IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 10, December 2014.
www.ijiset.com
ISSN 2348 – 7968

Building a Query Builder Interface for Accessing Multiple Databases

Mrs.Vasuki.M[1], Ms.Rajeswari.R[2] and Ms.Anandhi.R[3]

[1] Assistant Professor, MCA, Sri Manakula Vinayagar Engineering College, Madagadipet, Pondicherry India

[2] MCA, Student, Sri Manakula Vinayagar Engineering College, Madagadipet, Pondicherry India

[3] MCA, Student, Sri Manakula Vinayagar Engineering College, Madagadipet, Pondicherry India

Abstract

In order to store the data, different people use different types of databases. If a common user wants to access data from all those databases it is very difficult for them because the user has to switch over to different front end applications or to the different database environments. This will take much time and it is more complex to the users. In order to solve this problem we create a new application called the Query builder to access data from different types of database such as MS-Access, SQL server, Oracle. This query builder will also act as a GUI for the user and contains various options which make the users work very simple. With this application the user can retrieve data from various databases by using a single query. All the statement of SQL's Data Manipulation language (DML) can be used here. Since this application is a Graphical User Interface (GUI) so it is a more user friendly. In addition, it also includes features like printing the results and creating new table from the obtained result.

Keywords: *Relational Database Management System, Structured Query Language, Open Database Connectivity, Online Transaction Processing.*

1. Introduction

Large amount of data are stored safely with the help of the databases. A database is said to be reliable if it has the capability of adding, editing and retrieving data efficiently. Here the retrieving or accessing of data is very important. A collection of programs that enables you to store, modify, and extract information from a database is called as Database Management System (DBMS). The structured query language (SQL) is one of the famous languages which is used to interact with the database. All the organizations store data by using different type of databases because of the advantages of each and every version of the databases and the variation in the release period of the databases. So if the user of the organization wants to access the data from these different types of databases it is difficult because the user has to switch over to the different front end applications or to the different database environments. So the query builder is created to access data from different types of database such as MS-Access, SQL server, Oracle as that of SQL. The basic difference is by using SQL, data can be accessed only form the Oracle database but by using the Query Builder data from database such as MS-Access, SQL server, Oracle This query builder will also act as a GUI for the user and contains various options which make the users work very simple.

QUERY form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and predefined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex. In natural sciences, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources [22], [13], [25]. Many web databases, such as Freebase and DBPedia, typically have thousands of structured web entities [4], [2]. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases. Many existing database management and development tools, such as Easy Query. Data are stored safely with the help of the databases. A database is said to be reliable if it has the capability of adding, editing and retrieving data efficiently. Here the retrieving or accessing of data is very important. A collection of programs that enables you to store, modify, and extract information from a database is called as Database Management System (DBMS). The structured query language (SQL) is one of the famous language which is used to interact with the database. All the organizations store data by using different type of databases because of the advantages of each and every version of the databases and the variation in the release period of the databases. So if the user of the

ISSN 2348 – 7968

organization wants to access the data from these different types of databases it is difficult because the user has to switch over to the different front end applications or to the different database environments. So the query builder is created to access data from different types of database such as MS-Access, SQL server, Oracle as that of SQL. The basic difference is by using SQL, data can be accessed only form the Oracle database but by using the Query Builder data from database such as MS-Access, SQL server, Oracle This query builder will also act as a GUI for the user and contains various options which make the users work very simple.

QUERY form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and predefined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex. In natural sciences, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources [22], [13], [25]. Many web databases, such as Freebase and DBPedia, typically have thousands of structured web entities.

1.1 Existing system:

Query Builder is an application which provides a graphical user interface (GUI) for working with queries in various database. Complex queries are created in simple manner against any back-end database engine. It has integrated data transfer features via which data may be transferred between different types of data sources. It is like a part of the RDBMS because it acts as a tool to access data from the database. It also has all the advantages of the RDBMS data access tool. This Query Builder is a GUI application for accessing data from different types of Databases like MS-Access, SQL server, Oracle. It acts as a Natural Language Interface for Structured Databases. With this application the user can retrieve data from various databases by using a single query. All the statement of SQL's Data Manipulation language can be used here. Since this application is a Graphical User Interface (GUI), so it is a more user friendly. In addition, it also includes features like printing of result and creating new table with existing result.

