Query Builder Documentation

Project Completed – 17th January 2017

Version 1.0

Authors-

1. Aditya Deshpande

2. Alisha Shahane

3. Darshana Gadre

4. Mrunmayi Deshpande

Table of Contents

1. Basic Classes

1.1 Main class-

- Sets “Nimbus” look and feel for the UI
- Creates object of Menu class which is the entry point of the code

1.2 MyConnection class-

- Contains a static method – connectDB() for establishing connection to the database as set in Menu class.

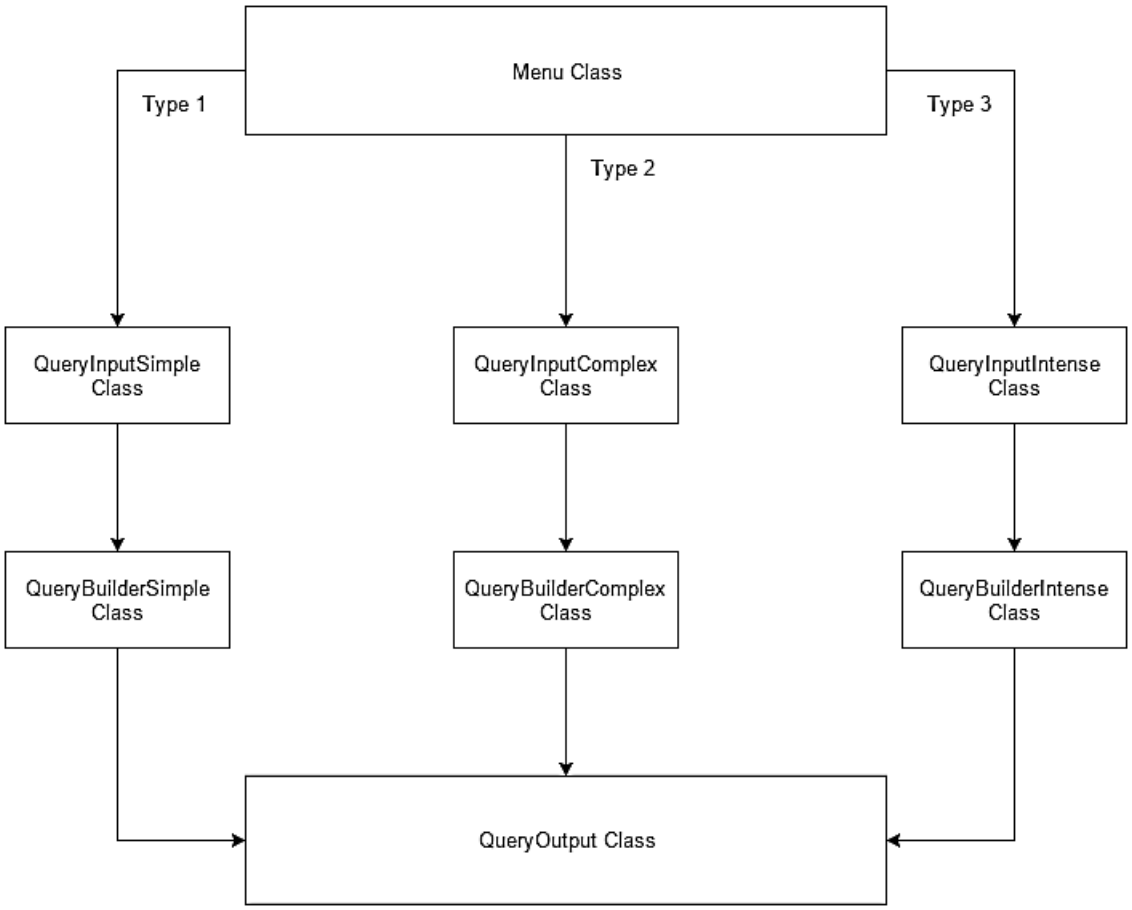
1.3 HeaderRenderer class-

- This class centrally aligns the table headings.

