(12)Indian Patent Application

---

(21) Application Number: 202041014478

(22) Filing Date: 31/03/2020 (43) Publication Date: 08/10/2021

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): Singh, Madhusudan
Halder, Kaushik
Vanapalli, Venkata Nirmal Ramesh Rayulu

(51) International Classifications: G06F 16/2452 G06F 16/242 G06F 16/33 G06F 16/22 G06F 16/28

(54) Title: A METHOD OF CONVERTING A NATURAL LANGUAGE QUERY TO A STRUCTURED QUERY LANGUAGE (SQL) QUERY

(57) Abstract: A method of converting a natural language query to a Structured Query Language (SQL) query is disclosed. The method may include receiving a natural language query from a user, and segregating the natural language query into a plurality of tokens. The method may further include identifying a database function associated with the plurality of tokens, using a database lookup table, and identifying one or more features associated with each of the plurality of tokens. The one or more features may include a table name, an attribute, and a data type. The method may further include determining a parameter associated with each the of the plurality of tokens, and creating one or more SQL queries based on the database function, the one or more features and the parameter associated with each the of the plurality of tokens.

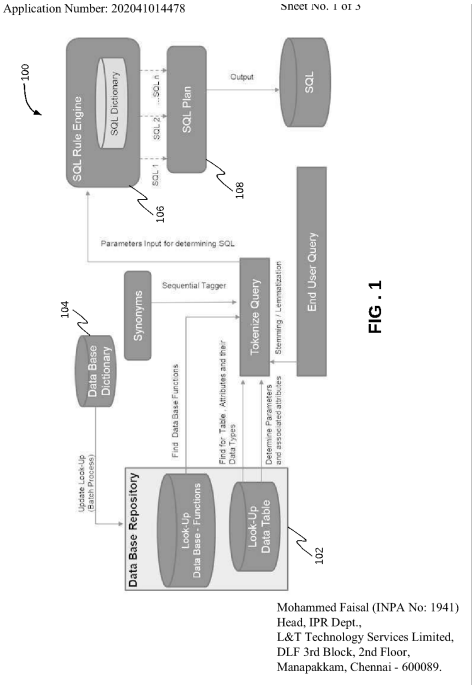


FIG . 1

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

FORM 2

THE PATENTS ACT 1970
(39 OF 1970)
&
The Patent Rules, 2003
Complete Specification
(See Section 10 and Rule 13)

1. TITLE OF THE INVENTION

A Method Of Converting A Natural Language Query To A Structured Query Language (SQL) Query

2. APPLICANT(S)

(a) NAME : **L & T TECHNOLOGY SERVICES LIMITED**

(b) NATIONALITY : **INDIAN**

(c) ADDRESS : DLF IT SEZ Park, 2nd Floor – Block 3

1/124, Mount Poonamallee Road

Ramapuram, Chennai – 600 089,

INDIA

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification particularly describes the invention and the manner in which it is to be performed.

DESCRIPTION

Technical Field

[001] This disclosure relates generally to structured query language (SQL) translations, and more particularly to a method and a system for converting a natural language query to a SQL query.

BACKGROUND

[002] Structured Query Language (SQL) is a common query language used to convert a plain text into a database understandable language. At present, the SQL language is implemented in relational database systems (RDBMS). Although, SQL is a powerful language, it may be too complex for many users because of its diverse nature and the need for users to know the underlying data schema.

[003] Some machine learning-based techniques are used to translate a natural language query to a SQL query based on a probabilistic approach. However, these machine learning-based techniques may require a large volume of data for processing, which may be time consuming and prove costly. Therefore, an effective, cost-efficient, and simple process of converting a natural language query to a SQL query is desired.

SUMMARY OF THE INVENTION

[004] In an embodiment, a method of converting a natural language query to a Structured Query Language (SQL) query is disclosed. The method may include receiving a natural language query from a user, and segregating the natural language query into a plurality of tokens. The method may further include identifying a database function associated with the plurality of tokens using a database lookup table, and identifying one or more features associated with each of the plurality of tokens. The one or more features may include a table name, an attribute, and a data type. The method may further include determining a parameter associated with each of the of the plurality of tokens, and creating one or more SQL queries based on the database function, the one or more features, and the parameters associated with each the of the plurality of tokens.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles.

