iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VI **Month of publication:** June 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Logical Spreadsheet Management System

Ramya R[1], Dinesh P S[2], Iniya Shree S[3]
[1]Department Of Computer Science and Engineering, Bannari Amman Institute of Technology, Sathy
[2]Department Of Computer Science and Engineering, Bannari Amman Institute of Technology, Sathy
[3]Department Of Computer Science and Engineering, Bannari Amman Institute of Technology, Sathy

Abstract-: ***Separate operators of the relational algebra were implemented in the spreadsheets. The ability to express relational queries and the users of spreadsheets can perform relational data transformations in the spreadsheet itself. End users can still work in the vanilla spreadsheet, benefit from its features like data analysis and visualization, while the complex formulas are generated by a tool that allows to express them in a better suited high-level language and avoids errors. Either they are not yet ready to migrate to a database, or they do not want to migrate at all. They can define their queries using a high-level language and then get their execution plans in a plain vanilla spreadsheet. No sophisticated database system, no spreadsheet plugging or macros are needed. The functions available in spreadsheets impose severe limitations on the algorithms one can implement. spreadsheet can implement, as we demonstrate, Depth-First-Search and Breadth-First-Search on graphs, thereby reaching beyond queries.***

Key words: relational algebra, graphs, data analysis

I. INTRODUCTION

Spread sheets are the desktop counterpart of databases and OLAP in enterprise-scale computing. They serve basically the same purpose data management and analysis, but at the opposite extreme of the data quantity scale. Spread sheets are very popular, and are often described as the very first killer app for personal computers. Today they are used to manage home budgets, but also to create, manage and examine extremely sophisticated models and data arising in business and research. Despite that encouragement, relatively little research has been devoted to spread sheets and consequently they are still poorly understood. In particular, 16 years later Excel users show up at community forums asking for help in performing database operations on their spread sheet data. Probably for the same group of users Google provides in its spread sheet a very useful QUERY function with SQL-like syntax. It is used to run Google Visualization API Query Language across data. However, this function does not permit joining relations, and is incompatible with other spread sheetsystems. The second notable fact is that spread sheet language of formulas of Excel has become a de facto standard. It is implemented in a large number of spread sheet systems, available for all major operating systems and hardware platforms, starting from handhelds and ending in the cloud, from proprietary to open source. Computer applications in the form of formula-only spreadsheets are therefore highly portable, probably to the extent comparable with Java byte code. Spreadsheet systems can be regarded as virtual machines, offered by various vendors, on which spread sheet applications can be run. It is therefore extremely surprising that those machines are predominantly programmed manually, with no compilers producing spreadsheet code from higher-level languages. It is an extended version of an earlier paper, which demonstrated as a proof of concept, that Excel and other spread sheets are capable of storing and querying relational data. Separate operators of the relational algebra were implemented in the spreadsheets. It offers a fully automated method to construct spreadsheet implementations for a wide class of relational data transformations. All operators of relational algebra have re-implemented to accept a variable number of input columns and to support NULL values. On top of these implementations it created a tool producing formulas only spreadsheet implementations of queries written in SQL.In the same way it addresses our second point: query compiler tool converts the high-level language into the language of spread sheetformulas. Thefull automation of the translation process reduces the number of human-introduced errors in the spreadsheet application. Users work invanilla spreadsheet, benefit from its features like data analysis and visualization, while the complex formulas are generated by the tool that allows expressing them in a better suited high level language and avoids errors.

II. LITERATURE SURVEY

Numerous works are there which discuss various methods to support high-level design of spreadsheets. Some works consider spreadsheets from the functional programming prospect. There are different databases known i.e. MySQL ,oracle, mongodb etc. Using which it can store our data in table formats. These databases are difficult to use and in need to install on system. Already a

work is there which can translate Relational Queries into Spreadsheets but there are some different problems like discontinuity not permit the joining relations and is incompatible with other spreadsheets and no ability to express relational queries, and the users of spreadsheets cannot perform relational data transformations in the spreadsheet itself. The problem of explicit relational algebra and SQL in spreadsheets has not been considered in the previous paper that adapt here.