2. Menu class- (JFrame)

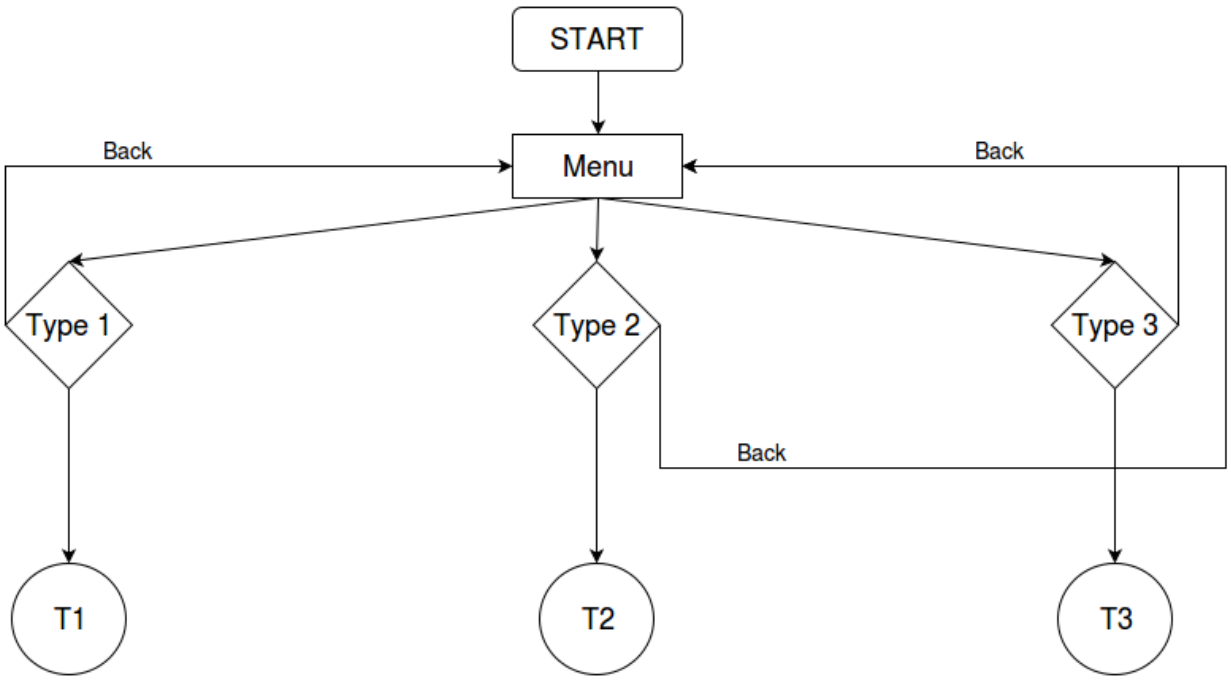
- Options for 3 different types of queries
- The static String db_name of MyConnection class is set to appropriate database name according to the type of query builder selected.
- Type 1 - single database contains variety of tables all with different schemas
    1. Can query single table at a time
    2. Builds normal query (multiple conditions separated by 'AND', 'OR' connectors)
    3. Builds nested query for example:
       (Column1 < 123 AND Column2 = 'abc') OR Column3 != 45
    4. Builds date query (if table has a column of datetime datatype then querying a particular date range). The specified data range is treated as a condition in where clause.
    5. This type of query builder appends all conditions specified by user to the where clause.
    6. The query output contains all columns of selected table.
- Type 2 – single database contains tables which have same schema and are updated at certain time interval (for example – a table for each month), and an index table which contains metadata of these tables.
    1. User does not specify a table name. Instead he specifies a date range (compulsorily) according to which tables which contain data for that date range are selected automatically at the backend.
    2. Rest of the working similar to Type 1. Additionally a single column can also be selected without any conditions for displaying.
- Type 3 - single database contains tables which have same schema and are updated at certain time interval (for example – a table for each month), and an index table which contains metadata of these tables.
    1. User does not specify a table name. Instead he specifies a date range (compulsorily) according to which tables which contain data for that date range are selected automatically at the backend.
    2. Unlike previous 2 types, this query builder has only aggregate functions for generating reports (for example – max(column1), min(column2), avg(column3) etc)
    3. Here user specifies a time interval in hours according to which the date range is sliced and each time slice is queried using threading.
    4. The output contains one row for each time slice.

