IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 5, Ver. V (Sep. – Oct. 2015), PP 01-07
www.iosrjournals.org

An application of flexible query interface to relational databases

Onwuachu Uzochukwu C, Oghenekaro, Linda U
Department of Computer Sciences, Imo state University, Owerri, Nigeria.
Department of Computer Sciences, University of Port Harcourt, Choba, Nigeria.

Abstract: *The use of databases has for long been for the computer elites in the society as well as multinationals that can pay for their services, and these has inhibited the scope and wide use of database applications. The aim of this paper is to develop yet another flexible query interface for relational databases that is user friendly and has the capability to adequately help users work with databases without a thorough knowledge of database programming. It also provides guidelines for users interested in learning the technicalities involved in database query writing. The proposed system uses an object oriented methodology and was implemented using Java programming language.From the result, the system shows a high level of flexibility in database query processing.*

Keywords: *Database, flexible query, Interface, object oriented and Java programming*.

I. Introduction

The absence of a flexible and intelligent database query interface for non-expert users has been an issue of concern for ages for the populace. A general information management system that is capable of managing several kinds of data, stored in the database is known as Database Management System (DBMS). The DBMS grants support for logical views of data that are separate from the physical views, i.e. how the data is actually stored in the database. By permitting applications to define, access, and update data through a Data Definition Language (DDL) and Data Manipulation Language (DML) combined into a declarative query language such as the relational query language SQL, the separation is accomplished (Ribeiro andMoreiro, 2003).

Structured Query Language (SQL) is an ANSI standard for accessing and manipulating the information stored in relational databases. It is comprehensively employed in industry and is supported by major database management systems (DBMS). Most of the languages used for manipulating relational database systems are based on the norms of SQL. They work on the basis of Boolean interpretation of the queries: a logical expression is the only accepted selection criterion and the response always encompasses only these tuples for what the expression results in a true value. But some user requirements may not be answered explicitly by a classic querying system. It is due to the fact that the requirements' characteristics cannot be expressed by regular query languages (Hallet, 2006).

In recent times, there is a rising demands for non-expert users to query relational databases in a more natural language encompassing linguistic variables and terms, instead of operating on the values of the attributes. Flexible query interface, a promising approach, enhances the users in database management. They work on the basis of Boolean interpretation of the queries: a logical expression is the only accepted selection criterion and the response always encompasses only these tuples for what the expression results in a true value [Neelu, etal 2009]. But some user requirements may not be answered explicitly by a classic querying system. It is due to the fact that the requirements' characteristics cannot be expressed by regular query languages. Many novel-generation database applications stipulate intelligent information management necessitating efficient interactions between the users and database. Flexible database systems, a promising approach, enhance the users in performing database queries(Zongmin 2007). The research and advancement of flexible query interface have lately emerged and have fascinated the attention of many people. It is to this end this research was done, so as to make database querying in distributed platforms even much more flexible.

II. Literature Review

Neelu etal, (2009), proposed an intelligent layer for database which is responsible for manipulating flexible queries. Initially, the flexible queries from users in their natural language are submitted to intelligent layer and this layer converts the amorphous query into a structured SQL query. The shaped query is executed and the results are presented to the user. Afterwards, on the basis of results, feedback and the acceptance or rejection of the results are requested from the user. It enables the design of a knowledge based self-learning system based the values obtained from user, which will aid the selection of appropriate SQL query, when a same flexible query is issued in the future. The experimental results demonstrate the effectiveness of the proposed intelligent database system.

Ben, (2014) proposed the data, information and knowledge based technology of Smart/ Intelligent User Interface (IUI) design, which interacts with users and systems in natural and other languages, utilizing the principles of Situational Control and Fuzzy Logic theories, Artificial Intelligence, Linguistics, Knowledge Base technologies and others. The proposed technology of IUI design was defined by multi-agents of (a) Situational Control and of data, information and knowledge, (b) modeling of Fuzzy Logic Inference, (c) Generalization, Representation and Explanation of knowledge, (c) Planning and Decision-making, (d) Dialog Control, (e) Reasoning and Systems Thinking, (f) Fuzzy Control of organizational unit in real-time, fuzzy conditions, heterogeneous domains, and (g) multi-lingual communication under uncertainty and in Fuzzy Environment.