1.2. Proposed system:

In this part a connection string is established for the databases, which is a simple process to the user. The connection is made to the database by using the components such as ADO, OLEDB and ODBC driver. This is made available to the user by means of a simple click .After the connection is established the list of tables available in the particular database is displayed in the explorer window. An explorer window is a simple window which is used to display the directory of files of a particular database. Now the connection or a pipeline is set between the database and the application and so we can interact with the structured query language with the database. The query is executed by the user by using simple actions. Here almost all the queries of the data manipulation language such as select, update, edit, delete can be used. The query can be formed by the user by doing various click events without typing it and the query can be displayed to the user. Here the executed query results can be printed, formed as a new table and viewed in maximum size window.

The GUI features can be provided to the user with the use of designing a menu bar and a tool bar. The menu bar will consist of various menus such as file and view. The tool bar will contain a short cut options for print, forming new results and maximizing the result window.

2. Literature Review

2.1 Angis Query:

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. In this query builder a single database is connected to this particular application and by using the application the user retrieves data from it by using a query. ACeDB is a database of genome mapping information for the nematode worm Caenorhabditis elegans. It is accessed through an object-oriented graphical user interface called as Angis query builder. ACeDB is a large program and the following exercises only demonstrate its basic features. However, the graphical nature of the program makes it easy to explore and experiment with. A complete mapping project brings together genetic mapping (the process of locating the genes responsible for given phenotypes, for example using recombination data) and the physical mapping of restriction enzyme sites. **ACeDB** allows the retrieval of these data at various levels, from whole chromosomes down to individual genes. This includes information about alleles, genetic loci, clones, contigs (assemblies of overlapping clones), sequences and much more. Each particular item of information is classified according to its type (e.g. chromosome, allele, author, journal, clone, gene etc.). Related items of information are linked together, so it is possible, for example, starting from a given gene, to retrieve the information available about its physical mapping, its genetic map location, its various alleles, the DNA clone on which it was mapped, the journal article describing it, the authors of this article, the laboratory they work. So the Angis query builder is used as

ISSN 2348 – 7968

an access tool for the retrieval of the information from the ACeDB database.

2.2 Query commands:

In the query command window, option for dealing with the result is designed. The options such as print, save, clear, undo are designed. Listed below are there features
Quit: closes the window.
Help: brings up a relevant on-line help window.
Save: To save the query result as a text format.
Print: sends the contents of the current window to your local printer. This feature may be disabled on some systems.Preserve, Set Auto Preserve, Unset Auto Preserve: these options determine whether the window remains when another window of the same type is called. For example, in this case, if you wanted to display the information about another journal article, this information would be displayed in the same window and would replace the current article, unless Preserve or Set Auto Preserve had been previously selected.
Update: this is used to change the contents of ACeDB's database, and may be disabled on some systems. Neighbors: lists the links available from the current window (that is, the items in bold type). Show selected object as text: displays the information linked to a highlighted item (same as double-clicking on the item). Mailer: sends the text of the window as an email message to a specified address (inoperative on some systems).

2.3. The Query Builder window:

The Query Builder utility greatly simplifies query formulation, especially for users unfamiliar with the organization of the database. With this utility, search commands can be chosen from menus rather than typed in. when the query builder window is opened then a query option is available as shown in the above figure. Here the list of fields available in the database is simply connected to the option button after filling the query then search button is clicked and the result is displayed.

3. Model and Customer Purchasing Behavior

In this part a connection string is established for the databases, which is a simple process to the user. The connection is made to the database by using the components such as ADO, OLEDB and ODBC driver.

To make the project work successfully completed, it is necessary that the problem in it must be well defined, which has been performed in this phase. The problem definition, System requirement, feasibility study has been performed, by means of which the project has been analyzed for its feasibility and found to be feasible in all factors. The various activities such as user interface, external interface, various attributes of the system such as reliability, efficiency, etc are well analyzed. And with the help of all these factors the Dataflow Diagram, Use Case Diagram and Activity diagram of the whole system are designed. The system is built by dividing into various modules and it is checked. After finishing the above activities a connection is set between the application and the database. Thus these three sub function forms the activity of using the database. This function deals with how the user handles the query and this main function consist of various actions which are performed by the user. These actions are Create, Save, Execute, Load and Edit Query. By using his action the user can obtain the result for various queries such as create, edit, delete, select and joins. In this part the results which are obtained after the execution of the query is handled in many ways. The result can be printed and various page set up can be seen. Here the executed result can be formed as a new table. It can also be modified then and there and executed once again. This is how our application works for accessing data from different type of databases.

