International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:1, 2015

SQL Generator Based On MVC Pattern

Chanchai Supaartagorn

***Abstract*—**Structured Query Language (SQL) is the standard de facto language to access and manipulate data in a relational database. Although SQL is a language that is simple and powerful, most novice users will have trouble with SQL syntax. Thus, we are presenting SQL generator tool which is capable of translating actions and displaying SQL commands and data sets simultaneously. The tool was developed based on Model-View-Controller (MVC) pattern. The MVC pattern is a widely used software design pattern that enforces the separation between the input, processing, and output of an application. Developers take full advantage of it to reduce the complexity in architectural design and to increase flexibility and reuse of code. In addition, we use White-Box testing for the code verification in the Model module.

***Keywords*—** MVC, relational database, SQL, White-Box testing.

I. Introduction

RELATION database is a set of tables containing data where there are connections between the data stored in one table and data stored in other tables, all of which is formally described and organized according to the relational model. Nowadays, a database plays an important role for every business. It is a tool for decision making and planning to achieve objectives or goals. Many businesses regularly make database transactions to store, query, sort and display information, as well as keeping records via a Relational Database Management System (RDBMS).

Typically, database application development is often done with SQL commands. The Structure Query Language (SQL) is a standard language for specifying access and modification to relational databases. SQL is a relatively simple language, but it's also very powerful. However, one of the difficult issues in database manipulation is coding SQL commands. Indeed, several studies suggest that traditional database query language is not very simple to use for users unskilled in database technologies, as a consequence of the fact that the interaction is based on a textual language such as SQL [1].

One way to solve the problem is to develop a tool that provides libraries of SQL commands for database management based on a framework. A framework usually implements the Model View Controller (MVC) pattern. The MVC pattern breaks an application into three associated modules: Model, View and Controller. The Model module is the business logic of the application and is the core of the application [2]. It contains the underlying classes. The View is the user interface of the controller. It is the public face of the user event's response [3]. The Controller component implements the flow of control between the View and the Model [4]. The MVC pattern is interesting to research because its simplicity makes it more acceptable to developers. In addition, MVC helps to reduce the complexity in architectural design and to increase flexibility and reuse of code [5].

This research proposed a tool for generating SQL commands based on the MVC design model, which will create an effective separation of event handling, underlying classes and user interface. The tool provides a graphical user interface for generating SQL commands easily. Users can view the SQL commands and the data sets in a simple format simultaneously on a local web page to facilitate selecting, filtering, sorting, grouping and joining data without knowing SQL commands. The system was developed with PHP language, working in a web-based application. In addition, the prototype of the system is available with MySQL database only. In future research, we plan to develop a system that supports multiple RDBMS.

The rest of this paper is organized as follows: Section II describes the related works. Section III presents the structural design of the SQL generator. Section IV discusses the Model testing. Section V shows the implementation of the SQL generator based on this framework. Section VI draws the conclusions and proposals for future research.

II. Related Work

SQL is a standard language which is a classical and powerful tool for querying relational databases. Although SQL is simple syntax, it is difficult for novice users to learn and understand the concepts, especially when considering its more complex syntax. There is some research identifying the reasons for these difficulties. For example, [6] noted that student experience many problems when learning SQL being: the burden of having to memorize database schemas and understand the syntax of SQL. While [1] found that SQL is very hard for unskilled users or novice users, who are not familiar with the syntax query language. While [7] discovered that many learners are often misleading of SQL commands. Furthermore, the declaration nature of SQL is very complex to grasp. And [8] identified that the most important part of learning and the practical application of SQL requires problem-solving abilities. We solve the difficulties by creating a tool, which automatically generates the SQL commands with a simple graphical interface and maps out each action to a specified command for the users. The unskilled users or novice users can use this tool without knowing SQL syntax.

