January 2023

Integrating UI Elements into Custom SQL Code

Joseph Albert F. S. Pingenot

Integrating UI Elements into Custom SQL Code

ABSTRACT

Database queries in the structured query language (SQL) comprise text characters. While professional programmers are comfortable with text-based programming, database users (who may not be proficient programmers) are generally more comfortable with graphical query interfaces that reduce or eliminate the need for coding with textual characters. This disclosure describes SQL syntax and corresponding parser and backend infrastructure that enables database users to effectively use graphical elements in SQL queries by automatically providing the location to inject a value from a user interface control (such as the text selected, or the number in the slider or combo box) and by causing the GUI to create a control for that value.

KEYWORDS

- Structured query language (SQL)
- Query UI
- Query parser
- Query execution
- Parameterized query
- Distributed database
- Graphical user interface (GUI)
- Log analytics
- Abstract symbol tree (AST)

BACKGROUND

Database queries in the structured query language (SQL) comprise text characters. While professional programmers are comfortable with text-based programming, database users (who

may not be proficient programmers) are generally more comfortable with graphical query interfaces that reduce or eliminate the need for coding with textual characters.

Some portions of a SQL query lend themselves naturally to graphical elements. For example, many SQL queries include a clause such as ‘WHERE field=value,’ where value is one of a finite number of choices or a numerical range (e.g., time, temperature, etc.). A finite number of choices can be selected from a dropdown list. A numerical range can be selected from a user interface element such as a slider, a knob, etc. This usually requires populating graphical elements prior to the user writing the query. This limits what the user can query and is inconvenient, resulting in effectively unusable graphical controls in the user interface.

DESCRIPTION

This disclosure describes SQL syntax and corresponding parser and backend infrastructure that enables database users to effectively use graphical elements in SQL queries by automatically providing the location to inject a value from a user interface control (such as the text selected, or the number in the slider or combo box) and by causing the GUI to create a control for that value.