Oussama, (2001), indentified that Database flexible querying is an alternative to the classic one for users. The use of Formal Concepts Analysis (FCA) makes it possible to make approximate answers that those turned over by a classic Database Management System (DBMS). Some applications do not need exact answers. However, flexible querying can be expensive in response time. This time is more significant when the flexible querying require the calculation of aggregate functions ("Sum", "Avg", "Count", "Var" etc.). So, he proposed an approach which tries to solve this problem by using Approximate Query Processing (AQP).
Donald, (1990), at the Unisys center for advanced information technology paoli, Pennsylvania developed an Intelligent Database Interface (IDI) with a cache-based interface designed to provide Artificial Intelligence systems with efficient access to one or more databases on one or more remote database management systems (DBMSs). It could be used to interface with a wide variety of different DBMSs with little or no modification since SQL was used to communicate with remote DBMSs and the implementation of the ID1 provides a high degree of portability. The query language of the ID1 is a restricted subset of function-free Horn clauses which is translated into SQL. Results from the ID1 are returned one tuple at a time and the ID1 manages a cache of result relations to improve efficiency. The ID1 is one of the key components of the Intelligent System Server (ISS) knowledge representation and reasoning system and is also being used to provide database services for the Unisys spoken language systems program.

Neelu, et al, (2010), in their paper discussed the mapping of natural language queries to SQL. They further proposed a general architecture for an intelligent database interface and also a real implementation of such a system which can be connected to any database. One of the main characteristics of this interface is domain-independence, which means that this interface can be used with any database. Another characteristic of this system is ease of configuration. The intelligent interface employs semantic matching technique to convert natural language query to SQL using dictionary and set of production rules. The dictionary consists of semantics sets for tables and columns. The shaped query is executed and the results are presented to the user. This interface was first tested using Supplier-Parts database and secondl y with Northwind database of SQL server 7.0.

Nittaya and Kittisak, (2012), presented a paper at the International Journal of Database Theory and Application which presents knowledge acquisition method focusing on association pattern mining, its implementation, and a systematic method of rewriting query with association patterns and materialized views. They research performed a preliminary efficiency tests of the system. The experimental results demonstrates the effectiveness of the system in answering queries sharing the same pattern as the available knowledge and the pre-computed views

Ribeiro, and Moreira, (2003), presented a paper which describes a fuzzy query interface for a business database. Hence, queries in natural language with pre-defined syntactical structures are performed, and the system uses a fuzzy natural language process to provide answers. This process uses the fuzzy translation rules of the meaning representation language PRUF. The interface was built for a relational database of the 500 biggest non-financial Portuguese companies. The attributes considered are the economic and financial indicators. Examples of pseudo natural language queries, such as "is company X very profitable? " or " are most private companies productive? ", are presented to show the capabilities of this human-oriented interface.

Antonio et al (2006), present an overview of Flexible query languages for relational databases which is the most important proposals for human-oriented query languages for relational databases, based on fuzzy sets theory. To highlight important issues concerning communication with databases, they propose two taxonomies: the first taxonomy deals with flexible query languages in crisp relational databases and the second deals with flexible query languages in fuzzy relational databases. It helps database designers and users understand and select the best approaches to solve their problems.

III. Methodology

Figure 1 describes the flowchart of the proposed system. The Login Interface requires a USERNAME and PASSWORD. The Flexible Query Interface provides the user with the server name of all the servers in the system. From the list of server provided by the system the user can now select the required server. The sever status will indicate connected immediately the user is connected to the server. All the created databases in the

server will appear for the user to select the required database. From the FQI, there is a button the list out all the tables that is found in database. The user is required to select from the list of table and FQI still gives the user an alternative to create tables when the desired table is not available. The user executes query if the desired query is already written and there is also an alternative for the user to write his/her own query. After successfully executing a query for queries that are often executed, you can save such query as a script file with the Save Script button for further execution.

