Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.6, No.6, 2015
www.iiste.org

MSGI: MySQL Graphical Interface

Dr. Ismail Al-Rawi
Arab Open University, (Kuwait Branch), P.O 830 Al Ardia, P.C 92400, Kuwait
*E-Mail: ism_49@hotmail.com

Abstract
The current development in data processing technology and the current revolution in office information systems require a very efficient database end user interface capable of satisfying the requirements of the increasing spectrum of end user. Studies in this respect have shown that the graphical specification for such an interface is more successful than the linear and the natural language specifications.In this paper, **MSGI** (MySQL Graphical Interface) design and implementation are presented. It is an easy to use and easy to learn graphical database end user interface, running on the PC with any platform, to provide a comprehensive facility for accessing databases managed and controlled by MySQL database management systems.
Keywords: Graphical End-User Interface, Database End-User Interface, MySQL Graphical Interface, Graphical Database Interface.

1. Introduction

The growth of computer technology and the acceptance of computer methods in simple daily life depend largely on the successful establishment of effective man machine communications (Al-Rawi, 1986). Therefore in recent years, the user has become the prime factor of system design, the screen instead of being peripheral devices, have become the mirror which reflects the whole service (Al-Rawi, 1998). MSGI syntax is very wide and covers most of the manipulation and the updating operations that are required by most relational database management systems. The syntax and the database structure are presented to the user on the screen in a very natural form, so that the user can formulate his query easily depending on the menu-selection technique, and assisted by the instruction statements and the different feedback methods (Al-Rawi, 2009). MSGI provides the user with the facility to reformat the accessed data and for report generation as well.

2. Motivations of using graphics techniques in MSGI

(1) Experimental studies on graphical interfaces have shown that the user is more successful in formulating his query in a graphical language (Gravell, 2010). A great deal of information can often be shown more precisely, accurately and clearly in a two dimensional syntax rather than in one.
(2) Restricting the representation to a specific graphical form instead of allowing any number of words and phrases will help in making the graphical language a finite state grammar. This will eliminate many of the anomalies and complexities of the non-finite state grammar, English.
(3) Fewer typing errors will occur because only a very small portion of any query expressed by graphical representation need be entered at the keyboard.
(4) Procedural and ordered manner of phrasing will be a very much reduced. This is a very important matter in achieving the easy to use property. Generally, procedural and ordered expressions are found in linear format specifications.
(5) The graphics is also easy to use in other language (French. Japanese. Arabic, etc.), rather than English (Natural language).
(6) Given the preceding five points and the absence of the other points of difficulties existing in English language based implementation. One can expect that the implementation effort is much reduced over the effort necessary to implement a natural language interface.

3. Interface Objectives

Before designing the interface the following primary goals were' set:

(1) Simplicity.
(2) Ease of use.
(3) Minimum training.
(4) Portability across different DBMSs.
(5) Cover most of the facilities of the underlying database languages.
(6) Suitability for use by all levels of user.

In addition to the above primary objectives, several properties have become apparent during the implementation of the interface:

(1) Friendliness to the casual users. By using the graphics facilities and different kinds of feedback, the interface provides the user with a pleasant environment in which to work.

(2) It provides another way of employing the interactive graphics technique to facilitate the access of the database, other than the ways which have been used in other graphical interfaces.

(3) It provides the users with a new look to the database structure. The database is presented to the user in a very simple, natural form. The user is entirely separated from the logical as well as the physical representation of the database. He is not required to remember any relation or attribute names or their relationship. They can all be displayed to him on the screen under his control.

(4) The user of MSGI is entirely separated from the database language and its complications. He is not concerned with how the data will be accessed, or about the formal representation of the query, or the DBMS. This is the main property of the unified interface. MSGI is not only suitable to be used by the entire spectrum of end-users, but also provides the same facilities to access databases controlled by different DBMSs (Al-Rawi. 2009).

(5) Extensibility: the generality of the interface, and the separation of the user and the interface from the underlying DBMS and the database language, makes it very easy to include other DBMS to operate underneath the interface. This is done simply by adding its analyzer to the system.

(6) It provides interactive graphical methods for accessing a large database which may reside on a large scale computer system from a minicomputer.

(7) MSG users avoid a lot of typing and its inherent problems.

(8) The output subsystem adds another useful facility to the interface, and extends the use of the accessed data for further processing.