[006] **FIG. 1** illustrates a block diagram of a system for converting a natural language query to a Structured Query Language (SQL) query, in accordance with an embodiment of the present disclosure.

[007] **FIG. 2A** illustrates a look-up data table, in accordance with an embodiment of the present disclosure.

[008] **FIG. 2B** illustrates a tabular form of a dictionary for deriving a SQL query from a natural language query, in accordance with an embodiment of the present disclosure.

[009] **FIG. 3** is a flowchart of a method of converting a natural language query to a SQL query, in accordance with an embodiment.

DETAILED DESCRIPTION OF THE DRAWINGS

[010] Exemplary embodiments are described with reference to the accompanying drawings. Wherever convenient, the same reference numbers are used throughout the drawings to refer to the same or like parts. While examples and features of disclosed principles are described herein, modifications, adaptations, and other implementations are possible without departing from the spirit and scope of the disclosed embodiments. It is intended that the following detailed description be considered as exemplary only, with the true scope and spirit being indicated by the following claims. Additional illustrative embodiments are listed below.

[011] Referring to FIG. 1, a block diagram of a system 100 for converting a natural language query to a Structured Query Language (SQL) query is illustrated, in accordance with an embodiment of the present disclosure. It may be noted that the system 100 may perform various functions including annotation, deriving one or more SQL queries based on a natural language query using a

skeleton-based approach, and SQL plan which may include selecting a most optimized SQL query of the one or more SQL queries.

[012] As shown in FIG. 1, the system 100 may include a database repository 102 which may include a database function lookup table (first lookup table) and a data-table lookup table (second lookup table). The system 100 may receive an end-user query which may be a natural language query. This natural language query may be tokenized to generate a plurality of tokens. Further, in some embodiments, various synonyms associated with the tokens may be determined, via sequential tagger. A database function associated with the plurality of tokens may be identified, using the first lookup table from the database repository 102. Further, one or more features associated with each of the plurality of tokens may be identified using the second lookup table from the database repository 102. The one or more features may include a table name, an attribute, and a data type. Furthermore, a parameter associated with each of the of the plurality of tokens may be determined using the second lookup table from the database repository 102. The system 100 may further include a database dictionary 104 which may update the first lookup table and second lookup table of the database repository 102.

[013] The system 100 may further include a SQL rule engine 106 which may create one or more SQL queries based on the database function, the one or more features, and the parameters associated with each the of the plurality of tokens. The system 100 may further include SQL plan 108 which may select an optimized SQL query from the one or more SQL queries.

[014] In some embodiments, the plurality of tokens may be annotated. It may be noted that an end user query may be plain text or a text in a form of a question. The process of annotation may include sequential analysis of the tokenized query by using a sequential tagger and use of corresponding synonyms to identify presence of tables and attributes in a database repository or a corpus. This corpus may further include a metadata of an object present in the database. The metadata of an object may be details include but not limited to, a table name, attributes and data type. The database dictionary 104 may include details related to the table. The database dictionary 104 may be configured to populate the database repository 102 of a look-up table.

[015] The process of annotation may further include matching the tokenized query with a database functions look-up table to determine functions. This be a one-time activity for a particular

database. The process of annotation may further include matching the tokenized query with a look-up table by using the database repository. A populated look-up table may act as a repository table. The details of the look-up data table may be fed from the database dictionary. The look-up data table may further be populated from a database based on an offline process. The offline process may include a feeding of the look-up table details form a user. An exemplary look-up table 200 is shown in FIG. 2A.

[016] As shown in FIG. 2A, the look-up data table 200 may include a column associated with each of a table name, an attribute, and a data-type. It may be noted that all details of the look-up table 200 may be stored in a database dictionary and may fed to a database repository during a populating of the look-up table 200. The populating of the look-up table 200 is important because if there is no access to the database dictionary, the details of table such as a table column name, attributes and their data type may be gathered form this look-up data table 200.