Nowadays, there are many tools for managing MySQL databases that provide an intuitive interface. For example, phpMyAdmin [9] is a free software tool, intended to handle the administration of MySQL over the web. It supports a wide

Asst. Prof. Chanchai Supaartagorn is with the Department of Mathematics Statistics and Computer, Faculty of Science, Ubon Ratchathani University, Ubonratchathani 34190 Thailand (phone: 6681-853-0650; e-mail: chanchai.s@ubu.ac.th).

range of operations on MySQL such as, creating, modifying or deleting database and tables. SQLyog [10] is a GUI tool that provides a powerful means to manage MySQL database. Moreover, it creates a complex queries using a drag and drop interface. Navicat [11] is a graphical database management system. Its advantage is the connect multiple, database administration tool, allowing users to connect MySQL, Oracle PostgreSQL, SQLite, and SQL server and is available on three platforms – Microsoft Windows, Mac OS and Linux. However, these tools are not suitable for users who yet need to learn SQL statement, especially the learning of the complex SQL statement.

In addition, there are many techniques of the tools that help a novice user to learn and comprehend the SQL syntax. For example, the SQL query by iconic metaphor [1] helps unskilled users to understand and learn the SQL syntax while composing simple visual queries. The queries can be formulated by simple clicks and the drag and drop of icons representing the elements of the database. The SQL query by Tree structure [12] helps users build the automatic generation of SQL queries through the Web. SQL queries may be constructed using a tree-like structure and are submitted to the SQL application server only when complete and validated syntactically, as well as semantically. In these researches, the system uses a simplified interface for supporting learning and understanding of SQL syntax. However, there is a problem in the development phase and the maintenance phase. The system mixes the code of access, the processing of business logic, and user interface layers together. We aim to alleviate the problem by creating a tool based on the MVC pattern. The advantages of the pattern are: 1) Loosely-coupled: many kinds of components can interact in a flexible way; 2) Parallel development: the duty is clear and it is possible to partition the whole system in to components so that different person could develop at the same time. The structure is clear so as to easily integrate and maintain; 3) Expandability: Controller can expand with the module; 4) Reusable quality: it can improve the reusable quality by encapsulated the business logic in the component [13]. In summary, we present SQL generator; a system that uses an intuitive interface for supporting the learning and the understanding of SQL languages. The main features are: Firstly, it uses a simple interface to operate the SQL queries and converts them into SQL commands immediately. The advantage is of helps users to eliminate common SQL mistakes. Secondly, we present the structural design of a framework based on the MVC pattern. The tool can be implemented easily, especially when maintaining the system in future work.

III. The Structural Design of the SQL Generator

In this section we describe the basic concept of the MVC pattern that was used for creating the SQL generator tool. In addition, we design the Model module that contains the underlying classes for manipulating the SQL commands.

A. The Architecture of the SQL Generator

The MVC pattern breaks an application into three modules: Model, View and Controller. Fig. 1 illustrates the architecture of SQL generator based on MVC pattern.

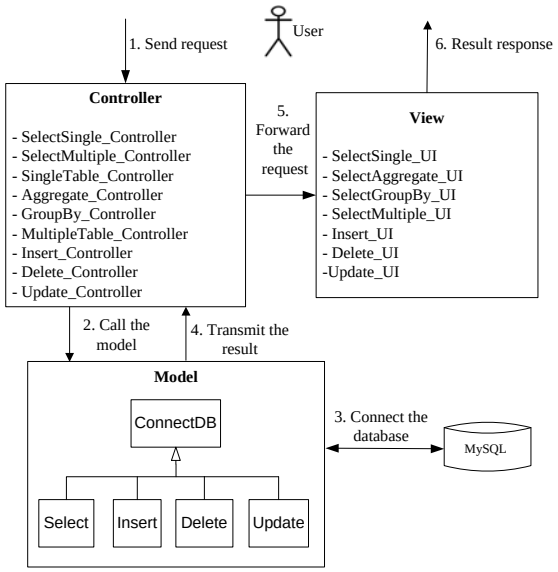

Fig. 1 Overall system architecture of the SQL generator

The Model module contains the underlying classes that are the core of the application. There are five classes in the model: ConnectDB class, Select class, Insert class, Delete class and Update class. Firstly, the ConnectDB class is used to connect the hosting and select the database. Second, the Select class is used to create Select statement in single table and multiple stables, Aggregate Function statement and Group by statement. Third, the Insert class is used to create Insert statement. Fourth, the Delete class is used to create Delete statement. Finally, the Update class is used to show the record that matches with the condition specified. In addition, it creates Update statement.

