連続的問合せを対象とした複数問合せ最適化方式の提案

Proposal of a Multiple Query Optimization Scheme for Continuous Queries

渡辺 陽介* 北川 博之†

Yousuke WATANABE Hiroyuki KITAGAWA

ストリーム型情報源からの情報取得の手段として連続的問合せがある．本稿では，多数の連続的問合せを効率的に実行するためのアプローチとして，連続的問合せに対する複数問合せ最適化方式を提案する．ストリーム型情報源に対する連続的問合せでは，ウインドウなどの時間条件が記述できること，利用者がその情報を必要とするタイミングを指定できることが求められる．このような連続的問合せは，同一の演算であっても実行タイミングによって全く異なる結果を生成し得るため，従来のバッチ処理などを想定した複数問合せ最適化手法をそのまま適用することは困難である．本提案手法の特徴は，問合せの参照範囲の違いを考慮し，参照範囲の近い同士の問合せをグループ化することによって効率的な実行処理プランを導出する点である．

Continuous queries are often used to get information from stream information sources. In this paper, we propose a novel multiple query optimization scheme for efficiently processing many continuous queries. In the advanced stream processing environments, we usually need to use temporal conditions as expressed by windows, and to deliver information when the user needs it. In this situation, the same operator may generate different outcome when it is evaluated at different time instances. Therefore, we cannot directly apply existing multiple query optimization techniques. The proposed scheme features grouping continuous queries based on their execution time and group-wise multiple query optimization.

1. はじめに

ネットワーク技術の発達により，データ放送やプッシュ型情報源のようなストリーム型情報源が新たな情報配信形態として注目されている．ストリーム型情報源の数や種類が増加してきたことにより，複数のストリーム型情報源に対する高度利用の重要性が高まっている．今後，多数のユーザがストリーム型情報源を利用するようになると，ストリーム型情報源処理システムに対して膨大な数の問合せが発生することが予想される．そのため，大量の問合せ要求を効率的に処理可能なストリーム型情報源の処理方式が求められている．

ストリーム型情報源から到着するデータを処理する枠組みとして，連続的問合せ(Continuous Query)[1,3,7]が注目されている．連続的問合せは，ストリーム型情報源から新規に到着した情報に対して逐次問合せ処理を実行し，生成された結果を配信するものである．本稿では，多数の連続的問合せを効率的に実行するための手段として，連続的問合せに対する複数問合せ最適化方式を提案する．

連続的問合せは，情報源から情報が到着するとそれに連動して実行される到着型問合せと，タイマーによって一定間隔で定期的に実行されるタイマー型問合せに分類される．従来の連続的問合せに対する研究の中でもこれらの両者を扱っているものはあったが，タイマー型問合せと到着型問合せに対して別個の実行方式や問合せ記述を用いる等，統一性を欠く場合が多かった．本研究では，定期的にアラームを送信するような仮想的なストリーム型情報源であるクロックストリームと，マスタ情報源の概念を導入することにより，タイマー型問合せをクロックストリームからの情報の到着に連動して実行される到着型問合せとして扱う．これにより両者を統一的に扱うことが可能となる．一方，連続的問合せ中で記述可能な条件についても，既存の研究ではそれぞれ異なって仮定のもとに研究が行われている．本研究で想定する複数データストリームの処理環境では，時間的に近いデータ同士を結合するなどの時間条件をデータ処理に用いることが一般に必要である．そこで，本研究では，ストリーム型情報源に対する結合演算としてウインドウ結合[2]を用いることを前提とし，その場合の連続的問合せの複数問合せ最適化について検討する．

一般に，複数問合せ最適化とは複数の問合せで共通な演算や中間結果を共有することを目的とするものである[4,6]．しかし，リレーショナルデータベース等を対象とした従来の複数問合せ最適化は，バッチ型の問合せ処理など，あらかじめ同時に実行されることが分かっている問合せ群のみを対象としてきた．連続的問合せは，外部のイベントに連動して実行されるため，あらかじめ問合せの実行タイミングを完全に把握することは困難である．したがって，従来の複数問合せ最適化手法を直接的に適用することはできない．しかし，同一ストリームの新規データ到着に連動して実行される問合せ群や，あらかじめ短時間の間に続いて実行されることが分かっている問合せ群の中では，演算の一部の共有化を図ることが可能である．本研究では，ストリーム型情報源からのデータ到着のパターンならびに各問合せの実行タイミングとその参照データ範囲に着目し，演算やその結果を共有できる可能性が高い問合せ同士をグループ化の対象とするような連続的問合せに対する複数問合せ最適化方法を提案する．