(9) It offers flexibility in query formulation, no matter what order is chosen to select relation or attribute names in any particular level of the query formulation. For instance, in Figure-1, if PART has to be joined with SUPPLIER in order to project PNAME and SNAME. It does not matter whether PART or SUPPLIER relation is selected first. Also the selection order of PNAME and SNAME makes no difference. This is one of the characteristics of an easy to use interface.

PART

P#	PNAME	COLOUR	WEIGHT	CITY

SUPPLIER

S#	SNAME	STATUS	CITY

Figure-1

4. MSGI Components

MSGI components are the hardware devices, software utilities, and the user interaction. These components operate together to form an integrated interface capable of achieving specified objectives and showing certain properties. Figure-2 shows the main components of **MSGI**.

(1) User: the user is any person who utilizes the interface for creating, retrieving, and updating the database. It does not matter whether he has used the computer directly or indirectly, or even if he has done any computer programming. The user is the element which activates the interface system. Unlike other computer programs which can be run in the absence of the user interaction, in MSGI the user is actually part of the system: his interaction with the system must continue until he gets the answer of his request.

(2) PHP Engine: this utility is represented by a set of interactive graphics routines and devices attached to the used programming language (PHP). It provides a sketching facility and interaction means between the user and MySQL.

(3) Operating system: It represents the utility routines, operations controlling the attached devices, compilers and microcodes. These work jointly to manage and control the machine and run the interaction facility between the real world (user. user program) and the database management systems through **MSGI**.

(4) Interface screen: it is the two dimensional visual part of the interface, which represents a work area where the real world is represented in the form of relations and attributes. In this area actions on the simulated real-world take place through the query formulation process. The user monitors the success of his interactions with the MSGI in this area through the output result and error massages.

(5) Analyzer: The analyzer is the main and most complicated part of the system. It receives the query primitives (**MSGI** representation) and starts analyzing, interpreting, and translating this representation into a formal representation of the specified database language (SQL). This representation of the query is then transmitted through PHP to MySQL (Chris, 2006), to be executed and accesses the data.

(6) Output-File-Manager (OFM): it is a software subsystem which receives the accessed data from MySQL, and allows the user to specify the form of the output he wants before displaying it on the screen (IDS, 1977).

The OFM is the topic of future work.

5. Screen Design

The screen is the most important component of any two dimensional language interfaces. It represents the communication bridge between the user and the system, and it is the means by which the idea of the interface is presented and through which it is implemented (Gali, 1982). Thus the design of the screen becomes one of the most important tasks the interface designer has to consider. A well-designed screen format can increase user processing speed, reduce user errors, and speed computer processing time. A poorly designed screen will have the opposite effect that is it will decrease user processing speed, provoke user mistakes, and complicate machine operations. A well-designed screen will increase user productivity, and a poorly designed one will degrade it.

Figure-3 shows the MSGI screen layout which is divided into five distinct areas. The main bulk of it represents the main display window which is located in the middle, and it has dimensions of (450 x 550) pixels. This window is defined by a GRAFIKS utility procedure call. Through this window the end user communicates with the system, the output results, error messages, and the formal query representation are displayed. The constant values are also entered into the system through this window. Any dialogue between the user and the system is carried out through this window. Size limitations of this window are avoided by using the scrolling technique. The output result can be scrolled up and down, left and right across the window. Thus, any part of the output can be viewed. Along the bottom of this window, there is a narrow rectangle containing four arrows in circles which indicate the scrolling direction. Scrolling is activated by pointing the cursor into any one of these circles and depressing the left button of the mouse.

The second area is the lower part of the screen; it is specified to contain an extensive commands and functions menu. Commands and functions are functionally grouped in this menu in order to be easily located. Elements of this menu represent the updating commands, ordering and group-by commands. ENTER and CLOSE commands and a number of built-in functions.

The third area represents the right hand side of the screen, where the relation. Names menu is located. This menu is not a static menu like the commands and functions menu, but is dynamic; the relation names in it can be scrolled up and down, by pointing to anyone of the two arrows in the control box at the bottom of it. And these relation names can be updated according to the status of the database. Below this menu there is a small box containing the READ command.

The fourth area is the upper part of the screen. It is specified to be occupied by three relation skeletons where each skeleton can contain up to eight attributes at any one time. It is possible to allow the name of each attribute in any skeleton to be scrolled left and right in order to display relations with higher degrees. The first cell in the skeleton represents the relation name. On the very top of the screen there is a one line instruction statement. A small box on the top right contains the asterisk character (*), the selection of which indicates that the whole relation attributes have to be projected.

