams are used for validation purposes. The model was validated and verified by using those scenario descriptions by manually comparing them with the outputs of each stage of the proposed method with the aid of Test Cases which are described in Section 3.1.1 below. Since the scenario descriptions are standard and well-written descriptions.

For example, in Table 1, the scenario for the university database taken from the database textbook in [15] contains 8 sentences. It includes 6 entities and 34 attributes. These entities are combined with 3 relationships. Similarly, the details of the 9 other scenario descriptions are given in Table 1.

3.1.1 Test cases

10 test cases were used to evaluate various stages of the methodology. Two of the test cases are given in Chart 2. which are used for the evaluation of extracting entities and attributes. As shown in Chart 2. for each test case user given scenario was used and the expected and actual outputs are compared. From the result field, it gives whether each test case identified all entities and attributes correctly or not.

3.2 Performance of the Key Element Extraction

Table 2. illustrates the accuracy of extracting the key elements of database tables which are entities, attributes and relationships. For each scenario description, Table 2. presents the actual number and the identified number of entities, attributes and relationships by the proposed model as (actual/identified). It can be seen from Table 2. that for all database tables except for the US house of a representative database, all entities are identified correctly. When it comes to the identification of attributes except for one attribute each from 4 databases (Company 1, Company 2, Course, US house of a representative databases) all attributes are identified correctly.

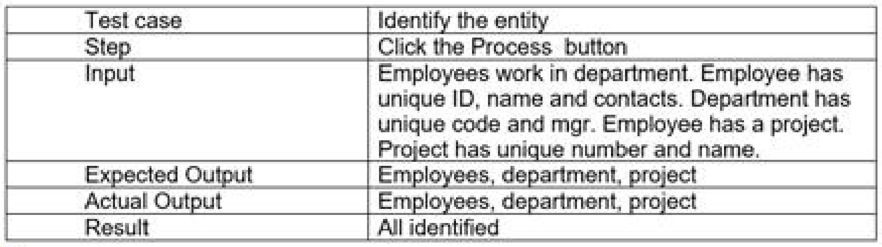
One issue noted is that some attributes are identified twice (e.g., Company database 1). Also, if the same attribute appears in two 2 entities, the system has missed the identification of one of them. For example, the attribute "name" is common for both department entity and employee entity in Company database 2, but it is identified only once. Further, except for one relationship each from US house representative and Museum databases, all relationships are identified correctly.

3.3 Performance of the Logical Schema and Script Generation

This section presents the performance of logical schema and script generation steps of the methodology. As explained in the methodology, the three scripts are generated using the logical schema. The logical schema is created based on the XML file. Table 3. shows the accuracy of automatic generation of logical schema, RDB scripts, ORDB scripts and NoSQL scripts. For each logical schema, manual cross-checking was done by comparing the generated schema line by line. To validate the database scripts, table names, attributes (column names), primary keys and foreign keys are manually cross-validated. From Table 3. it can be seen that the accuracy of generating the logical schema is 90% or less. The reason for this is that rarely the foreign keys are not identified correctly from the automated ER model because some of the useful keywords are missed due to the grammar checker. The accuracy of logical schema depends on the number of foreign keys and the number of primary keys. Another reason for less accuracy is the fact that cardinality ratios such as M:N or 1:M but for 1:1 not correctly identified. For instance, for the university database, one foreign key is not detected and because of that, it gives 85% of accuracy.

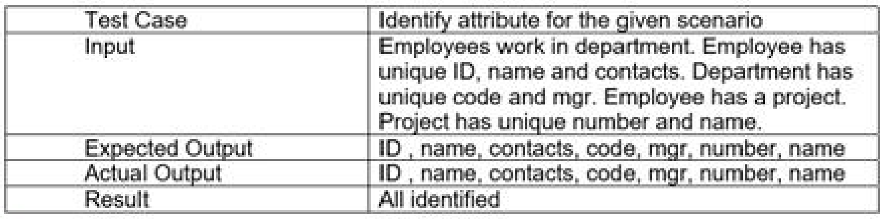
Table 1. The details of the 10 scenario descriptions