3. Architectural Overview Diagram

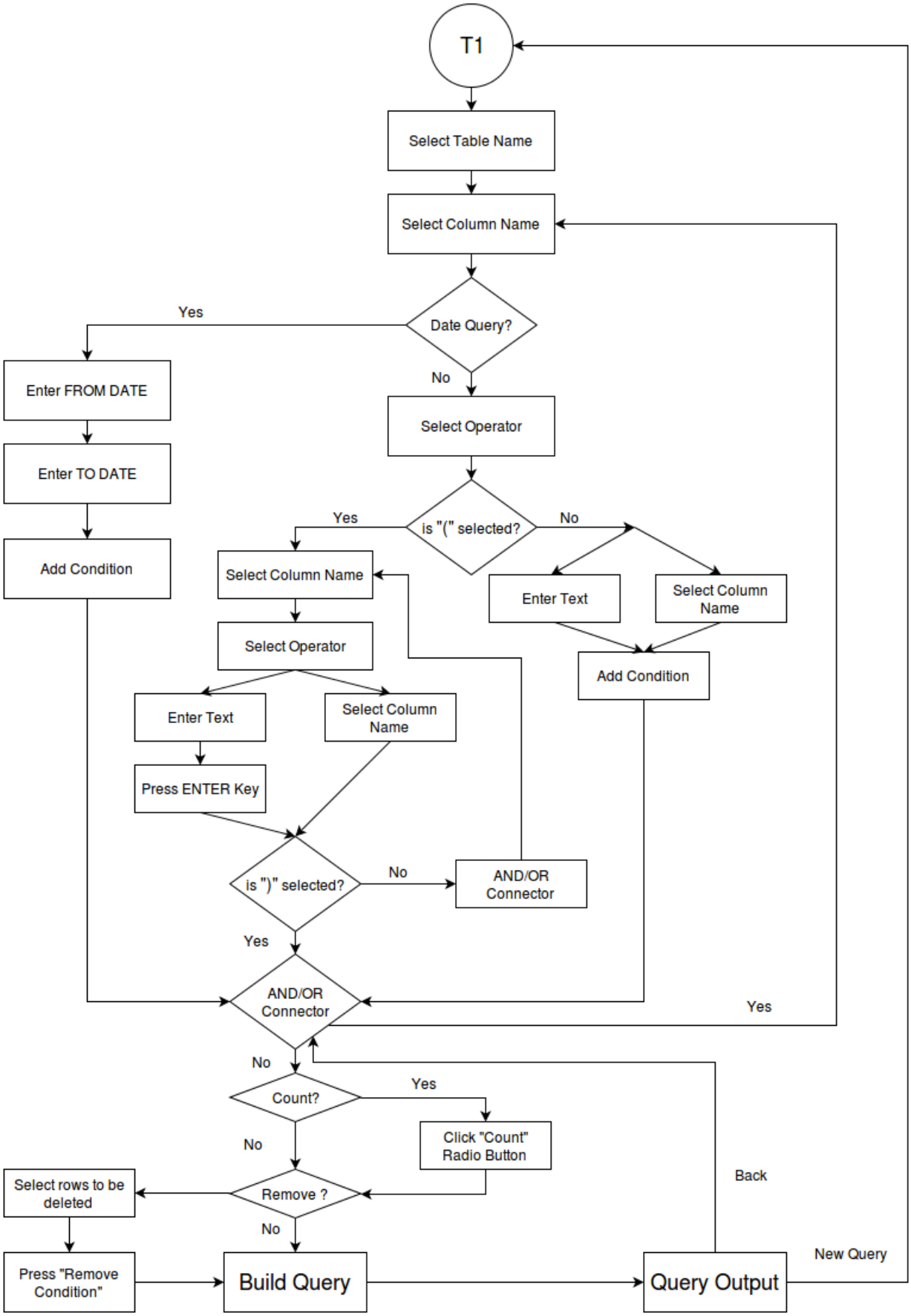


4. Flow Diagrams

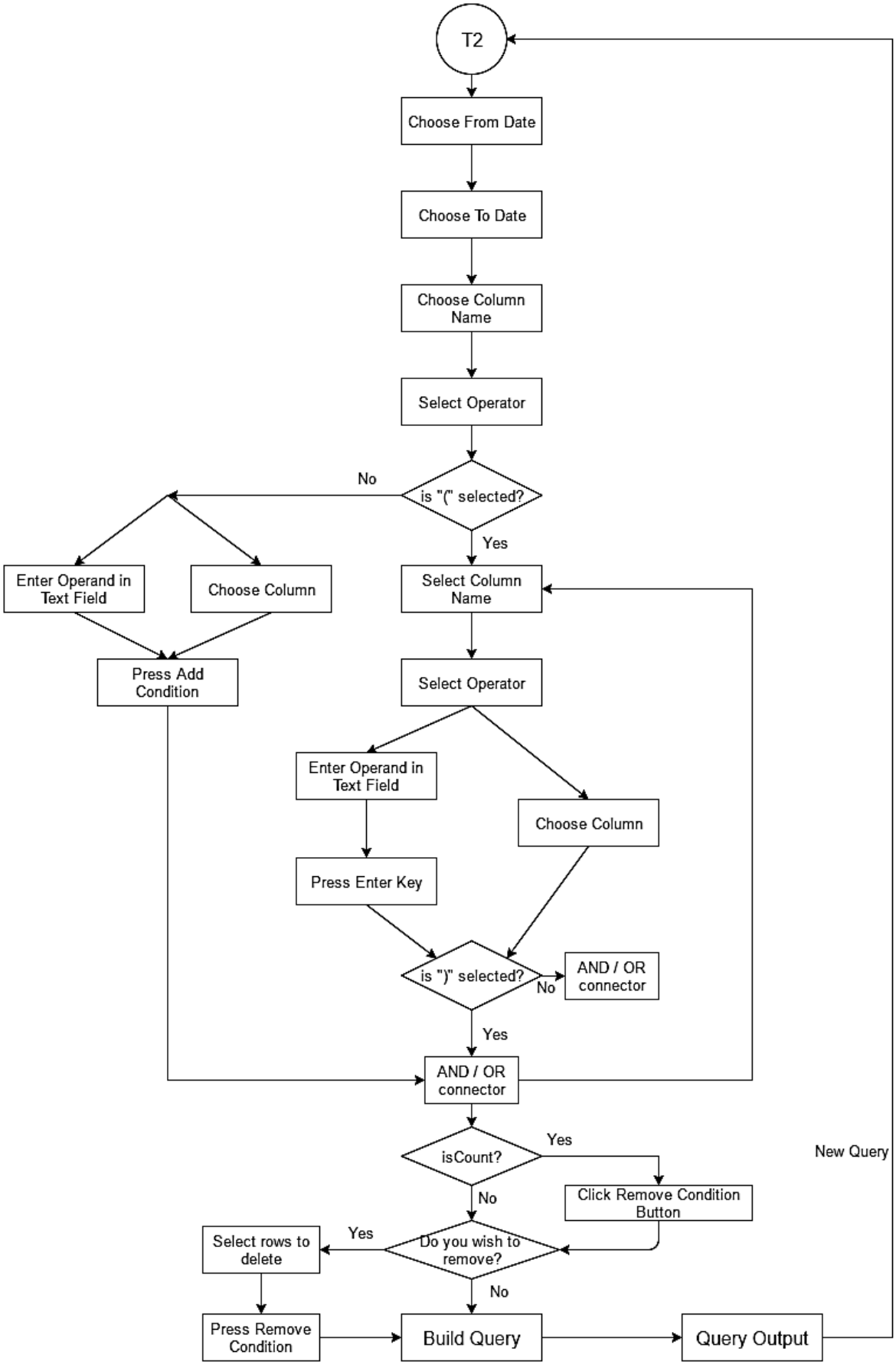
4.1 Menu



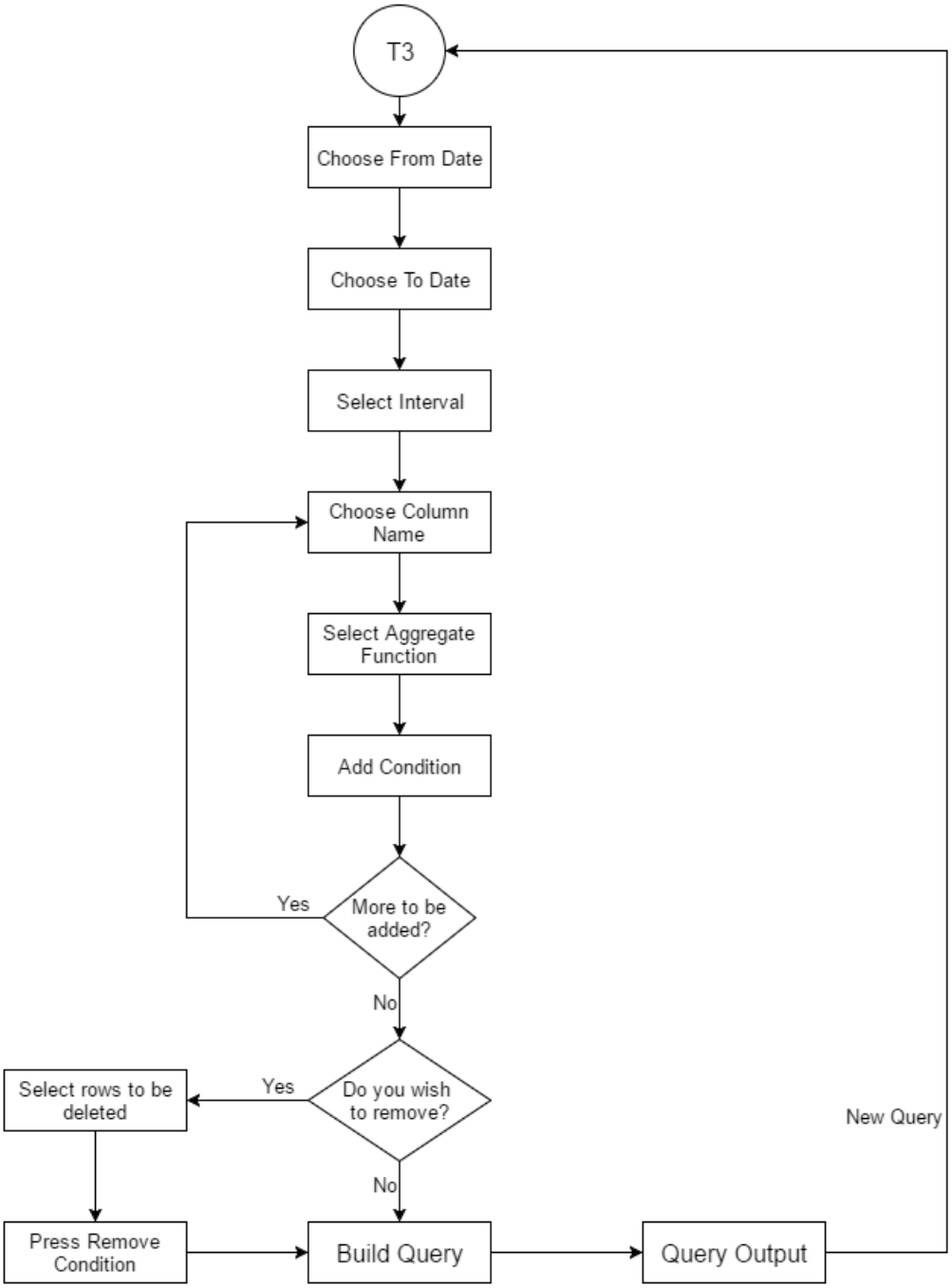
4.2 Type 1 QueryBuilder



4.2 Type 2 QueryBuilder

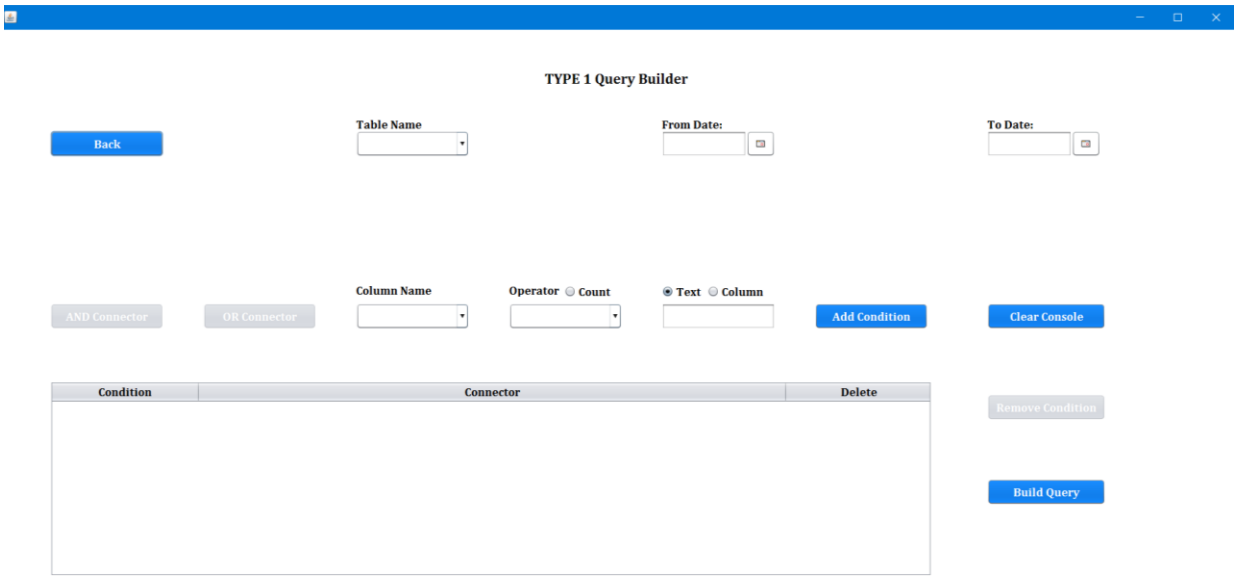


4.3 Type 3 QueryBuilder



5. QueryInputSimple class- (JFrame)

- This class only handles the GUI part for Type 1 queries.
- It initializes the UI components and creates an instance of QueryBuilderSimple class to handle the backend working.
- This class contains all the action listener declarations for the various UI components but the actual logic is written in methods of QueryBuilderSimple class.



6. QueryBuilderSimple class-

- This class contains the logic for actually building Type 1 queries.

6.1 Basic Methods

6.1.1 **void** getConsole()
// Function for getting user inputs from console

6.1.2 **void** setConsole()
// Function for setting combo box values on console
-Sets the operator combo box(hard coded).

-Sets the column and table combo boxes dynamically.

6.1.3 DefaultTableModel setTable()
// Function for initializing table model