[017] FIG. 2B shows a tabular form of a SQL dictionary 202 for deriving a SQL query from a natural language query, using a skeleton-based approach. The skeleton may include various SQL rules which may be applied for a semantic parsing of the natural language query and selecting a matched skeleton from an SQL dictionary. For example, the tokenized query and the skeleton may be operated by the WHERE parser and then by the SELECT parser. First the query may be parsed for a WHERE clause and then for a SELECT statement. It may be noted that a change in the SQL dictionary present in a SQL rule engine does not impact a processing of other layers. The SQL rule engine may generate a multiple SQL queries.

[018] The SQL dictionary 202 may include classes along with its attributes and various rules. The SQL dictionary may be configured to build an appropriate SQL skeleton in order to provide matching with the input parameters. It may be noted that the various rules that are being used for matching purpose may be stored in the database repository.

[019] Referring back to FIG.1, it may be noted that the sequential analysis of the tokenized query may be performed using a sequential tagger and synonyms. The sequential tagger may further include a part-of-speech (POS) tagger for making up a word into a text on a basis of their grammar, and may use synonyms to identify whether the user end query in the form of tables and attributes is

present in the database repository or not. The synonyms may help to convert similar words that are used by the user into words that may exist in the database having same meaning.

[020] The annotation is done to find out which all terms in the statements (either attributes or functions) are actually getting matched with the look-up database table and the look-up database functions. It may be noted that the look-up database functions may be configured to identify whether the functions which are being used throughout the process actually exist or not. Further, if they exist, then the look-up database functions may be used to incorporate the related functions in a look-up database table.

[021] Referring now to FIG. 3, a flowchart of a method 300 of converting a natural language query to a SQL query is illustrated, in accordance with an embodiment. At step 302, an input text like a natural language query may be tokenized. At step 304, synonyms associated with the tokens may be determined. At step 306, the tokens may be matched with look-up database functions to determine functions of database. The look-up table may receive feed/input from database based on an offline process. At step 308, the tokens may be matched with look-up table to determine table name, attributes and their data types. This process may acknowledge to all types of databases; however, it may acknowledge one database at a single point of time. In other words, the tokenized query is analyzed sequentially for the presence of tables and attributes with the help of repository. At step 310, parameters and its associated attributes may be determined. Further, if multiple SQL queries are generated, then the most optimized one may be selected, and the results on the command line may be displayed.

[022] One or more techniques for automatically converting natural language query into the SQL query are disclosed. The techniques make use of semantic parsing of the input query. Further, the techniques use a pipelining technique to generate an SQL query from a natural language query. The techniques are semantic-based which may use a domain specific mapping corpus to relate a database attributes with their physical names. The techniques may further use a skeleton based architecture to derive the SQL query from the natural language query. This may help in determining any syntax errors even before executing the query, and recommending the end user only the parameter that is left out, thereby giving out only syntactical correct SQL queries. Moreover, the techniques provide for determining a most optimized SQL query from multiple SQL queries.

[023] It is intended that the disclosure and examples be considered as exemplary only, with a true scope and spirit of disclosed embodiments being indicated by the following claims.

CLAIMS

We claim:

1. A semantic-based method of converting a natural language query to a Structured Query Language (SQL) query, the method comprising:

segregating a natural language query into a plurality of tokens;

identifying a database function associated with the plurality of tokens, using a first database lookup table, wherein the database lookup table is based on a database dictionary;

identifying one or more features associated with each of the plurality of tokens using a second database lookup table based on a domain-specific mapping corpus, wherein the one or more features comprise a table name, an attribute, and a data type;

determining a parameter associated with each the of the plurality of tokens; and

creating one or more SQL queries based on mapping of the database function, the one or more features and the parameter associated with each the of the plurality of tokens.

2. The method as claimed in claim 1, further comprising selecting an optimized SQL query from the one or more SQL queries.