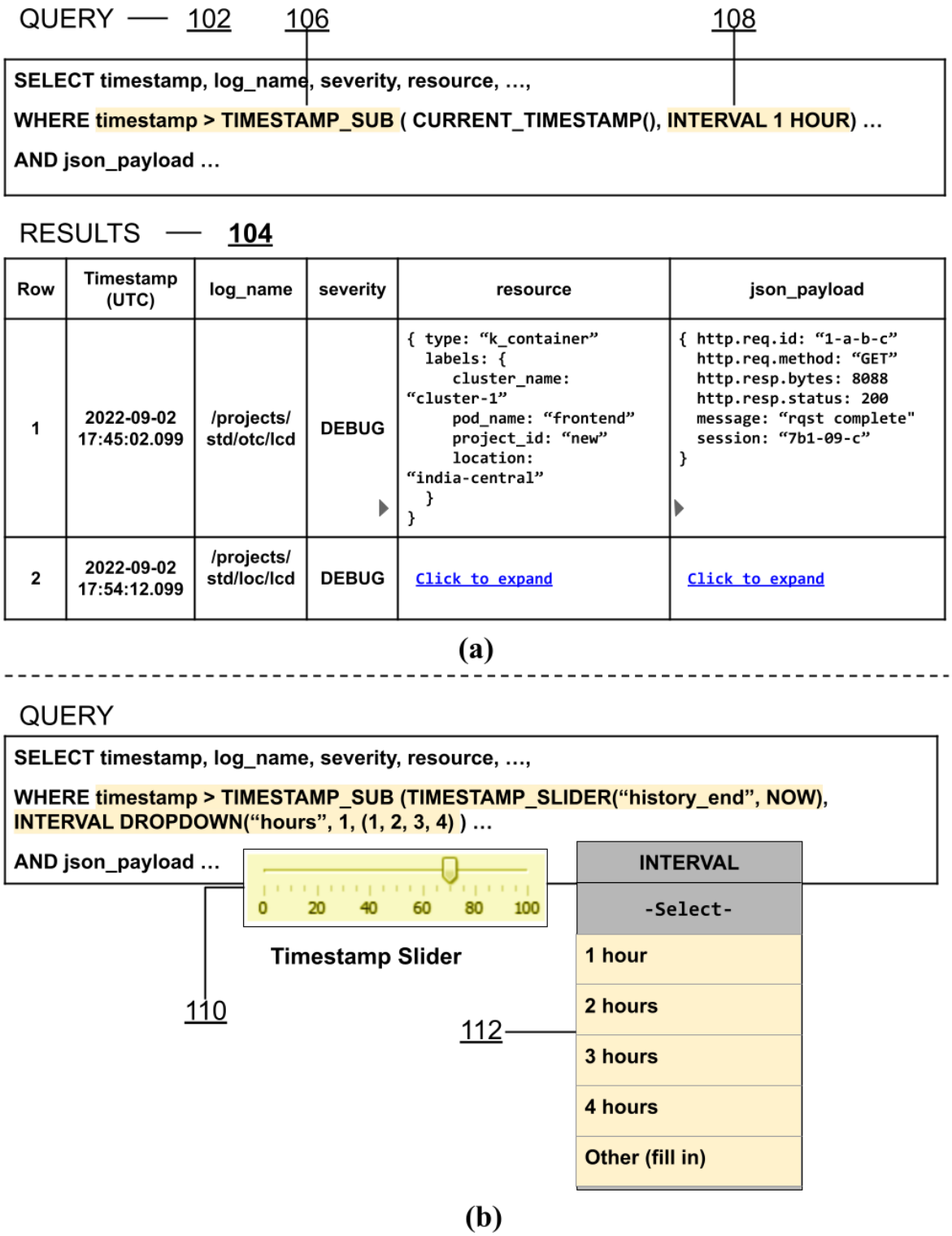

Fig. 1: Integrating UI Elements into Custom SQL Code: (a) A conventional query-result pair; (b) A query augmented with graphical control elements

Fig. 1 illustrates integrating UI elements into custom SQL code. Fig. 1(a) illustrates a conventional SQL query (102), as input by a database user, and the corresponding result (104), as returned by the database. The SQL query is text-heavy, but has certain fields, e.g., a field describing a numerical range specified by a 'WHERE field=value' clause (106), a field with a finite number of choices (108), which lend themselves to control by graphical elements.

Per the techniques of this disclosure, as illustrated in Fig. 1(b), fields amenable to control by graphical elements are augmented in the query input interface. For example, the database user can be presented with user interface elements to simplify providing values for certain fields. In Fig. 1(b), a slider (110) can be used to specify the timestamp and a dropdown menu (112) can be used to specify the interval. The UI controls enable the user to graphically set query fields. The database user is given a syntactic facility to specify that certain query fields be controlled graphically. For example, text such as 'slider/my_slider/1/10' in the query can be a signal that the field that precedes the text is controllable by a slider named 'my_slider' with a range between 1 and 10.