The fifth area is the left hand side of the screen which contains three different menus; the top one contains all the arithmetic and logical operators, the one below contains the relational operators; UNION. INTERSECT, and DIFFERENCE. The bottom menu is a small window, containing the query elements which have been selected during the building operation of the query. These elements in the window can also be scrolled up and down to be viewed by the user. The purpose of this menu is to provide another way of query feedback to the user, which allows him to trace the query as a sequence of primitives displayed according to their selection order.

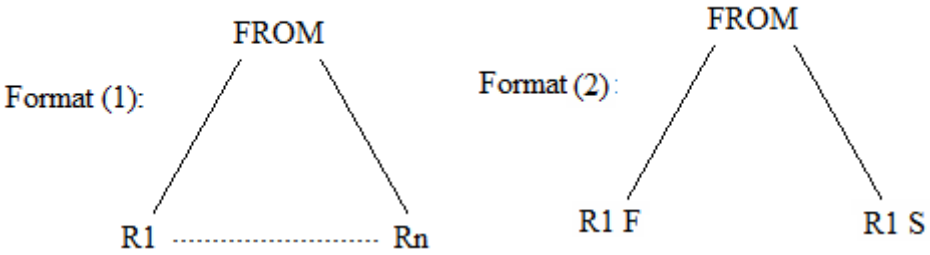
6. SQL Analyzer

The SQL analyzer consist of two sets of subroutines: the first set is concerned with the analysis and the translation of the retrieving queries to SQL representation, while the second set is integrated functionally with the first set of subroutines to perform the analysis and the translation of the updating queries.

The target of the analysis process for any retrieving query and most of the updating ones is achieved by the construction of the three basic clauses of the mapping block 'FROM-SELECT-WHERE'.

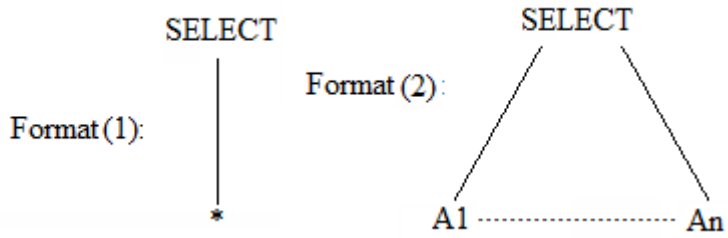
The syntax formats of each clause that are supported by SQL-analyzer are presented in form of binary trees:

(1) FROM Clause

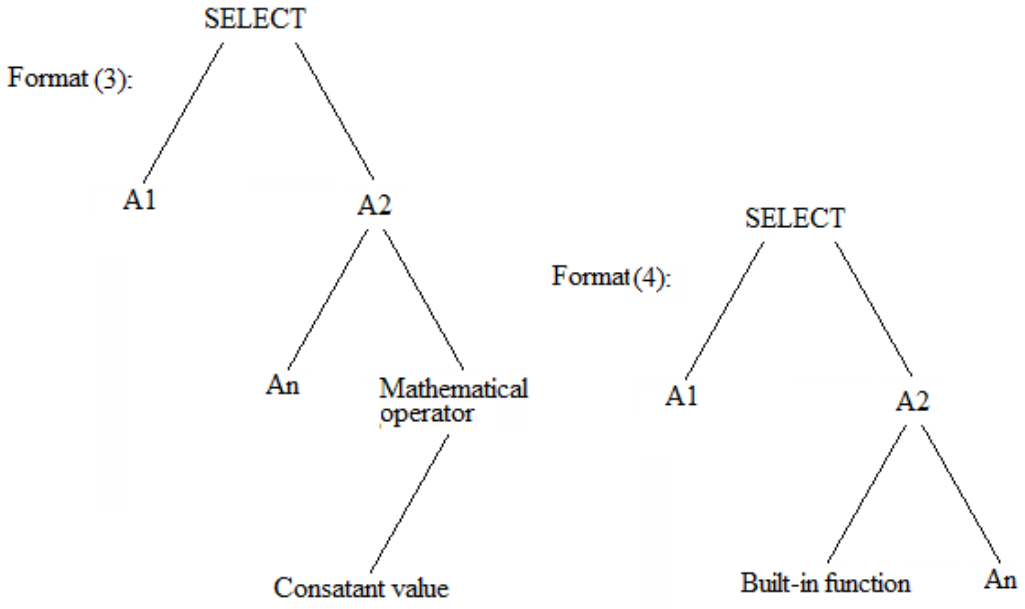


In this format (1) the maximum number of relations which can be selected at anyone level of the query is five, (n=1. 2, 5). This number is restricted by the number of viewports which can be created in the available memory. Format (2) represents a selection of a relation joined with itself.

