ストリームの動的特性変化を考慮した連続的問合せ最適化方式

渡辺 陽介[†]，北川 博之[††]

概要

ネットワークの発達に伴って，時々刻々と変化する情報をオンラインで提供するデータストリームと呼ばれる情報源の利用が増加し，ストリームに対するフィルタリングや複数ストリームの統合などストリームに対する問合せ処理の重要性が高まっている．我々の研究グループでは，多数のストリーム型情報源に対する大量の連続的問合せが与えられた際に，その効率的実行を実現するための連続的問合せの複数問合せ最適化方式を提案してきた．本手法は，情報の到着ログを元に連続的問合せの実行をシミュレートし，実行タイミングや参照範囲が近い同士の問合せ同士をクラスタリングすることによって，効率的な実行処理プランを導出するというものである．しかし，データの到着パターンが変化すると，古い到着ログを元に計算したクラスタでは最適な処理を行うことができなくなるため，問合せ最適化を再実行する必要がある．本稿ではこれまでの手法を拡張し，到着パターンが変化する環境における動的な連続的問合せの最適化手法を提案する．

Adaptable Optimization of Multiple Continuous Queries over Data Streams with Variable Arrival Patterns

Yousuke WATANABE[†], Hiroyuki KITAGAWA[††]

Abstract

Recent development of network technologies has enabled us to use a variety of data streams, and a demand for query processings on data streams has been increasing. Based on these backgrounds, our research group has proposed a novel query optimization method for multiple continuous queries over data streams. The proposed scheme forms clusters of continuous queries based on their execution time, and applies a groupwise multiple query optimization technique. However, once data arrival patterns have changed, clusters based on old patterns are no longer optimal. In this paper, we extend our scheme to cope with variable data arrival patterns.

1　まえがき

ネットワークを用いた情報流通が一般化したことにより，我々は様々な情報を容易に利用することが可能となった．情報源の形態も多様化し，静的な情報を提供する RDB のような従来の情報源に加えて，ニュースや天気予報，株価など時々刻々と変化する情報をオンラインで提供する，ストリーム型の情報源が用いられるようになってきた．ストリーム型情報源の代表的な例としては，データ放送やメールマガジン，メッセージングシステムなどが挙げられる．現在，ストリーム型情報源の増加によって，到着情報のフィルタリングや情報源の統合利用のような，ストリーム型情報源に対する問合せ処理を実現するシステムの需要が高まっている．特に，ストリーム型情報源の利用者の増加に伴って，多数のユーザの大量な要求にも応じられる効率的な問合せ処理を実現する処理方式が求められている．

このような背景から，我々の研究グループでは連続的問合せ (Continuous Query)[2, 4, 8] を対象とした複数問合せ最適化方式 [9] を提案してきた．連続的問合せは，ストリーム型情報源から到着した情報に対して繰り返し問合せ処理を適用し，前回の実行時からの差分となる結果を生成するものである．複数問合せ最適化 [5, 7] は，同時に実行される複数の問合せ中に共通に出現する演算を共有化することで，問合せ全体の処理量を削減するための手法である．しかし，ストリームを扱うことを想定した連続的問合せでは，時間の経過に合わせた到着情報の処理を行わなければならないため，問合せの実行タイミングの明示的な指定や，問合せが結果を生成するために参照する情報の時間範囲を与えるための時間条件記述が一般に必要となる．問合せごとに実行タイミングとその参照範囲が異なることから，従来の RDB における複数問合せ最適化をそのまま適用することはできないが，実行タイミングと参照範囲が近い問合

[†] 筑波大学 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[††] 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

せ同士であれば処理結果の一部を共有化をすることは可能である．我々の提案方式は，情報源の到着ログを元に各連続的問合せの実行タイミングと参照範囲を調べ，近いタイミングで共通な結果を多く生成する演算の処理結果を共有するというものである．

本稿では，これまでの方式を拡張し，動的に到着パターンの変化するストリーム型情報源を考慮した動的な複数問合せ最適化手法を提案する．これまで本研究では，ある程度定常的な情報の到着パターンをもつストリーム型情報源を対象にしていたため，一度の最適化を行うだけで十分な効果が得られたが，一般的なストリーム型情報源においては，到着ログを用いて得られた情報通りに将来も情報が到着することは保証されていない．到着パターンがログを解析した時点から変化してしまえば，効率の良い処理を提供することはできない．提案手法では，到着パターンの変化に合わせて複数問合せ最適化を繰り返し適用することで，効率の良い処理を提供し続けることが可能である．