The View module is the user interface of the controller. It is the public face of the user event's response [3]. Developers can design a view with HTML, cascading style sheets (CSS), Javascript, etc. There are seven interfaces in the View module: SelectSingle_UI, SelectAggregate_UI, SelectGroupBy_UI, SelectMultiple_UI, Insert_UI, Delete_UI and Update_UI.

The controller module contains code to handle the user action and invoke changes in the model. There are nine parts in the controller: SelectSingle_Controller, SelectMultiple_Controller, SingleTable_Controller, Aggregate_Controller, GroupBy_Controller, MultipleTable_Controller, Insert_Controller, Delete_Controller, and Update_Controller.

The operational process of the architecture can be broken down into six steps as follows: 1) A user sends a request to the Controller 2) The Controller analyses the request and calls the Model 3) The Model will perform the necessary class and connect the MySQL database 4) The Model transmits the result to the Controller 5) The Controller forwards the request to the View and 6) The request is complete when the result responds for the user.

B. The Design of Model

We use a class diagram to illustrate the Model module. A Class diagram is a graphical model that shows the relationship between classes and shows what attributes reside in the class. They are useful for showing how models work. The Class diagram of the module is shown in Fig. 2.

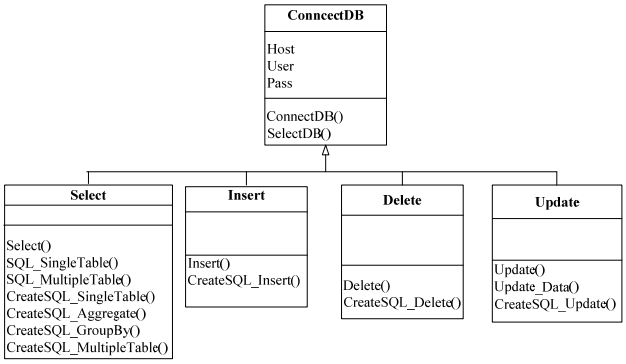

Fig. 2 Class diagram showing the Model module of SQL generator

The ConnectDB class is superclass or parent class. There are three attributes (host, user and pass) and two methods (ConnectDB() and SelectDB()). The ConnectDB() method is the constructor that automatically executes at the time of object instantiation. It is used to initialize the host name attribute, user name attribute and password attribute. The SelectDB() method is used to connect the hosting and select of the database.

The Select class is a subclass or child class that inherits from the ConnectDB class. There are seven methods: The Select() method is the constructor that initializes the table name for the select statement. The SQL_SingleTable() method and SQL_MultipleTable() method is used to retrieve the field name inside the single table and multiple tables respectively. The CreateSQL_SingleTable() method and CreateSQL_MultipleTable() method is used to create the SQL statement in single table and multiple table respectively. The CreateSQL_Aggregate() method is used to create the aggregate function statement. The CreateSQL_MultipleTable() method is used in conjunction with the aggregate functions to group the result set. In addition, the Select class also inherits the attributes and method from its superclass.

The Insert class is a subclass or child class that inherits from the ConnectDB class. There are two methods: The Insert() method is the constructor that initializes the table name for the insert statement. The CreateSQL_Insert() method is used to create the insert statement. In addition, the Insert class also inherits the attributes and method from its superclass.

The Delete class is a subclass or child class that inherits from the ConnectDB class. There are two methods: The Delete() method is the constructor that initializes the table name for the delete statement. The CreateSQL_Insert() method is used to create the delete statement. In addition, the Delete class also inherits the attributes and method from its superclass.

The Update class is a subclass or child class that inherits from the ConnectDB class. There are three methods: The Update() method is the constructor that initializes the table name for update statement. The Update_Data() method is used to show the record that matches with the condition specified. The CreateSQL_Update() method is used to create the update statement. In addition, the Update class also inherits the attributes and method from its superclass.

