ISSN 2395-1621

TRQ in to Spreadsheet using Graph Traversing with Graphical Representation

[#1]Tejashri Awate, [#2]Pooja Dhormale, [#3]Priti Kumbhojkar, [#4]Anushree Reddy, [#5]Ms.Sheetal Kusal

[1]awatetejashri44@gmail.com
[2]pooja.dhormale@gmail.com
[3]kpriti2009@gmail.com
[4]anushreereddy30@gmail.com

[#12345]Department of Information Technology
G.H.Raisoni Collage of Engineering and Technology,
Pune, India

ABSTRACT

Spreadsheets are among the most commonly used applications for data management and analysis. They combine data processing with very several supplementary features: statistics, visualization, reporting, linear programming solvers, Web queries periodically downloading data from external sources, etc. However, the spreadsheet ensample of computation still lacks sufficient analysis. In this paper we determine that a spreadsheet can implement all data transformations definable in SQL, simply by utilizing spreadsheet formulas. We provide a query compiler, which translates any given SQL query into a worksheet of the same definition, including NULL values. Thereby database operations become available to the users who do not want to immigrate to a database. They can define their queries using a high-level language and then get their execution plans in a plain vanilla spreadsheet. The functions available in spreadsheets introduce limitations on the algorithms one can implement. In this paper we offer O(nlog2 n) sorting spreadsheet, using a non-constant number of rows, and, specially, Depth-First-Search and BreadthFirst-Search on graphs.

ARTICLE INFO

Article History

Received: 28th May 2016

Received in revised form :

28th May 2016

Accepted: 1st June 2016

Published online :

2nd June 2016

Keywords: **Relational Databases, Spreadsheets, Query languages.**

I. INTRODUCTION

SPREADSHEETS 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. Spreadsheets are very popular, and are often describedas 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. A lot of users today find the true databases complex enough that they simply go into either the word processor, with the tabletype capabilities, or into the spreadsheet, which is a little more typical, and use that as their way of structuring data. And, of course, we get a huge discontinuity because, as we want to do database-type operations, the spreadsheet isn't set up for that. And so then we have to learn a lot of new commands and move your data into another location. What we'd like to see is that even if we start out in the spreadsheet, there's a very simple way then to bring in software that uses that data in a richer fashion, and so you don't see a discontinuity when you want to move up and do new things.

Probably for the same group of users Google provides in its spreadsheet 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 spreadsheet systems.The second notable fact is that spreadsheet language of formulas of Excel has become a de facto standard. It is implemented in a large number of spreadsheet 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 bytecode. Spreadsheet systems can be regarded as virtual machines, offered by various vendors, on which spreadsheet 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.

II. LITERATURE SURVEY

We envisage the main group of potential users of our work to be characterized by the following: They are experienced and relatively proficient users of spreadsheets, mainly Microsoft Excel. They are approaching the limits of spreadsheet abilities. Either they are not yet ready to migrate to a database, or they do not want to migrate at all. One of us (J.Ty.) was active at the MrExcel.com forum where users of Excel can exchange tips and solutions, performing a kind of participant observation. It has turned out that requests to help in joining datasets are not uncommon. One type of requests for help comes from users who are aware of database operations and clearly state what they need, as in the threads [2], [6]. The other type of requests comes from users, who describe the operation they need in plain words. The first discussion [3] concerns social research. Its initiator needs to self join a table on employment in companies, to detect pairs who work together in the same company. This task can be very concisely formulated as a query in SQL or the relational algebra. The ability to compile such a query into Excel formulas will significantly reduce the amount of necessary user work. In the second discussion [5] a user needs to join two tables of sensor data: humidity and temperature, both keyed by date. This meteorological application amounts to a textbook outer join: matching humidity and temperature records are to be identified and non-matched records are to be retained. The topic of the third discussion [4] is the problem of assembling services from components. The requesting user needs to join an association table of components with the service table that relates services with components. This time the problem reduces to an interesting inner join that leads to a non-4NF result. To summarize, real users do need to join Excel data tables in various ways (self-, outer-, inner-, non-NF). A compiler of queries can be a valuable ally in their efforts. It is instructive to have a look at the thread [6]. The user wants help in performing a join, and wants to do that in MsQuery. However, the Excel-only solution turns out to require just two formulas per data sheet and a few clicks to remove duplicates. A formula-only solution would be more complex, but with the advantage of automatic recalculation upon data change.