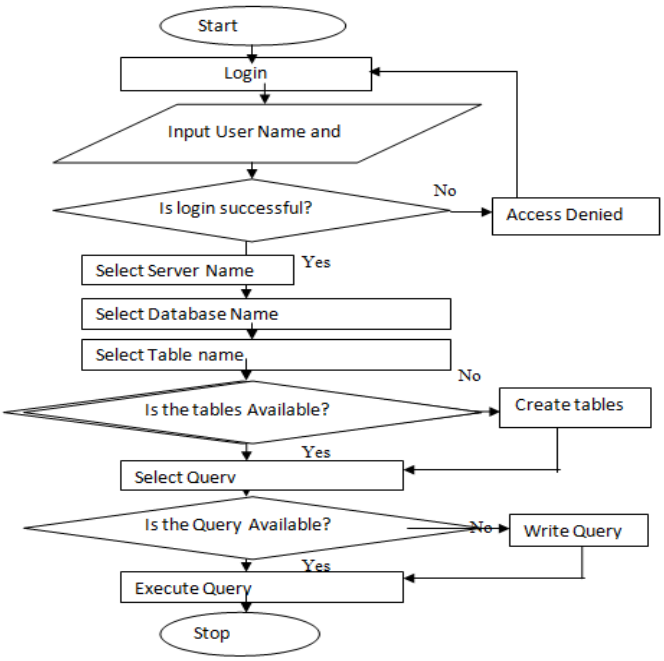

Figure 1. Flowchart of the Proposed System

In figure 2 the login class connects the user to the login section, and then in the login session helps the user in Flexible Query Interface. The user can as well access the database directly from the login section. The query processor will always query the database to get the required information.

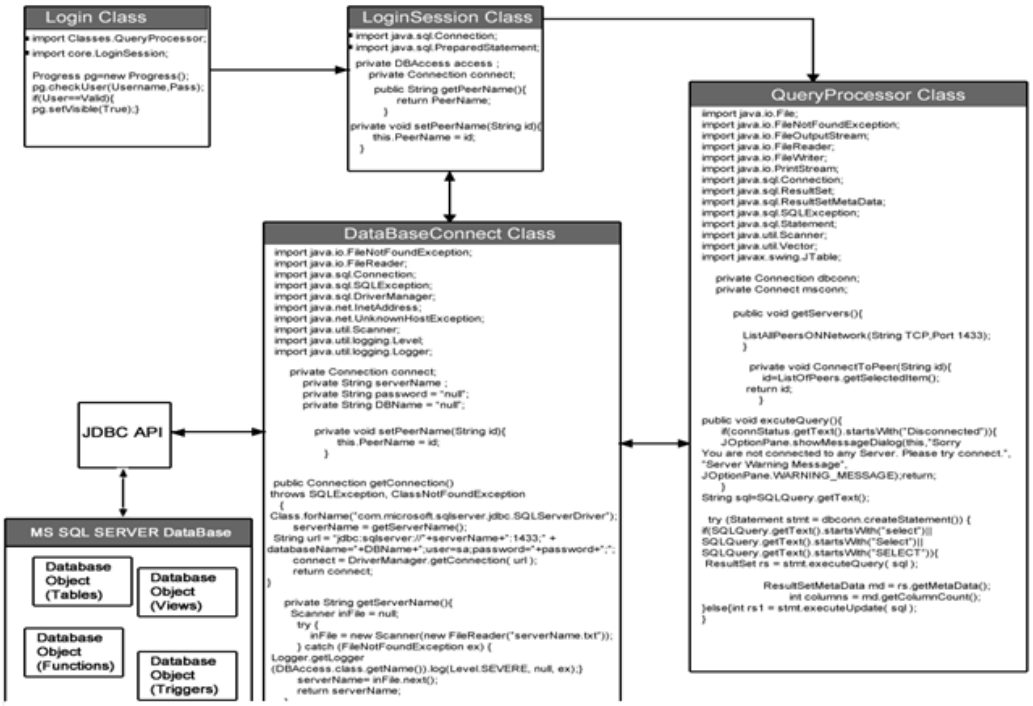

Figure 2: UML Class Diagram of the Proposed System

IV. Experiment And Results

The user needs to have the username and password before he/she can access the system. Immediately the username and the password are entered to the system, the login button takes you to the FQI platform. Figure 3 shows the Flexible Query Interface Login Module**.**