The proposed work is an extension of the set of spreadsheet functions by a carefully designed database function, whereby the user can specify and later execute SQL queries in a spreadsheet-like style, one step at a time. These additional operators are executed by a classical database engine running in the background. Our augmentation means that exactly the same functionality can be achieved by the spreadsheet itself. The work named Query by Excel to extend SQL by spreadsheet-inspired functionality, allowing the user to treat database tables as if they were located in a spreadsheet and define calculations over rows and columns by formulas approximating those found in spreadsheets. In another work, a spreadsheet interface is offered for specifying these computation,which had to be specified in an SQL-like code in the earlier papers.

Andrew Witkowski (2003), suggested that automatically translation of Excel computation into SQL. Though spreadsheets and Excel offer an attractive user interface and easy computable model, they lack parallelization and do not offer scalable computation. Thus the system called QueryByExcel (QBX) is been introduced. It combines interactive skill of Excel with Relational computation facility which also allows scalability.

Main goal was to translate Excel computation to SQL and end user extensions to Excel formulas, menus to perform relational operations on relational database management system. Some of the features of QBX are:

A. Users build and edit model using Excel formulas. The model will automatically be translated into SQL and relational views.
B. Users can easily perform relational operations with Excel without writing a SQL statement.
C. The operators are translated to SQL applying scalability of relational database system to Excel.

Computerized spreadsheet system works successfully for automatic calculation of values using formulas. But it limits usefulness by having restrictions on formulas which are used for calculations must be a function.He has also suggested and proposed spreadsheet like computation in relational database system through extensions to SQL, allowing OLAP tools to handle the user interface. Spreadsheet is a successful analytical tool for business data but it has scalability problem. SQL doesn't support for dimensional array based computations which are good in OLAP scale computation while spreadsheets and specialized MOLAP engines are much good for constructing formulas for mathematical modelling. Main objective was to bring the benefits of additional programming language features to a system, maintain the compatibility and the usability advantage by end users. Thus the focus is on end users than professional users.

III. EXISTING METHOD

An interface for spreadsheet is more intuitive for many non-technical database users compared to traditional alternatives like visual query builders. The construction of such a direct manipulation interface may appear straightforward, but there are some significant challenges. First, individual direct manipulation operations cannot be too complex, so expressive power has to be achieved through composing (long) sequences of small operations. Second, all intermediate results are visible to the user, so grouping and ordering are material after every small step. Third, users often find the need to modify previously specified queries. Since manipulations are specified one step at a time, there is no actual query expression to modify. Suitable means must be provided to address this need. Fourth, the order in which manipulations are performed by the user should not affect the results obtained, to avoid user confusion.

It address the aforementioned challenges by designing a new spreadsheet algebra that: i) operates on recursively grouped multi-sets, ii) contains a selectively designed set of operators capable of expressing at least all single-block SQL queries and can be intuitively implemented in a spreadsheet, iii) enables query modification by the notion of modifiable query state, and iv) requires no ordering in unary data manipulation operators since they are all designed to commute. A prototype has been built for the implementation of the spreadsheet algebra and show, through user studies with non-technical subjects that the resultant query interface is easier to use than a standard commercial visual query builder.

A. Spreadsheets in RDBMS for OLAP

One of the critical deficiencies of SQL is lack of support for n-dimensional array-based computations which are frequent in OLAP environments. Relational OLAP (ROLAP) applications have to emulate them using joins, recently introduced SQL Window Functions [18] and complex and inefficient CASE expressions. The chosen place in SQL to specify calculations is the SELECT query, which is extremely limiting and forces the user to generate queries using nested views, sub queries and complex joins. SQL-

