Interface for Composing Queries for Complex Databases for Inexperienced Users

Rodolfo A. Pazos R.(✉), Alan G. Aguirre L., Marco A. Aguirre L., and José A. Martínez F.

Instituto Tecnológico de Cd. Madero, Tecnológico Nacional de México, Cd. Madero, Mexico
r_pazos_r@yahoo.com.mx, li.aguirre.lam@hotmail.com, marco.aguirre@itcm.edu.mx, jose.mtz@gmail.com

Abstract. In most business activities, decision-making has a very important role, since it may benefit or harm the business. Nowadays decision-making is based on information obtained from databases, which are only accessible directly by computer experts; however, the end-user that requires information from a database is not always a computer expert, so the need arises to allow inexperienced users to obtain information directly from a database. To this end, several tools are commercially available such as visual query building and natural language interfaces to databases (NLIDBs). However, the first kind of tools requires at least a basic level of knowledge of some formal query language, while NLIDBs, despite the fact that users do not require training for using the interface, have not obtained the desired performance due to problems inherent to natural language processing. In this paper an intuitive interface is presented, which allows inexperienced users to easily compose queries in SQL, without the need of training on its operation nor having knowledge of SQL.

1 Introduction

Natural interfaces allow users to access information in a database by a query formulated in natural language (NL) or by multimodal interfaces. Examples of such interfaces are described in [1, 2]. However, the use of natural language to formulate a query to a DB can lead to some problems concerning the process of translating the query to SQL [3], which could cause that users could not obtain the desired result.

An alternative to NLIDBs are the tools for query composition for databases, which aim at obtaining information from a DB by formulating an SQL query by using a friendly graphical interface. They also facilitate the composition of an SQL query using various methods of composition; however, they require users to have some degree of knowledge of SQL and the database schema; as a result, such interfaces are difficult to use for inexperienced users.

This paper presents a human-computer interface that facilitates mid and higher managers to compose an SQL query to obtain information from a database, which is necessary for their decision-making tasks. Unlike other interfaces, the design of this interface allows composing a query without the need to have any knowledge of neither SQL nor the database schema. The design of the interface involves two major aspects:

E. Onieva et al. (Eds.): HAIS 2015, LNAI 9121, pp. 61–72, 2015.
DOI: 10.1007/978-3-319-19644-2_6

the human-computer interaction, which is presented in this paper; and the semantic information dictionary, which is based on a semantically enriched database model (described in [4]).

This interface has been customized for and tested with Spanish-speaking users, but its design allows its customization for other European languages: English, French, Italian, Portuguese.

2 Related Work

As mentioned in Sect. 1, there are a large number of interfaces that allow the composition of SQL queries, such as COBASE [5], WYSIWYM [6], TAICHI [7] and Query by Example of Microsoft Access,[1] among others.

Table 1 shows a comparison of the characteristics of the interfaces mentioned before and those of the interface described in this paper. As shown, most interfaces require a degree of knowledge of SQL, which could lead the user to face difficulties in using these interfaces. Furthermore, none of the interfaces explains the contents of the database to the user, so the user can not be sure that the information that he/she needs is in the database.

Table 1. Characteristics of some interfaces for query composition

Interface	Domain independence	Method of query composition	Explanation of the DB contents	Need of SQL knowledge	Used in complex DBs
CoBase	✓	Selection	×	✓	×
WYSIWYM	×	• NL • Templates	×	×	✓
TAICHI	✓	• NL • Drag & Drop	×	✓	×
MS Access	✓	• Drag & Drop • SQL templates	×	✓	✓
Proposed interface	✓	Selection	✓	×	✓

Unlike the interfaces presented in Table 1, a NLIDB requires the user to formulate a NL query in order to generate an SQL query. Such task requires a semantic processing of the NL query. An example of a semantic processing is presented by Agrawal [2] in 2013. Unfortunately, while using a NLIDB, sometimes the user may request information that is not stored in the database, this may happen because the user does not know the database schema; moreover, the user may think that the NLIDB can answer any questions, which is not so.

[1] http://products.office.com/en-us/access.

3 Description of the Query Composition Interface

The proposed interface aims at allowing an inexperienced user to compose SQL queries through a three-step composition process: selection of the topic of interest, selection of the elements of interest, and specification of the search conditions.