Figure 3: Flexible Query Interface Login Module

Figure 4 shows Flexible Query Interface module. The FQI provides the user with the server name of all the servers in the system. From the list of server provided by the system the user can now select the required server. The sever status will indicate connected immediately the user is connected to the server. All the created databases in the server will appear for the user to select the required database. From the FQI, there is a button the list out the entire table that is found in database. The user is required to select from the list of table and FQI still gives the user an alternative to create tables when the desired table is not available. The user executes query if the desired query is already written and there is also an alternative for the user to write his/her own query.

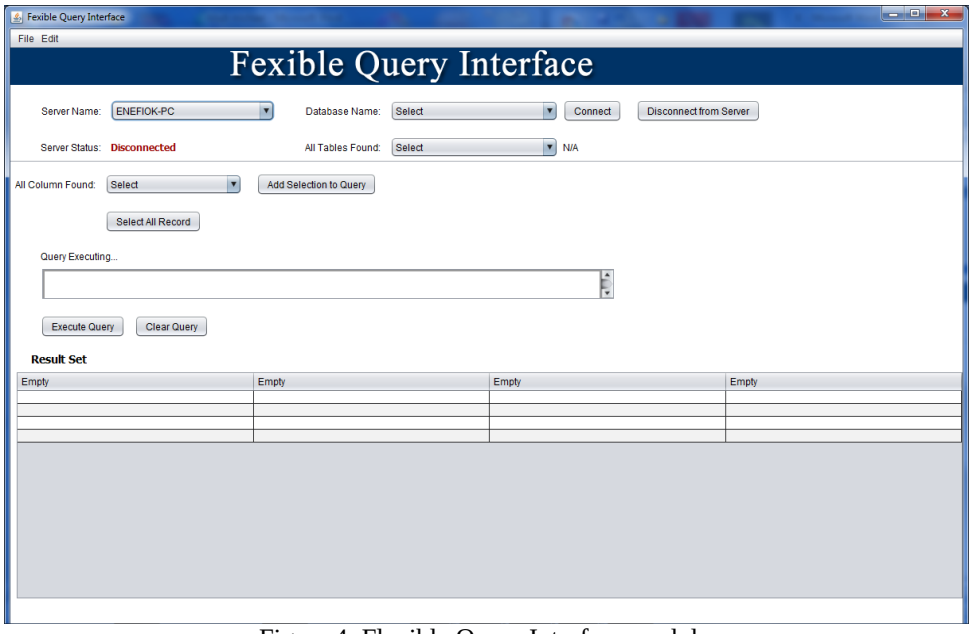

Figure 4. Flexible Query Interface module

Figure 5 shows the flexible query interface with the query result. After successfully executing a query for queries that are often executed, you can save such query as a script file with the Save Script button for further execution.The Execute Query from Script button allows you to execute queries that are save on script file. After every successful execution of queries (e.g SELECT STATEMENT) the result are displayed on the Result Set Table but for other queries like the DDL or DML, a message dialog box displays the success/error message.