3. The method as claimed in claim 1, further comprising determining one or more synonyms associated with each of the plurality of tokens.

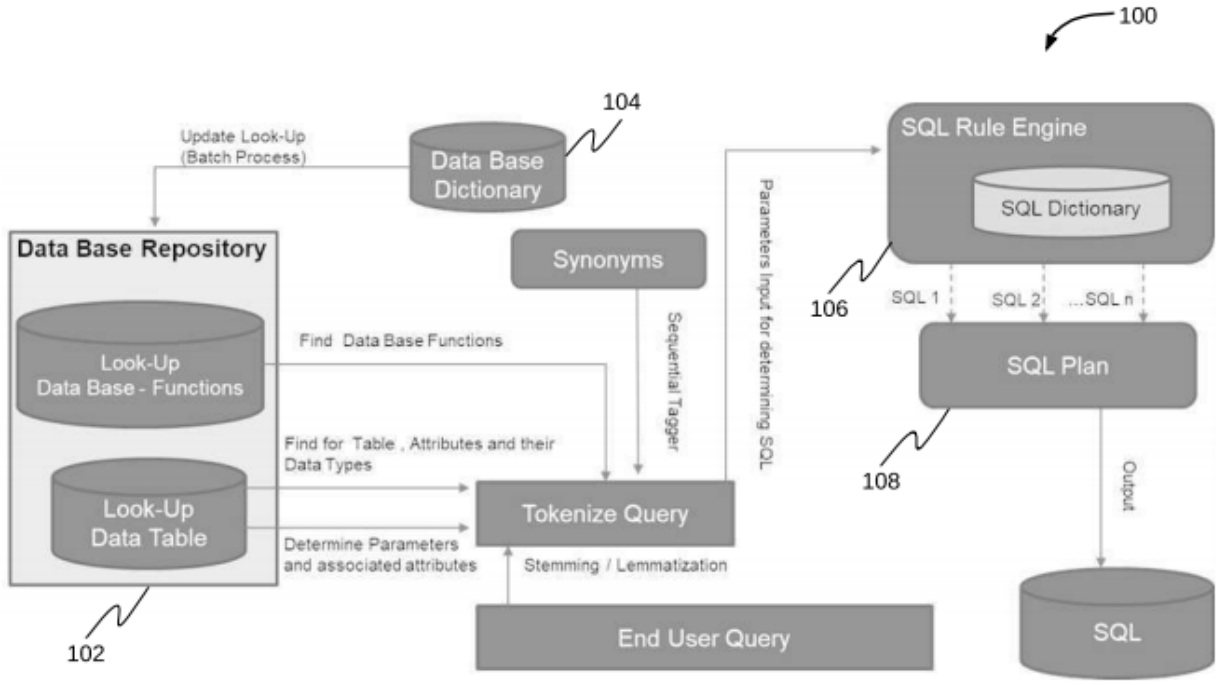
Dated this 31st day of March 2020

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

ABSTRACT

A METHOD OF CONVERTING A NATURAL LANGUAGE QUERY TO A STRUCTURED QUERY LANGUAGE (SQL) QUERY

A method of converting a natural language query to a Structured Query Language (SQL) query is disclosed. The method may include receiving a natural language query from a user, and segregating the natural language query into a plurality of tokens. The method may further include identifying a database function associated with the plurality of tokens, using a database lookup table, and identifying one or more features associated with each of the plurality of tokens. The one or more features may include a table name, an attribute, and a data type. The method may further include determining a parameter associated with each the of the plurality of tokens, and creating one or more SQL queries based on the database function, the one or more features and the parameter associated with each the of the plurality of tokens.