query optimizer is pre-occupied with determining efficient join orders and choosing optimal access methods and largely disregards optimization of complex numerical formulas. Execution methods concentrated on efficient computation of a cube [11], [16] rather than on random access structures for inter-row calculations which has created a gap that has been filled by spreadsheets and MOLAP engines, which are good at formulas for mathematical modeling but lack the formalism of the relational model, are difficult to manage, and exhibit scalability problems. This paper presents SQL extensions involving array based calculations for complex modeling. In addition, we present optimizations, access structures and execution models for processing them Spreadsheets are a common tool in end-user programming. But even while important decisions are based on spreadsheet computations, spreadsheets are poorly documented software and the differences between simple on-shot computations and large, long-living sheets are not well understood. Like other software, production spreadsheets are subject to repeated maintenance cycles. Consequently, as with conventional software, short maintenance cycles and poor documentation tend to decrease their quality. An approach to help maintainers to understand the structure of large spreadsheets as well as to zoom into certain parts of the spreadsheet is introduced here..

To cope with large sheets, our approach features two levels of abstraction: logical areas and semantic classes. Our claim of translating SQL queries into spreadsheets does not mean, that it can translate the algorithms typical RDBMS systems employ to implement SQL. In particular, most of the algorithms used are quadratic and hence inefficient if used on large datasets. Moreover our translation tool in its present form does not perform optimization. Translating SQL queries into spread sheets does not mean, that can translate the algorithms typical RDBMS systems employ to implement SQL. In most of the algorithms the use of quadratic and hence inefficient if used on large data sets. So far, no system exists based on translating SQL Queries into spreadsheet database. This is required for the developers working on migration of legacy system to newer ones. Recently this work is done manually.

The people who are working on converting legacy script to newer technology required to do this work manually that is incompetent and mostly time overriding. This also increases the probability of error because of increased human interaction. So, there is need to need to automate this process. To triumph over this problem SQL Parser is the option. The input is taken from the user in the form .sql file format. After that, it splits the query and processes it to form the output which will be shown in spreadsheet (Microsoft® Excel). It is called as Mapping Document. Our claim of translating SQL queries into spreadsheets does not mean, that they can translate the algorithms typical RDBMS systems employ to implement SQL. In particular, most of the algorithms they use are quadratic, and hence inefficient if used on large data sets. Moreover, our translation tool in its present form does not perform optimization.

B. Finding High-Level Structures in Spreadsheet Programs

Spreadsheets are a common tool in end-user programming. But even while important decisions are based on spreadsheet computations, spreadsheets are poorly documented software and the differences between simple on-shot computations and large, long-living sheets are not well understood. Like other software, production spreadsheets are subject to repeated maintenance cycles. Consequently, as with conventional software, short maintenance cycles and poor documentation tend to decrease their quality. Maintainers can understand the structure of large spreadsheets as well as to zoom into certain parts of the spreadsheet.

To cope with large sheets, our approach features two levels of abstraction: logical areas and semantic classes. Our claim of translating SQL queries into spreadsheets does not mean, that it can translate the algorithms typical RDBMS systems employ to implement SQL. In particular, most of the algorithms used are quadratic and hence inefficient if used on large datasets. Moreover our translation tool in its present form does not perform optimisation. Translating SQL queries into spread sheets does not mean, that can translate the algorithms typical RDBMS systems employ to implement SQL. In most of the algorithms the use of quadratic and hence inefficient if used on large data sets.

The developers which are working on converting legacy script to newer technology required to do this work manually which is inefficient and mostly time consuming. It also increases the probability of error because of increased human interaction. So, there is need to need to automate this process. To overcome this problem SQL Parser is the option. It takes the input from the user in the form .sql file format. After that it Splits the query and Processes it to form the output which will be shown in spreadsheet (Microsoft® Excel). It is called as Mapping Document. Our claim of translating SQL queries into spreadsheets does not mean, that they can translate the algorithms typical RDBMS systems employ to implement SQL. In particular, most of the algorithms they use are quadratic, and hence inefficient if used on large data sets. Moreover, our translation tool in its present form does not perform optimization.