(2) SELECT Clause



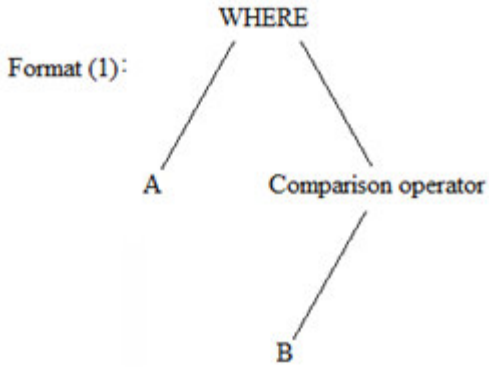
The asterisk in Format (1) means that all the attributes of the selected relations in FROM clause are to be projected.
In the current implementation of Format (2), up to eight attributes can be projected from the selected relations at anyone level of the query. This has to be extended to an unlimited number of attributes, after providing the left, right scrolling facility to the relation skeleton.



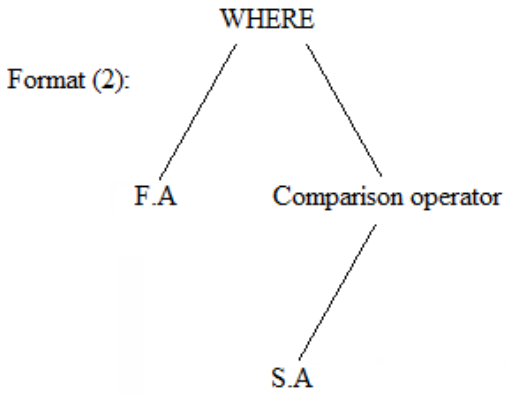
In Format (3) the last selected attribute may be applied to any mathematical operation.
Format (4) shows that, a built-in function can be applied to the last selected attribute.

(3) **WHERE Clause**

Format (1):

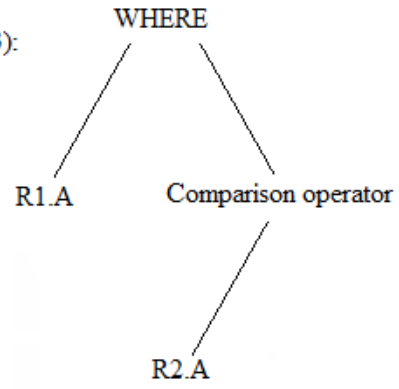


Format (2):

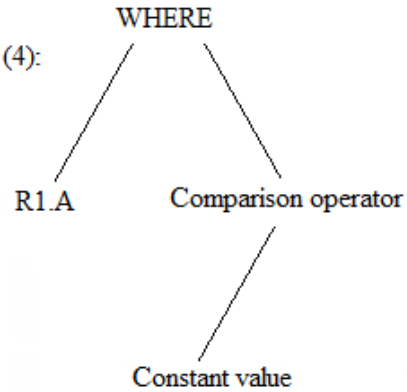


'A' in Format (1) represents any attribute name in the selected relations, and 'B' represents any constant value.
In Format (2) 'F.A' and 'S.A' are two values for the same attribute in two different tuples in one relation.

Format (3):

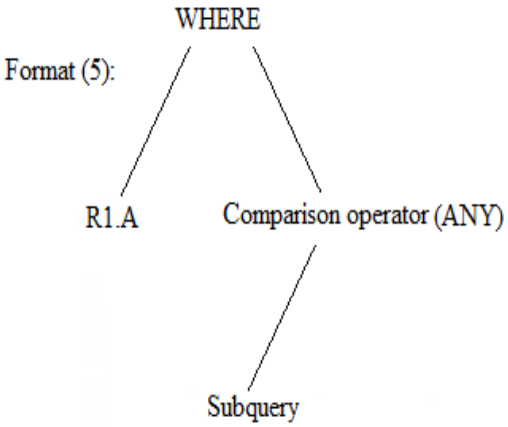


Format (4):

