International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 4– No.6, December 2012 – www.ijais.org

Optimizing the Intelligent Generic Query Mode and its Interface for Relational Database Applications

Ali El-Matarawy
Faculty of Computers and
Information, Cairo University

ABSTRACT

This research presents an optimization for generic query mode for any database application. It is an enhancement for a previously research "Improving the Query Mode and its Interface for Relational Database Applications". The research improved its intelligence more than expected. In the previous research the algorithm was intelligent in the sense that it answers most of the possible queries or questions that may arise in the user's mind without the need or support of the application developer but there was a restriction for that which is the user can't query about any computed field. This research added this feature to the algorithm regardless the depth of the computed field. The meaning of the depth of the computed field is how many times of the computed field in an equation needs to be transformed so that it contains no computed fields. It has been implemented using PowerBuilder (release 11.5) as a front end tool and Adaptive Server Anywhere (one of Sybase products) as a database engine. This research describes the design of the optimized intelligent generic query mode and its interface.

Keywords

Query mode, entry mode, relational database applications, optimizing query mode, generic query mode, intelligent query mode, database field, computed field.

1. INTRODUCTION

A query is a question or task a user asks of a database [1]. Query mode is the mode in which the user can ask the database some questions to get answers for them while data entry mode is the mode in which the user creates or edits records [2] in the underlying tables in the database. The basic steps involved in performing a query are [3]:

Entering query criteria

The information defining the query is entered into the Query datawindow (form).

Performing the query

The query is performed and all records in the database which match the information defining the query and which the user is privileged to view are retrieved.

Displaying the retrieved records

The matching records are displayed and can now be stored, copied, edited, re-ordered, reported on and deleted.

According to the history of developing database applications and its query mode, those query modes can be classified into three types.

The first query mode was so simple and very poor either in its efficiency in replying to many queries that the user may need at sometimes, or in its interface, its idea was based on displaying a menu (called query menu) to the user contains the available or allowed criteria to retrieve data. The main two disadvantages for using this type is first you can't get an answer for any question is not included in the query menu, and if you want that you should call the application developer and asking him to develop the new query you want, second is for its interface, it always needs from the user to read the query menu and sometimes it is too long to know the location of required query in the query menu, this will be sometimes so difficult.

The second query mode is too much better than the first one. Its interface depends on the data entry menu itself. When the user enters this query mode, the application clears all of previous data and asks the user to write criteria for only some or all of the displayed fields, and after that the user asks the application to execute the query with only the criteria fields he filled, and then the matched records will be displayed to him. The main advantages of this technique are first, you can ask the application for any query you want related to the displayed fields, second it is easier in use since it omitted the long query menu, third it allowed answering many queries to the user that may he needs, so it decreases the calling back to the developer.

The disadvantages for this query mode are first, the user can't constitute a query containing fields outside the entry menu and are found in other tables with a direct or indirect relation to the updated table in the entry menu, second, its interface needs from the user to know some SQL syntax such as logic relational words, logic operators, some symbols such as "*" or "%" to express some characters he doesn't know in the criteria field, third there is a great ambiguous of using many criteria fields in the mind of the user because all of the criteria fields is ANDED together or ORED together and in most of applications he should first specify the type of the logic relation among all fields and can't specify an OR relation for some fields and an AND relation for other fields.

The third query mode is a combined mode between the first one and the second one, this is because the second query mode is applied only to the entry menu you are using and the user would like to get some data is related to the updated table in the entry menu but at the same time is in a relation with other tables that has some fields the user may be would like to specify a criteria values for them, the third query mode is the most common used in any relational database application. This new query mode has been submitted in [3] and it is called "Intelligent Generic Query Mode" (IGQM).

The rest of this paper is organized as following: After defining and comparing between database field and computed field in section 2, some related work is introduced in section 3. In sections 4 the new approach for optimizing the intelligent and generic query mode is explained. In section 6 the interface that the new approach has used is presented. Section 7 introduces the conclusion and future work. Section 8 and section 9 are the acknowledgements and references for this research.

