D-028

ストリームデータに対するストリーム処理エンジンを用いた OLAP 処理
OLAP for Stream Data Using Stream Processing Engine

中挾 晃介 † 北川 博之 ‡ 天笠 俊之 ‡
Kosuke Nakabasami Hiroyuki Kitagawa Toshiyuki Amagasa

1. はじめに

近年，センサデータやマイクロブログ等，連続的に生成され，配信されるようなストリームデータの増加に伴い，ストリーム処理エンジンが多数開発されてきている．一方，ストリームデータを集約し，そのデータの傾向などの分析を行いたいというニーズが増加している．この代表例として，データを多次元のデータキューブとして捉えて分析を行う多次元データ分析（OLAP 処理[1]）がある．ストリームデータに対して OLAP 処理を行う研究として，Jiawei Han らはストリームデータに対する OLAP 処理を容易にすることを目的とした Stream Cube と呼ばれるアーキテクチャを提案している[2]．ただし，ストリーム処理エンジンを活かした OLAP 処理については十分検討が行われていない．このため，OLAP 処理のために個別のシステムを一から開発する必要があったり，他のストリームデータ応用との連携が十分にとれないなどの問題がある．

そこで本研究では，ストリーム処理エンジンを用いた OLAP 処理を実現するための手法を検討する．具体的には，エンジン側に連続的問合せを登録しておき，これらの登録しておいた問合せから OLAP 処理の結果として必要なデータを得る．OLAP 処理では，分析対象のデータの次元（属性）や次元内の階層の全ての組合せを頂点とする lattice を考える．lattice の各頂点は，OLAP 処理において集約する対象の次元の組合せに対応する．最も単純には，lattice 内の全ての頂点の次元の組合せの集約を実行すればよい．しかし一般には，lattice の全頂点数は膨大な数になることが多く，これらの全てをエンジンに登録することは，処理性能上難しい場合が多い．また，必ずしも全ての頂点に対応する集約値を常時取得し続けなくても，ユーザ要求に応じて導出できればよいという場合も多いと考えられる．そこで本研究では，lattice 内の一部の頂点のみを適切に選択し連続的問合せとして登録することで，効率的な OLAP 処理を実現することを目的とする．

2. 提案手法

2.1 処理の効率化

1 章で述べたように，データの次元，階層の集約の全組合せを登録することは難しいため，エンジンにどの集約問合せを登録するかを，コストが最小になるように選択する．

また，各集約間には依存関係が存在する．そのため，選択された連続的問合せを独立に評価せずに，それらを依存関係に基づいて結び付け，先に評価された集約結果を利用することで，処理を効率化できる．そこで，各集約間の評価順を表す処理のパイプラインを構築する．

† 筑波大学システム情報工学研究科コンピュータサイエンス専攻 Graduate School of Systems and Information Engineering, University of Tsukuba
‡ 筑波大学システム情報系情報工学域 Faculty of Engineering, Information and Systems, University of Tsukuba

2.2 コストモデル

本節では，エンジンに登録する集約問合せを決めるアルゴリズムに用いるコストモデルを説明する．

例として，以下のようなスキーマを持つデータ内の二つの次元から lattice を構成することを考える．

売上情報（商品 ID, 商品名, ジャンル,
顧客 ID, 顧客名, 地域, 売上額）

この売上情報の lattice は，図 1 のようになる．本手法において，処理にかかる全体のコストは lattice 内のそれぞれの集約問合せにおいてかかるコストの総和と考える．lattice 内には，エンジンに登録する集約問合せと，エンジンに直接集約問合せを登録せずに，必要に応じてエンジンに登録する集約問合せの結果から必要とする集約結果を導出する（以降，オンデマンドに評価すると言う）集約問合せの二種類の問合せが存在すると考える．ここで前者を$Q_r = \{r_1, \cdots, r_n\}$とし，後者を$Q_d = \{d_1, \cdots, d_m\}$とする．