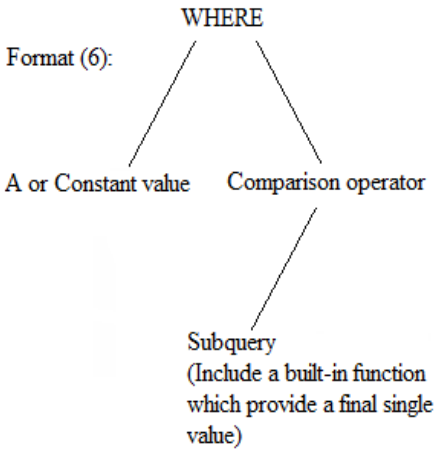


In Format (3) 'Rl.A' and 'R2,A' are the same attribute name in two different relations,
In Format (4) 'A' is a common attribute name in relation Rl and other relations joined with Rl. 'R1.A' indicates the value of the attribute A in the relation Rl to be compared with the constant value.

Format (5):

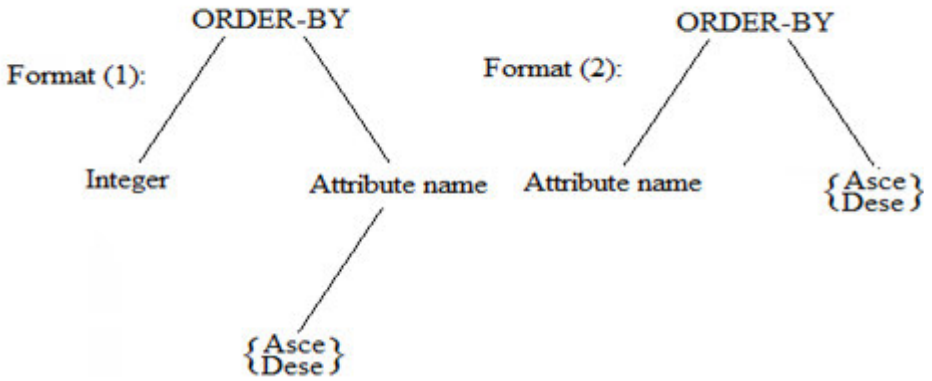


Format (6):



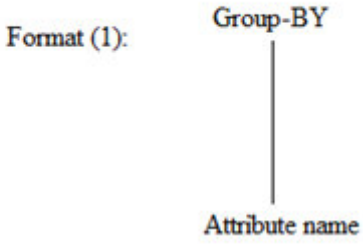
In the structure of Format (5), the clause represents a condition in which a value of attribute A in relation Rl is compared with a set of values generated by a subquery.
The comparison operator (comparison operator plus ANY) is interpreted as the condition F (=, ^=, >, <, <=, >=) ANY ({s1, s2, s3….}) evaluate to true if and only if the value F is (=, ^=, >, <, <=, >=) to at least one value in the set {s1, s2, s3...}.
Format (6) represents a condition in which an attribute value or a constant value is compared with one single value produced by a subquery which includes a built-in function.

(4) ORDER BY Clause

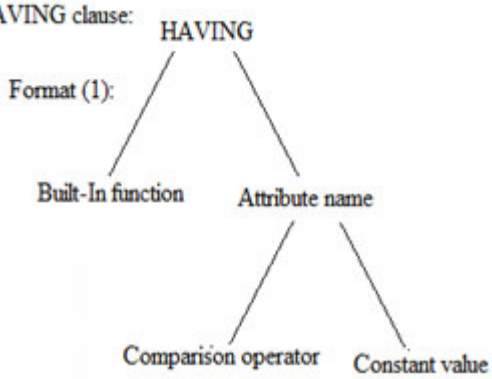


In the above two formats, {Asce, Desc} means that one of the options has to be selected to specify the sorting order of the projected attributes.

(5) GROUP BY Clause



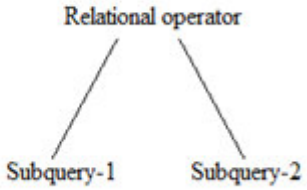
(6) HAVING clause:



The GROUP-BY clause consists of the GROUP-BY command and an attribute name as an argument.
In fact the HAVING clause represents a WHERE clause, its left operand represents a single value produced by applying a built-in function to one of the selected attributes. The right operand represents a single value, which may be specified directly as a constant value or produced by a nested subquery. Having clause is usually follows the GROUP-BY clause.

(7) Logical Operator Format

(8) Relational Operator Format



In the logical operator format, a number of WHERE clauses linked together with logical operators, and can be repeated for a number of times in one query.
In the relational operator format, the relational operator is applied to two sets of data values generated by subquery-1 and subquery-2. The result will be a third set of data values.

7. Query Formulation Process

The query formulation process is presented through a variety of retrieving and updating examples. Throughout these examples, the following three relation database is considered.

PART (PARTNUM, PNAME, COLOUR, WEIGHT, CITY)
SUPPLIER (SUPNUM, SUPNAME, STATUS, CITY)
SHIPMENT (PARTNUM, SUPNUM, QTY)

In MSGI representation the line arrows indicate the selected elements in the query while the dotted arrows specify the typed constant values.

7.1. Retrieving Operations:

Q1: Get supplier number for all suppliers in Paris Figure-4.

```
——> SUPPLIER
——> SUPNUM
——> =
——> CITY
----> 'Paris'
——> ENTER
```

```
SELECT SUPNUM
FROM SUPPLIER
WHERE CITY = 'Paris'
```

Q2: Get supplier numbers for suppliers in Paris with status < 50.

```
——> SUPPLIER
——> SUPNUM
——> =
——> CITY
------> 'Paris'
——> AND
——> <
——> STATUS
——> 50
——> ENTER
```

```
SELECT SUPNUM
FROM SUPPLIER
WHERE CITY = 'Paris'
AND STATUS < 50
```

Q3: Get supplier names for suppliers who supply at least one red part Figure-5.

```
——> SUPPLIER
——> SUPNAME
——> =
——> SUPNUM ( in SUPPLER)
——> SHIPMENT
——> SUPNUM (in SHIPMENT)
——> =
——> PARTNUM ( in SHIPMENT)
——> PART
——> PARTNUM (in PART)
——> =
——> COLOUR
--------> 'Red'
——> ENTER
```

```
SELECT SUPNAME
FROM SUPPLIER
WHERE SUPNUM IN
    (SELECT SUPNUM
     FROM SHIPMENT
     WHERE PARTNUM IN
        (SELECT PARTNUM
         FROM PART
         WHERE COLOUR = 'Red'))
```

Q4: Get part numbers for parts that either weigh more than 18 pounds or are currently supplied by supplier S2 (or both).

```
——> PART
——> PARTNUM
——> >
——> WEIGHT
--------> 18
——> UNION
——> SHIPMENT
——> PARTNUM
——> =
——> SUPNUM
--------> 'S2'
——> ENTER
```

```
SELECT PARTNUM
FROM PART
WHERE WEIGHT > 18

UNION

SELECT PARTNUM
FROM SHIPMENT
WHERE SUPNUM = 'S2'
```


7.2. Updating Operations:

PDBIS provides the main update operations which are supported by SQL-DML, or SQL-DDL. In PDBIS, the construction of updating queries may require the user to answer certain questions conducted by the system.

Q1: Change the color of part P2 to yellow, increase its weight by 5, and set its city to 'unknown' (null) Figure-6.

```
UPDATE          SET COLOUR = 'Yallow'
PART            WEIGHT = Weight + 10
COLOUR          CITY = Null
WEIGHT          WHERE PARTNUM = 'P2'
CITY
=
PARTNUM
'P2'
ENTER

ENTER NEW COLOUR: 'YALLOW'
ENTER NEW WEIGHT: Weight + 10
ENTER NEW CITY: Null
```

Q2: the following example shows the view definition in PDBIS Figure-7.

```
DEFINE          DEFINE VIEW VIEW1 (VPARTNUM, VPNAME,
PART            VCITY) AS
PARTNUM         SELECT PARTNUM, PNAME, CITY
PNAME           FROM PART
CITY            WHERE PARTNUM = 'P2'
=
PARTNUM
'P2'
ENTER

ENTER VIEW NAME: VIEW1
```

8. CONCLUSION

This paper has described the design and the implementation of MSGI, as

A graphical database end user interface to enables the non-computer professional to access information stored and managed by MYSQL database management system.

MSGI has managed to employ the powerful graphics in the programming language to present the database to the user in its most natural form, and separate the user from the physical structure of the data, and the data manipulation language. The way in which the database is displayed on the screen free the user form knowing the database structure and its relationships.