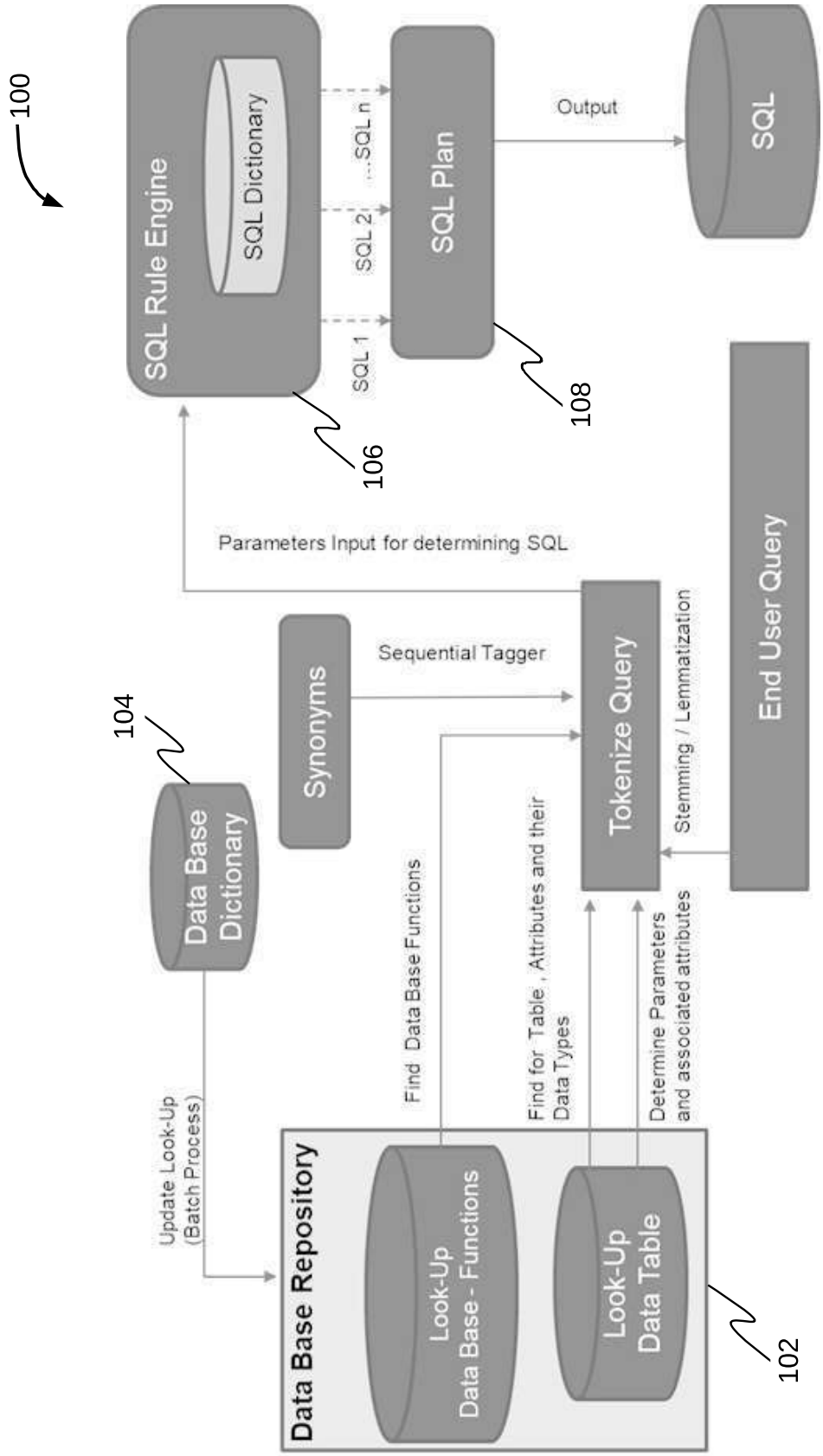


FIG . 1

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

200

Sl. No#	Data Base	Table Name	Attribute	Data Type
1	Production Plant	Plant-X001-Master	Motor	Varchar(20)
2	Production Plant	Plant-X001-Master	Turbine	Varchar(20)
3	Production Plant	Plant-X001-Transaction	Power generation	Varchar(20)
4	Production Plant	Plant-X001-Transaction	RPM	Varchar(20)
5	Production Plant	Plant-X001-Transaction	Failure Details	Varchar(20)
6	Production Plant	Plant-X001-Transaction	Failure -Date Time	Date-Time