IV. The Solution for Model Testing

We use White-Box testing to analyses code in the Model module. White-Box testing can examine the design documents and the code, as well as observing algorithms and their internal data [14]. Branch/Decision coverage technique is one of several techniques for White-Box testing. This testing aims to ensure that each possible branch from each decision point is executed at least once. We show an example of CreateSQL_SingleTable() method to test the quality of software below.

```
function CreateSQL_SingleTable($field_value,$condition='',$order='')
{       $sql = "Show Columns From ".$this->table[0].";";
        $result = mysql_query($sql);
        $count = 0;
        $count_field = 0;
        While($dbarr = mysql_fetch_row($result))
        {     if ($field_value[$count] == true)
                    $count_field++;
              $count++;                  }
        $count_index = 0;
        $comma = 0;
        $sql = "Show Columns From ".$this->table[0].";";
        $result = mysql_query($sql);
        $sql_comm = "Select ";
        While($dbarr = mysql_fetch_row($result))
        {    $field[$count_index] = $dbarr[0];
             if ($field_value[$count_index] == true)
             {   $sql_comm = $sql_comm.$field[$count_index];
                 $comma++;
                 if ($count_field != $comma)
                    $sql_comm = $sql_comm.", ";                  }
             $count_index++;
        } // end while
            $sql_comm = $sql_comm."<br>";
            if (($condition=='') AND ($order==''))
               $sql_comm = $sql_comm." From ".$this->table[0].";";
            elseif (($condition!='') AND ($order==''))
            {  $sql_comm = $sql_comm." From ".$this->table[0]."<br>";
               $sql_comm = $sql_comm." Where ".$condition.";";             }
            elseif (($condition=='') AND ($order!=''))
            {  $sql_comm = $sql_comm." From ".$this->table[0]."<br>";
               $sql_comm = $sql_comm." Order By ".$order.";";             }
            else
            {  $sql_comm = $sql_comm." From ".$this->table[0]."<br>";
               $sql_comm = $sql_comm." Where ".$condition."<br>";
               $sql_comm = $sql_comm." Order By ".$order.";";           }
               return $sql_comm;
} //end function CreateSQL_SingleTable
```

To help to do this systematically, we draw a control flow graph of the code, as shown in Fig. 3.

This graph has a shade node representing the eight decisions (A, B, C, D E, F G and H) where the code can make the nine branches (paths 1, 2, 3, 4, 5, 6, 7, 8 and 9). We created an employee database and employee table for this testing. The details of the employee table are shown in Table I.

We devised a test case to make sure that every decision and branch was taken. The following tests in Table II ensure Branch/Decision coverage.

From the test case, we can conclude 100% decision coverage and 100% branch coverage. We have used this same testing with the rest of the methods. All reachable code in the method is executed.

TABLE I
THE DETAILS OF EMPLOYEE TABLE

employeeID	name	job	salary	departmentID
1111	Somchai	Programmer	15000	128
2222	Wichit	DBA	13500	42
3333	Somjai	Programmer	16500	128
4444	Aphitsit	System Administrator	12000	130
4445	Youngyut	Programmer	20000	128

TABLE II
TEST CASE OF CREATESQL_SINGLETABLE() METHOD

Test Case	$field_value	$condition	$order	$this->table[0]	Decision	Branch
1	$field_value = array("0"=> "employeeID", "1"=> "name", "2"=> "job", "3"=> "salary")	Null	Null	Employee	A,B,C,D,E,F	1,2,3,4,5,6
2	$field_value = array("0"=> "employeeID", "1"=> "name", "2"=> "job", "3"=> "salary")	$condition = "salary>=15000"	Null	Employee	A,B,C,D,E,F,G	1,2,3,4,5,7
3	$field_value = array("0"=> "employeeID", "1"=> "name", "2"=> "job", "3"=> "salary")	Null	$order = "name ASC"	Employee	A,B,C,D,E,F,G,H	1,2,3,4,5,8
4	$field_value = array("0"=> "employeeID", "1"=> "name", "2"=> "job", "3"=> "salary")	$condition = "salary>=15000"	$order = "name ASC"	Employee	A,B,C,D,E,F,G,H	1,2,3,4,5,9