6.1.4 **boolean** isDate()
// Function to determine whether user is specifying all inputs for date query
-Stores the entered dates in the form of a string.

6.2 Methods for constructing queries

6.2.1 **void** buildQuery()
// Constructing and executing final query
-If count radio button is selected, it fires a query to count the rows in the database which match given conditions.

-Creates the where clause for the query by appending the rows in the table.

-Fires the query and send result set to QueryOutput.

6.2.2 String buildSubQuery()
// Function for constructing where clause of the sub queries
-Creates a subquery for min/max operators and adds it as a row in the table.

-Example: select * from tb_name where col_name=(select max(col_name) from tb_name)

6.2.3. String buildDateQuery()

// Function for constructing where clause of the date query

-Creates a datequery for specified date range and adds it as a row in the table.

-Example: select * from tb_name where col_name between fromDate and toDate

6.3 Methods for handling combo box events

6.3.1 **void** cbx_columnActionPerformed(ActionEvent arg0)

// Column CBX

-In case of nested query dynamically adds any selected column to table.

-Validations/Error Handling

-Popups/highlight to guide the user.

6.3.2 **void** cbx_operatorActionPerformed(ActionEvent arg0)

// Operator CBX

-Sets ifNested and isSubquery flags according to operator selected.

-In case of nested query dynamically adds any selected operator to table.

-Validations/Error Handling

-Popups/highlight to guide the user.

6.3.3 **void** cbx_colopActionPerformed(ActionEvent arg0)

// Colop CBX

-In case of nested query dynamically adds any selected column to table.

-Validations/Error Handling

-Popups/highlight to guide the user.

6.4 Methods for handling radio button events

6.4.1 **void** rdb_textStateChanged(ChangeEvent arg0)

// Text RDB

-Used when user wants to compare a column to an operand of his choice

-Example: col_name >= 90

-Hides colop combo box and enables operand text field.

6.4.2 **void** rdb_colopStateChanged(ChangeEvent arg0)
// Colop RDB

-Used when user wants to compare a column with another column

-Example: col_name1 >= col_name2

-Hides operand text field and enables colop combo box.