まず，エンジンに登録した問合せに対する集約処理におけるコストを考える．これを$C_1(r_i)$とする．$C_1(r_i)$は，単位時間あたりにその集約演算処理に入力されるタプル数と 1 タプル当りの集約処理時間の積に比例したコストと考える．ここで，集約計算自体の処理はタプルが到着する度に集約属性に応じてハッシュ等を用いてタプルを振り分け集約すると考える．

次に，オンデマンドに評価する集約問合せの処理コストを考える．オンデマンドに評価する集約問合せは，ユーザからその集約が求められたときに限り，lattice 上で自分自身の上流にあるQ_r内の問合せの中で，その集約結果として保持しているタプル数が最も少ない問合せから自身の集約結果を導出する．よってコスト$C_2(d_i)$は，その上流にある連続的問合せの集約結果タプル数に比例した処理コストに自身の問合せの参照頻度$\mathrm{P}(d_i)$（単位時間あたりに自身の問合せを実行する回数）をかけたものとする．

さらに，エンジンに登録した連続的問合せが，集約結果の保持に必要なメモリ空間をコストと考え，その集約結果のタプル数を空間コスト$S_1(r_i)$とする．

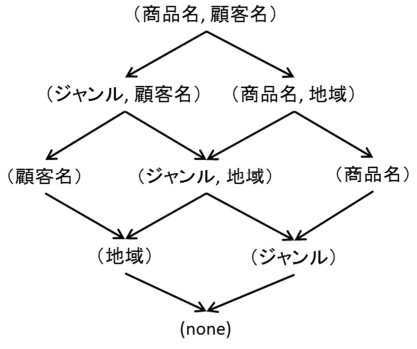

図 1 売上情報を表す lattice

以上から，処理コストCおよび空間コストSを以下のように定義する．

$$C = \sum_{i=1}^{n} C_1(r_i) + \sum_{i=1}^{m} C_2(d_i) \quad (1)$$

$$S = \sum_{i=1}^{n} S_1(r_i) \quad (2)$$

2.3 連続的問合せ群を決定するためのアルゴリズム

2.1 節において述べた，コストが最小となるような最適な連続的問合せの選択は，一般に組合せ最適化問題となるため，本研究では貪欲アルゴリズムを用いる．このアルゴリズムは，ストリームデータの入力レート，集約処理の対象となるタプル数，各集約問合せの集約率及び参照頻度，空間コストの上限が与えられた条件下で先述の問題を解く．このアルゴリズムは，Harinarayan らのアルゴリズム[3]を応用している．なお，擬似コードなど詳細については我々の別の論文[4]を参照されたい．

具体的には，以下のような流れとなる．

(1) まず lattice 内の全ての頂点をオンデマンドに評価する問合せであるとみなす．
(2) オンデマンドに評価する問合せの中から，ある一つの頂点をエンジンに登録する集約問合せとして選んだ時のコストを 2.2 節のコスト式を用いて計算する．
(3) (2)を lattice 内のオンデマンドに評価する全ての問合せに対して行ったとき，最も処理コストCの値が小さくなった集約問合せをエンジンに登録する問合せとして選ぶ．
(4) (2)，(3)を空間コストSの累積値が最初に設定した空間コストの上限を超えるまで繰り返す．

2.4 処理パイプラインの決定

前節のアルゴリズムにおいて決定した連続的問合せ群をどの順番で評価を行うか表すパイプラインを構築する．

具体的には，自分の上流にある連続的問合せの中で，集約結果として保持しているタプル数が最も少ないものに向けてエッジを張っていく．これは，タプル数が少ない方が，集約対象となるタプル数が少なくなり，集約における演算量が少なくなるためである．以上の手法で処理のパイプラインが構築される．