2. DATABASE FIELDS VERSUS COMPUTED FIELDS

Database fields (attribute) is a property or description of an entity [4], its definition are saved in the database table and its corresponding data are located also in the database as rows or records of a table while a computed field is a field which derives its data from calculation of other fields (database fields, computed fields or both). The data are not entered into a computed field by the user and it is not located in any table of the database, its definition is located in the definition of the datawindow (form) fields and its corresponding data is located in memory during the running of the database application.

3. RELATED WORK

As mentioned in [5] there are different approaches to query formulation, focusing on the usability of these approaches for non-IT people.

Query-by-form is an old practice; users can fill in and submit a form, where all fields in this form are seen as query variables. This way of data access is simple; however, it is neither flexible nor expressive. For each query, a form needs to be developed, and any change to the query implies changing the form.

Query-by-example allows users to formulate their queries as filling a table [6]. The names of the queried relations and fields are selected first; then users can enter their keywords. Although this approach is claimed to be easy to learn by non-IT people, however, it was not used by such people. In our opinion, this is because users are still required to understand the relational structure, which is difficult for non-IT people.

Conceptual query languages are an alternative approach to query formulation. As many databases are modeled conceptually using EER or ORM diagrams, one can also query these databases starting from those diagrams. Users can select some concepts from a given conceptual diagram, and their selection is automatically translated into SQL queries. This scenario was implemented by several EER-based [7, 8] and ORM-based [9, 10] approaches. ConQuer [11] is another ORM-based language, but it has some nice features indeed. Instead of starting from a conceptual diagram that may not exist, it starts from the logical schema and converts it into lists of concepts and relations. Users can then drag-drop from these lists to formulate their queries. What users drag-drop become a tree of facts, and this tree is seen as a query. Although this drag-drop scenario is not simple, 90 however structuring a query as a tree-pattern looks intuitive indeed.

The new query mode submitted in [3] titled IGQM (Intelligent Generic Query Mode) which has omitted all the disadvantages of the previous query modes mentioned in section 1. Notice that one can combine the IGQM query mode with the first one. Using this query mode the user can make a query with some fields that are not found in the current entry screen and are found in other table(s). This feature enables the application to approximately answer all questions of the user by 100%, hence the user can ask the application for some questions in his mind without asking the developer of the application to hardcode them in the application, this safe time and empower users and improve the quality and efficiency of service provided by applications that use the intelligent generic query mode. Some questions will not be answered at once but this is seemed to be logical and will be explained later in this research. Also, its interface is more flexible than the second one, since it does not depend on SQL language. It can be in the native language of the user.

Simply the new technique enabled to the user the features of defining his own question in his natural language by selecting database fields which are displayed in the screen either those fields in the master form or in any sub-form and define corresponding criteria to those fields with logic operators and logic relation.

The new technique has been implemented by adding some tables for saving information about the database application and an algorithm to construct the SQL dynamically of the user defined query.

4. OPTIMIZING IGQM

In IGQM, as mentioned in section 3, the user can define his own query by selecting database fields only not a computed fields, what the research has added to the IGQM is giving the ability to the user to define his own query from database fields located in a table with its corresponding data and computed fields located in a datawindow (form) with its corresponding data located in memory.

Simply it can be said that the new technique made the computed field as like a database field without changing of its location definition or its corresponding data location. This has been done by saving information about all computed fields in a separate table of the database application, the main information are its name and its equation so its name can be replaced by its equation in the definition of the user query. A major problem now arises which is that the computed field may depends on other computed fields for any unknown depth of this dependency, to build or construct a correct SQL the equation of the computed field should include only database fields, so a routine has been written for replacing any computed field in the equation of the selected computed field by the user by its equation and constitute a new equation for the selected computed field by the user, this routine loops until the equation doesn't contain any computed field.