6.5 Methods for handling text field events

6.5.1 **void** tf_operandActionPerformed(ActionEvent arg0)
// Operand TF

-In case of nested query dynamically adds any entered operand to table.

-Popups/highlight to guide the user.

6.5.2 **void** tf_operandKeyPressed(KeyEvent arg0)
//Operand TF

-Popups/highlight to guide the user.

6.6 Methods for handling button click events

6.6.1 **void** btn_addConditionActionPerformed(java.awt.event.ActionEvent evt)
// Add Condition Button

-Validation/Error Handling

-In case of nested query dynamically sets delete check box 'false' in the table.

-In case of date query buildDateQuery() function is called. It adds a row only if it is the first row in the table.

-In case of subquery buildSubQuery() function is called. It adds a row only if it is the first row in the table.

-In case of normal query, column_name + operator + colop/operand is added to table.

6.6.2 **void** btn_andActionPerformed(java.awt.event.ActionEvent evt)
// AND Button

-In case of nested query dynamically adds 'and' to table.

-If not nested creates a new row with AND connector.

-Popups/highlight to guide the user.

6.6.3 **void** btn_orActionPerformed(java.awt.event.ActionEvent evt)
// OR Button

-In case of nested query dynamically adds 'or' to table.

-If not nested creates a new row with OR connector.

-Popups/highlight to guide the user.

6.6.4 **void** btn_rmConActionPerformed(java.awt.event.ActionEvent evt)
// Remove Condition Button

-Removes the rows where the delete checkbox is selected.

6.6.5 **void** btn_clearActionPerformed(java.awt.event.ActionEvent evt)
// CLEAR Button

-Creates a new instance of QueryInputIntense.

6.6.6 **void** btn_executeActionPerformed(java.awt.event.ActionEvent evt)
// Build Query Button

-Calls the buildQuery() function.

-Hides the current JFrame.

6.7 Miscellaneous Methods

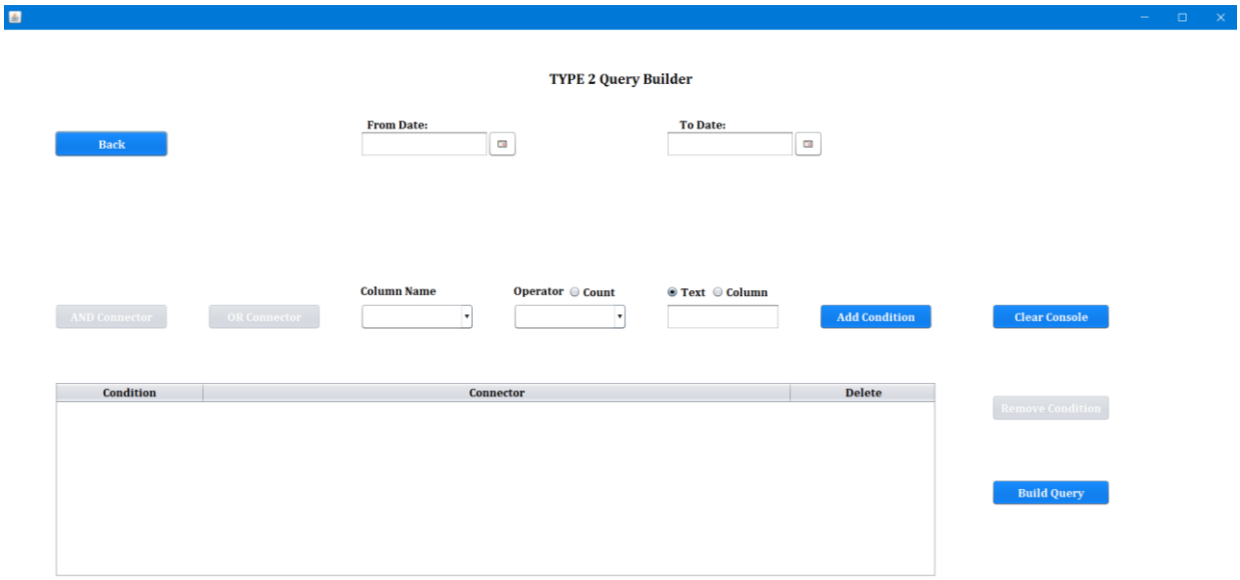
6.7.1 **void** mouseMoved(MouseEvent arg0)
// Mouse Moved on Frame

-Sets the toDate equal to fromDate in the beginning.

-Ensures toDate >= fromDate

7. QueryInputComplex class- (JFrame)

- This class only handles the GUI part for Type 2 queries.
- It initializes the UI components and creates an instance of QueryBuilderComplex class to handle the backend working.
- This class contains all the action listener declarations for the various UI components but the actual logic is written in methods of QueryBuilderComplex class.



8. QueryBuilderComplex class-

- This class contains the logic for actually building Type 2 queries.