IV. PROPOSED METHODOLOGY

A. System description

One of them is an efficient sorting algorithm, implemented by spreadsheet formulas, improving on the quadratic sorting. The present algorithm is O(n $\log^2$ n),but requires 4 log n columns to sort n items. The other two algorithms go beyond standard SQL. It also presents a recursive implementation of Breadth-First-Search for directed acyclic graphs and an iterative implementation of Depth-First-Search for arbitrary graphs. This sheds additional light on real computational capabilities of spreadsheets and their ability to express recursive queries.

Spreadsheets act as a computational tool for data storage and on querying the relational data. Spreadsheet is capable of implementing all data transformations defined in SQL, based on building querying formulas. The main contribution is building an automated method which implements all the operators of relational algebra into spreadsheet, Microsoft Excel. Next is introducing a tool or compiler called as Query-Converter from high level language into formula based spreadsheet language. The tool implements the complex formulas given by the user in SQL or relational algebra query, reducing human generating errors. The objective is to translate the database queries into formula based spreadsheets. To build a tool for translating queries into spreadsheets compatible for other spreadsheets offered by various vendors of operating system and to implement some of the traversing algorithms such as BFS and DFS.

B. Advantages

1) To develop optimisations for SQL queries translated into spreadsheets.
2) Implemented a few specific algorithms:alinearithmic sorting procedure and two graph traversing algorithms:BFS and DFS.
3) Computer applications in the form of formula-only spreadsheets are therefore highly portable, probably to the extent comparable with Java bytecode. Spreadsheet systems can be regarded as virtual machines, offered by various vendors, on which spreadsheet applications can be run.
4) Reduce amount of user work.
5) Automatic recalculations upon data change.
6) Efficient complexity formula.
7) Very portable spreadsheets.

C. The Database Architecture

The Database Architecture of relational database has following procedure:

1) The tool Query-Converter implements the SQL or relational query in the .xlsx file form.
2) The result obtained contains a single worksheet including number of columns for data-tables and those columns performing the computation.
3) There are two rows of formulas to be implemented in which the second row contains all the detail information of the formulas that the user enters.
   When the user enters data into tables, the tool automatically computes the query and output it in the form of columns. The list of operators for implementation is: Sorting, Duplicate removal, selection, projection, union, difference, Cartesian product, grouping with aggregation, and two additional operators are semi-join and join.Query- Converter is an automated tool converts the SQL query into the spreadsheets.

4) The steps are as follows:

a) The SQL expression is translated into relational algebra expression as per the algorithm below.

b) Then relational algebra expression translated into spreadsheet as per operator implementations.
c) Translate the subquery-free part. De-correlate the exists subqueries
d) De-correlate the not exists subqueries.
e) Apply the project.

D. System Architecture

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

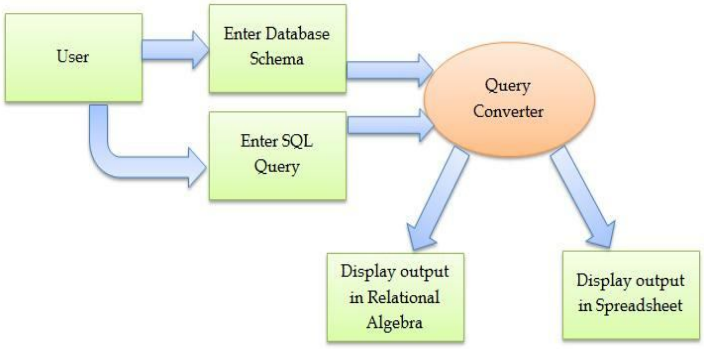


Fig.1 Query-Converter

It explains End user needs to first define the database schema.User has to give the database query (SQL) as an input to the system.The system analyses the query and gives output in two ways i.e. the relational algebra and in the form of single worksheet of the spreadsheet.Finally the system generates the output and displays.