It is notable to mention that using the optimized IGQM in a database application raises the number of queries that the user may generate from function of billions to function of trillions queries. One screen of this application consists of 4 sub-forms, each sub form base on a table consists of 8 database fields i.e. total no of fields in the four sub-forms equal to 32, hence the user if he used the IGQM can get approximately the combinatorial of ${}^{32}C_1 + {}^{32}C_2 + \ldots + {}^{32}C_{16}$, this summation will give billions of queries that the user may generate. In this database application, each sub-form has 52 computed fields, i.e. the total fields in the four sub-forms equal to 240, hence the user if he used the optimized IGQM, he can get approximately the combinatorial of ${}^{240}C_1 + {}^{240}C_2 + \ldots + {}^{240}C_{240}$, this summation will give trillions of queries that the user may generate.

A brief comparison between the new technique and "Query by Form" query mode is presented in the following table (1).

Table (1): Comparison between the new query mode and "Query by Form" query mode

Seq.	OIGQM	Data Entry Query Mode
1	The user can define his criteria to any fields in the main screen fields or its sub-forms fields. This increase the number of possible queries.	The user can define his criteria to only the fields of the screen he switched it to query mode, this decrease the number of possible queries.
2	No need to use SQL symbols or characters such as "*", "?,…,etc.	The user should know some SQL wild used character such as "*", "?",,…,etc.
3	The user uses his natural language in defining logic operators	The user should use logic operators such as ">","<",…,etc.
4	The user can use any field either in the main form or in sub-form.	The user can use only the fields in the screen in which he switched to its query mode
5	No ambiguity when the user wants to use either "AND" or "OR" logic relation when defining fields criteria	There is an ambiguous use when the user wants to use either "AND" or "OR" logic relation when defining fields criteria

5. INTERFACE OF OIGQM

In the following sections a description of the interface of the optimized intelligent generic query mode, section 5.1 describes briefly the data entry mode which is the same for the IGQM. The data entry mode is implemented to reflect the relation types among tables in the database which enables the user the innovation in creating his own query, section 5.2 describes the interface of the intelligent generic query mode which also it is the same as the IGQM interface.

5.1 The Data Entry Mode

Before explaining the optimized intelligent generic query mode, a brief explanation about the data entry mode should be introduced. It is designed to reflect the relations among tables in the database, so it used master-detail interface [12, 13] or main-subform forms like mentioned in MS-ACCESS. This interface has been designed up to 4 levels of master-detail (i.e. in other words up to 4 tables will be entered in one screen. The screen consists of 4 parts, each part is for one table of the four tables and each part is in a relation one to many to the part below it directly, Fig. (1) shows one screen which can be used for entering data in 3 tables, the upper part shows the table entry level – 1 and the middle part shows the table entry level – 2, the bottom part shows the data entry level – 3), the number of levels can be extended but in this case the usability of application will be lost since the user will have to enter many tables data in one screen. Also notice that the optimized intelligent generic query mode can be applied in any data entry interface either in a screen contains a table or in a screen contains any number of tables, this interface enables the user to understand the relation between different tables data and consequently he can generate more power queries he needs.

In Fig.(1) the three screens are in tabular form, this style of the form allows many records to be displayed to the screen at

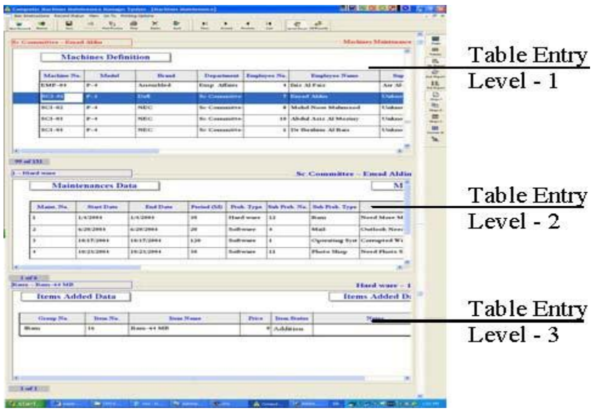

Fig. (1) Data Entry Interface

a time but notice that this is only an example, the user can alter it to a normal form in which only one record can be displayed at a time.