Database	Purpose of Database	(Entities, attributes, relationships)
University	To store the information of a university student and courses.	(6, 34, 3)
Company 1	To keep track of employees, departments and projects	(3, 17, 3)
Company 2	To store the details of departments inside the company	(4, 11, 2)
Course	To store information about the courses offered in a semester	(3, 12, 3)
Olympic	To keep track of sports complexes and events	(3, 8, 2)
Soap	To store details of sales of the specific soap type	(5, 14, 1)
US house of a representative	To store the details of on the house of representatives.	(4, 11, 3)
Museum	To store the details of objects inside a museum.	(11, 34, 6)
Library	To record the operations of a library	(4, 15, 6)
Airport	To keep track of airplanes, owners, airport employees, and pilots saved for this database.	(8, 21, 5)



Test case	Identify the entity
Step	Click the Process button
Input	Employees work in department. Employee has unique ID, name and contacts. Department has unique code and mgr. Employee has a project. Project has unique number and name.
Expected Output	Employees, department, project
Actual Output	Employees, department, project
Result	All identified

a)



Test Case	Identify attribute for the given scenario
Input	Employees work in department. Employee has unique ID, name and contacts. Department has unique code and mgr. Employee has a project. Project has unique number and name.
Expected Output	ID , name, contacts, code, mgr, number, name
Actual Output	ID , name, contacts, code, mgr, number, name
Result	All identified

b)

Chart 2. Two of the test cases used to evaluate the identification of a) entities, b) attributes for the given scenario

Table 2. Performance of the identification of key elements of the 10 scenario descriptions

Scenario (Database)	Entities	Attributes	Relationships
	(actual/identified)	(actual/identified)	(actual/identified)
University	6/6	34/34	3/3
Company 1	3/3	18/17	3/3
Company 2	4/4	10/11	2/2
Course	3/3	11/12	3/3
Olympic	3/3	8/8	2/2
Soap	5/5	14/14	1/1
US house of a representative	3/4	10/11	2/3
Museum	11/11	34/34	5/6
Library	4/4	15/15	6/6
Airport	8/8	21/21	5/5

Table 3. Performance of Logical schema, RDB script, ORDB script and NoSQL scripts

Database	Logical Schema	RDB Script	ORDB Script	NoSQL Script
University	85%	90%	95%	95%
Company 1	90%	90%	95%	95%
Company 2	85%	85%	90%	90%
Course	80%	70%	75%	75%
Olympic	80%	75%	75%	70%
Soap	85%	90%	90%	90%
US house of a representative	75%	70%	70%	65%
Museum	80%	85%	85%	85%
Library	90%	85%	80%	80%
Airport	70%	85%	85%	90%
Mean Accuracy	82.0%	82.5%	84.0%	83.5%

The accuracy of the generation of RDB, ORDB and NoSQL database scripts is calculated using the table counts. RDB scripts depend on both logical schema and the key elements. For the first scenario, one table is not generated due to a foreign key issue and therefore it gives 90% of accuracy for the RDB script. Similarly, the accuracies of the RDB scripts for other database scenarios are slightly less than 90% for the same reason. Overall, according to Table 3. the accuracies of RDB scripts are less than ORDB and NoSQL scripts since ORDB and NoSQL scripts are generated mainly using the key database elements only. ORDB and NoSQL scripts give a better accuracy value because of having almost 100% accuracy for key element extractions as given in Table 1. According to the results shown in Table 3. the mean accuracy values are 82.5%, 84.0%, and 83.5% for RDB, ORDB and NoSQL scripts, respectively. Also, the mean accuracy for logical schema generation is 82.0%.

Since this study introduces a novel method, there is no existing work to compare the results covering the entire process. Extracting key database elements and generating three types of database scripts follows an entirely different approach compared to related works summarized in Section 1. Overall, the accuracies of various stages of the method presented in Table 2. and Table 3. are promising.