V. CONCLUSION AND FUTURE WORK

The proposed automated tool Query-Converter makes it easy for users to analyze and manage the data on large scale reducing the human generated errors while writing the database queries. Also certain drawbacks of spread sheets and SQL are overcome, improving performance of the system. The main goal is to translate the queries into spread sheets which generate results in the spread sheet form. The discussed graph traversal algorithms can be used to implement recursive queries over hierarchies which are encountered in practice. It can be believed that the discussed approach is not only applicable to end-user programmers (who are generally having more difficulty in learning and using new programming languages), but also to the professional users to design extension of programming languages and environments.SQL have established that it can be automatically converted into spreadsheet code. Apart from SQL, have also applied a few specific algorithms: a linearithmic sorting procedure and two graph traversing algorithms: BFS and DFS.

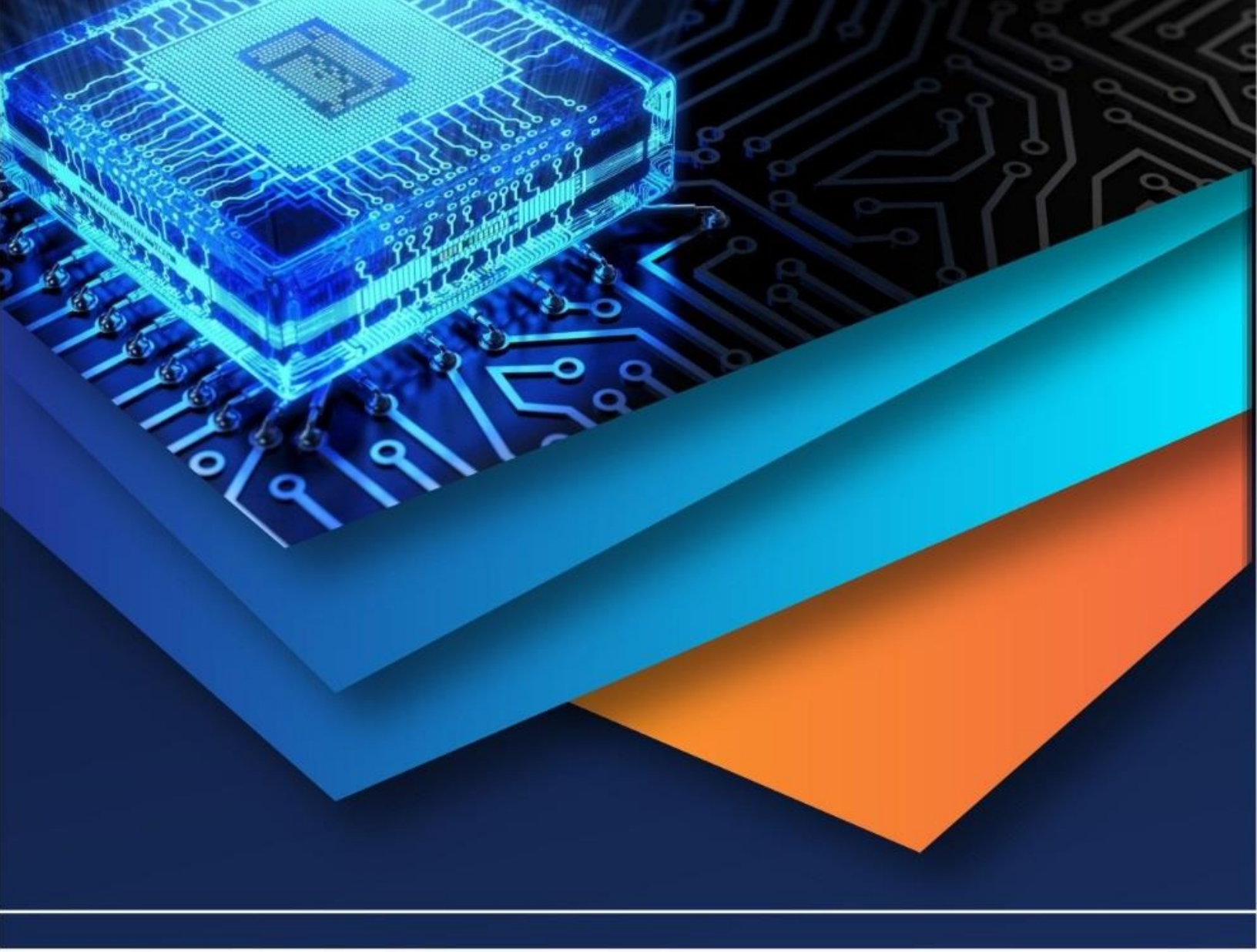
As the next steps plan to develop optimizations for SQL queries translated into spreadsheets. The interesting part is whether spreadsheets can naturally implement other models of databases, like semi-structured or object-relational ones.. This requires translating Google specific functions QUERY, SORT, FILTER and UNIQUE which are not recognized by other spreadsheet systems. SQL Parser is a tool to analyze the SQL queries using various mathematical and functional aspects of language. This will create a final output as an understandable document in .csv or .xlsx format. Hence, this tool will be useful for programmers and business analyst in order to understand the functionalities of the queries and SQL based systems in general.

