CellSQL: 大規模データ処理のための表計算ソフトウェア

菊地悠平† 田浦健次朗†

1. はじめに

1.1 背景

近年の計算機の高性能化、広範な利用により、計算機の扱うべきデータの量は年々爆発的に増加している。このような巨大データ処理の処理に有効な技術の一つとして、データベースシステム、特に SQL などを用いた RDBMS があるが、このようなシステムの問題点として学習コストが高く容易には利用できないという問題がある。また、必要な処理が複雑なものである場合、処理を試行錯誤するための少数のテストデータはユーザがそれぞれ用意する必要がある。以上のような背景から、巨大なデータを平易で直感的な方法で試行錯誤を重ねつつ操作・処理できるようになることは、計算機上におけるデータの効率的な処理に大いに寄与すると考える。

1.2 目的

本研究は，データベースに格納されたデータをスプレッドシートを用いて表現し，SQL に関する深い知識を必要とせずにデータベースを操作できるような GUI を実装することを目的とする．GUI 上でのデータの表示においてはデータの一部のみを取得し用いるため，例えばデータの処理方法を試行錯誤したい場合などに実際のデータのサブセットを用いることができ，データ全体がメモリ容量を超えるほどの大きさである場合でも，不必要に巨大なデータを操作する必要がない．また，このようにして GUI 上で行った操作のマクロとしての保存・再利用を可能とすることで，少数のデータに対する操作により生成したマクロを，他の巨大データに対し同様に適用する・繰り返し処理を行う，といったことが可能になる．

2. 関連研究

2.1 Related Worksheets

データベースの GUI 上でのスプレッドシートを用いた効果的な表現を目指した研究として，Related Worksheet[1] がある．この研究は，GUI 上における分かりやすいデータの配置方法などに重点が置かれ，メモリ容量を超えるような巨大データの効率的な処理には注目していない．そのため，本研究の目的とするような，少数データの操作による処理のマクロ化及び生成されたマクロの再利用という機能を用いた，効率的な巨大データの操作を行うことはできない．

2.2 IBM BigSheets

巨大データの表形式での表現を目的としたアプリケーションとして，IBM BigSheets[3] がある．このアプリケーションは，バックエンドとして Hadoop を用いており，MapReduce 処理を GUI から記述しその結果をスプレッドシート上に表現することで，巨大データの GUI による対話的な解析が可能である．このアプリケーションは本研究が目標とするものに近いが、本研究の目的の一つである、対象とするデータから少数のサブセットを自動的に取り出し効率的な試行錯誤に利用する、という機能にこのアプリケーションは欠ける。また, Hadoop の現在の実装では，タスクの起動オーバーヘッドが大きく，対話的な処理の実現には不向きである．本研究はこのような点を差別化できると考える。

3. 提案システム

本研究では，データベースをスプレッドシートを用いて操作するソフトウェア CellSQL（図 1）を設計・実装した．このアプリケーションでは，ローカルディスクから SQLite 形式で保存されたデータを取得し，それを表として表示することで，スプレッドシートを用いたデータの操作，計算を可能にする．データの取得時にはデータ全体ではなくはじめに得られた数件のデータのみを読み込みスプレッドシート上に表示する。また、データの一部をサンプルとして抽出し一時テーブルを作成することができ、これを用いることで巨大なデータをそのまま試行錯誤の際に用いずに済む。そのようにして読み込まれたデータに対し GUI 上での操作によってカラムの値の計算・行の抽出などの処理

† 東京大学
University of Tokyo

を行い、またそのような操作をマクロとして記録し再利用・自動化することが可能である。さらに、Python スクリプトを用いてユーザがマクロを記述することが可能であり、その中で GUI 上の操作から作成したマクロを実行することで、SQL のみでは難しい場合の多い繰り返し・条件分岐といった制御構造を記述・自動化できる。

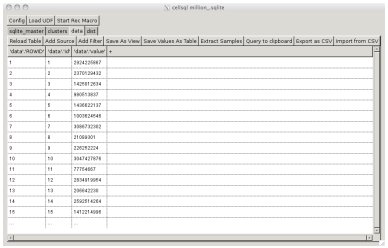

図 1　CellSQL

4. 性能評価

本研究では、巨大なデータに対し、スプレッドシート上でサンプル抽出を利用した効率的な処理の試行錯誤を可能とすること、 またそのようにして実行された処理をマクロとして記録し再利用・自動化することを目的とする。そのため本研究では性能評価として、物理メモリ容量を超えるサイズの巨大データに対しその一部をサンプルとして抽出することで 長い処理時間を必要とせずマクロの生成が可能であること、それを他のデータに対し再利用可能であることを確かた。またその実行時間を直接 SQL を書き下した場合と比較することで、マクロ機能によって生成される処理が, 通常行われるようなデータベース操作で記述するような SQL と比べどれほど実行時のオーバヘッドを持つかを確かめた。

4.1 日本語文書の tf-idf 値の算出

単語と文書の関係の深さを示す指標である tf-idf 値[4] を算出した。まず、 GUI 上での操作からマクロを生成する。この操作の対象とするデータとしては、Wikipedia のダンプデータを用いた。評価実験に用いた計算機のメモリ容量は 2G であるため、ダンプデータはこれより大きなサイズとして 3.9G とした。次に、このようにして生成されたマクロを別のデータベースである青空文庫のデータに対し適用した。その結果、GUI 上での操作はいずれもおよそ 5 秒以下で処理が完了し、生成されたマクロは他のデータに対し正常に適用する事ができた。また、その青空文庫に対する処理時間は、マクロと SQL とでそれぞれ表 1 のようになり、ほぼ同等の時間で処理が行えていることが確かめられた。

CellSQL	37 分 38 秒
SQLite	37 分 58 秒

表 1　tf-idf 値の算出に要した時間

4.2 k-means

前節とほぼ同様の実験を、 k-means クラスタリング[5] についても行った。こちらも計算は正常に終了し、処理に要した時間もマクロと SQL とでほとんど同等となった（表 2)。

CellSQL	36 分 22 秒
SQLite	36 分 9 秒

表 2　k-means クラスタリングに要した時間

5. まとめと今後の課題

本研究で設計・実装した CellSQL を用いることにより、 対象とするデータが巨大であっても、 その一部を対象とした操作を GUI 上で行うことでマクロを生成し、そのマクロを他のデータに適応可能であることが確かめられた。今後の課題としては、より平易な操作が可能となるような機能・ UI の改良、リモートで動作する Paralite[2] などに接続することによる処理の並列・分散化などが挙げられる。

参 考 文 献

1) Bakke, E. and Benson, E.: The Schema-Independent Database UI: A Proposed Holy Grail and Some Suggestions, *CIDR*, pp. 219–222 (2011).
2) Chen, T. and Taura, K.: ParaLite: Supporting Collective Queries in Database System to Parallelize User-Defined Executable, *CCGRID*, pp. 474–481 (2012).
3) IBM: BigSheets, http://www-01.ibm.com /software/ebusiness/jstart/bigsheets/.
4) Jones, K. S.: A statistical interpretation of term specificity and its application in retrieval, *Journal of Documentation*, Vol. 28, pp. 11–21 (1972).
5) MacQueen, J. B.: Some Methods for Classification and Analysis of MultiVariate Observations, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (Cam, L. M. Le and Neyman, J.(ed.)), Vol. 1, University of California Press, pp. 281–297 (1967).