2 連続的問合せ

まず，本研究が対象とする連続的問合せについて説明する．

2.1 基本モデル

本研究ではストリーム型情報源がリレーションとしてモデル化されていることを想定する．情報源から送られてきたデータを，そのリレーションの 1 タプルとして扱い，情報の到着時刻 TS がタプルに付加されるものとする．

連続的問合せは，以下のような形式で記述する．

CREATE *query_name*
MASTER *master_source*,...
[**WINDOW SIZE** *window_size*]
SELECT–FROM–WHERE *sql*

CREATE 節には，問合せの識別子となる名前を与える．MASTER 節は，問合せ実行のきっかけとなるストリーム型情報源（マスタ情報源）を記述する．指定されたマスタ情報源から情報が到着するとその問合せが実行される．マスタ情報源には任意のストリーム型情報源が指定可能なほか，システムが提供するクロックストリームを記述することができる．クロックストリームは指定された時刻に定期的にアラームを配信する仮想的なストリーム型情報源である．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 [3] 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．SQL 文によって記述された各演算は，新規に到着した情報を用いて新たに生成可能な処理結果を差分的に出力する．

```
CREATE Example_1
MASTER Clock_12
WINDOW SIZE 8 * hour
SELECT *
FROM Quote, News
WHERE Quote.name = News.company
```

図 1: 問合せ 1

ウインドウ結合は複数のストリーム型情報源に対する統合演算として提案されてたものである [3]．ストリーム型情報源の情報は，一般には時間の経過と共に重要度が低下し，また，離れた時期に届いたものほど関連が少ないことが多いため，非常に新しい情報と非常に古い情報を統合したいという要求は現実には少ない．さらに，ストリーム型情報源では，サービス提供中は逐次にタプルが到着するため，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扱わなければならない．ウインドウ結合では，到着してからウインドウ幅以内の時間しか経過していない最近のタプルのみを結合演算の適用対象とすることが可能である．

2.2 問合せ例

図 1 は連続的問合せの例で,「毎日 12 時に過去 8 時間分の株価情報 Quote とニュース情報源 News から届いた情報を統合して提供して欲しい」という要求を表している．ただし，マスタ情報源である Clock_12 は毎日 12 時にアラームを配信するようなクロックストリームを表しており，hour は 1 時間の単位を表す定数である．

この問合せを処理木にすると図 2 のようになる．処理木はルートノード，演算ノード，デルタリレーションから構成される．ルートは処理結果の出力を表し，

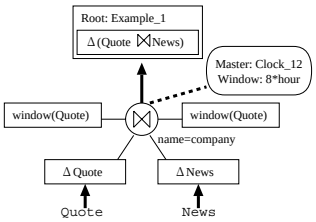

図 2: 問合せ 1 の処理木

ここに格納されたタプルはユーザへ配信される．Δで始まる名前を持ったはノードはデルタリレーションを表している．デルタリレーションは各演算にとっての入出力キューに相当し，一度演算の入力として使われたタプルは入力側のデルタリレーションから除かれ，処理結果は出力側のデルタリレーションまたはルートへ格納される．演算ノードは，それぞれマスタ情報源の情報とウインドウの情報を保持している．図中の ⋈ はウインドウ結合を表す．また，ウインドウ結合の処理結果は，新規到着情報を格納したデルタリレーションだけでは生成できない．図中の $window(Quote)$ や $window(News)$ は，新規情報ではないがウインドウの時間条件の範囲内に含まれるタプルを保持するためのウインドウリレーションである．ウインドウ結合の結果は以下のような処理によって生成される．

$$\Delta Quote \bowtie window(News)$$
$$\bigcup window(Quote) \bowtie \Delta News$$
$$\bigcup \Delta Quote \bowtie \Delta News$$