The graphical interface contains controls that most users are familiar with; therefore, the composition of a query does not require the user to receive training to use the interface, as the experimental results show (Sect. 5).

To this end, the interface uses a classification of tables that belong to the database and builds a graphic structure (composition tree) to represent the database schema based on the classification of tables and the relations between them.

In addition, the interface displays descriptions of the tables and columns of the DB keeping their names hidden, thus the user can get a better idea of what is stored in the database than by just looking at the names of tables and columns.

In the next subsections the classification of tables and the composition tree are explained, which are two of the most important aspects of the interface.

3.1 Classification of Tables

The classification of tables allows the interface to group tables of the database according to their level of relevance to query composition. In addition, the classification of tables is used to build the query tree.

Table 2 shows the different types of tables proposed for classification, they are listed in decreasing order of relevance to query composition.

Table 2. Classification of tables

Type of table	Description
1. Base tables	Main tables that store information that is frequently used for querying the database.
2. Views	Virtual tables that are obtained from a Select statement that involves base tables and are used for providing information that can not be directly obtained from base tables.
2. Catalogs	Tables that are used mainly for obtaining a NL description for a key or ID.
3. M-to-N relations	Tables that contain foreign keys that belong to other tables (T_i and T_j) and are used for implementing M-to-N relations between T_i and T_j.
4. Satellite tables	Tables that are disconnected from the rest of the tables. These tables are used by internal processes of the applications that use the database.

In query composition, base tables have the highest relevance, also views and catalogs are highly likely to be used for composition, while tables that implement M-to-N relations are only used for the construction of the composition tree; finally, satellite tables are seldom used for query composition.

3.2 Composition Tree

The composition tree is the most important component of the interface. It is used by the user for selecting the columns belonging to the tables of the database that will be used in query composition.

In the composition tree, a fragment of the DB (tables and columns) is presented using NL descriptions instead of names. Each table is represented as an expandable node, while the columns of each table are represented with simple nodes, which can not be expanded and are the only nodes that can be selected by the user.

The construction of the composition tree is carried out by the interface in the second step of the query composition process (selection of the elements of interest) from a table selected by the user in the first step.

Algorithm 1 shows the pseudocode for the construction of the composition tree, where *CT* is the composition tree, *n* is the relation node (of the composition tree) that represents a table, and *R* is a set of tables that are related with table *t*. The construction consists of initializing *CT* by inserting the root table T_r and applying the recursive function shown at line 1, which requires a table *t* as input. Subsequently, the interface obtains a set *R* of related tables of *t*, and for each table *r* in *R* its type is evaluated as follows:

- If the related table is a base table, it applies the recursive function to insert the columns and relations in the composition tree *CT* (line 8).
- If the related table is a catalog table, the nodes corresponding to the columns of this table are inserted into the parent node (line 11).
- If the related table is an M-to-N relation, the table related to this one is obtained (line 14), and the function to insert related tables is applied (line 15), thus, hiding the M-to-N relation table.

Algorithm 1. Pseudocode for constructing a composition tree

```
1:  insertRelations(t)
2:      p  //Parent node of t
3:      n ← insertRelationNode(CT_p, t)
4:      insertColumnNode(CT_n, t)
5:      R ← getRelatedTables(t)
6:      for each r from R do
7:          if isBaseTable (r)
8:              insertRelations(r)
9:          endif
10:         if isCatalog(r)
11:             insertColumnNodes(CT_n, r)
12:         endif
13:         if isMtoN (r)
14:             r' ← getRelatedTable(r)
15:             insertRelations(r')
16:         endif
17:     endfor
18: end
```

3.3 Query Composition Process

The query composition process is performed by the user using the interface. This process consists of three steps: selection of the topic of interest, selection of the elements of interest, and specification of the search conditions.

In the step of selection of the topic of interest, the interface displays the tables in the database grouped by the classification shown in Table 2. In this step, the interface starts displaying the tables commonly used for composition and, at the end, those tables that are used less frequently.

For example, the classification of tables proposed for the ATIS database is shown in Table 3, which shows table descriptions in Spanish (along with their translation to English). In this case, the interface would show first the base tables, then the catalog tables, next the M-to-N relation tables, and finally the satellite tables.