Database Connection The database such as MS-Access, SQL server and Oracle are connected successfully to our application by using the data access components such as ADO, OLEDB and ODBC. Each database is explored and the various tables in the database are made to display in our application. Query handling the queries in the Data Manipulation Language such as Select, Update, Delete and Insert are built in such a way that the user can access data without typing the query. In addition the queries in the Data Definition Language such as Create and dropping of table are also built. The executed results of the above queries are also verified. Result Handling the Executed results of the queries is displayed to the user using the data grid and the system is designed in such a way that the user can view the result then and there. The Executed results are also be formed as a new table and added to the database. Printing of the result which is the main function is also built and the results are printed.

GUI Features The menu bar is designed with our application which consists of menus such as file and view. A tool bar is also designed which consist of short cut options for print, forming new table with existing result and maximizing the result window that makes work easy for our user.
Selecting a database when the system is started if a user wants to access the data from the database first he has to

ISSN 2348 – 7968

select the type of database from which the data is going to be retrieved. This process is done in this sub function of selecting the database.
Setting the connection after finishing the above activities a connection is set between the application and the database. Thus these three sub function forms the activity of using the database.
Using Query this function deals with how the user handles the query and this main function consist of various actions which are performed by the user. These actions are Create, Save, Execute, Load and Edit Query. By using his action the user can obtain the result for various queries such as create, edit, delete, select and joins.
Using Result in this part the results which are obtained after the execution of the query is handled in many ways. The result can be printed and various page set up can be seen. Here the executed result can be formed as a new table. It can also be modified then and there and executed once again. This is how our application works for accessing data from different type of databases.
Selecting a table after the database is selected the user must select the table in which the data is stored. This action is done by the user in this particular function.

4. Query Forms

Query forms are designed to return the user's desired result. There are two traditional measures to evaluate the quality of the query results: *precision* and *recall* [30]. Query forms are able to produce different queries by different inputs, and different queries can output different query results and achieve different *precisions* and *recalls*, soweuse *expected precision* and *expected recall* to evaluate the expected performance of the query form. Intuitively, *expected precision* is the expected proportion of the query results which are interested by the current user. *Expected recall* is the expected proportion of user interested data instances which are returned by the current query form. The user interest is estimated based on the user's click-through on query results displayed by the query form. For example, if some data instances are clicked by the user, these data instances must have high user interests. Then, the query form components which can capture these data instances should be ranked higher than other components. Next we introduce some notations and then define expected precision and recall. **Notations:** Table 2 lists the symbols used in this paper. Let F be a query form with selection condition OF and projection attribute set AF.Let$D$ be the collection of instances in x (R_F). N is the number of data instances in D.Let d be an instance in D with a set of attributes A ={A1, $A2, \ldots,$ $A_n\}$, where n = |A|.Weuse $d^{\wedge}{}_F$ to denote the projection of instance d on attribute set A_F and we call it a projected instance. $P(d)$ is the occurrence probability of d in D. $P(oFld)$ is the probability of d satisfies OF. $P(oFld)$ e{0, 1}. $P(oFld)$ = 1if d is returned by F and $P(oFld)$ = 0 otherwise.

4.1 Linguistic description and rule based

We define the crisp inputs e(t), g(e(t)) and output u(t) with the linguistic variables $\tilde{e}$(t), $\tilde{g}$ (e (t)) and $\tilde{u}$ (t), respectively. There are N (N = 1, 2, 3,) LVs (Linguistic Values) assigned to each of these linguistic variables. Specifically, we let Pi ={$\tilde{p}$i j : j= 1, 2,…,N} be the input LVs with i = 1 for $\tilde{e}$(t) and i = 2 for $\tilde{g}$(e (t)),: and let $\bar{U}$ ={$\bar{U}$ j: j = 1, 2, … ,N} for $\tilde{u}$ (t). For example, when N = 9, we can assign the following values for both the inputs e(t) and g(e(t)). $\bar{P}$i1 ="Negative Very Large (NV)," $\bar{P}$i2i ="Negative Large (NL)," $\bar{P}$3i ="Negative Medium (NM)," $\bar{P}$4ii ="Negative Small (NS)," $\bar{P}$5i ="Zero (ZR)," $\bar{P}$6i ="Positive Small (PS)," $\bar{P}$7i ="Positive Medium (PM)," $\bar{P}$8i ="Positive Large (PL),"and $\bar{P}$9i ="Positive Very Large (PV)," i = 1, 2. Similarly, we can designate the output when N = 9 with the following linguistic values. U1="Zero (ZR)," U2="Extremely Small (ES)," U3="Very Small (VS)," U4="Small (SM),"U5="Medium (MD)," U6="Big (BG)," U7="Very Big (VB)," U8="Extremely Big (EB)," and U9="Maximum (MX)."