8.1 Basic Methods

8.1.1 **void** getConsole()
// Function for getting user inputs from console

8.1.2 **void** setConsole()
// Function for setting combo box values on console

8.1.3 DefaultTableModel setTable()
// Function for initializing table model

8.1.4 **boolean** getDate()
// Function for getting the FROM, TO dates and tables corresponding to them

-Converts the user entered date in the JDatePicker to epoch timestamps.

-Fires an index query on the database to get table names which are to be queried according to date range specified.

-Also contains error handling if no data is available for given date range.

8.2 Methods for constructing queries

8.2.1 **void** buildQuery()
// Constructing and executing final query

-In case of simple query, fires a query and sends result set to QueryOutput.

Example: select col_name from tb_name

-If count radio button is selected, it fires a query to count the rows in the database which match given conditions.

-Creates the where clause for the query by appending the rows in the table.

-Fires the query and send result set to QueryOutput.

8.2.2 String buildSubQuery()
// Function for constructing where clause of the sub queries

-Creates a subquery for min/max operators and adds it as a row in the table.

-Example: select * from tb_name where col_name=(select max(col_name) from tb_name)

8.3 Methods for handling combo box events

8.3.1 **void** cbx_columnActionPerformed(ActionEvent arg0)
// Column CBX

-In case of nested query dynamically adds any selected column to table.

-Popups/highlight to guide the user.

8.3.2 **void** cbx_operatorActionPerformed(ActionEvent arg0)
// Operator CBX

-In case of nested query dynamically adds any entered operator to table.

-Validations/Error Handling

-Popups/highlight to guide the user.

8.3.3 **void** cbx_colopActionPerformed(ActionEvent arg0)
// Colop CBX

-In case of nested query dynamically adds any selected column to table.

-Popups/highlight to guide the user.

8.4 Methods for handling radio button events

8.4.1 **void** rdb_textStateChanged(ChangeEvent arg0)
// Text RDB

-Used when user wants to compare a column to an operand of his choice

-Example: col_name >= 90

-Hides colop combo box and enables operand text field.

8.4.2 **void** rdb_colopStateChanged(ChangeEvent arg0)
// Colop RDB

-Used when user wants to compare a column with another column

-Example: col_name1 >= col_name2

-Hides operand text field and enables colop combo box.

8.5 Methods for handling text field events

8.5.1 **void** tf_operandActionPerformed(ActionEvent arg0)
// Operand TF

-In case of nested query dynamically adds any entered operand to table.

-Popups/highlight to guide the user.

8.5.2 **void** tf_operandKeyPressed(KeyEvent arg0)
//Operand TF

-Popups/highlight to guide the user.

8.6 Methods for handling button click events

8.6.1 **void** btn_addConditionActionPerformed(java.awt.event.ActionEvent evt)
// Add Condition Button

-Validation/Error Handling

-In case of nested query dynamically sets delete check box 'false' in the table.

-In case of date query buildDateQuery() function is called. It adds a row only if it is the first row in the table.

-In case of subquery buildSubQuery() function is called. It adds a row only if it is the first row in the table.

-In case of normal query, column_name + operator + colop/operand is added to table.

8.6.2 **void** btn_andActionPerformed(java.awt.event.ActionEvent evt)
// AND Button

-In case of nested query dynamically adds 'and' to table.

-If not nested creates a new row with AND connector.

-Popups/highlight to guide the user.

8.6.3 **void** btn_orActionPerformed(java.awt.event.ActionEvent evt)
// OR Button

-In case of nested query dynamically adds 'or' to table.

-If not nested creates a new row with AND connector.

-Popups/highlight to guide the user.

8.6.4 **void** btn_rmConActionPerformed(java.awt.event.ActionEvent evt)
// Remove Condition Button

-Removes the rows where the delete checkbox is selected.

8.6.5 **void** btn_clearActionPerformed(java.awt.event.ActionEvent evt)
// CLEAR Button

-Creates a new instance of QueryInputIntense.

8.6.6 **void** btn_executeActionPerformed(java.awt.event.ActionEvent evt)
// Build Query Button

-Calls the buildQuery() function.

-Hides the current JFrame.