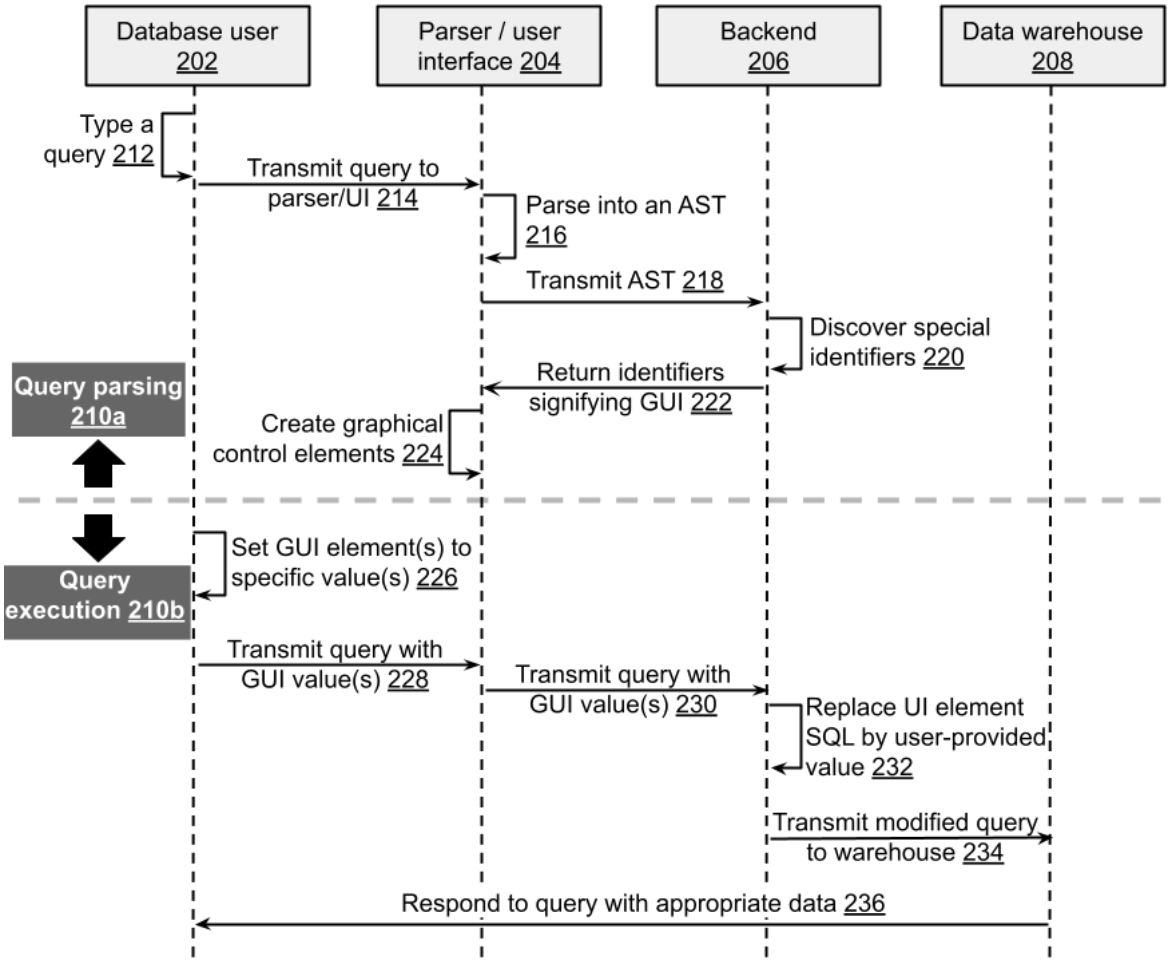

Fig. 2: An example mechanism to integrate UI elements into custom SQL code

Fig. 2 illustrates an example mechanism to integrate UI elements into custom SQL code. A database user (202) types a query (212). In a query-parsing stage (210a), the query is parsed (216) into an abstract syntax tree (AST) by a parser (204), which can be part of a user interface (UI) module. Parsing (214) can be done, e.g., in JavaScript, TypeScript, etc., or it can be done via a server API call.

The AST is transmitted (218) to the backend (206) and traversed to discover (220) special identifiers in the query such as ‘slider/my_slider/1/10.’ Alternatively, a custom function such as ‘slider(`my_slider`, 1, 10),’ e.g., with function-name ‘slider,’ UI-element name ‘my_slider,’ and arguments ‘1’ and ‘10’ may be found. These identifiers and/or functions describe controls that

the UI can present (in this example, a slider in the range 1 through 10, inclusive). The backend collects and canonizes elements signifying graphical control and returns them to the UI (222), possibly through intermediate layers. Using a web application framework, e.g., Angular, the UI creates graphical control elements (224) based on the canonized elements returned by the backend.

In a query execution stage (210b), the database user can manipulate the graphical elements (e.g., slider, dropdown menu, knob, etc.) presented within the SQL query user interface to set query fields to specific values (226). Query execution can be done in JavaScript (or Typescript) or via an API call. The query, including the GUI element names and values, is transmitted to the parser/ user-interface (228). For example, if the user selected '3' in 'my_slider,' then the relevant part of the SQL query includes 'my_slider: 3.' The SQL query with graphical elements and values reaches the backend (230), where it is resolved, e.g., the UI element SQL is replaced with the value of the UI component provided by the user (232), e.g., 'slider(`my_slider`, 1, 10)' is replaced with "3". The modified query is sent (234) to the data warehouse (208), which responds (236) with data relevant to the query. Alternatively, the query is returned to the user for debugging. The described techniques can be used in log analytics, distributed or cloud-based databases, or other applications that support structured queries.

CONCLUSION

This disclosure describes SQL syntax and corresponding parser and backend infrastructure that enables database users to effectively use graphical elements in SQL queries by automatically providing the location to inject a value from a user interface control (such as the text selected, or the number in the slider or combo box) and by causing the GUI to create a control for that value.