III. PROPOSED SYSTEM

Following are the steps we can perform to convert relational query into spreadsheet.

1. Input to the browser is query.
2. Using jsp create java servlets pages ,pagesdynamically for HTML and ASP.
3. After that it gives to the JAVA bean for communication,it usesful for entity,massageS.
4. further the databse allow to retrieve data from database.

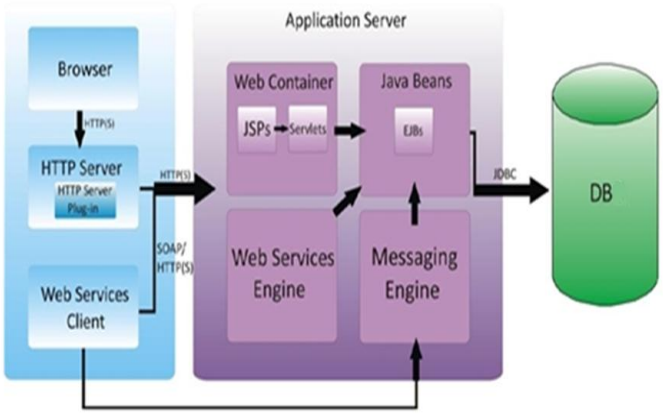

Fig 1. System architecture

While we have identified the extent of modifications needed for databases to effectively support an interface, our current implementation and discussion focuses on enhancing some core components. Naturally, there are other components that require modification, such as the transaction manager, and we leave them for future work. Interface Storage Manager. In this unified framework, a spread-sheet not only has tabular data, corresponding to relational tables in the underlying database, but also has other interface data, e.g., formulae or data entered by the user. This interface data requires special treatment as it does not have a schema. The interface storage component stores this data as a collection of cells. To enable efficient retrieval for a given range, the component groups the cells together by proximity and splits the groups into data blocks as required by the underlying storage. To enable efficient access, the blocks are further indexed by a two-dimensional indexing method. Relational Storage Manager. Our unification semantics demand that the schema changes to the tabular data, which we persist in the database as relational tables, should be very efficient, almost as efficient as changes to tuples. With an insight to reduce the disk blocks to update during a schema change, the relational storage manager uses a hybrid of column-store and row-store to physically store the table. Here, data is structured along a collection of attribute groups, thereby radically reducing the disk blocks that need an update during a schema change.
Interface Manager. The interface manager keeps close tabs on the data presented to the user. For every data item, e.g., the output of a query, a table imported from the database, that is displayed on the interface, the presentation manager assigns a context; a context comprises a positional address along with a reference to the sheet. This context can then be utilized to enable functionalities such as two-way sync and relative addressing. Along with positional addressing, the interface manager allows a two-way synchronization for the tables displayed on the inter-face. Since primary keys are a natural way to identify tuples in a relational database, the interface manager maintains a mapping be-tween a tuple's key attribute and its corresponding location. This enables translation of an update on the interface, having a locational context, to the underlying relational database, which requires a key to uniquely identify a tuple. Compute Engine. To optimally support interface interactions and

data updates, we introduce a new component termed as "compute engine". By using ideas like shared computation, the compute engine enables efficient handling of formulae and queries with positional referencing, e.g., DBSQL. It performs computations asynchronously, free from a user's context, as updates are made to either the interface or the database. It further improves the interface's interactivity by prioritizing the computation for visible cells.