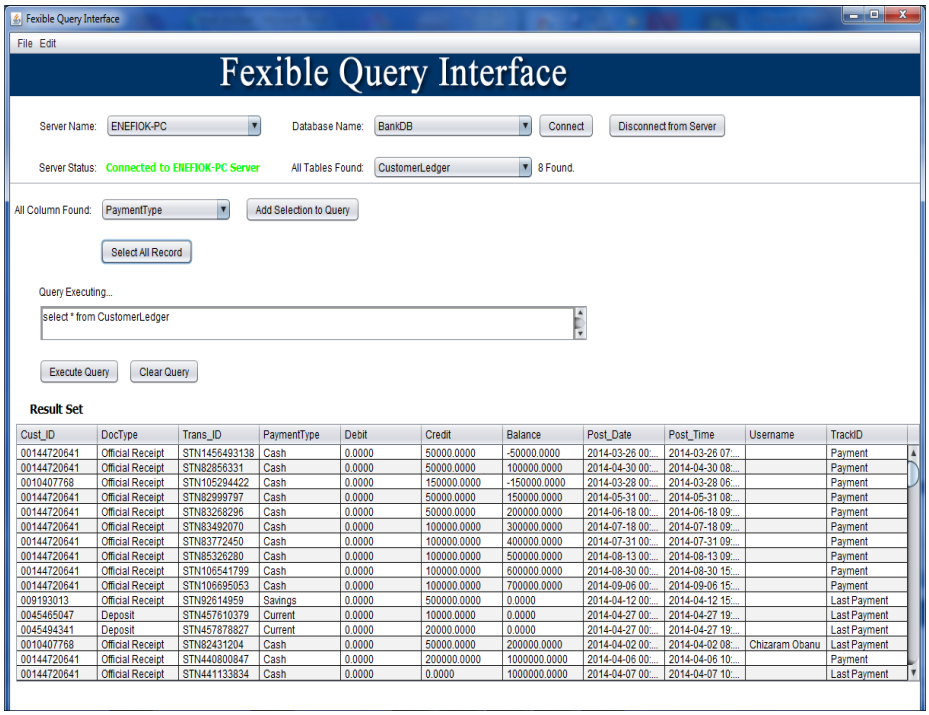

Figure 5. The Flexible Query Interface with the Query Result

Result Set

Account_No	Cust_ID	Account_Type	Branch	Balance
0042321962	CST1	Corporate		
0061280944	CST2			
0061310530	CST3			
0061341126	CST4		Head-Office	
0061386846	CST5		Head-Office	
0061460477	CST6		Head-Office	
0061484299	CST7	Savings	Head-Office	5000
00138299430	CST8	Savings	N/A	20000
00144720641	CST9	Savings	N/A	50000
0010407768	CST2	Current	Head-Office	50000
009193013	009193013	Savings	N/A	500000
0045465047	0045465047	Current	N/A	10000
0045494341	0045494341	Current	N/A	20000

Table 1 the Result Set for the query (select*from accounts)

Table 2 the Result Set for the query (select*from computeprofit)

Result Set

AccountNo	AcountType	Profit	DateUpdated
0010407768	Current	400.0000	2014-04-02 00:00:00.0
00144720641	Savings	8760082.1918	2014-04-26 00:00:00.0
009193013	Savings	50.0000	2014-04-12 00:00:00.0
0045465047	Current	65.7534	2014-04-27 00:00:00.0
0045494341	Current	657.5342	2014-04-27 00:00:00.0

Table 3 the Result Set for the query (select*from customer_transaction)

Result Set

Account_No	Trans_Type	depositor	Amount	Trans_Time	Trans_Date
0061484299	Deposit	Self	5000	16:35:38 PM	05/12/2013
00138299430	Deposit	Self	20000	13:33:05 PM	03/25/2014
00144720641	Deposit	Self	50000	07:20:51 AM	03/26/2014
0010407768	Deposit	Self	50000	06:18:43 AM	03/28/2014
009193013	Deposit	Self	500000	15:32:48 PM	04/12/2014
0045465047	Deposit	Self	10000	19:04:27 PM	04/27/2014
0045494341	Deposit	Self	20000	19:08:54 PM	04/27/2014

Table 4 the Result Set for the query (select*from customerLedger)