3. 実際のストリーム処理エンジンを用いたコストモデルの推定

本研究では，実際のストリーム処理エンジンを用いて，どのように処理コストおよび空間コストがかかるかを調べ，より現実に近いコストモデルを推定する．ここで，実際の商用のストリーム処理エンジンとして，株式会社日立製作所が 2008 年に開発した μ Cosminexus Stream Data Platform[5]を用いる．このエンジンに対し，入力レートや集約演算対象となるタプル数（ウィンドウ幅），集約率，タプルサイズを変えたときに，1 タプルあたりの処理時間やメモリ使用量にどのような影響があるかを調べる．

図 2 の左上図は入力レート（x 軸）と 1 タプルあたりの CPU 時間（左 y 軸），物理メモリ量（右 y 軸）の関係を表

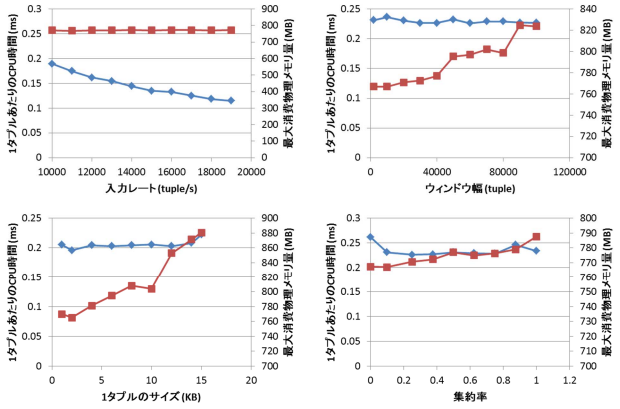

図 2　実験結果

している．図から，入力レートが大きくなるほど 1 タプルあたりの処理にかかる CPU 時間は小さくなるが，処理時間が 0 になることはなく，また傾きも小さいため，入力レートに対して一定であると推測できる．一方，右上図，左下図，右下図はそれぞれウィンドウ幅，1 タプルのサイズ，集約率と 1 タプルあたりの CPU 時間，物理メモリ量の関係を表している．図から，ウィンドウ幅および 1 タプルのサイズが大きくなるほど，使用する物理メモリ量が大きくなることが分かる．

特に空間コストに関して，(3)式で近似を行える．ここで，使用メモリ量をS(MB)，1 タプルあたりのサイズをT(MB)，ウィンドウ幅をw(tuple)，連続的問合せの数をn(個)とする．aおよびbは定数項である．

$$S = \sum_{i=1}^{n} (w_i T_i + a) + b \quad (3)$$

4. まとめと今後の課題

本稿では，ストリーム処理エンジンを用いた OLAP 処理を実現する手法と，処理の効率化のためのアプローチを示した．また，実際のストリーム処理エンジンに適用する際のコストモデルを検討した．今後は，実ストリーム処理エンジンに対する提案手法の適用を行う．また，時間次元に対する集約を加味することで，コストにどのように影響するかを考える．

謝辞

本研究の一部は，文部科学省・未来社会実現のための ICT 基盤技術の研究開発「実社会ビッグデータ利活用のためのデータ統合・解析技術の研究開発」による．

参考文献

[1] J. Han et al. “Data Mining: Concepts and Techniques”, Morgan Kaufmann, Third Edition, Jul. 2011.
[2] J. Han et al. “Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams”, Distributed and Parallel Databases, 18(2):173-197, Sep. 20 2005.
[3] V. Harinarayan et al. “Implementing Data Cubes Efficiently”, In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD ’96), pages 205-216, ACM, Montreal, Canada, June 1996.
[4] 中挾ら, “ストリーム処理エンジンを用いたストリームデータの OLAP 処理”, 第 6 回データ工学と情報マネジメントに関するフォーラム(DEIM2014), D5-4, 2014 年 3 月 3 日～3 月 5 日.
[5] μ Cosminexus Stream Data Platform, http://www.hitachi.co.jp/Prod/comp/soft1/cosminexus/sdp/