8.7 Miscellaneous Methods

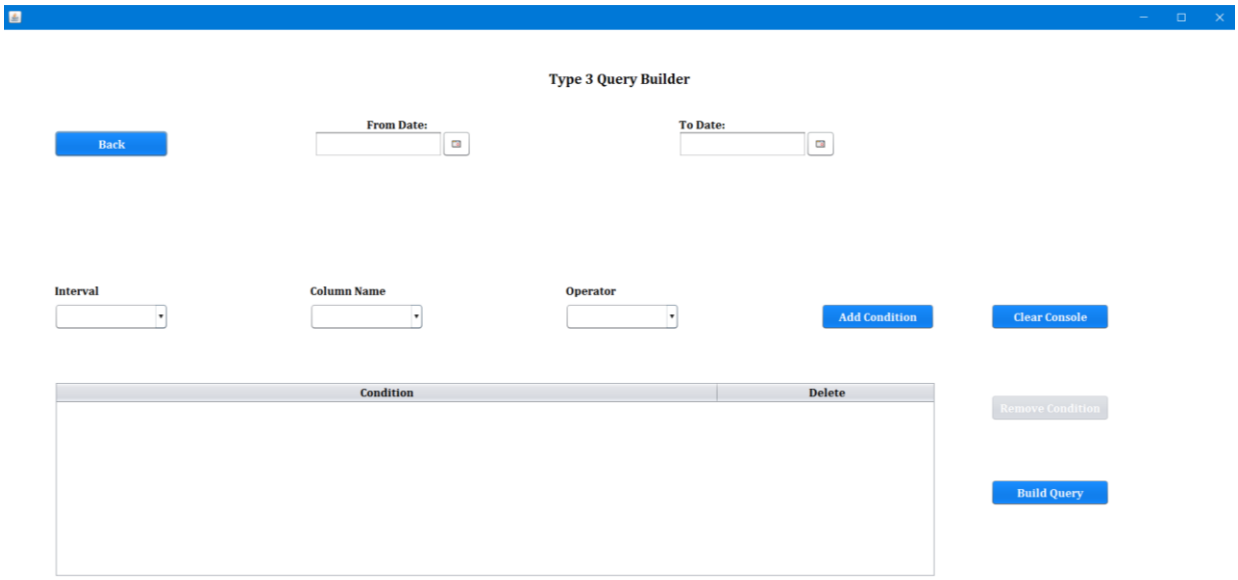
8.7.1 **void** mouseMoved(MouseEvent arg0)
// Mouse Moved on Frame

-Sets the toDate equal to fromDate in the beginning.

-Ensures toDate >= fromDate

9. QueryInputIntense class- (JFrame)

- This class only handles the GUI part for Type 3 queries.
- It initializes the UI components and creates an instance of QueryBuilderIntense class to handle the backend working.
- This class contains all the action listener declarations for the various UI components but the actual logic is written in methods of QueryBuilderIntense class.



10. QueryBuilderIntense class-

- This class contains the logic for actually building Type 3 queries.
- This class implements Runnable interface for multi-threading.

10.1 Basic Methods

```
10.1.1 void getConsole()
// Function for getting user inputs from console

10.1.2 void setConsole()
// Function for setting combo box values on console

10.1.3 DefaultTableModel setTable()
// Function for initializing table model

10.1.4 boolean getDate()
 // Function for getting the FROM, TO dates and tables
corresponding to them
     -Converts the user entered date in the JDatePicker to epoch
     timestamps.

     -Fires an index query on the database to get table names
     which are to be queried according to date range specified.

     -Also contains error handling if no data is available for
     given date range.
```

10.2 Methods for constructing queries

```
10.2.1 void buildQuery()
// Constructing and executing final query
     -The select clause is formed by appending the rows of the
     table.

     -Based on interval, fromDate, toDate chunks are created and
     where clause is formed for each chunk based on timestamp
     column.

     -A thread is created to query each chunk, and all threads
     are joined and the final result set vector is sent to
     QueryOutput.


10.2.2 void queryChunks(String query)
// Every thread executes this method to query a single chunk
```

10.3 Methods for handling combo box events

10.3.1 **void** cbx_columnActionPerformed(ActionEvent arg0)
// Column CBX
Popups/highlight to guide the user.

10.3.2 **void** cbx_operatorActionPerformed(ActionEvent arg0)
// Operator CBX
Popups/highlight to guide the user.

10.3.3 **void** cbx_intervalActionPerformed(ActionEvent arg0)
// Interval CBX
Popups/highlight to guide the user.

10.4 Methods for handling button click events

10.4.1 **void**
btn_addConditionActionPerformed(java.awt.event.ActionEvent evt)
// Add Condition Button
-Error handling for validations.

-Adding row to the table: operator + (ColumnName)

10.4.2 **void** btn_rmConActionPerformed(java.awt.event.ActionEvent
evt)
// Remove Condition Button
-Removes the rows where the delete checkbox is selected.

10.4.3 **void** btn_clearActionPerformed(java.awt.event.ActionEvent
evt)
// CLEAR Button
-Creates a new instance of QueryInputIntense.

10.4.4 **void**
btn_executeActionPerformed(java.awt.event.ActionEvent evt)
// Build Query Button
-Calls the buildQuery() function.

-Hides the current JFrame.

10.5 Miscellaneous Methods

```
10.5.1 void mouseMoved(MouseEvent arg0)
// Mouse Moved on Frame
     -Sets the toDate equal to fromDate in the beginning.

     -Ensures toDate >= fromDate


10.5.2 public void run()
// Overridden for runnable interface
     -Calls he queryChunks() function.
```

11. QueryOutput class- (JFrame)

- Displays output of the built query in JTable.
- This class contains 3 constructors for each type of query builder. Constructor signature-

  (`ResultSet rs, String query, QueryInput qi`)

  ```
  -rs contains the output of the query

  -query is the final query executed on the db

  -qi is reference of the JFrame class (for back button)
  ```

- This class contains 2 methods for filling the JTable as QueryBuilder types 1 & 2 have a single result set but output of type 3 is a vector of result sets.
- Back button handling – closes the current JFrame and makes calling JFrame visible
- NewQuery button handling – closes the current JFrame and creates a new instance of calling JFrame