Result Set

Cust_ID	DocType	Trans_ID	PaymentT...	Debit	Credit	Balance	Post_Date	Post_Time	Username	TrackID
00144720...	Official Re...	STN1456...	Cash	0.0000	50000.00...	-50000.00...	2014-03-...	2014-03-...		Payment
00144720...	Official Re...	STN8285...	Cash	0.0000	50000.00...	100000.0...	2014-04-...	2014-04-...		Payment
00104077...	Official Re...	STN1052...	Cash	0.0000	150000.0...	-150000.0...	2014-03-...	2014-03-...		Payment
00144720...	Official Re...	STN8299...	Cash	0.0000	50000.00...	150000.0...	2014-05-...	2014-05-...		Payment
00144720...	Official Re...	STN8326...	Cash	0.0000	50000.00...	200000.0...	2014-06-...	2014-06-...		Payment
00144720...	Official Re...	STN8349...	Cash	0.0000	100000.0...	300000.0...	2014-07-...	2014-07-...		Payment
00144720...	Official Re...	STN8377...	Cash	0.0000	100000.0...	400000.0...	2014-07-...	2014-07-...		Payment
00144720...	Official Re...	STN8532...	Cash	0.0000	100000.0...	500000.0...	2014-08-...	2014-08-...		Payment
00144720...	Official Re...	STN1065...	Cash	0.0000	100000.0...	600000.0...	2014-08-...	2014-08-...		Payment
00144720...	Official Re...	STN1066...	Cash	0.0000	100000.0...	700000.0...	2014-09-...	2014-09-...		Payment
009193013	Official Re...	STN9261...	Savings	0.0000	500000.0...	0.0000	2014-04-...	2014-04-...		Last Pay...
00454650...	Deposit	STN4576...	Current	0.0000	10000.00...	0.0000	2014-04-...	2014-04-...		Last Pay...
00454943...	Deposit	STN4578...	Current	0.0000	20000.00...	0.0000	2014-04-...	2014-04-...		Last Pay...
00104077...	Official Re...	STN8243...	Cash	0.0000	50000.00...	200000.0...	2014-04-...	2014-04-...	Chizaram ...	Last Pay...

Table 5 the Result Set for the query (select*from customer)

Result Set

Cust_ID	Title	Surname	FirstName	MiddleName	NickName	Gender	MaritalStatus
0010407768	Mr.	George	Kent	Kelvin	Kelvin	Male	Married
00144720641	Dr.	Uche	Marizu	N	Doc	Male	Married
0045465047	Mr.	Promise	r	r		Male	Single
0045494341	Mr.	Jude	Opuh	p	k	Male	Single
009193013	Prof.	Jaja	Dickson			Male	Single

Table 6 the Result Set for the query (select*from loan)

Result Set

Cust_ID	LoanType	LoanAmo...	TransDate
00144720...	No Wahala	100000.0...	2014-04-...
00454650...	No Wahala	2000.0000	2014-04-...
00454943...	Short Ter...	1000000....	2014-04-...

Once the output requirements are determined, the system designer can decide what to include in the system and how to structure it so that they require output can be produced. For the proposed software, it is necessary that the output reports be compatible in format with the existing reports. The output must be concerned to the overall performance and the system's working, as it should. It consists of developing specifications and procedures for data preparation, those steps necessary to put the inputs and the desired output, i.e. maximum user friendly. Proper messages and appropriate directions can control errors committed by users.

V. Result Discussion

In table 1 the Result Set for the query (select*from accounts) was displayed by just clicking a button. The columns that were displayed include the account number, customer identity, account type, branch and balance. In table 2 the Result Set for the query (select*from computeprofit) was displayed by just clicking a button. The columns that were displayed include the account number, account type, profit and DateUpdated. In table 3 the Result Set for the query (select*from customer_transaction) was displayed by just clicking a button.

The columns that were displayed include the account number, transaction type, depositor, amount, transaction time and transaction date. In table 4 the Result Set for the query (select*from customerledger) was displayed by just clicking a button. The columns that were displayed include the account number, document type, transaction identity, payment type, debit, credit, balance, post_time, username and track identity. In table 5 the Result Set for the query (select*from customer) was displayed by just clicking a button. The columns that were displayed include the account number, title, surname, midname, gender and maritalstatus. In table 6 the Result Set for the query (select*from loan) was displayed by just clicking a button. The columns that were displayed include the account number, loan type, loan amount and transdate.

VI. Conclusion

This research represents a first step toward the design of a complete flexible query system. It is yet another flexible query interface for relational databases that is user friendly and has the capability to adequately help users work with databases without a thorough knowledge of database programming. A system designed for an efficient and flexible database query processing model in distributed system. It supports a more diverse and richer set of queries, and presents the techniques for flexible query processing.The algorithms of query processing were described in unstructured systems. The main concern in unstructured systems is how to processing the query to obtain high quality answers while minimizing the communication cost. This paper describes how flexible query interface can be used to produce an efficient query system for a relational database.

References

[1] Antonio Rosado, Rita A. Ribeiro, SlawomirZadrozny, JanuszKacprzyk.(2006), Flexible query languages for relational databases: An overview. In: Flexible databases supporting imprecision and uncertainty, Gloria Bordogna and Giuseppe Psaila (eds), Studies in Fuzziness and Soft Computing Series, Vol 203, Springer (2006).

[2] .Ben K., Kazar O., Caplat G., [2014] "intelligent query processing for semantic interoperable information systems" Department 1of computer science, university of Mohammed KhiderBiskra, Algeria

[3] Brodie M. [2008], "Future Intelligent Information Systems: AI and Database Technologies Working Together" in Readings in Artificial Intelligence and Databases, Morgan Kaufman, San Mateo, CA.

[4] Donald P. McKay and Timothy W. Finin [1990], "The Intelligent Database Interface: Integrating AI and Database systems", In Proceedings of the 1990 National Conference on Artificial Intelligence: 677-684.

[5] Hallett C., [2006] "Generic Querying of Relational Databases using Natural Language Generation Techniques", Proceedings of the Fourth International Natural Language Generation Conference, pages 95-102.

[6] Huanliang Sun, YubinBao, Faxin Zhao, Ge Yu and Daling Wang [2004], "CD-Trees: An Efficient Index Structure for Outlier Detection", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 3129, 2004.

[7] Kacprzyk, J., Zadrozny, S. (2001). Computing with words in intelligent database querying: standalone and Internet-based applications, Information Sciences, 134, Elsevier, pp.71-109

[8] Murugan K., and Ravichandran T.[2012]" event matching approach based human language interface in databases" . International journal of computational intelligence research (IJCIR)

[9] Neelu N., Sanjay S., and Mahesh M.[2009] "Design of an intelligent layer for flexible querying in database" International Journal on Computer Science and Engineering Vol.1(2), 30-39.

[10] Neelu N., Sanjay S., and Mahesh M.[2010], "An Intelligent Interface for relational databases" International Journal on Computer Science and Engineering Vol.1(5), 330-340

[11] Nittaya K. and Kittisak K.,[2012] "Semantic-based query answering supported association patterns and materialized views ". Data engineering research unit, School of computer engineering, Suranaree University of Technology, NakhonRatchasima 30000, thialand.

[12] OussamaTuli, Minyar S., Habib O.[2001] "Intelligent Database Flexible Querying by Approximate Query Processing (AQP)". Faculty of science of Tunis, Campus Universitaire 1060 Tunis, Tunisia

[13] R. A. Ribeiro, A. M. Moreira.(2003), Fuzzy query interface for a business database. International Journal of Human-Computer Interfaces Vol. 58, No 4, 363-391

[14] TorstenHothorn, David A. James, and Brian D. Ripley. [2001]" R/S interfaces to databases. In Proceedings of the Distributed Statistical Computing 2001Workshop, http://www.ci.tuwien.ac.at/Conferences/DSC-2001, 2001.Vienna University of Technology.

[15] Zongmin M,[2007] "Intelligent Databases: Technologies andApplications", IGI publishing, 320 pages, 2007.