MSGI syntax is complete and covers all significant functions, operators, and operations that are required in most relational DBMSs. This syntax is well suited to the expression of a wide range of SQL queries against a MYSQL database, and this is especially true as queries become more complex.

Providing three different styles of feedback have successfully provided all the expected feedback advantages to all classes of MSGI user. By having more than one style of feedback; any negative aspect which might appear in one style will be covered by the other, so that at the end the three different styles act as one successful integrated feedback method.

MSGI is one of the very few database interface designs which provide the facility for report generation, graph display, and to have the result of the query printed in different output formats. By this utility the user can control the output of his query and specify the format which is most suitable for that particular requirement, therefore the database efficiency is increased by better utilizing the accessed data. By implementing this utility; **MSGI** can be categorized as a multifunctional interface, which allows the user to perform several different functions using the same utility. This certainly increases the productivity and positively affects the cost.

References

Al-Raw, 1998: PDPIS: Perq Database Interface System.
Journal of Science and Technology, V3, No2, Dec 1998.
AL-Rawi, 1986: “PDPIS: A unified Database End-User Interface”
Ph.D. Theses, University of Bath, England 1986.

Al-Rawi, 2009: D-PDBIS: Distributed Database Interface System
International Journal of Computer Science and Systems (IJCSS). Second Issue 2009.
R. Gravell, 2010: Top 10 MySQL GUI Tools.
Database Journal May-2010.
Chris Newman, 2006: Sams Teach Yourself MySQL in 10 minutes.
Publisher: Sams- May-2006.
IDS, 1977: IDS/FF End User Language Manual.
N. KRQ03A-E, 1977.
Gali, 82: Handbook of screen format design.
Q.E.D. Information Science Inc.
Wellesley, Massachusetts 02181-0501.