2. 連続的問合せ

2.1 基本モデル

```
CREATE    query_name
MASTER    master_source,...
WINDOW SIZE    windowsize
SELECT-FROM-WHERE    sql
```

図 1 問合せ記述形式

Fig.1 Syntax of continuous query

本研究では，ストリーム型情報源をリレーションとしてモデル化する．ストリーム情報源から送られてきたデータを，対応するリレーションの 1 タプルとして扱う．また，そのデータの到着時刻 TS がタプルに付加されるものとする．

連続的問合せは，図 1 のような形式で記述する．CREATE

* 学生会員 筑波大学システム情報工学研究科
watanabe@kde.is.tsukuba.ac.jp
† 正会員 筑波大学電子・情報工学系
kitagawa@is.tsukuba.ac.jp

節には，問合せの識別子となる名前を記述する．MASTER 節には，問合せ実行のきっかけとなる情報源を指定する．本稿では MASTER 節に記述された情報源をマスタ情報源と呼ぶ．連続的問合せは，いずれかのマスタ情報源から情報が到着した時点で実行される．WINDOW SIZE 節には，複数のストリーム型情報源間でウインドウ結合[2]を実行する際のウインドウの範囲を記述する．ウインドウとウインドウ結合については以下で述べる．SELECT-FROM-WHERE は SQL の構文と同様である．ただし本研究では，WHERE 節は AND で結合された条件のみとし，ネストした問合せや集約演算，OR などは考慮していない．連続的問合せでは，1 回の実行でユーザに返される処理結果は，前回の実行の後に届いた新規情報に基づく差分である．

ウインドウ結合：ストリーム型情報源では，サービス提供中はタプルが逐次到着するため，過去に到着したすべての情報を結合演算の処理対象とした場合，非常に大量のタプルを処理対象としなければならない．このようなことを背景に，複数のストリーム型情報源に対する統合演算としてウインドウ結合[2]が提案されている．本研究における連続的問合せにおいても，ウインドウ結合を考慮の対象とする．ウインドウとは，結合演算の適用対象となるタプルを選択するための時間範囲のことで，論理的には，マスタ情報源のタプルの到着時刻 t を用いた以下の選択演算を各情報源 I に対して行った後に，通常の結合演算を行うことに相当する．

$$\sigma_{I.TS\in[t-windowsize,t]}(I)$$

クロックストリーム：これまでに述べた問合せ記述の枠組みを用いることで，到着型問合せは直接記述可能である．本研究では，クロックストリームの概念を導入することで，タイマー型問合せも上記の枠組みのもとに扱う．クロックストリームは処理系が提供するストリーム型情報源であり，決まった時刻に周期的にアラームを送信する．クロックストリームをマスタ情報源として用いることで，問合せを定期的に実行するタイマー型問合せを記述することができる．従来の連続的問合せでは，タイマー型問合せと到着型問合せは別の種類の問合せとして扱われていたが，本研究における連続的問合せではこれらを統一的に取り扱うができる．

連続的問合せの例を示す．図 2 は「毎日 0 時に過去 9 時間分の Stream_1 と Stream_2 と Stream_3 の情報を統合して提供して欲しい」という意味の問合せである．この問合せにおけるマスタ情報源は毎日 0 時にアラームを送信するクロックストリーム Clock_0 である．また，ウインドウの大きさは 9 時間である．ただし，hour は 1 時間を表す定数とする．

2.2 問合せの実行モデル

連続的問合せは，マスタ情報源の情報が到着すると実行され，前回の実行後に届いた新規情報に基づく差分がユーザへ配信される．前回の実行後に送られてきたタプルを保持するために，新規タプルをその情報源に対応したデルタリレーションへ格納する．問合せはデルタリレーション中のタプルに対して適用され，演算を適用し終わったデルタリレーションは毎回消費される．以下では，デルタリレーションを Δ(source_name, query_name)と表記する．複数の問合せがあるときは問合せごとにデルタリレーションが用意され，到着したタプルはそれぞれのデルタリレーションに格納される．

ウインドウ結合を計算するためには，ウインドウの時間範囲に含まれるタプルをすべて保管しておく必要がある．そのための保管場所としてウインドウリレーションを用いる．ウインドウリレーションは window(source_name, query_name)と表記する．結合演算は，各情報源のデルタリレーションとウインドウリレーションを用いて行われる．例えば，Stream_1 と Stream_2 のウインドウ結合では，新規の処理結果は次のようにして得られる．

Δ(Stream_1, qname) ⋈ window(Stream_2,qname)
∪ window(Stream_1,qname) ⋈ Δ(Stream_2, qname)
∪ Δ(Stream_1, qname) ⋈ Δ(Stream_2, qname)

処理を適用し終わったデルタリレーションのタプルは，ウインドウリレーションに移される．ウインドウリレーションに含まれているタプルは時間が経つと，問合せのウインドウの時間範囲から外れ，不要な情報になる．ウインドウリレーション中の不要なタプルは，ウインドウ結合の実行前に削除される．デルタリレーションに格納されているタプルでも，マスタ情報源の到着を待っている間にウインドウの時間範囲外へ出てしまうことがあるが，これも同様に削除される．

```
CREATE   Example_1
MASTER   Clock_0
WINDOW SIZE   9*hour
SELECT   *
FROM   Stream_1, Stream_2, Stream_3
```

図 2　連続的問合せの例（問合せ 1）
Fig.2 Example of continuous queries

3. 複数問合せ最適化手法

3.1 基本的アイデア

以降では，複数の連続的問合せに対する問合せ最適化方式について述べる．

まず，提案方式の基本的アイデアを述べる．例として，図 2,3,4 のような 3 つの問合せの最適化を考える．図 3,4 の問合せの意味は図 2 とほぼ同様であるが，マスタ情報源であるクロックストリームの違いから，問合せ 1 は毎日 0 時，問合せ 2 は毎日 23 時，問合せ 3 は毎日 12 時にそれぞれ実行される．これら 3 つの問合せは，共通する処理として Stream_2 と Stream_3 の結合（Stream_2⋈Stream_3）を含んでおり，この演算の計算結果を共有することで効率化が期待できる．

しかし，各問合せがウインドウ結合において参照するデータの範囲を調べると，図 5 のような関係になっていることがわかる．問合せ 1 と問合せ 2 は，問合せの実行タイミングが近く，参照する範囲に重なりがあるが，問合せ 3 は実行タイミングが離れているため，ほかの 2 つの問合せの参照範囲との重なりが存在しない．このことから，問合せ 1 が生成する Stream_2⋈Stream_3 の処理結果と問合せ 2 が生成する処理結果には同じものが含まれる可能性が高いが，問合せ 3 の生成する結果には他の 2 つと同じものが含まれる可能性が全くないことがいえる．複数問合せ最適化の効果は，他の問合せが生成した結果をどの程度再利用できるかによって決まるため，問合せ 1 と問合せ 2 の演算を共有することは処理効率の向上を見込めるが，問合せ 3 と他の 2 つで演算を共有することでは処理効率の向上はありえない．逆に，再利用できない無駄な中間結果のスキャンが発生する分だけ，効率低下を招く可能性がある．よってこの場合，問合せ 1 と問合せ 2 だけ共有し，問合せ 3 を別に処理する方式が望ましい．

このように，実行タイミングが異なる問合せを最適化するためには，問合せが参照するデータの範囲を計算し，生成される結果がどの程度共有できそうかを調べることが必要である．本研究の連続的問合せの複数最適化方式は，この点を考慮し，次の 2 つのステップから構成される．

（1） 与えられた問合せの集合から，参照範囲の重なりが大きい問合せを集めたクラスタを生成する．
（2） クラスタごとに共有可能な演算を探し，最適な問合せ実行プランを導出する．

```
CREATE   Example_2
MASTER   Clock_23
WINDOW SIZE   9*hour
SELECT   *
FROM   Stream_2, Stream_3, Stream4
```

図 3 問合せ 2
Fig.3 Query Example_2

```
CREATE   Example_3
MASTER   Clock_12
WINDOW SIZE   9*hour
SELECT   *
FROM   Stream_2, Stream_3, Stream_5
```

図 4 問合せ 3
Fig.4 Query Example_3

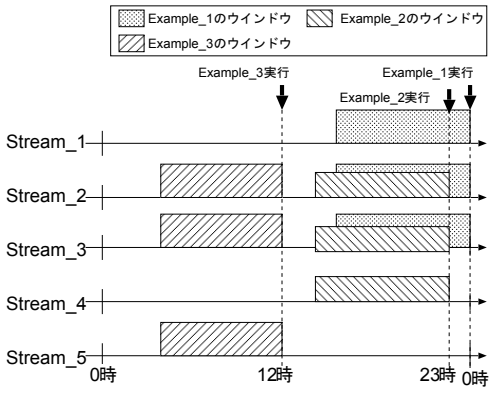

図 5 問合せ参照範囲
Fig.5 Reference ranges of queries

3.2 問合せのクラスタリング

はじめに，クラスタ作成の基準となる，問合せ同士の参照範囲に基づく類似度の概念を定義する．一般に，外部のストリーム型情報源をマスタ情報源とする場合，その到着時刻を完全には予測できないことが多い．この場合，問合せ同士の参照範囲の重なりをあらかじめ正確に調べることは不可能である．そこで本研究では，各情報源からのデータ到着ログを用いて問合せの実行タイミングをシミュレートし，参照範囲の重なりを予測する．

まず，ある期間 T におけるデータ到着ログを解析し，期間内に届いた各情報源のタプルの到着時刻の集合を得る．各情報源 $I_i(1 \le i \le n)$からの情報の到着時刻集合を S_i とする．次に，問合せ $Q_j(1 \le j \le m)$をこれらの到着時刻情報に基づいて実行した場合に，Q_j が参照する各 S_i の到着時刻集合 USE_{ij} を求める．これは，Q_j の実行時刻の集合 MS_j の各時刻 s において Q_j を実行したときに，各回の実行で Q_j のウインドウの参照範囲内に含まれた S_i の要素の集合和を計算することによって得られる．

$$USE_{ij} = \bigcup_{s \in MS_j} \{ t \mid t \in S_i \wedge t \in [s - windowsize_j, s] \}$$

Q_j の実行時刻の集合 MS_j は，Q_j のマスタ情報源すべての到着時刻集合の和である．また，$windowsize_j$ は問合せ Q_j の WINDOW SIZE 節で指定されたウインドウの大きさであり，指定がない場合には無限大とする．

USE_{ij} を元に，問合せ Q_a, Q_b の参照範囲の類似度 $similarity(Q_a, Q_b)$を計算する．本研究では，以下の式により類似度を求める．

$$similarity(Q_a, Q_b) = \min_{I_i \in \mathrm{Re}f(Q_a) \cap \mathrm{Re}f(Q_b)} \frac{|USE_{ia} \cap USE_{ib}|}{|USE_{ia} \cup USE_{ib}|}$$

この式は，Q_a, Q_b が共に使用する情報源において，最低でもどの程度参照範囲に共通部分があるかを表している．ただし，Ref(Q)は問合せ Q の FROM 節に記述された情報源の集合を表すものとする．共通に利用する情報源がない場合($Ref(Q_a) \cap Ref(Q_b) = \phi$)の類似度は 0 とする．同じマスタ情報源を持ち，かつウインドウサイズが等しい問合せ同士の類似度は 1 になる．

クラスタ生成のため，全ての問合せのペアについて類似度を計算する．m 個の問合せに対しては，計算結果として m×m の対称行列 A（類似度行列）が得られる．計算された類似度行列から，類似度の高い問合せ同士のクラスタリングを行う．ただし，クラスタ内に含まれる問合せの類似度の最小値が閾値 θ 以上になるようにクラスタを生成する．クラスタ生成のアルゴリズムは，ここでは通常の階層的クラスタリング手法[5]を用いる．

クラスタ生成のパラメタ θ を変化させることによって，クラスタの大きさを調整することができるが，これはどの程度の演算結果の再利用性が保証されたら演算を共有するか，ということを決定する基準となる．θ を 1 にした場合は，参照範囲が全く同一である問合せの共通部分式のみを共有することに相当する．また，θ を 0 にした場合は，巨大なクラスタが 1 つ生成される．これは類似度を気にせずに，問合せの共通部分式のみを調べて共有する従来の複数問合せ最適化の手法をそのまま当てはめた場合に相当する．

3.3 クラスタ内での共通演算の共有

3.2 節で述べた通り，ユーザから与えられた問合せのうち，参照範囲の類似度が高いもの同士がクラスタを構成する．クラスタ内の問合せの共通演算の処理結果は，ほぼ共有できることが保証されているため，あとは実際にクラスタ内の問合せから共通演算を探索し，最適な問合せプランをクラスタごとに導出する．この部分の処理には，従来の複数問合せ最適化手法[4]に基づく手法を適用する．ただし，[4]では最適プランを導出するために問合せ処理コストの見積りが可能であることを仮定している．しかし，ストリーム型情報源では，一般にコスト見積りに必要な各種の統計情報を得ることは難しく，正確なコスト計算を行うことができない．そのため，本手法では共有される演算数を最大とするようなプランを，最適プランとして選択することとする．

例として，図 2,3 で示した問合せ 1,2 のみを含むクラスタを用いて説明する．この 2 つの問合せは Stream_1⋈Stream_2⋈Stream_3 と Stream_2⋈Stream_3⋈Stream_4 という演算を含んでいる．問合せ 1 では，最終的な結果を得るための計算順序として，(Stream_1⋈Stream_2)⋈Stream_3, Stream_1⋈(Stream_2⋈Stream_3)等の複数の候補が存在する．問合せ 2 にも同様に複数の候補がある．この場合は問合せ 1 を Stream_1⋈(Stream_2⋈Stream_3)，問合せ 2 を (Stream_2⋈Stream_3)⋈Stream_4 の順で計算することで，Stream_2⋈Stream_3 が共有できるため，最適な問合せプランとして導出される．共有された Stream_2⋈Stream_3 は，

各問合せのマスタ情報源にしたがって，Clock_0 のデータが到着した時点と Clock_23 のデータが到着した時点の両方で実行されるが，対応するデルタリレーションおよびウインドウリレーションも共有され，新規到着タプルは毎回消費されるので，重複したデータを生成することはない．

4. 評価実験

本研究の提案手法の有効性を示すため，プロトタイプシステムを実装し，評価実験を行った．

実験環境：プロトタイプシステムはユーザから与えられた連続的問合せの集合を，共通演算を共有した実行プランに変換する．そして，情報の新規到着を検出すると，到着情報を RDB へ格納し，対応する実行プランに従って RDB に SQL 問合せを発行する．このときの SQL 問合せには，結果を生成するための処理の他に，デルタリレーションやウインドウリレーションの更新処理が含まれる．SQL 実行によって得られた処理結果は直ちにユーザへ配信される．ただし，今回の評価実験では，結果の配信までは行わずにログへ出力して処理を完了するものとした．

本実験で用いたストリーム型情報源はすべて擬似的に構築したものである．通常のストリーム型情報源として，10 秒おきにデータを配信するものを 2 種類用意した．クロックストリームは 10 種類用意し，それぞれ異なるタイミングで 5 分おきにアラームを配信するものとした．

最適化の対象とする連続的問合せの集合は，共通のテンプレートを元に人工的に生成したものを用いた．各問合せの MASTER 節には，前述の 10 種類のクロックストリームからランダムに選んだ 1 つを与え，WINDOW SIZE 節には，30 秒から 300 秒の範囲の整数値をランダムに選択して指定するものとした．SELECT-FROM-WHERE の部分はすべて同一とした．

実験：すべての問合せを独立に実行した場合と，複数問合せ最適化を適用して実行した場合の比較を行った．本実験ではストリーム型情報源からの情報を 15 分間受信し，期間終了までに発生した処理の合計時間を測定した．ここでは複数問合せ最適化のクラスタリングのパラメータ θ を 0.5 に固定とした．問合せ数を変化させたときのそれぞれの総処理時間の変化を図 6 に示す．図 6 から，問合せを独立に実行する場合よりも複数問合せ最適化を適用した場合の方がはるかに高速であることが確認できる．

5. まとめと今後の課題

本稿では，多数の連続的問合せが大量に与えられる状況を想定し，複数問合せ最適化方式を提案した．本提案手法は，実行タイミングの違いによる問合せの参照範囲の違いを考慮し，参照範囲の類似度が高いものをクラスタ化した後，クラスタ単位で共通演算の共有を行うというものである．評価実験で複数の連続的問合せ処理における本提案手法の有効性を確認した．

今後の課題としては，より詳細な評価実験がある．クラスタリングパラメータ θ を変化させた場合の処理効率の変化などについて検討する予定である．

[謝辞]

本研究の一部は，日本学術振興会科学研究費補助金基盤研究(B)(12480067)による．

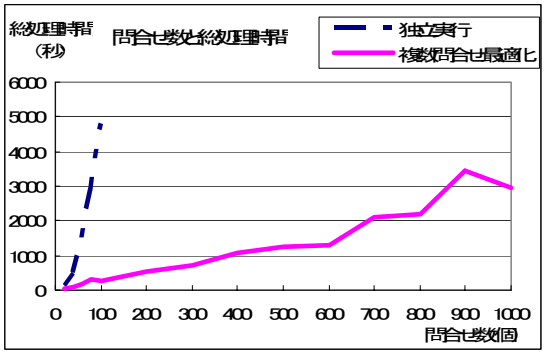

図 6　実験結果
Fig.6 Result of experiment

[文献]

[1] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. “NiagaraCQ: A Scalable Continuous Query System for Internet Databases,” Proc. SIGMOD 2000, pp. 379-390.

[2] J. Kang, J. F. Naughton, and S. D. Viglas. “Evaluating Window Joins over Unbounded Streams,” Proc. ICDE2003.

[3] S. Madden, M. Shah, J. M. Hellerstein, and V. Raman. “Continuously Adaptive Continuous Queries over Streams,” Proc. SIGMOD 2002, pp. 49-60.

[4] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. “Efficient and Extensible Algorithms for Multi Query Optimization,” Proc. SIGMOD 2000, pp. 249-260.

[5] G. Salton. “Automatic Information Organization and Retrieval,” McGraw-Hill Book Company, 1968.

[6] T. K. Sellis. “Multiple-Query Optimization,” ACM Transactions on Database Systems, 13(1), 1988, pp. 23-52.

[7] D. Terry, D. Goldberg, and D. Nichols. “Continuous Queries over Append-Only Databases,” Proc. SIGMOD 1992, pp. 321-330.

渡辺 陽介　Yousuke WATANABE

2001 筑波大・第三学群情報学類卒．同年 同大学大学院博士課程システム情報工学研究科入学．異種分散情報源統合システムにおける配信型情報源の統合利用に関する研究に従事．日本データベース学会，情報処理学会各学生会員．

北川 博之　Hiroyuki KITAGAWA

1978 東大・理・物理卒．1980 同大大学院理学系研究科修士課程了．日本電気（株）勤務の後，1988 筑波大学電子・情報工学系講師．同助教授を経て，現在，筑波大学電子・情報工学系教授．理学博士（東京大学）．異種情報源統合，文書データベース，WWW の高度利用などの研究に従事．著書「データベースシステム」（昭晃堂），「The Unnormalized Relational Data Model」（共著，Springer-Verlag）等．日本データベース学会，情報処理学会，日本ソフトウェア科学会，ACM,IEEE-CS 各会員．