Table I illustrates the controller rule base using N = 9. The rule base is the set of linguistic rules used to map the inputs to the output using the "If. . . Then. . . " format, e.g. "If e(t) is ZR (Zero) and g(e(t)) is PS (Positive Small), Then u(t) is BG (Big)." In the following sections, we refer to a rule in this table by the notation ($\bar{P}^j{}_1$, $\bar{P}^k{}_2$, $\bar{U}^l$), where j, k, l = 1, 2. . . N, e.g. ($\bar{P}^5{}_1$, $\bar{P}^2{}_2$, $\bar{U}^2$) = (ZR, NL, ES).

b) $\pi^*(q) := \pi(p^*(q), q)$ is increasing with respect to q and decreasing with respect to h and the customers' degree risk aversion.
Proposition 4.1 shows that the online retailer should price higher and will get a higher profit if its website's ability to complete transactions without any problems is higher, customers' transaction cost is lower, or if customers are less risk averse.

We introduce the notion of hazard rate order for random variables. Given two nonnegative random variables X and Y with PDFs $fX(x)$ and fY (y), the corresponding CDFs are $FX(x)$ and FY (y).We define that X is no more than Y according to hazard rate order (denoted by $X \leq_{hr} Y$, or equivalently $FX(\cdot) \leq_{hr} FY$ $(\cdot)$) if $rX(v) := fX(v)/FX(v) \geq rY$ $(v) := fY$ $(v)/FY$ (v) for all $v \geq 0$, or equivalently, $FX(v)/FY$ (v) is decreasing in v over $[0,+\infty)$. Hazard rate order, which often appears in decision theory, is an effective way to compare two nonnegative variables. For example, if X follows a uniform distribution

ISSN 2348 – 7968

random over [0, AX] and Y follows a uniform distribution [0, AY], then $X \leq_{hr} Y$ is equivalent to $AX \leq AY$.

Proposition 4.2:

Suppose that $pFX\pi^*F(p^*FY$and $\pi^*FY)$are respectively the optimal price and the single-period profit corresponding to CDF $FX(\cdot)$(CDFFY $(\cdot)$). If both $FX(\cdot)$ andFY $(\cdot)$ have increasing hazard rates and $FX(\cdot) \leq_{hr} FY$ $(\cdot)$, then $p^*FY \geq p^*FX$ and $\pi^*FY \geq \pi^*FX$

Proposition 4.2 indicates that both the optimal price and single-period profit are higher if customer valuations for the the customers). For simplicity, we say that y is increasing (decreasing) with respect to customer valuations if $FX(\cdot) \leq_{hr} FY$ $(\cdot)$ implies that $yX \leq yY$ $(yX \geq yY$), where yX and yY are two quantities corresponding to CDFs $FX(\cdot)$ and FY $(\cdot)$, respectively.

4.2 Upgrading thresholds change with model parameters:

It investigates how the firm's optimal total discounted profit and the upgrading thresholds change with model parameters.

TABLE I

LINGUISTIC RULE 1

	T for Period 1	T for Period 2	T for Period 3	T for Period 4	T for Period 5
A=35	0.4401	0.5662	0.6918	0.8171	0.9423
A=40	0.4483	0.574	0.6994	0.8245	0.9496
A=45	0.4546	0.5799	0.7051	0.8303	0.9554
A=50	0.4593	0.5847	0.7098	0.8348	0.9599

The optimal threshold T changes with model parameters. To this end, we do some numerical examples in which $F(x)$ is the CDF of uniform distribution over[0,A], n = 5, N = 1000, $u(x)$ = 1 − exp(ax), Ki = 500, i =1, 2, . . . 5, $q1$ = 0.5, $q2$ = 0.625, $q3$ = 0.75, $q4$ = 0.875, $q5$ 1, and β = 0.9. First, we set h = 4 and a = 0.01 and investigate how threshold T changes with A. The results are listed in Table I. It demonstrates that T increases as A increases, which means that the firm is more likely to upgrade its web system when customer valuations for the product are higher. This is because it is more profitable to upgrade the web system when the product is more attractive to the customers.