5.2 OIGQM Interface

This interface is so simple and is found in many of applications, some is related to database and the other is not. Instead of clearing the fields in the data entry menu it hides it and displays another menu. Each row of the menu constitutes a user question. Each row consists of 4 fields. The first field is the field title either for a database field or a computed field. A user can select a field title from a combo box as shown in Fig.(2).

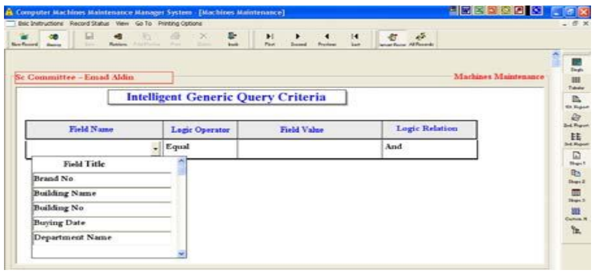

Fig. (2): Field Title Selection

The combo box contains all fields titles (either titles for database fields or computed fields) of the current level of the table that the user of the application is using; also it contains all fields titles (either titles for database fields or computed fields) in the table data entry levels below it. To make this clear to the user of the application, each table level fields is represented by different color, the black color is for the first table level fields titles, the blue color is for the second table level fields titles, the red color is for the third table level fields titles and the green color is for the fourth table level fields titles.

The second field is a combo box containing all logic operators, they are displayed in the native language of the application and not in SQL syntax, and this is shown in Fig.(3). As you can see in this figure the user select the logic operator in the native application language which is such as English in this figure but notice that this is can be in any natural language, it depends on the development issues. This avoid the user of knowing some SQL symbols such as ">", "<","<>", … , etc.

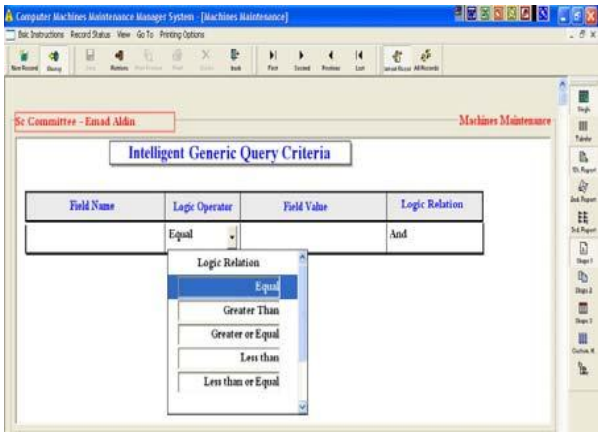

Fig. (3): Logic Operator Selection

Consequently the user don't need to know some information about some SQL syntax, he can understand it directly because it is written in his native language, so he can understand them.

The intelligent generic query mode is responsible to convert those fields contents to the correct SQL syntax, the third field is the field value the user wants to specify for the field title he select before. The fourth field is used to specify the logic relation "AND" or "OR" with the subsequent rows of the current row as shown in Fig. (4), also this field is displayed in the native language of the application and not in SQL syntax, so again it is so simple for understanding.