Implementation

Application Creation & Inserting Details:

In this phase, we are going to develop an ecommerce application for mobile shop. In this application owner as to maintain the details of the each product (incoming and outgoing). To maintain all products details by using excel sheet. All the product information's are stored in excel sheet for easy access. Owner has to insert all the product details, customer details, workers details and bill details in their own excel database.

Implementing algebraic notation:

Once excel database has been created, we need to implement the algebraic notation on it. Here we are going to perform selection, union, difference and duplicate removal information. Translating all relational queries into excel sheet for performing notation. All the database will not allow duplicate records into their database. Every application has to maintain removing of duplicate values. In this application, algebraic notations are used when entering product details, viewing product details and entering bill details. Performing Aggregation function In excel sheet we need to implement aggregation functions. To perform aggregation we need to transform relational queries for excel sheet. For quick searching on product we are going to apply aggregation function. Aggregation functions are max, min, count, sum, less than, greater than etc,

Sorting & Searching details:

In this phase, we are going to implement sorting and searching operations on excel sheet. Sorting operations will be apply on showing workers performance of every month. For sorting we are implementing both ascending and descending order. For searching details, we are implementing BFS and DFS searching algorithm.

IV. MAIN FEATURES

In our proposed system, we offer a fully automated method to construct spread sheet implementations for a wide class of relational data transformations.

We have re-implemented all operators of relational algebra to accept a
variable number of input columns and to support NULL values.

The full automation of the translation process reduces the number of human introduced errors in the spread sheet application.

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.

V. QUERY

QUERY The MySQL server provides a database management system with querying and connectivity capabilities, as well as the ability to have excellent data structure and integration with many different platforms. It can handle large databases reliably and quickly in high-demanding production environments. The MySQL server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases. The MySQL server works in a client and server system. This system includes a multiple-threaded SQL server that supports varied backends, different client programs and libraries, administrative tools, and many application programming interfaces (API).

VI. CONCLUSION

We have demonstrated that SQL can be automatically translated into spreadsheet code, including NULL values. Thus, we have shown the power of the spreadsheet paradigm, which subsumes the paradigm of relational databases. Apart from sql implemented a few specific algorithms: linearithmic sorting procedure and two graph traversing algorithms: BFS and DFS.

REFERENCES

1)B. Gates, "Investing in research, SIGMOD Conference 1998 Keynote Speech, video," ACM SIGMOD Digital Symposium Collection, vol. 1, no. 2, 1999.

2)SirMille. (2012, February) inner/outer/full join tables?[Online].Available: http://www.mrexcel.com/forum/excel-questions/612236-inner-outer-full-join-tables.html information: DOI 10.1109/TKDE.2015.2397440, IEEE Transactions on Knowledge and Data Engineering TKDE, VOL. 1, NO. 1, JANUARY 2025 14

3)xil. (2012, May) Connecting list of users with list of companies - please help. [Online]. Available: http://www.mrexcel.com/forum/excel-questions/ 642466-connecting-list-users-list-companies-please-help.html

4) Stefanaalten. (2013, May) Combining two lists into one big listAvailablehttp://www.mrexcel.com/forum/excel-questions/ 704217-combining-two-lists-into-one-big-list.htm mzalikhan. (2012, Apr.) Double V-look up. [Online]. Available: http://www.mrexcel.com/forum/excel-questions/ 625750-double-v-look-up.html,G. Inc. (accessed 2014-05-21) Query
Language Reference (Version 0.7). [Online]. Available: https://developers.google.com/chart/interactive/docs/querylanguage.

5) IntrnationalConference on Management of data, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 195–

206.[Online].Available:http://doi.acm.org/10.1145/1807167.1807191

6) Microsoft Corporation. (accessed 2014/05/21) Excel Home Page - Microsoft Office Online. http://office.microsoft.com/en-us/excel-help excel-functions-alphabetical-list-HA010277524.aspx?CTT=1.

7). Liu and H. V. Jagadish, "A spreadsheet algebra for a direct data manipulation query interface," in ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering.