Table 3. Proposed classification of tables for the ATIS database

Base tables	Catalogs	Relations M-to-N	Satellite tables
Avión (Aircraft) Aerolínea (Airline) Aeropuerto (Airport) Servicios de aeropuerto (Airport service) Ciudad (City) Clase compuesta (Compound class) Tarifa (Fare) Vuelo de un aeropuerto a otro (Flight from an airport to another) Conexión de vuelo (Flight connection)	Clase de servicio (Class of service) Servicio de comida (Food service) Clase de restricción (Class of restriction) Estado (State) Zona horaria (Time zone) Servicio de transporte del aeropuerto (Transport service of the airport)	Vuelo – Tarifa (Flight – Fare) Restricción de empresa (Airline restriction) Segmento de conexión (Connection leg) Empresa doble (Dual carrier) Servicio terrestre (Ground service) Parada (Stop)	Descripción de código (Code description) Día (Day) Días de vuelo (Flight days) Nombres de mes (Month names) Intervalo de tiempo (Time interval)

In this step, the selected table will be used to build the composition tree. This table represents the main topic of which the user is interested in finding information.

After choosing the topic of interest of the query, the user will be directed to the selection of the elements of interest, where the interface displays the composition tree and asks the user to select the elements that he/she wants to be displayed as the result of the query. In this step the interface obtains the information needed to build the Select clause of the query.

For each element that the user wants to know in the database, he/she must add it to a list containing the elements of interest. Next, the interface stores a vector of tree nodes that represents the path from the root of the tree to the selected element. Each node is represented as a vector with four positions, where the first position stores the description of the table to which the node belongs, the second stores the description of

the column representing the node, the third stores the relation between the previous node and the current node, and the fourth represents the type of table to which the node belongs. Figure 1 shows how the information concerning the generation of a path is stored.

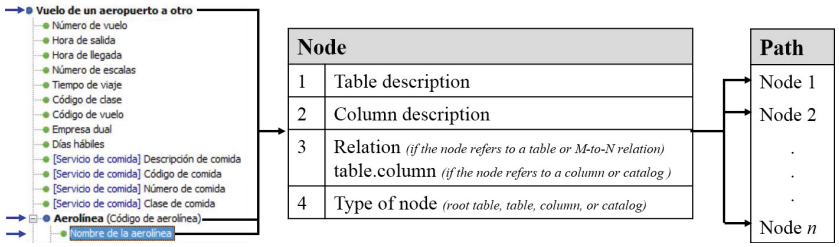

Fig. 1. Generation of the path information for a selected node

Once the user has finished adding items to the list of elements of interest, the interface will have a set of vectors representing the paths of each element of interest. This set of paths is stored in a vector with *m* positions, where *m* is the total number of elements added by the user, as shown in Fig. 2.

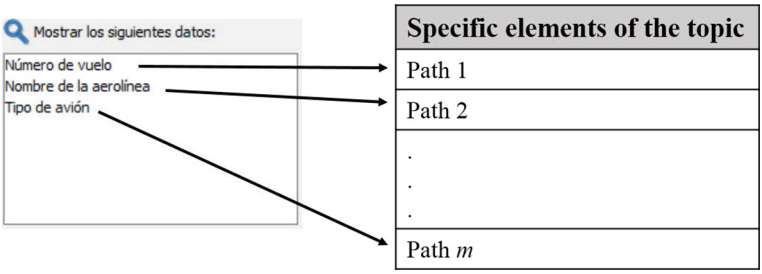

Fig. 2. Information of the paths of the elements of interest

Once the user has defined the elements of interest, if the user wants to obtain a dataset with specific information, he/she can use the interface to enter the search conditions to discriminate the data and get a result with specific characteristics. In this step the interface obtains the information needed to construct the Where clause of the query.

The information obtained by the interface concerning the search conditions is shown in Fig. 3. This figure shows that the information of a search condition is constituted by a column node or catalog, a comparison operator (=, <, >, <=, > =, <>), the description of the comparison operator, the value of the search condition, and the path from the root node to the selected node (see Fig. 1). After the user has finished defining search conditions, the interface will have a list of search conditions.