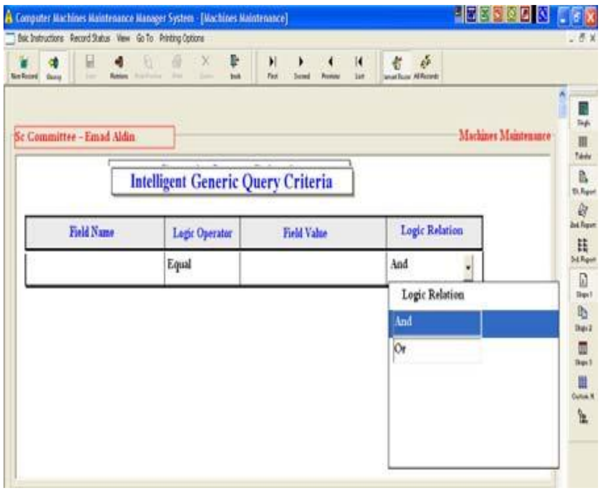
Computer Machines Maintenance Manager System - [Machines Maintenance]
Sc Committee - Emad Aldin
Machines Maintenance
Intelligent Generic Query Criteria
Field Name
Logic Operator
Field Value
Logic Relation
Equal
And
Logic Relation
And
Or

Fig. (4): Logic Relation Selection

6. CONCLUSIONS & FUTURE WORK

The Optimized intelligent generic query mode increases the efficiency of answering more and more questions that comes from the user in an easy and simple interface, this efficiency has been increased mainly due to increasing the number of queries that the user may generate by adding all computed fields in the database application as well as the database fields to be available to the user when he defines his own queries, finally the more computed fields in the database applications, the more efficiency the user gains. Our future work is to generate intelligent generic statistical query (IGSQ) for any database field or computed field in the database application, also a combine of the IGSQ with the optimized IGNQM to increase the number of statistical report the user can gain.

7. ACKNOWLEDGMENTS

Thanks to Dr. Reem Bahgat for her constant support and for Dr. Mohamad El-Sharkawy for his comments.

8. REFERENCES

[1] Albert K.W. Yeung, G. Brent Hall, Spatial Database Systems, Design, Implementation and Project Management, Published by Springer

[2] Turban E., & Aronson J., (1998), Decision Support Systems and Intelligent Systems, 5th Ed'n, Prentice Hall, New Jersey, ISBN 0-13-740937-0

[3] Ali El-Matarwy, Improving the Query Mode and its Interface for Relational Database Applications, First National Symposium on Information Technology, journal 2005, King Saud University faculty of Computer Science Information System.

[4] Database Management Systems, Solutions manual, Third Edition, Raghu Ramakrishnan, University of Wisconsin, Madison, WI, USA, Johannes Gehrke, Cornell University, Ithaca, NY, USA and Jeff Derstadt, Scott Selikoff, and Lin Zhu, Cornell University, Ithaca, NY, USA.

[5] Mustafa J., Marios D. Dikaiakos, MashQL: A Query-by-Diagram Topping SPARQL Towards Semantic Data Mashups, University of Cyprus mjarrar.

[6] Zloof, M.: Query-by-Example: A Data Base Language. IBM Systems Journal, 16(4). 1977.

[7] Czejdo, B., Elmasri, R., Rusinkiewicz, M., and Embley, D.:An algebraic language for graphical query formulation using an EER model. In Proceedings of the ACM Conference on Computer Science. 1987.

[8] Parent, C., and Spaccapietra, S.: About Complex Entities, Complex Objects and Object-Oriented Data Models. In Proceedings of the IFIP 8.1 conference. 1989.

[9] De Troyer, O., Meersman, R., and Verlinden, P.: RIDL on the CRIS Case: A Workbench for NIAM. In Proceedings of the IFIP.8.1 Conference. 1988.

[10] Hofstede, A., Proper, H., and van der Weide, T.: Computer Supported Query Formulation in an Evolving Context. In Proceedings of the ADC. 1995.

[11] Bloesch, A. and Halpin, T.: Conceptual Queries usingConQuer–II. In Proceedings of the ER. LNCS, Springer. 1997.

[12] Schwabe D., Rossi G. and Barbosa D.J. Systematic Hypermedia Application Design with OOHDM. Proc. ACM Conference on Hypertext. pp.166. 1996.

[13] Paolini P. and Fraternali P. A Model-Driven Development of Web Applications: the Autoweb System . ACM Transactions on Office Information Systems, 2000, Vol. 18, Number 4.