TABLE II

LINGUISTIC RULE 2

	T for Period 1	T for Period 2	T for Period 3	T for Period 4	T for Period 5
h=2	0.4497	0.5748	0.6999	0.8249	0.9499
h=4	0.4483	0.574	0.6994	0.8245	0.9496
h=6	0.4458	0.5726	0.6984	0.8239	0.9492
h=8	0.4409	0.5703	0.6971	0.8229	0.9485

set a = 0.01 and A = 40 and investigate how T changes with h. The results are listed in Table II. We find that T becomes smaller when customer transaction cost becomes higher. This indicates that the firm is less likely to upgrade its web system when customer transaction cost is higher. It is also very natural because it is less profitable to upgrade the web system when customer transaction cost is higher, which results in smaller population of customers who try to buy the product.

TABLE III

LINGUISTIC RULE 3

	T for Period 1	T for Period 2	T for Period 3	T for Period 4	T for Period 5
a=0.01	0.4483	0.574	0.6994	0.8245	0.9496
a=0.03	0.4501	0.5751	0.7001	0.8251	0.95
a=0.05	0.4522	0.5764	0.7009	0.8256	0.9505
a=0.07	0.455	0.5778	0.7018	0.8262	0.9509

set h = 4 and A = 40 and investigate how T changes with customers' degree of risk aversion. The results are listed in Table III, in which larger a represents higher degree of risk aversion. The result demonstrates that the firm is more likely to upgrade the web system when facing more risk-averse customers. The explanation is as follows. When customers are more risk averse, they are more sensitive to the possibility of transaction failure. Thus, upgrading the web system, which increases TSP, attracts more customers to buy the product. This means that upgrading the web system is more profitable when customers are more risk averse; thus, T increases with customers' degree of risk aversion. Further numerical analysis demonstrates that T has no explicit relationship with period number i(i = 1, 2, . . . , 5). The relationship between threshold T and period number i depends on the interactions of several parameters such as qi and Ki(i = 1, 2, . . . , 5).

5.

ISSN 2348 – 7968

An interesting question is: What happens if the firm ignores query decisions based on the belief that user think $q = 1$? In this section, we will answer this question. From Part (a) of Proposition 4.1, we have

$$p^*(1) > p^*(q), \quad \text{for any } 0 < q < 1$$

which shows that the user will price higher if it ignores strategic behavior. Let $\tilde{\pi}(q) := \pi(p^*(1), q)$ be the online retailer's single-period profit with TSP q when ignoring customer strategic behavior. Similarly as in Section IV, let $\tilde{V}n+1(q) = 0$ for any $q \in [q, qn]$ and $Vi(qi) = \max_{q \in [qi, \bar{q}i]} \{-Ki\,\delta(q - qi) + \pi(q) + \beta\, Vi+1(q)\}$

Then, $\tilde{V}i(qi)$ stands for the online retailer's total discounted profit from period i to period n with TSP qi when ignoring customer strategic behavior. Denote the online retailer's profit loss rate of ignoring customer strategic behavior by

$$r = \frac{V1(q) - \tilde{V}1(q)}{v1(q)}$$

It investigate how large r is and how r changes with model parameters. Let $F(x)$ be the CDF of uniform distribution over $[0,A]$, $u(x) = 1 - \exp(ax)$, $n = 3$, $N = 1000$, $Ki = 1000$, $i = 1, 2, 3$, $q0 = 0.5$, $q1 = 0.6$, $q2 = 0.7$, $q3 = 0.8$, and $\beta = 0.9$

keywords queries. The relevance score between the keywords and the query form [12] can be incorporated into the ranking of form components at each step. The basic advantage of this application is that the user can work with the databases without having the basic knowledge of forming the query. Since all the basic queries are built here it makes the users work easy. One of the defects of this application is that we are not able to form queries for in depth concepts such as Views and trigger. In Future we can enhance this application by adding some other databases and other features related to it.

These results are robust since they remain true for several extensions and variations of the model.