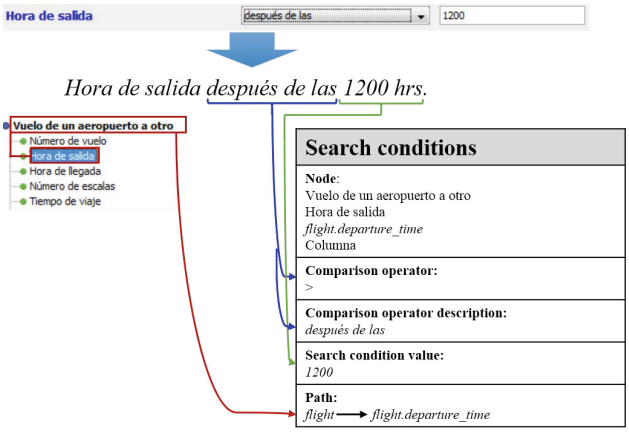

Fig. 3. Definition of a search condition

Once the three steps of the composition process have been completed, the interface will be able to generate the SQL expression from the elements of interest and search conditions already defined. The query is constructed as follows:

- Select clause. From the list of elements of interest, for each element, the last node of the path (the column selected by the user) is obtained and defined as an element of the clause.
- Where clause. From the list of search conditions, for each condition specified by the user, the name of the column, the comparison operator and the value of the condition are taken to build a condition of the clause. Subsequently the joins between tables are defined from the paths of all the elements of the list of elements of interest and the paths from the list of search conditions.

4 Composition Example

Considering the classification of tables proposed in Subsect. 3.1, the following types of queries that the user could compose using the interface were considered:

- Queries that involve one base table.
- Queries that involve two base tables.
- Queries that involve three base tables.
- Queries that involve two base tables and one M-to-N table.

To illustrate the composition of a query, consider the next query that involves two base tables directly connected:

Dame el número de vuelo y nombre de aerolínea de los vuelos que salen antes de las 1500 h (Give me the flight and airline name of the flights departing before 1500 h).

As mentioned previously, this interface was customized for Spanish; thus, the relevant information for this example is presented in Spanish along with its translation to English.

It is worth noting that the user must consider three aspects to compose the query: the topic on which he/she wishes to obtain information, the specific elements of the topic that are of interest, and if the query requires the specification of search conditions.

First, the user defines the topic that the query will deal with. In this case, the topic of interest is *Vuelo de un aeropuerto a otro* (*Flight from an airport to another*), which involves table *flight*. Later, the elements of interest of the main topic are selected. In the example a flight number and airline name are required. Therefore, the node *Número de vuelo* (*Flight number*) of the root node *Vuelo de un aeropuerto a otro* (*Flight from an airport to another*) is selected. For the second element, the user should extend the node *Aerolínea* (*Airline*) and select the node *Nombre de la aerolínea* (*Airline name*).

The sample query requires to obtain information with specific conditions; therefore, the search conditions are defined. For this example, it is required that the flight departs before 1500 h, then, the node *Hora de salida* (*Departure time*) is selected from the node *Vuelo de un aeropuerto a otro* (*Flight from an airport to another*). For the selected node, it is necessary to specify the description of the comparison operator *antes de las* (*before*) and the value *1500* without the measurement unit (*hrs*). Finally, the interface will display the SQL query and its result as a table whose columns are *Número de vuelo* (*flight number*) and *Nombre de aerolínea* (*Airline name*).

5 Experimental Results

The experiments performed on the composition interface aim at measuring the ease of use in conjunction with its functionality. This is done in order to determine if the interface is friendly and functional enough to compose the queries mentioned in Sect. 4 to a complex database. For this purpose, the following parameters are measured:

- Amount of time that a user spends in composing a query.
- Number of attempts by a user to compose a query correctly.

Considering the abovementioned parameters, information can be obtained on the difficulty involved in composing a query for a specific type of query.

5.1 Description of the Experimental Setting

The experiments were conducted in a one-hour session with 17 students majoring in engineering in computer science. They were provided a user manual with one day in advance that details how the composition interface is used. The users were given a set of 20 natural language queries related to the ATIS database sorted by level of difficulty, which are separated in groups of five queries, ordered as follows:

- Queries from 1 to 5. Queries that involve one base table.
- Queries from 6 to 10. Queries that involve two base tables.
- Queries from 11 to 15. Queries that involve three base tables.

- Queries from 16 to 20. Queries that involve two base tables and one M-to-N relation table.

It is worth mentioning that the students were never provided with a diagram of the database schema nor the correct SQL statement for each query; moreover, some of them were not familiar with the domain of the database.

5.2 Results

The experimental results are shown in Table 4. These results were obtained from 20 queries composed by 17 users, discarding the attempts of queries that were not composed correctly.

Table 4. Experimental results for each query

Query No.	Attempts			Times			No. of correct compositions
	Minimum	Maximum	Average	Minimum (sec.)	Maximum (sec.)	Average (min.)	
1	1	10	2.35	42	609	2.78	17
2	1	6	1.47	43	343	1.55	17
3	1	3	1.25	27	303	1.49	16
4	1	2	1.12	31	73	0.76	17
5	1	7	1.59	28	285	1.17	17
6	1	2	1.06	24	105	0.71	17
7	1	1	1.00	20	85	0.59	16
8	1	5	2.00	24	280	1.36	17
9	1	1	1.00	27	57	0.59	17
10	1	6	2.15	41	514	2.55	13
11	1	1	1.00	34	97	0.83	17
12	1	2	1.06	39	123	0.98	16
13	1	2	1.06	46	182	1.23	16
14	1	4	1.24	24	185	0.91	17
15	1	4	1.18	55	257	1.63	17
16	1	1	1.00	19	75	0.49	17
17	1	2	1.06	31	99	0.82	17
18	1	2	1.24	26	110	0.84	17
19	1	3	1.25	25	237	1.30	16
20	1	8	4.64	90	970	6.93	11
Average			**1.48**			**1.47**	

The meaning of the columns of Table 4 is explained next. The first column indicates the number of query, the second column indicates the minimum number of attempts it took to compose the query, the third column indicates the maximum number of attempts, the fourth column shows the average number of attempts. The fifth column indicates the minimum time in seconds that was spent in composing the query, the sixth column indicates the maximum time, the seventh column shows the average time in

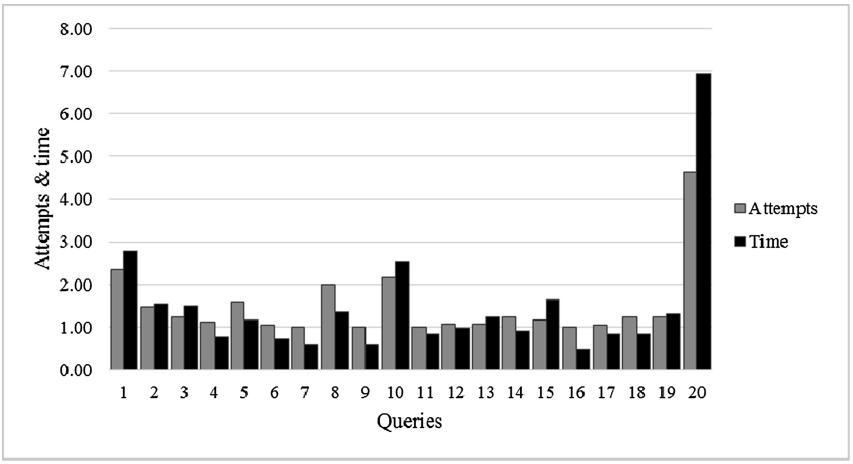

Fig. 4. Average attempts for each query

minutes; finally, the eighth column shows the number of users that were able to correctly compose the query.

As shown in the plot of Fig. 4, most of the queries could be answered by the users in one or two attempts in less than 2 min. However, for query number 1 a higher average of attempts and time occurs. This is so because it is the first query that the users must compose and they face the learning process of the operation of the interface. Later, the attempts and the average time were declining because users got used to the operation of the interface. It is important to remark that by the fourth query, the users have already learned to operate the interface.

The queries from number 4 to 19 were composed by most users in the first attempt, except for queries 8 and 10, and the average time for this set of queries was 55 s.

Query number 10 (*Dame los códigos de clase de tarifa y tipos de tarifa de temporada de la clase de servicio con rango 12 – Give me the codes of fare class and the types of season fare of the service class with rank 12*) required a larger number of attempts because the query requires that the user has more knowledge about the topic at hand.