各リレーション中のタプルは，現在時刻から問合せのウインドウ幅分だけ古くなった時点で期限切れ状態になる．期限切れタプルの廃棄処理はガベージコレクタが定期的に行う．

3 連続的問合せの複数問合せ最適化手法

本節では，我々が提案した連続的問合せの複数問合せ最適化方式 [9] について例を用いて概説する．ここでは，2.2 節で示した図 1 と，図 3，図 4 のような 3 つの問合せを用いる．これら 3 つの問合せは共通する処理として $Quote \bowtie News$ を含んでいるが，マスタ情報源の違いから，問合せ 1 は毎日 12 時，問合せ 2 は毎日 13 時，問合せ 3 は毎日 0 時にそれぞれ実行される（図 5）．問合せ 1 と問合せ 2 の実行タイミングは近く，参照する範囲に重なりが存在するが，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せず，演算の結果を共有しても意味はない．我々の提案した手法では，このような実行タイミングとウインドウ範囲の近さによる各問合せの参照範囲の重なり度合いを問合せ同士の類似度として数値化し，クラスタリングによる分類を行っている．

```
CREATE Example_2
MASTER Clock_13
WINDOW SIZE 8 * hour
SELECT *
FROM Quote, News
WHERE Quote.name = News.company
```

図 3: 問合せ 2

```
CREATE Example_3
MASTER Clock_0
WINDOW SIZE 8 * hour
SELECT *
FROM Quote, News
WHERE Quote.name = News.company
```

図 4: 問合せ 3

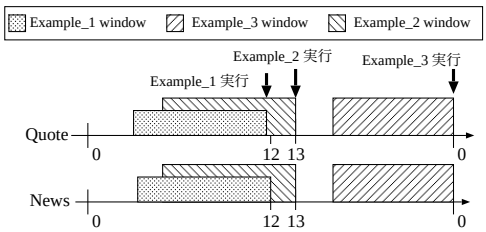

図 5: 参照範囲の比較

3.1 問合せの類似度とクラスタリング

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．ただし，外部の一般的なストリーム型情報源では，情報の到着パターンが時間の経過と共にログのパターンから変化する場合が存在する．4 節で述べる動的な問合せ最適化方式ではこの点を考慮し，動的な類似度の再計算を行う．

以下で類似度の概念について述べる．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用意する．情報源 $I_i(1 \leq i \leq n)$ に対する到着時刻集合を S_{I_i} とする．次に，問合せ $Q_j(1 \leq j \leq m)$ 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 が参照するタプルの到着時刻の集合を求める．これは Q_j のマスタ情報源の到着時刻集合の和集合 MS_j の各到着時刻について 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 の到着時刻の集合を計算することによって得られる．

$$USE_{I_iQ_j} = \bigcup_{s \in MS_j} \{t | t \in S_{I_i} \wedge t \in [s - windowsize_j, s]\}$$

ただし，$windowsize_j$ 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，WINDOW SIZE 節を持たない問合せの場合には無限大とする．

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

$$sim(Q_a, Q_b) = \min_{I_i \in Ref(Q_a) \cap Ref(Q_b)} \frac{|USE_{I_iQ_a} \cap USE_{I_iQ_b}|}{|USE_{I_iQ_a} \cup USE_{I_iQ_b}|}$$

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．また，共通に利用する情報源がない場合 $(Ref(Q_a \cap Ref(Q_b) = \emptyset)$ の類似度は 0 とする．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と，m 個の問合せに対しては，計算結果として $m \times m$ の対称行列（類似度行列）が得られる．例において，ログから図 6 のような $USE_{I_iQ_j}$ が得られたならば，類似度行列は図 7 のようになる．行列の (k, l) 成分は類似度 $sim(Q_k, Q_l)$ の値に対応している．

計算された類似度行列を用いてクラスタリングを行い，類似度の高い問合せ同士のクラスタを生成する．クラスタ生成のアルゴリズムは，通常の階層的クラスタリング手法 [6] と同様であるので，詳細は省略する．ここでは，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．図 7 の場合に閾値 0.4 でクラスタリングを適用した場合は，$\{Example1, Example2\}, \{Example3\}$ のような 2 つのクラスタが生成される．

3.2 クラスタ内の共通演算の共有

問合せを類似度に基づいてクラスタリングしたことにより，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，あとはクラスタごと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導出する．この部分の処理には，従来の複数問合せ最適化 [5, 7] を適用する．ただし，最適な問合せ実行プランの候補となるパスが複数存在する場合は，従来のコスト見積りによるパス選択は行わず，共有される演算数を最大とするようなパスを最適プランとして選択する．

$$\begin{aligned}
USE_{Example_1, Quote} &= \{t2, t4\} \\
USE_{Example_2, Quote} &= \{t2\} \\
USE_{Example_3, Quote} &= \{t3, t5\} \\
USE_{Example_1, News} &= \{t6, t8\} \\
USE_{Example_2, News} &= \{t6, t8\} \\
USE_{Example_3, News} &= \{t7, t9\}
\end{aligned}$$

図 6: 例における参照情報

$$\begin{bmatrix} 1.00 & 0.50 & 0.00 \\ 0.50 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

図 7: 類似度行列

単純な例として，問合せ 1（図 1），問合せ 2（図 3）の 2 つの問合せを含むクラスタにおいての演算の共有処理を説明する．

まず，各問合せの処理木からマージ可能な演算ノードを探索する．問合せ 1 は図 2 のような構造の処理木を持ち，問合せ 2 もほぼ同様な構造をもっている．この例においては共通な演算が 1 つずつしか存在しないため，ほぼ自明なマージ結果が得られる（図 8）．このとき，マージされた演算は両方の処理木のマスタ情報源の和集合をマスタ情報源としてもち，ウインドウ幅は最大のウインドウをもつ演算のものに調整される．図 8 におけるウインドウ結合の演算ノードは Clock_12 が到着したときと，Clock_13 が到着したときの両方で実行され，それぞれのルートノードへ処理結果を出力することを表している．

共通演算のマージが完了したら，実行プランの選択を行う．一般に，問合せが複数の演算から構成される場合は，その実行順序の候補が複数あるため，ルートノードへ到達する経路は複数となる．先ほども述べた通り，ここ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．図 8 においては，パスは各ルートノードに 1 つしか存在しないため，実行プランは自明である．

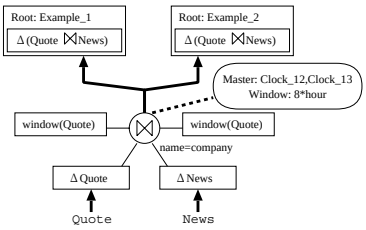

図 8: マージ後の処理木

4 動的な複数問合せ最適化

本節では動的な複数問合せ最適化方式について述べる．

ストリームからの到着パターンが変化する環境においては，問合せ間の類似度が大きく変動する．ストリームに対して最適な処理を続けるためには，絶えずクラスタ内の問合せ間の類似度を監視し，類似度が閾値 θ を下回る期間が一定期間を越えるようなら，クラスタの再構成と実行プランの再導出を行うことになる．

しかし，実行中のシステムにおいて，クラスタの再構成処理を行うためには，以下のような問題が存在する．

- **各問合せの参照情報 $USE_{I_iQ_j}$ の維持**
  実行中のシステムでは，類似度の計算に必要な各問合せの参照情報 $USE_{I_iQ_j}$ は，実際に行われた処理をトレースすることで容易に得ることができる．しかし，$USE_{I_iQ_j}$ の維持には問合せ数と到着タプル数に応じた記憶領域を必要とするため，大量の問合せが与えられた場合には，主記憶上に配置できないサイズになりうる．

- **計算時間**
  クラスタの再構成中は問合せ処理を一時中断し，その間に届いた情報は到着キューへ蓄積される．短時間で再構成を終了することができない場合，長大な到着キューが必要となる．しかし，問合せ間の類似度の計算および，クラスタリングの処理は問合せ数に対してそれぞれ二乗以上の処理コストであるし，クラスタ内の各問合せの処理木をマージする処理のコストは各処理木中の演算数の二乗であるので，大量の問合せに対するクラスタリングでは，再構成にかかる時間が長くなってしまう．

- **問合せ結果の整合性の保証**
  各問合せはそれぞれ異なった実行タイミングを持つので，稼動中のシステムでは問合せごとに到着情報に対する処理の進行状況が異なっている．処理の進行状況が異なった状態で問合せのクラスタを再構成し，演算を共有した場合，それまでの処理の進行状況が無視され，処理結果の欠如や重複を招いてしまう可能性がある．クラスタの再構成を行うためには，処理状況の同期が必要不可欠である．また，ウインドウリレーションに蓄積された情報がクラスタ再構成と共に失われてしまうとウインドウ結合の結果が欠落してしまう．ウインドウリレーション内のタプルは，再構成されたクラスタにも継承されなければならない．

4.1 ベースグループ

全ての問合せに関する参照情報を保持することは現実的でなく，また，稼動中のシステム上で全ての問合せのペアに対し類似度の計算とクラスタリングを行うことは非常にコストが高い．そこで，本研究では必要な記憶領域と計算量の削減のため，クラスタリングの最小単位を問合せではなく，ベースグループという問合せの集合に変更する．ベースグループの基本的なアイデアは，どのような到着ログに対しても必ず高い類似度を持つことが予想される問合せ同士をあらかじめ１つにまとめておくというものである．例えば，まったく同じ要求を記述した問合せ同士などでは，クラスタリングにより必ず同じクラスタへ分類されることが明らかである．

本研究で用いるベースグループは以下の条件を満たす問合せの集合 BG と定義する．

$$\begin{aligned}&\forall q_i, q_j \in BG \\ &\quad (Master(q_i) = Master(q_j) \\ &\quad \wedge Ref(q_i) \cap Ref(q_j) \neq \phi \\ &\quad \wedge \frac{min(windowsize_{q_i}, windowsize_{q_j})}{max(windowsize_{q_i}, windowsize_{q_j})} \geq \theta)\end{aligned}$$

この式は，同一のベースグループに属する問合せは，同一のマスタ情報源をもち，さらに共通の情報源を閾値 θ 以上の割合で参照していなければならないことを表している．ただし，θ は階層的クラスタリングにおいてクラスタ生成を行う際に使用したパラメ

タ θ と同一である．ベースグループの生成は，与えられた問合せ集合に対してシステムの起動時などに一度だけ行う．

次に，ベースグループ $BG_k(1 \leq k \leq K)$ が参照する情報の集合 $USE_{I_iBG_k}$ について述べる．$USE_{I_iBG_k}$ はベースグループ内の各問合せ Q_j の $USE_{I_iQ_j}$ の和集合とする．

$$USE_{I_iBG_k} = \bigcup_{Q_j \in BG} USE_{I_iQ_j}$$

ベースグループ同士の類似度 $sim(BG_a, BG_b)$ は，問合せの類似度の計算式をほぼそのまま適用する．

$$sim(BG_a, BG_b) = \min_{I_i \in Ref(BG_a) \cap Ref(BG_b)} \frac{|USE_{I_iBG_a} \cap USE_{I_iBG_b}|}{|USE_{I_iBG_a} \cup USE_{I_iBG_b}|}$$

また，稼動中のシステムではベースグループ内の各問合せの実行をトレースすることで，新たな参照情報 ΔUSE_q を差分的に得ることができる．これを USE_{I_iBG} へ追加するには単純に以下の式を用いる．

$$USE'_{I_iBG} = USE_{I_iBG} \cup \Delta USE_{I_iq}$$

ベースグループをクラスタリングの最小単位とすることで，参照情報の管理に必要な記憶領域や類似度計算およびクラスタリングの処理コストは，問合せ数ではなく，ベースグループの数によって決まることになる．一般に，ベースグループ数は問合せ数よりもはるかに少ないことが期待されるので，大幅な計算量の削減が見込まれる．

また，クラスタの再構成の度に発生する実行プラン再導出の処理も高速化することができる．ベースグループ内の問合せ同士についてはベースグループを生成した段階であらかじめ行っておくことが可能であるので，クラスタ再構成時には，ベースグループ間の処理木のマージと，最適な実行プランの選択のみを行うだけで済む．

4.2 問合せ結果の整合性の保証

クラスタの再構成によって処理結果が変化しないことを保証するためには，問合せ間の処理の進行状況の同期をとらなければならない．ここでは，もっとも単純な手法として，結果の配信処理を除いた全演算の実行を行い，全ての問合せの処理の進行状況を最新の到着情報まで完了した状態に統一することで同期を取るものとする．結果の配信処理だけはユーザの要求に従う必要があるので，マスタ情報源が到着した時点で行われる．同期の処理では全クラスタ中の全演算を実行するため，一般にクラスタ数に比例した処理時間がかかる．これについては，クラスタごとに内部の処理が独立していることを活かし，マルチスレッドによる並列処理等によって処理時間を削減する．

また，ウインドウ結合による処理結果を保証するためには，各演算におけるウインドウリレーション中のタプルを，再構成されたクラスタにも引き継がれなければならない．本研究では，処理木の結合演算ノードから処理内容に基づいたキーを生成し，再構成前のクラスタ中のウインドウリレーションをキーと対にして保存する．クラスタ再構成後は，各結合演算と同一のキーをもつウインドウリレーションを探し，再びそれを割り当てる処理を行う．一般には，再構成前と再構成後で，クラスタの数は同一ではないため，割り当てられるウインドウリレーションの数が不足することがある．その場合には，ウインドウリレーションを複製して割り当てを行う．

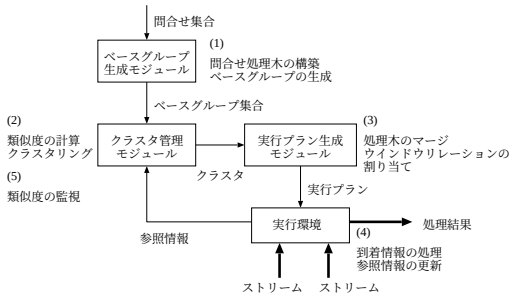

図 9: システム構成

4.3 処理概要

動的な連続的問合せの複数問合せ最適化の処理手順を示す（図 9）．

1. ベースグループ生成モジュールは与えられた問合せの集合をベースグループへ分類する．この段階で，ベースグループ内の問合せの処理木に出現する共通演算はあらかじめマージしておく．
2. クラスタ管理モジュールは参照情報を元に，与えられたベースグループの集合からクラスタを生成する．初回の起動で，まだ参照情報が得ら

CPU	UltraSPARC-II 296MHz × 2
メモリ	1GB
OS	Solaris 2.6
開発言語	Java (J2SE 1.4.0, J2EE 1.3.1)
使用 RDBMS	PostgreSQL 7.3.1

表 1: 実験環境

れていない場合は，すべてのベースグループを含むクラスタを 1 つ生成する．

3. 実行プラン生成モジュールは，同一クラスタに属すベースグループの処理木に出現する共通演算をマージし，最適な実行プランを生成する．生成された実行プランは実行環境に渡される．

4. 実行環境上では，ストリーム型情報源から到着した情報に対して，実行プランに基づいた処理が行われる．演算の実行時に参照されたタプルの情報はベースグループ単位で逐次記録される．

5. クラスタ管理モジュールは定期的に参照情報からクラスタ内の類似度を計算し，現在のクラスタが閾値 θ 以上の類似度を保っているかをチェックする．もし，類似度が θ 未満である状態が一定期間続いた場合は，クラスタの再構成のため，(2) 以降の処理を行う．

5 評価実験

本節では本研究において行った評価実験について述べる．

まず，実験システムは表 1 のような環境で実装を行った．図 9 の実行環境に対応する部分は，Java プログラムと PostgreSQL の組合せにより実現した．新規情報が到着すると，対応する実行プランに基づいて PostgreSQL へ必要な SQL 問合せを発行する．

次に，実験に用いたデータの内容について述べる．問合せは図 10 のテンプレートを元にランダムに 300 個を生成した．マスタ情報源には $M_0, \ldots, M_9$ の 10 種類のうちの 1 つが指定される．各マスタ情報源からは 10 単位時間を 1 周期として必ず 1 つのデータが到着するものとした．それぞれのマスタ情報源の到着パターンは，時刻 $t = 0, \ldots, 10$ では図 11 に従い，時刻 $t = 11, \ldots$ では図 12 に変化する．また，ストリーム型情報源 Quote と News は 1 単位時間のうちにそれぞれ 24 個のデータが定常的に到着するものとした．なお，本実験では 1 単位時間は 2 分とした．

CREATE Q_i $(0 \leq i \leq 299)$
MASTER M_j $(0 \leq j \leq 9)$
WINDOW SIZE $unittime * k$ $(k \in \{3,4,5\})$
SELECT *
FROM Quote, News
WHERE Quote.name = News.company