REFERENCES

[1] A. Witkowski, S. Bellamkonda, T. Bozkaya, N. Folkert, A. Gupta, L. Sheng, and S.Subramanian, Business modeling using SQL spreadsheets, in VLDB 2003: Proceedings of the 29th international conference on Very large data bases. VLDB Endowment, 2003, pp. 11171120.
[2] A. Witkowski, S. Bellamkonda, T. Bozkaya, G. Dorman, N. Folkert, A. Gupta,L. Shen, and S. Subramanian, Spreadsheets in RDBMS for OLAP, in SIGMOD 03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2003, pp. 5263
[3] B. Liu and H. V. Jagadish, "A spread sheet algebra for a direct data manipulation query interface," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 417–428
[4] B. Gates, "Investing in research, SIGMOD conference 1998 keynote speech, video," ACM SIGMOD Digital Symp. Collection, vol. 1, no. 2, 1999
[5] G. Inc. Query Language Reference (Version 0.7) [Online].Available: https://developers.google.com/chart/interactive/docs/querylanguage, accessed 2014.
[6] H. Garcia-Molina, J. D. Ullman, and J. Widom, Database System Implementation. Prentice-Hall, 2000
[7] J. Tyszkiewicz, "Spread sheet as a relational database engine," in Proc. ACM SIGMOD Int. Conf. Manag. Data, 2010, pp. 195–206
[8] J. V. den Bussche and S. Vansummeren, Translating SQL into the relational algebra, Lecture material, Universiteit Limburg, lecture INFO-H-417: Database Systems Architecture.
[9] Lakshmanan, S. N. Subramanian, N. Goyal, and R. Krishnamurthy, "On query spread sheets," in Proc. Int. Conf.Data Eng., 1998, pp. 134–141.
[10] Michael Kassoff The Knowledge Engineering Review, Vol. 22:3, 281–295.2007, Cambridge University Pressdoi:10.1017/S0269888907001154 Printed in the United Kingdom.
[11]] P. W. P. J. Grefen and R. A. de By, A multi-set extended relational algebra – a formal approach to a practical issue, in ICDE. IEEE Computer Society
[12] R. Abraham and M. Erwig, "Type inference for spreadsheets, "in Proc. 8th ACM SIGPLAN Symp. Principles Prac. Declarative Program.2006, pp. 73–84.
[13] Simon Peyton Jones, Margaret Burnett, Alan Blackwell. Proc International Conference on Functional Programming, Uppsala, Sept 2003 (ICFP'03), pp 165-176, 12 pages.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653

[14] SirMille.(2012,Feb.).inner/outer/full join tables? [Online]. Available: http://www.mrexcel.com/forum/excel-questions/612236-inner-outer-full-join-tables.html.
[15] Stefanaalten.(2013,May).Combining two lists into one big list[Online]. Available: http://www.mrexcel.com/forum/excelquestions/704217-combining-two-lists-into-one-big-list.html.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY