D-027

差分計算を用いたトランザクショナルストリーム処理の実現

Transactional Stream Processing using Incremental Computation

西村 和也 †　　北川 博之 ‡
Kazuya Nishimura　　Hiroyuki Kitagawa

1. はじめに

近年，センサデータをはじめとしたストリームデータに対する処理要求が増加しており，その処理の枠組みとして，連続的問合せを実行する**データストリーム処理システム(DSMS)**が開発されている．連続的問合せを効率的に実行する処理方式として，差分計算がある．ストリーム処理において外部データベースとの結合処理などが行われる場合，外部データベースが更新されると，連続的問合せの処理結果の一貫性が保たれないという問題が発生する．本研究では，差分計算の枠組みにおいて，このような一貫性を保証するトランザクショナルストリーム処理の実現手法を提案する．

2. データストリーム処理システム

従来，静的なデータベースへの問合せにはデータベース処理システム(DBMS)が用いられてきた．しかし，情報源から連続的に到達するストリームデータに対して，連続的に問合せを実行する目的には適さない．そこでこのような連続的問合せ処理を実行可能なシステムとして STREAM[2]，Storm 等の DSMS が開発されてきた．DSMS において連続的問合せを記述するための言語としては CQL(Continuous Query Language)[3]等が存在している．CQL で記述された問合せを DSMS に登録することでストリームデータに対する連続的な問合せ処理を実行することが可能になる．DSMS における基本的な処理の流れは次のようなものである．まず，**ウィンドウ演算**によってストリームデータから範囲を指定して最新のデータを切り出す．次に選択，結合などの**関係演算**を適用することで分析を行う．最後に**ストリーム化演算**によって，関係演算として得られる結果をストリームデータ化して出力する．

2.1. ナイーブな処理

図 1 のような演算子木で DSMS のナイーブな処理を行うと考える．この擬似演算子木は，ウィンドウ演算 W でストリームデータを切り出し，外部リソースのデータと結合演算を行い，関係演算によって得られた結果を全て出力するストリーム化演算 RSTREAM で出力するという処理を表している．

ウィンドウ幅 $N>1$ のウィンドウ演算に新規のタプルが到着する度に外部リソース中のタプルと結合演算を

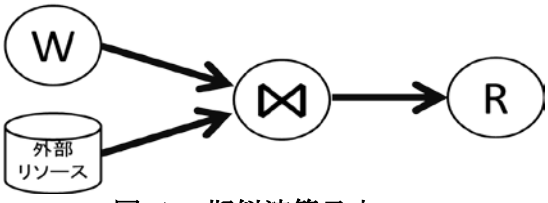

図 1．擬似演算子木

行う．ここでは仮に各タプルが 1 対 1 で結合し，N 個のタプルが毎回出力されるとする．この時，1 回目と 2 回目の処理の結果として得られるタプル群の中で内容が異なるタプルは，新規のタプル，ウィンドウ演算の範囲から外れたタプルと結合処理を行なった結果として得られる 2 つのタプルのみであり，残りの $N-2$ 個のタプルは重複することになる

このようなナイーブな処理を実現しようとすると，演算子ごと，処理系全体での結果で重複する部分が発生する事がある．そこで，DSMS では**差分計算**と呼ばれる処理方式がしばしば用いられる．

2.2. 差分計算

差分計算方式の目的は，ナイーブな処理の過程で発生する各処理での重複をする計算を行わず，前回の処理結果との差分のみを計算することで計算量を減らした上で，ナイーブな処理と同一の結果を得ることである．そのため，差分計算方式では演算子自身が次に評価されるときに必要となる情報を**シノプシス(Synopsis)**と呼ばれる領域に保持する．このシノプシス中に保持されたデータを利用し，新たに追加，範囲から外れた情報を差分として計算していくのが差分計算方式である．この方式では，前回から新たにタプル s が演算子に入力されたという情報を $\langle s,+\rangle$，タプル s' が演算の範囲から外れたという情報を $\langle s',-\rangle$ のように表現して計算を行う．

3. 外部リソース参照による一貫性の問題

DSMS ではストリームデータのみを用いた分析処理以外にも，DBMS 等の外部リソース中に蓄積されたユーザデータ等を参照した処理を行いたい場合がある．このような場合には，DSMS の処理とは独立して外部リソース内のデータが更新されうる．連続的問合せの実行中に外部リソースの更新処理があった場合に，ナイーブな方式であれば問題は起こらないが，差分計算の枠組みで処理を行うと，結果として，一貫性が保たれない場合が存在する。

図 2 は，その例を表している．時刻 t でタプルが到着し，この処理全体を評価した場合の出力が図中の Table1 である．ただしウィンドウ幅を 2 と仮定する．その後，外部リソースが更新され，その後の時刻 $t+1$ でタプルが到着した場合の処理結果が Table2 である．

†筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻，Department of Computer Science, Graduate School of SIE, 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.

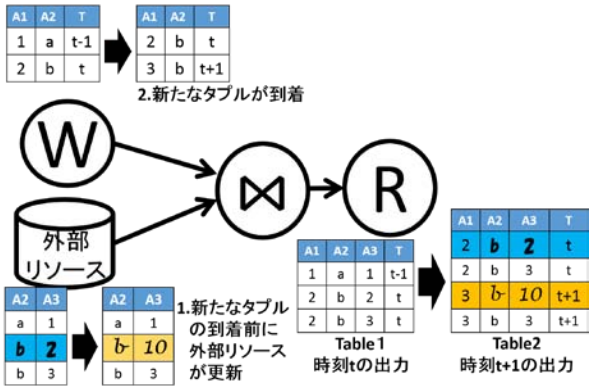

図 2．一貫性問題が発生する例

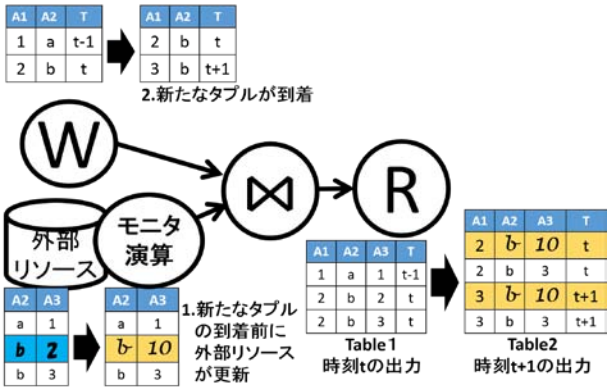

図 3．モニタ演算子の導入

ストリーム化演算 RSTREAM のシノプシスには，それぞれ表を出力するための情報が保持されている．

ここで，ウィンドウに追加された，あるいは外れたタプルが計算されていく一方で，変化がないタプルについては計算が行われない．従って，ストリーム化演算 RSTREAM のシノプシスに外部リソースの更新情報が伝播されず，図２中の Table2 が示すように，外部リソース更新前の $(b,2)$ を参照した結果と，更新後の $(b,10)$ を参照した結果が混在する結果となる．

このように，差分計算処理の最中に外部リソースが更新された場合に，一回の問合せ結果の中で一貫性が保たれなくなってしまう．

4. トランザクショナルストリーム処理

小山田ら[1]はストリーム処理の最中に，外部リソースが更新された場合，連続的問合せの一つの処理結果内で一貫して同一の状態の外部リソースを参照しないという問題に対して，トランザクショナルストリーム処理という概念を提案した．これは，外部リソースが更新されうる状況下で，「連続的問合せの各処理結果が，外部リソースの一つのスナップショットを一貫して参照する」ことを保証するというものである．つまり，連続的問合せの参照処理をトランザクションであると考え，一つのトランザクションの中では，ただ一つの外部リソースのスナップショットのみが参照されるということである．

この概念の実現のために，小山田らは外部リソースの参照処理群を連続的問合せ由来トランザクションと位置付けた上で，外部リソースの更新を行うトランザクションとの同時実行制御を行うロッキングプロトコルを提案している．本研究では，差分計算の枠組みの中でトランザクショナルストリーム処理を実現するための手法を提案する．

5. 提案手法

そこで，差分計算方式においてトランザクショナルストリーム処理を実現するための手法として，本研究では，**モニタ演算子**という演算子を新たに導入する．３節の例では，外部リソースの変更が RSTREAM へ通知されず，シノプシスが更新されないことで，一貫性の問題が発生していた．

モニタ演算子は外部リソースに発生した変更情報を検知して，その情報をデータストリームとして出力することで，差分計算の枠組みにもとづいてシステム全体へ外部リソースの変更を反映する．つまり，モニタ演算は特殊なウィンドウ演算のように振る舞う．図３中のような更新処理の場合，$(b,2)$ というタプルがモニタ演算の範囲から外れ，$(b,10)$ というタプルがモニタ演算に到着したと考え，$\langle b,2,-\rangle,\langle b,10,+\rangle$ という差分の情報を下流の演算子に伝えて差分計算を行うことで，RSTREAM のシノプシスを更新する事ができる．その結果，ウィンドウ演算に新たなタプルが到着した時に計算を行うと，一つの連続的問合せの処理結果として得られる表は図３中の Table2 であり，その表中のタプルがすべて外部リソース更新後のスナップショットのみを一貫して参照していることが確認できる．

以上から，モニタ演算子を導入することで，「連続的問合せの各処理結果が外部リソースの一つのスナップショットを一貫して参照していること」が保証できると考えられる．

6. まとめと今後の課題

本論文では，ストリーム処理システムの実装で用いられる差分計算方式においてトランザクショナルストリーム処理を実現するための手法として，モニタ演算子を導入することを提案した．現在，外部リソースとしてリレーショナルデータベースを用いて，モニタ演算子のプロトタイプを実装し，所属研究室で開発中の DSMS と連携し，手法の有効性を検証する実験を進めている．

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] Masafumi Oyamada, Hideyuki Kawashima and Hiroyuki Kitagawa, “Data Stream Processing with Currency Control”, ACM SIGAPP Appl, Comput. Rev., Vol13, No.2, pp.54-65 (2013)
[2] A.Arasu, et al. “STREAM: The Stanfold Data Stream Management System”, Technical report, Department of Computer science, Stanfold University, http://ilpub.stanfold.edu:8090/641 (2004)
[3] A.Arasu, et al. “The CQL Continuous Query Language: Semantics Foundations and Query Execution”. VLDB Journal, Vol.15, No.2, pp.121-142 (2006)