First, in this paper, we assume that the customers' valuation follow a distribution with increasing hazard rate, and for the situation where upgrade cost $Ci(q - qi)$ is a general concave function, our discussions are limited to the risk-neutral customers (i.e., $u(x) = x$) and the uniform valuation distribution. It will be worthwhile to relax these limitations to get the optimal upgrading policies. Second, this paper focuses on pricing and web system upgrading problems of a monopolist. It is more interesting to study similar problems when online retailers face competition from rivals. Third, this paper assumes that the TSP does not depend on the number of customers who purchase the product. It is more interesting to study similar problems when the TSP depends on the number of customers who try to purchase the product

6. Future Enhancement

In this paper we propose a dynamic query form generation approach which helps users dynamically generate query forms. The key idea is to use a probabilistic model to rank form components based on user preferences. We capture user preference using both historical queries and runtime feedback such as click-through. Experimental results show that the dynamic approach often leads to higher success rate and simpler query forms compared with a static approach. The ranking of form components also makes it easier for users to customize query forms. As future work, we will study how our approach can be extended to non relational data. As for the future work, we plan to develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, we can add a text-box for users to input some

7. Conclusion

The basic advantage of this application is that the user can work with the databases without having the basic knowledge of forming the query. Since all the basic queries are built here it makes the users work easy. One of the defects of this application is that we are not able to form queries for in depth concepts such as Views and trigger. In Future we can enhance this application by adding some other databases and other features related to it.

References

[1]. Abraham Silberschatz, Henry F.Korth, S. Sudarshan, 'Database *System Concepts*' (pages: 75-162), McGraw-Hill Higher Education, 2002.

ISSN 2348 – 7968

[2]. S. Abiteboul, 'Querying *semi-structured* data', In Intl. Conf. on Database Theory, 1997.

[3]. S. Abiteboul, R. Hull, and V. Vianu, 'Foundations *of Databases'* (Pages: 143 – 187), Addison Wesley, 1995.

[4]. S. Abiteboul and V. Vianu,' *Regular path queries with constraints'*, In Proc.ACM Symp.on Principles of Database Systems, 1997.

[5]. S. Adali, K. Candan, Y. Papakonstantinou, and V. Subrahmanian, '*Query caching and optimization in distributed mediator systems'*, In Proc. ACMSIGMOD Conf. on the Management of Data, 1996.

[6]. R. Agrawal, G. Psaila, E. Wimmers, and M. Zaot, ' *Querying shapes of histories*', In Proc. Intl. Conf. on Very Large Databases, 1995.

[7]. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, '*When is nearest* neighbor *meaningful? In IEEE International Conference on Database*. Theory', 1999.

[8]. M. Carey, D. Chamberlin, S. Narayanan, B. Vance, D. Doole, S. Rielau, R. Swagerman,and N. Mattos. O-O, ' *what's happening to DB2?'* In Proc. ACM SIGMOD Conf. on the Management of Data, 1999.

[9]. U. Chakravarthy, '*Architectures and monitoring techniques for active* d*atabases: An evaluation. Data and Knowledge Engineering'*, In Proc. Intl.Conf. on Very Large Databases, 1995.

[10]. S. Chaudhuri and K. Shim, 'Optimization *queries with aggregate views'*, InIntl. Conf. on Extending Database Technology, 1996.

Prof.Mrs.M.Vasuki is currently working as Assistant Professor in the Department of Master Of Computer Application at Sri.Manakula Vinayagar Engineering College, Madagadipet, Pondicherry, India. She received Master of Computer Application (MCA) degree in 1994 from Bharathidasan University, Karur, and completed M.Phil in 2005 from Manonmaniam Sundaranar University, Thirunelveli, and Completed M.Tech in 2012 from SRM University, Chennai, Tamil Nadu, India. Her research interests are Database Management Systems.

R.Rajeswari obtained her BSC degree from Idhaya College of Arts and Science, Pondicherry, India. She is currently pursuing her MCA degree in at Sri.Manakula Vinayagar Engineering College, Madagadipet, India. Her areas of research interest accumulate in the areas of Database Management Systems and Data Mining.

R.Anandhi obtained her BCA degree from Bharathidasan University, Tamil Nadu, India. She is currently pursuing her MCA degree in at Sri.Manakula Vinayagar Engineering College, Madagadipet, India. Her areas of research interest accumulate in the areas of Database Management Systems and Data Mining.