4. CONCLUSION

This study is focused on developing a novel method for the automatic generation of database scripts for a given scenario description from the requirement specification of a software system. The proposed method first extracts relevant information to create an entity-relation model from scenario descriptions using natural language processing techniques and then uses a few new algorithms to convert the entity-relational model to the logical schema and finally to a SQL database script. Another aim of this study was to develop a web application integrating all the algorithms introduced to automate the database script generation process. This web application was used to test the method which supports three popular database types, namely, relational (RDB), object-relational (ORDB) and not only SQL (NoSQL). Through an experiment conducted using 10 scenario descriptions related to 10 diverse disciplines, it was verified that the proposed method could generate database scripts of RDB, ORDB and NoSQL databases with accuracies of 82.5%, 84.0% and 83.5%, respectively.

Overall, this method could help to overcome some of the challenging steps of the database design in practice. One of the major outcomes of this work is the newly developed web application that can be used for educational purposes. Educators and learners can use this application for teaching and learning the database technology.

Therefore, the major contributions of this study include the introduction of novel algorithms to automate the entire process of converting a scenario description of the requirement specification to a database script of one of RDB, ORDB and NoSQL types, and the development of a new web-based tool that can be used to educate and train students and software engineers on the database technology.

One limitation of the proposed method is the extraction of attributes having the same name appearing in different entities that can be fixed using advanced natural language processing (NLP) techniques. The research work presented here is focused only on scenario descriptions provided in textual format, however the proposed model can be extended to support scenario descriptions given in spoken language format too. Further, this method generates only the entity-relational and class models and it can be improved to automatically generate entity relationship diagrams and class diagrams in the future.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Anonymous. Database Designer for MYSQL. A handy tool with intuitive interface that allows you to build a clear and effective database structure visually. Available:http://www.microolap.com/products/database/mysql-designer.
2. Dedhia R, Jain A, Deulkar K. Techniques to automatically generate Entity Relationship Diagram. International Journal of innovation and advancement in computer science IJIACS. 2015;4:10. ISSN: 2347-8616.
3. Xu S, Li Y, Lu S. ERDraw. An XML-based ER-diagram Drawing and Translation Tool; 2003.
4. Btoush E, Hammad M. Generating ER Diagrams from Requirement Specifications Based On Natural Language Processing. International Journal of Database Theory and Application. 2015;8:61-70. DOI: 10.14257/ijdta.2015.8.2.07.
5. Uma M, Sneha V, Sneha G, Bhuvana J, Bharathi B. Formation of SQL from natural language query using NLP. 2019 International Conference on Computational Intelligence in Data Science (ICCIDS). 1996;1-5. DOI: 10.1109/ICCIDS.2019.8862080.
6. Luisa M. NL-OOPS: From natural language to object oriented requirements using the natural language processing system LOLITA. Natural Language Engineering. 2. 1996;161-187. DOI:10.1017/S1351324996001337.
7. Li K, Dewar RG, Pooley RJ. Object-oriented analysis using natural language processing. 2005;75-76.
8. More P, Phalnikar R. Generating UML diagrams from natural language specifications. International Journal of Applied Information Systems. 2012;1:19-23. DOI: 10.5120/ijais12-450222.
9. Herchi H, Abdessalem W. From user requirements to UML class diagram; 2012 ;arXiv preprint arXiv:1211.0713.
10. Smith JBL, Burgoyne JA, Fujinaga I, De Roure D, Downie JS, 2011, October. Design and creation of a large-scale database of structural annotations. In ISMIR. 2011;11: 555-560.
11. Fokkema IF, Den Dunnen JT, Taschner PE, 2005. LOVD: easy creation of a locus-specific sequence variation database using an “LSDB-in-a-box” approach. Human mutation, 2005;26(2):63-68.
12. Connors KA, Beasley A, Barron MG, Belanger SE, Bonnell M, Brill JL, et al. (2019). Creation of a curated aquatic toxicology database: EnviroTox. Environ mental Toxicology and Chemistry. 2019; 38(5):1062-1073.
13. Russell SJ, Norvig P. Artificial Intelligence, A modern approach. 3rd ed; 1995.

14. Rich E, Knight K. Artificial Intelligence. 2nd ed; 2009.

15. Elmasri R, Navathe SB. Fundamentals of database system. 6th ed; 2011.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/66504