Additionally, query number 20 has a high number of attempts because it involves a large number of tables (three base tables and one M-to-N relation table), and the required information by the query needs to be searched in the deepest levels of the composition tree.

In summary, the experiments indicate that users can compose a query of any type in about one minute with an average of approximately 1.5 attempts.

Some comments left by the users about the interface were the following: “At first, it was hard to understand how to use it. However, it became easier to use it by just using it”. “It works just fine for me”. “As you use it, it becomes easier to get the queries right”. As noted, the comments about the interface were mostly positive and some of them confirm the conclusions drawn from the experimental results.

6 Conclusions

The experiments carried out on the composition interface show that users require to use the query composition interface a small number of times (about three) to learn how to compose queries quickly and efficiently. This is so because the interface is intuitive enough for users that do not know the schema and domain of the DB so as to allow composing queries in about one minute.

Note that a third party designed the natural language queries used in these experiments. Therefore, the interface will perform better when a user devises his/her own queries, because the user knows specifically what information he/she needs from the DB.

Table 4 shows that the average number of attempts is 1 to 2 per query and that a user takes an average of 1.47 min per query, indicating that the interface allows users to compose queries properly in a reasonable time.

One of the main aspects that enable the good performance of the interface is the use of the composition tree. This mechanism allows displaying a fragment of the database schema (specifically, the fragment of interest for each particular query), so the users can make use of several tables at once without this being a problem to compose queries. Therefore, the number of tables involved in a query does not greatly increase the difficulty of the composition of a query when using the composition tree.

In summary, the proposed interface has proven useful for the composition of queries that include three different types of tables. This is because the interface allows the user to view a section of the database schema through the composition tree, which provides information about each element of the database schema using natural language descriptions that are easy to understand.

The results in Table 4 show that from a total of 20 queries, 13 could be correctly composed by all of the 17 users. In addition, another 5 queries were composed correctly by 16 users, leaving 2 queries with 13 and 11 correct compositions respectively. This shows that most of the queries could be composed correctly by the users. Additionally, the low number of correct compositions for queries number 10 and number 20 was due to the lack of knowledge about the domain of the database by the users.

It is important to point out that 11 out of 17 users composed correctly all the queries showing that the interface can be used for composing queries by most of the users. This is remarkable considering that neither a diagram of the database schema nor the correct SQL statements were provided to the users involved in these experiments.

References

1. Chai, J., Pan, S., Zhou, M.: MIND: a context-based multimodal interpretation framework in conversational systems. In: Van Kuppevelt, J.C.J., Dybkjær, L., Bernsen, N.O. (eds.) Advances in Natural Multimodal Dialogue Systems Text, Speech and Language Technology, vol. 30, pp. 265–285. Springer, Netherlands (2005)

2. Agrawal, A., Kakde, O.: Semantic analysis of natural language queries using domain ontology for information access from database. Int. J. Intell. Syst. Appl. **5**, 81–90 (2013)
3. Pazos, R., Aguirre, M., Gonzalez, J., Carpio, J.: Features and pitfalls that users should seek in natural language interfaces to databases. In: Castillo, O., Melin, P., Pedrycz, W., Kacprzyk, J. (eds.) Studies in Computational Intelligence, vol. 547, pp. 617–630. Springer, Heidelberg (2014)
4. Pazos, R., González, J., Aguirre, M.: Semantic model for improving the performance of natural language interfaces to databases. In: Batyrshin, I., Sidorov, G. (eds.) MICAI 2011. LNCS, vol. 7094, pp. 227–290. Springer, Heidelberg (2011)
5. Zhang, G., Chu, W., Meng, F., Kong, G.: Query formulation from high-level concepts for relational databases. In: User Interfaces to Data Intensive Systems, pp. 64–74. IEEE Computer Society, Los Alamitos, CA (1999)
6. Hallet, C., Scott, D., Power, R.: Composing questions through conceptual authoring. Computational linguistics **33**(1), 105–133 (2007)
7. Pan, S., Zhou, M., Houck, K., Kissa, P.: Natural language aided visual query building for complex data access. In: 22nd Innovative Applications of Artificial Intelligence Conference (IAAI-10), pp. 1821–1826. Association for the Advancement of Artificial Intelligence, Palo Alto, CA (2010)