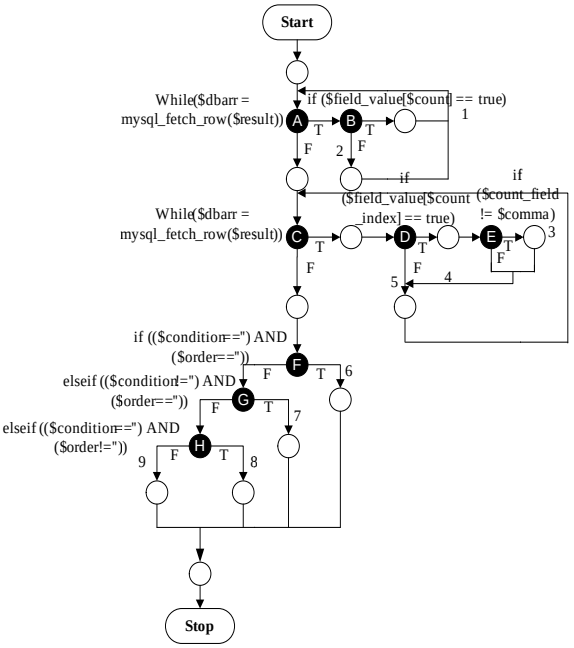

Fig. 3 The control flow graph of CreateSQL_SingleTable() method

V. SYSTEM IMPLEMENTATION

In order to use the SQL generator in an easy way, in our proposed system, major consideration must be given to the layout of the user interface. When a user runs the SQL generator through a web browser, the system prepares the tools for the View module and Controller module and loads the layout of the user interface. The interface has two main components: the database manager and the tabbed panels.

Users must enter the MySQL database details (username and password). Then, the database manager will show in the left-hand frame of the system. This is used for database navigation. Users will see the database name of the system. Tables will also show up under database. The database manager interface is shown in Fig. 4.

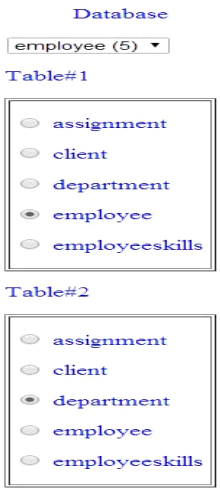

Fig. 4 The interface of the database manager

Fig. 5 The interface of tabbed panels

There are eight parts of the tabbed panels: Select SingleTable, Select Aggregation, Select GroupBy, Select MultipleTable, Insert, Delete Update and Exit. We show an example of Select SingleTable that creates the simple SQL syntax of single table. We now click the field name that wants to retrieve from the database. Then we click the condition options to extract only records that fulfill a specified criterion. After that, we click the ordering options to sort the result-set in ascending order or descending order. In the last step, we click the Create SQL button to convert user interface action to the proper SQL syntax. The tabbed panels interface is shown in Fig. 5.

We show an example of select from single table: List employeeID, name, job and salary which salary is greater than or equal to 15000 and order the result by employeeID in ascending order. The corresponding SQL and output data automatically generates as shown in Figs. 6 and 7 respectively.

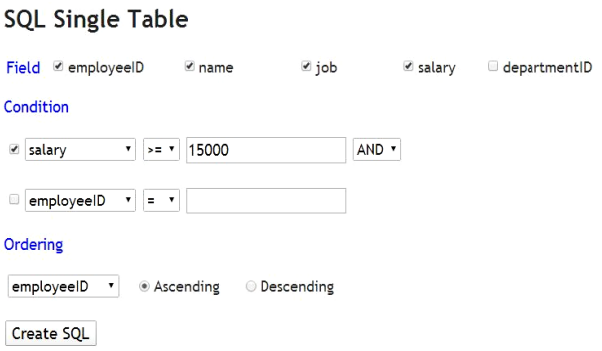

Fig. 6 The example of select from single table

```
Select employeeID, name, job, salary
From employee
Where salary >= 15000
Order By employeeID Asc;
```

employeeID	name	job	salary
1111	Somchai	Programmer	15000
3333	Somjai	Programmer	16500
4445	Youngyut	Programmer	20000