図 10: 問合せテンプレート

図 11: 学習に用いたマスタ情報源の到着パターン

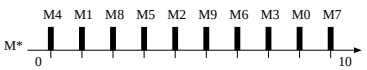

図 12: 変化したマスタ情報源の到着パターン

以上のようなデータに対し，本実験では (1) 動的にクラスタリングを行う方式，(2) 初期のログ情報を用いて生成したクラスタをそのまま使い続ける方式，(3) 類似度を考慮せずに全問合せを 1 クラスタとして実行する方式の 3 種類を処理時間で比較した．ログから図 11 の到着パターンを読み出し，(1),(2) の方式それぞれでクラスタを生成する．そして，(1) の方式では $(t = 20)$ においてクラスタの再構成を行うようにした．

実験結果は図 13 である．まず，(3) の方式は他の 2 つに比べ，常に処理時間が長いが，これは問合せ間の類似度が考慮されておらず，不要な処理結果が大量に生成されているためである．方式 (1) と (2) は $t = 20$ までは同様の処理が行われる．$10 < t \leq 20$ では到着パターンが変化したため，$0 < t \leq 10$ よりも全体的に処理時間が増加する．$t = 20$ において，クラスタリングの再構成を行ったことにより，$t > 20$ では (1) の方が (2) よりも処理時間が短くなっている．

6 関連研究

NiagaraCQ[2] でも，連続的問合せに対する複数問合せ最適化を行っている．彼らの複数問合せ最適化方式は，連続的問合せがインクリメンタルに追加・削除される状況を想定しており，既に存在する問合せ

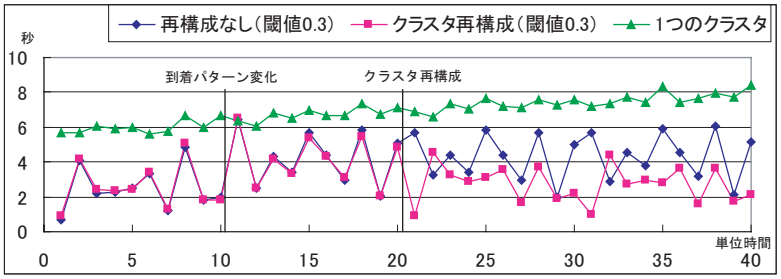

図 13: 実験結果

に対する実行プランをベースに処理の方法を決定することが特徴である．NiagaraCQ で対象とする連続的問合せは，ウインドウ結合のような時間条件を伴う演算や問合せの実行タイミングを考慮していない．そのため，上記のインクリメンタルな処理を除いては，基本的には従来の複数問合せ最適化が適用できる状況となっている．しかし，ストリーム型情報源に対する問合せにおいて時間条件を全く考慮しないのは非現実的であり，その場合には NiagaraCQ の最適化方式を直接適用することはできない．

CACQ[4] も大量の連続的問合せを処理することを目的としたシステムである．CACQ では eddy[1] を用いて，情報源の性質の変化やフィルタ条件の変化に対して演算の適用順序を変えるという，適応性の高い問合せ処理を実現している．ただし，eddy は各演算の選択率の変化には対応可能だが，本研究が対象とするような情報の到着パターンの変化については特に考慮されていない．

7 むすび

本稿では，我々の研究グループが提案した連続的問合せに対する複数問合せ最適化方式を拡張し，情報の到着パターンの変化に適応可能な動的な複数問合せ最適化方式を提案した．

今後の課題としては，より大規模な評価実験及び実データを用いた評価実験が挙げられる．

謝辞

本研究の一部は，日本学術振興会科学研究費基盤研究 (B)(15300027)，特別研究員奨励費 (0330)，文部科学省科学研究費特定領域研究 (2)(15017207) による．

参考文献

[1] R. Avnur, and J. M. Hellerstein. ”Eddies: Continuously Adaptive Query Processing,” SIGMOD 2000, pp. 261–272.

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

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

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

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

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

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

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

[9] 渡辺陽介，北川博之．“連続的問合せに対する複数問合せ最適化”，電子情報通信学会 第 14 回データ工学ワークショップ (DEWS2003), 2003.