FIG . 2A

202

Sl. No#	Attributes	Functions	From clause	Where Clause	Group By Clause	Having Clause	2nd attribute for Join	Inner Join	Left Outer Join	Right Outer Join	Cross Join	2nd Table	Union	Intersection	Inner Clause	Output - Example
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	Select X from T
2	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	Select F(X) from T Group By X1

FIG . 2B

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

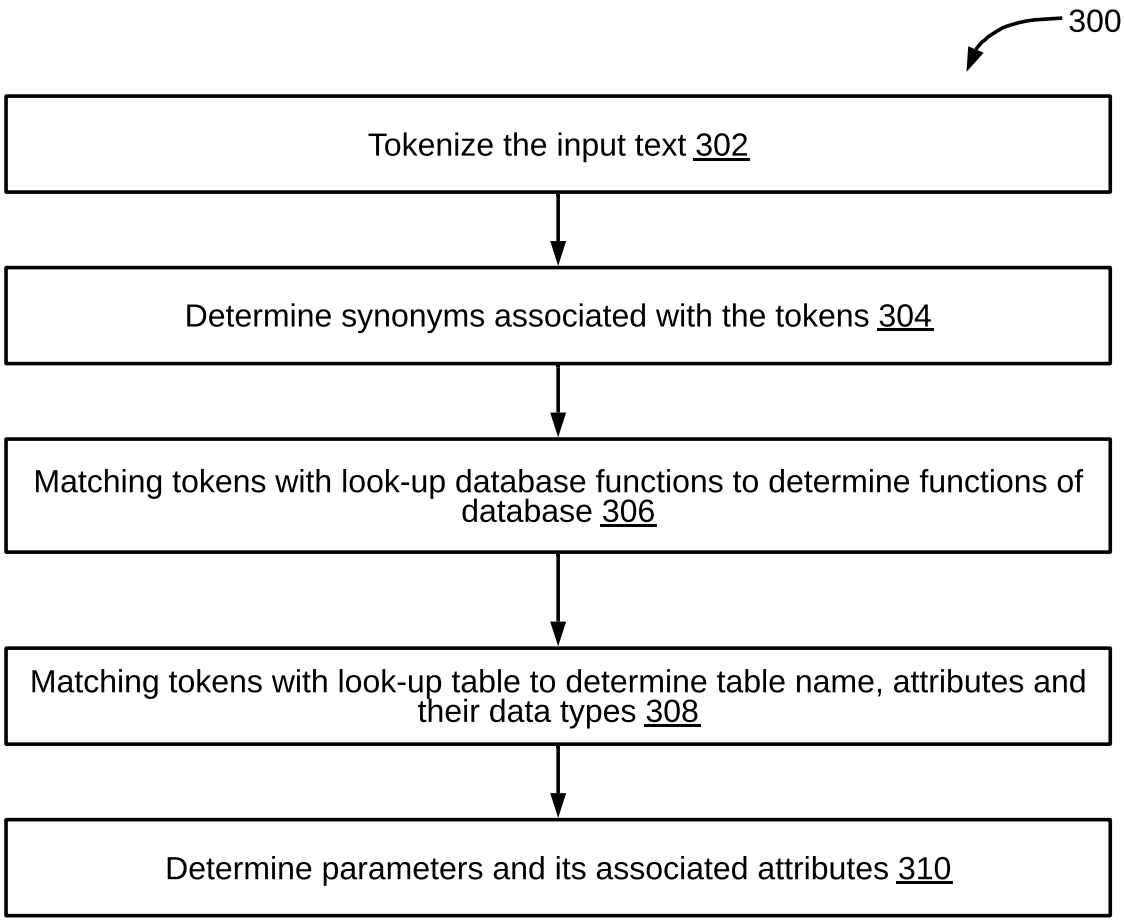


FIG. 3

Mohammed Faisal (INPA No: 1941)
Head, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.