Fig. 7 SQL syntax and output data

VI. Conclusions and Future Research

The Structure Query Language (SQL) is the set of instructions used to manipulate with a relational database. Learning SQL is an important step in developing the application. However, the crucial problem in database manipulation is coding the SQL commands. SQL is not very simple to learn and use for novice users. They often have trouble with SQL syntax. To remedy this problem was the main objective of this research. The SQL generator is proposed, which is capable of supporting the teaching and understanding of the SQL syntax. It can create SQL commands and display the data sets simultaneously. Moreover, its implementation was based on the MVC pattern. The MVC pattern breaks the system into three associated modules: Model, View and Controller. Each of these components handles a discreet set of tasks. The MVC design pattern is a well-established and compelling approach to building software. The developers lay out, like separation of content from display. The SQL generator takes full advantage of its loosely-coupled, expandability and reusable quality.

In addition, we can use the White-Box testing to examine the Model module in order to guarantee the system tool quality. Lastly, we show the system implementation and an example of SQL commands that were created from the SQL generator.

In future research, we will include refinement and enrichment of the Model, View and Controller. The system will cover complex SQL syntax and increased functionality. In addition, we will provide a function for connecting to various databases, such as Oracle, SQL Server and so on.

References

[1] Aversano L, Canfora G, De Lucia A, and Stefanucci S, "Understanding SQL through Iconic Interfaces, " in *Computer Software and Applications Conference*, Oxford, England, 2002, pp. 703–708.
[2] J. Li, G. Ma, G.Feng, and Y. Ma, "Research on Web Application of Struts Framework based on MVC pattern," in *International Workshop on Web-Based Internet Computing for Science and Engineering*, 2006, pp. 1029–1032.
[3] Armando Padilla, "Beginning Zend Framework," New York, 2009, pp. 55.
[4] Karam M, Keirouz W, and Hage R, "An abstract Model for Testing MVC and Workflow Based Web Applications," in *International Conference on Telecommunications and International Conference on Internet and Web Applications*, Guadeloupe, 2006, pp. 206–212.
[5] W. Cui, L. Huang, L. Liang, and J. Li, "The Research of PHP Development Framework Based on MVC Pattern," in *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, Korea, 2009, pp. 947–949.
[6] A. Mitrovic, "A Knowledge-Based Teaching System for SQL," in *World Conference on Educational Multimedia Hypermedia and Telecommunications*, 1998, pp. 1027–1032.
[7] S. Sadiq, M. Orlowska, W Sadiq, and J. Lin, "SQLator – An Online SQL Learning Workbench," in *Proceedings of the 9th annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2004, Leeds, UK, pp. 223–227.
[8] G. Russel, and A. Cumming, "Improving the student learning experience for SQL using automatic marking," in *IADIS International Conference Cognition and Exploratory Learning in Digital Age*, 2004, Lisbon, Portugal, pp. 281–288.
[9] PhpMyAdmin, On-line. Available from Internet, http://www.phpmyadmin.net, accessed 2 June 2014.
[10] SQLyog, On-line. Available from Internet, http://www.webyog.com, accessed 2 June 2014.
[11] Navicat, On-line. Available from Internet, http://www.navicat.com, accessed 2 June 2014.
[12] N. Rishel, K.Naboulsi, O. Wolfson, and B. Ehlmann, "An Efficient Web-based Semantic SQL Query Generator," in *Proceedings 19th IEEE International Conference on Distributed Computing Systems*, 1999, Texas, U.S.A., pp. 23–30.
[13] Z. Yanqiu, and C. Chuan, "Designing JSP/Servlet+EJB Web Applications Based on MVC Design Mode," in *Computer Engineering*, 2001, pp. 71–73.
[14] Timothy C, and Robert L, "Object-Oriented Software Engineering," McGraw-Hill, 2005.

Chanchai Supaartagorn is currently a lecturer and an Assistant Professor at the Department of Mathematics Statistics and Computer, Faculty of Science, Ubon Ratchathani University, Ubonratchathani, Thailand. He received a M.Sc. in Computer Science, Mahidol University, Thailand, 1998. B.Sc. in Computer Science, Kasetsart University, Thailand, 1995. His research interests include Model-View-Controller (MVC), Business Data Model, E-Commerce and Web-Based Technology.