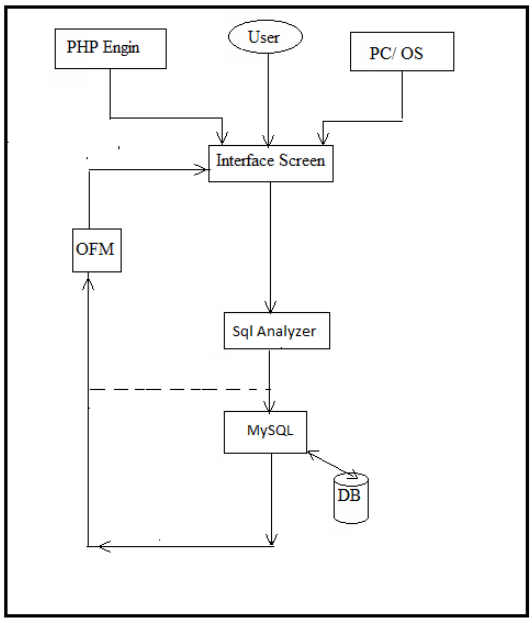


Figure-2

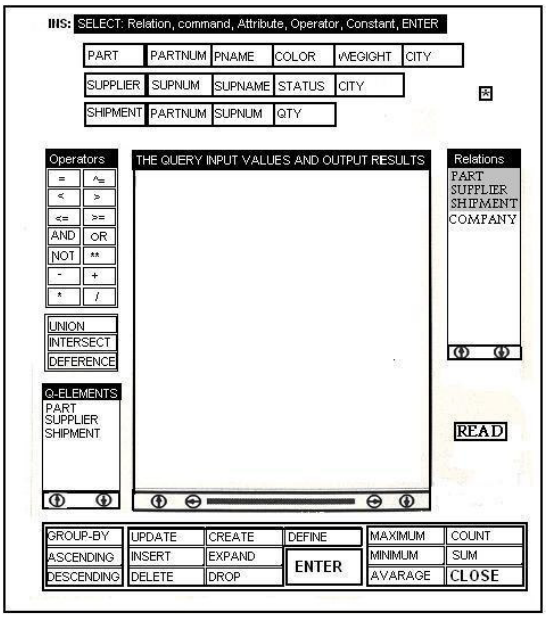


Figure-3

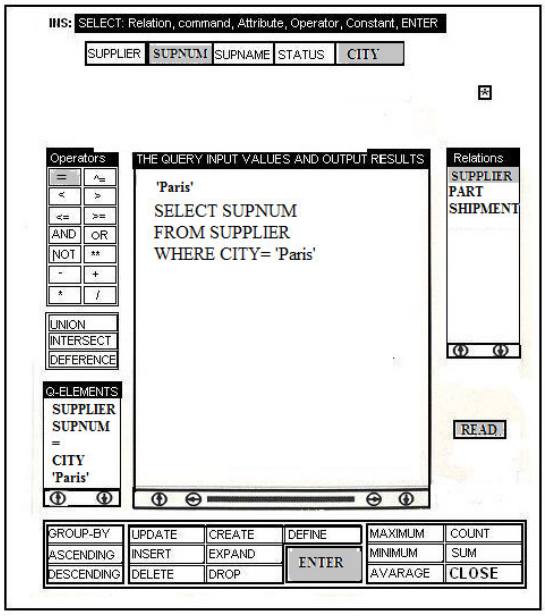


Figure-4

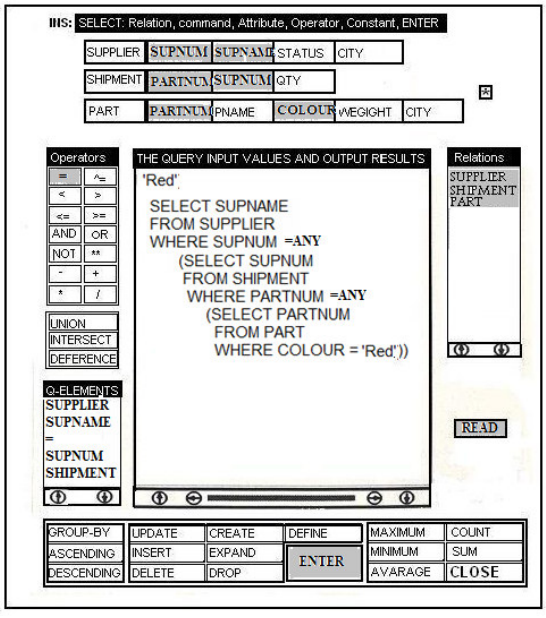


Figure-5

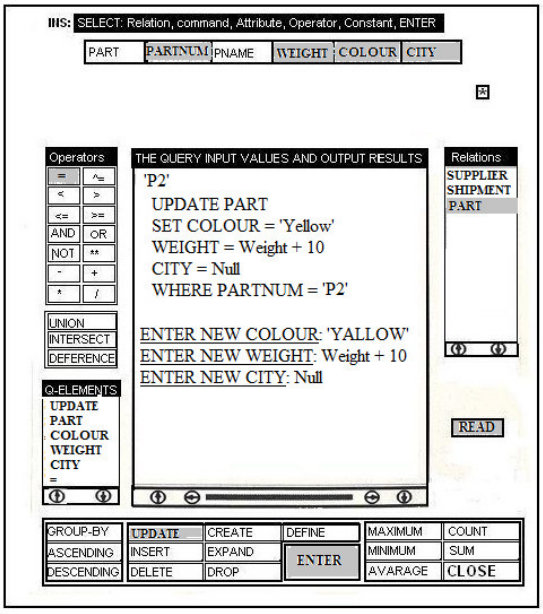


Figure-6

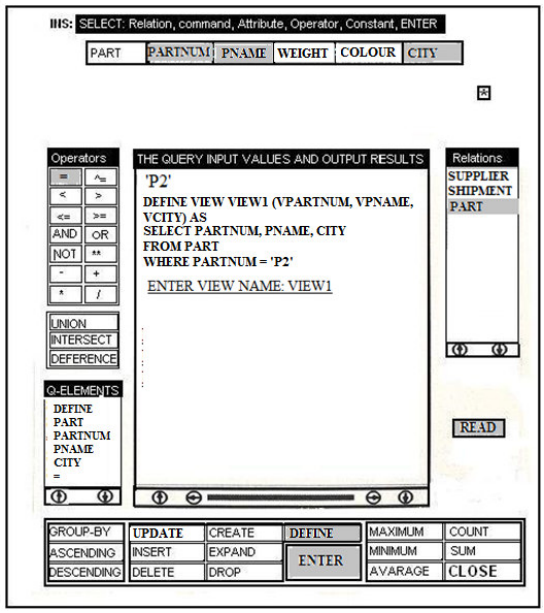


Figure-7

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar