社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DBS－140 (II) (66)
2006／7／13

ストリーム管理システムにおける永続化要求の妥当性評価

山田真一[†], 渡辺陽介[††], 北川博之[†,†††], 天笠俊之[†,†††]

概要

デバイス技術やネットワーク技術の発達により，実世界から得られるデータストリームが増加し，それらに対する問合せ処理が重要となっている．ストリームに対する問合せ要求の１つとして，新規到着データに対して連続的に処理した結果を逐次 DB に格納してほしいという蓄積要求がある．我々の研究グループでは，ストリーム処理エンジンと DB を連携させることで，この種の蓄積問合せを処理するシステムの開発を行っている．蓄積要求の記述にはシステムの性能，ストリームデータの到着頻度やデータ量，DB の特性を考慮する必要があるが，一般に利用者がそれらの特徴を正確に判断することは難しい．本研究では，蓄積問合せの作成を支援するため，システムが処理可能な問合せを利用者に提示する機能の開発を目的とする．本稿ではその前段階として，ストリームに対する連続的問合せが与えられたときに，その問合せを処理し続けることができるかを判定する手法を提案する．本稿で提案する手法は，ストリームの到着レートを考慮したコストモデルから処理結果の出力レートを計算し，処理結果が DB に書込可能かを判定する．本稿ではまた，提案手法が書込判定に利用可能かを確認するために行った予備実験について述べる．

Feasibility Evaluation of Data Persistency Requirements in a Stream Management System

Shinichi YAMADA[†], Yousuke WATANABE[††], Hiroyuki KITAGAWA[†,†††] and Toshiyuki AMAGASA[†,†††]

Abstract

Today, the amount of data delivered as data streams is increasing, and query processing over data streams has become important. An example of requirements is storing results generated by continuous queires to databases. We are developing a system to fulfill such requirements by combining a stream processing engine and DBMSs. The system performs query processing of data streams and data archiving. To specify feasible persistent requirements, users must consider characteristics of data streams and DBMSs. However, it is difficult for users to decide which queires are feasible in the system. In this paper, we propose a feasibility evaluation method to detect queires which exceed system capacity. The proposed method determines the capability of our system by using rate-based cost model.

1 はじめに

近年，デバイス技術やネットワーク技術の発達により，ニュースや株価情報，センサから得られる実世界情報など，最新の情報を逐次オンラインで提供するストリーム型情報源が増加してきており，ストリームに対する問合せ要求が複雑化している．ストリームに対する問合せ要求としては，新規到着デー

† 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
†† 科学技術振興機構 戦略的創造研究推進事業
Core Research for Evolutional Science and Technology, Japan Science and Technology Agency
††† 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

タに対する連続的な問合せ処理要求の他に，その処理結果を履歴データとして蓄積する要求や新規到着データと履歴データを統合した問合せ処理要求など，ストリームの履歴データを蓄積し，活用する要求が考えられる．ストリームに対する問合せを実現するシステムとしてストリーム処理エンジン [2, 4, 5, 6] があるが，それらはいずれも新規到着データに対する連続的問合せ処理のみが考慮されている．そこで我々の研究グループでは，ストリーム処理エンジンと DB を連携させることで，ストリームの履歴データに対する問合せ要求を実現するストリーム管理システム Harmonica[1] の開発を行っている．

一般に DB にデータを書込む処理はストリーム処理に比べて非常に遅いので，Harmonica が履歴データの蓄積要求を処理し続けるためには，ストリーム情報源の到着頻度や配信データ量，蓄積に用いる DB の特性を考慮にいれて，データが損なわれることの無いような蓄積要求問合せを発行することが必要である．しかし，利用者がストリーム情報源や DB の特徴を正確に判断し，問合せを記述することは非常に困難である．

本研究では，利用者による問合せ記述を支援するため，システムがストリーム情報源や DB の特性を考慮し，処理可能な蓄積要求問合せを利用者に提示・推薦する機能の開発を行う．この機能の実現のためには，利用者から与えられた問合せが処理可能かを判定する機能，利用者の要求に基づいて処理可能な問合せを提示する機能が必要となる．本稿では，利用者の問合せが処理可能かを判定する機能に焦点を当てて，その判定手法について述べる．本稿で提案する判定手法は，ストリームの到着レートを考慮したコストモデルからストリーム処理結果の出力レートを求め，処理結果がその出力レートで DB に書込むことができるかを判定する．本稿ではまた，提案判定手法が実際の DB において書込可能の判定ができるかを確認するために行った予備実験についても述べる．

以下は，次のような構成になっている．まず，2 節で本研究で開発を行っているストリーム管理システムの概要について説明する．3 節で提案する判定手法について述べ，4 節で具体的な問合せの判定例について述べる．5 節で行った予備実験について説明した後で，6 節で関連研究を紹介する．最後に 7 節で，まとめと今後の課題について述べる．

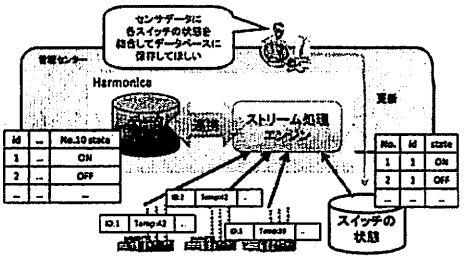

図 1: Harmonica システムの利用例

2 ストリーム管理システムの概要

本節では，我々の研究グループで開発を行っているストリーム管理システム Harmonica[1] の概要について述べる．

Harmonica はストリーム処理エンジンと DB を連携することで，新規到着ストリームデータに対する連続的問合せ処理や，履歴ストリームデータの蓄積，新規到着ストリームデータと履歴ストリームデータに対する統合処理を実現する．現在プロトタイプシステムでは，我々の研究グループで開発を行っているストリーム処理エンジン StreamSpinner[2] と MySQL[7] を用いて構成されている．

2.1 利用例

Harmonica の具体的な利用例として，ガスタービン発電所を各種センサを使ってモニタリングしている場合について説明する（図 1）．ガスタービン発電所には，ガスタービン内の温度や投入した燃料によるガスタービンの回転数などを逐次モニタリングするために，大量のセンサが付属している．複数のガスタービンから取得された各センサデータが管理センターに毎秒集められており，管理センター内の DB にガスタービンを制御するための各スイッチの状態が記録されているものとする．

このような状況では，例えば，センサから配信されるデータに各スイッチの状態を結合して，DB に保存してほしいという要求が考えられる．しかし，ガスタービン発電所からは大量のデータが次々と到着するため，DB に要求通り書き込み続けることができるかは管理者自らも正確に判断することは難しい．

Harmonica は，ストリームの到着レートや DB の特性を考慮して，利用者から与えられた問合せが処

```
MASTER Turbine
INSERT INTO MyTable VALUES (
    SELECT Turbine.*, Switches.*
    FROM Turbine[now], Switches
    WHERE Turbine.id = Switches.id
)
```

図 2: 問合せ例

理可能であるかを判定する．蓄積要求問合せを処理することができないと判定した場合には，同じような結果を生成するより書込み頻度の低い問合せを利用者に提示する．本稿では，利用者からの問合せが処理可能であるかを判定する機能について焦点を充てて説明する．

2.2 問合せ記述

Harmonica で扱う問合せについて説明する．Harmonica では**連続的問合せ**を処理する．連続的問合せとは，ストリームから新規に到着したデータに対して繰り返し問合せ処理を実行し，前回実行時からの差分の生成を繰り返す問合せである．連続的問合せは**マスタ情報源**と**ウインドウ**を指定することで行う．マスタ情報源とは，問合せ実行のきっかけを与えるストリームのことを言い，ウインドウとは演算の適用対象となるタプルを指定するための時間幅やタプル幅のことを言う．Harmonica ではまた，ストリームをタイムスタンプ型の属性を持ったリレーションとしてモデル化を行っているため，問合せには SQL ライクな問合せ言語を使用する．図 2 に 2.1 で挙げた，センサから配信されるデータに各スイッチの状態を結合して，DB に保存してほしいという要求の問合せ例を示す．Turbine はガスタービンから配信されるセンサデータのストリームであり，Switches はスイッチの状態を保存しておくリレーションである．Turbine のウインドウは時間幅で定義されており，その大きさは 0 (now) で，現在到着したデータのみが演算の処理対象となる．MASTER 節ではマスタ情報源を指定する．この例では Turbine が指定されているので，センサデータが到着する度に繰り返し処理を行う．次節で，問合せ処理可能性を判定するための手法について説明する．

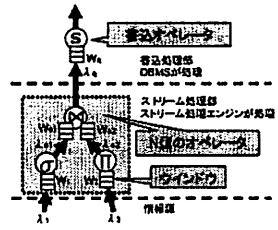

図 3: 問合せ処理

図 4: 処理可能な問合せのサイクル

3 問合せ処理可能性の判定手法

3.1 問合せ処理可能性の定義

2 節で紹介したように Harmonica はストリーム処理エンジンと DBMS を連携して開発されている．したがって図 2 に示すような問合せ要求は実際には，ストリーム処理エンジンが処理する**ストリーム処理部**と DBMS が書き込み処理を行う**書込処理部**に分けられている（図 3）．まずは，それぞれの処理部で処理可能であるための条件について説明する．ここでは，説明のためにストリームデータは一定間隔で到着し，時間によって頻度やデータ量が変化しないものとする．図 4 に処理可能な問合せのサイクルを示す．ここでいう処理時間というのは，ストリーム処理部での処理にかかる時間のことであり，書込時間というのは書込処理部での処理にかかる時間を示している．図 4 から分かるように，到着ストリームデータに対して連続的問合せを処理し続けるためには，データが到着してから次のデータが到着するまでの間にストリーム処理部での処理が完了することが必要である．したがって，ストリーム処理部での処理可能は以下のように定義できる．

定義 3.1 利用者からの問合せを q，q におけるストリーム処理部での演算を O_i，単位時間当たりに到着するタプル数を演算 O_i で処理する時間を τ_{O_i} とする．このとき，以下の条件を満たす q を**ストリーム処理可能**と定義する．

$$\sum_{i=1}^{n} \tau_{O_i} < 1 \qquad (1)$$

以降，単位時間当たりに到着するタプル数のことを**到着レート**と呼ぶ．ストリーム処理可能でない問合せは，データが到着したときには，まだ前回の処理を行っている状態となり，到着したデータの処理開始時間が遅れていく．最終的には，遅延が無限大に広がっていき破たんする．

同様に，ストリーム処理の結果を逐次 DB に書き込むためには，書き込むべき処理結果が生成されたときに，前回の書き込みが完了していればいいことが分かる．したがって，書込処理部での処理可能は次のように定義できる．

定義 3.2 利用者からの問合せを q，単位時間当たりに q のストリーム処理部によって出力されるタプル数を λ_q，単位時間当たりに DB に書き込むことができるタプル数を λ_S とする．このとき，以下の条件を満たす q を**書込処理可能**と定義する．

$$\lambda_q < \lambda_S \tag{2}$$

以降，λ_S，λ_q のことをそれぞれ**書込レート**，**出力レート**と呼ぶ．

以上より，問合せ処理可能は次のように定義できる．

定義 3.3 問合せ q がストリーム処理可能かつ書込処理可能のとき q を**問合せ処理可能**と定義する．

3.2 判定アルゴリズム

本研究で提案する判定アルゴリズムについて説明する．利用者から与えられた問合せ q が問合せ処理可能かを判定するアルゴリズムは以下のようになる．まず，q から処理木を生成する．次に，ストリームの到着レートと処理木から各演算の処理時間と q の出力レートを計算し，ストリーム処理可能の判定を行う．もし，q がストリーム処理可能ならば，先に計算しておいた出力レートと DBMS の書込レートを比較して書込処理可能の判定を行う．もし，q が書込処理可能ならば，q を問合せ処理可能と判定する．

本研究ではストリーム処理可能の判定に，Ahmed のコストモデル [3] を利用する．また，書込処理可能の判定に利用できるように Ahmed のコストモデルを拡張する．以下，3.3 で問合せ処理可能の判定のためのコストモデルについて説明し，3.4 で書込レートの推定手法について説明する．

表 1: 問合せ処理可能の判定に用いる変数

C_σ	選択演算，射影演算のコスト（時間/タプル）
$C_\bowtie$	結合演算，直積演算のコスト
f	タプルの選択率
λ_i	入力 i の到着レート（タプル/時間）
W	ウインドウに含まれるタプル数
λ_S	データベース S の書込レート
λ_q	問合せ q の出力レート
SS_q	問合せ q における出力結果のスキーマ

3.3 コストモデル

問合せ処理可能の判定に用いるコストモデルについて説明する．表 1 に，問合せ処理可能の判定に用いる変数の一覧を示す．前半の 5 つが Ahmed らによるコストモデルに従って定義しているストリーム処理可能の判定のための変数で，後半の 3 つが DB への書込処理可能判定のために本研究で新たに導入した変数である．

Ahmed らのコストモデルは，ストリームの到着レートとウインドウから，ストリーム処理における各演算の出力レートの推定を行い，出力レートと演算のコストから全体の処理にかかる時間を計測する手法である．ウインドウとしては，タプル幅のウインドウと時間幅のウインドウを考慮する．タプル幅のウインドウは，変数 W によってそのタプル数が示される．一方，時間幅のウインドウの場合は，$W = T * \lambda_i$ によって与える．ただし，T は時間幅の大きさを表している．また本研究では，情報源としてストリーム型情報源の他に DB を利用することを考えている．DB が情報源の場合には，到着レート $\lambda = 0$，ウインドウのサイズ $W =$ (DB 内の対象となる全タプル数) として考える．

本研究では，演算として選択・射影・結合・直積演算を想定している．以下では，各演算の出力レート，出力タプル数，処理時間の見積りについて説明する．

3.3.1 選択演算・射影演算

選択演算の場合，演算の出力レート λ_o とウインドウ W_o は，それぞれ選択率 f を用いて，$\lambda_o = f\lambda_i$，$W_o = fW_i$ と表すことができる．射影演算の場合は，演算の前後でタプル数に変化が無いので，選択演算

における選択率 $f=1$ の場合と等価であると考えることができる．また単位時間当たりの演算の処理時間 τ は，入力レート λ_i を用いて $\tau = C_\sigma \lambda_i$ と表せる．C_σ は1タプル当たりの選択演算にかかる処理時間でストリーム処理エンジンから取得する．

3.3.2 結合演算・直積演算

結合演算は2つのリレーションから1つのリレーションを出力する演算である．直積演算の場合は，選択率 $f=1$ の結合演算と見なすことができる．入力となる2つのストリームの到着レートをそれぞれ λ_L，λ_R，それぞれのウインドウを W_L，W_R とする．まず，L側の入力だけについて考えると，単位時間の間に λ_L のタプルが到着する．結合対象となるR側のウインドウには W_R のタプルが存在するので，単位時間にL側の入力によって出力されるタプルは $f\lambda_L W_R$．R側についても同様に考えることができ，単位時間にR側の入力によって出力されるタプルは $f\lambda_R W_L$．したがって，結合演算の出力レートは

$$\lambda_o = f(\lambda_L W_R + \lambda_R W_L) \tag{3}$$

となる．また，出力されるタプル数 W_o は，ウインドウ W_L の1タプルに対して fW_R のタプルが出力されるので $W_o = fW_L W_R$ となる．

結合演算の処理時間も出力レートの場合と同様に考えると，L側の到着タプルに対する結合処理に対する単位時間当たりの処理時間 τ_L は $\tau_L = C_\bowtie \lambda_L$．R側の到着による単位時間当たりの処理時間 τ_R は $\tau_R = C_\bowtie \lambda_R$．したがって，単位時間当たりの結合演算全体にかかる処理時間 τ は

$$\tau = \tau_L + \tau_R = C_\bowtie(\lambda_L + \lambda_R) \tag{4}$$

となる．$C_\bowtie$ は1タプル当たりの結合演算にかかる処理時間である．$C_\bowtie$ の中には，ストリームデータの到着によるウインドウ内のタプルの更新処理にかかる時間も含んでいる．$C_\bowtie$ はストリーム処理エンジンから取得する．

3.4 書込レートの推定

Harmonicaシステムでは，それを利用するコンピュータや内部で利用するDBMSに特に制約を設けていない．システムが起動するコンピュータによって，ディスクの書込速度が異なり，使用するDBMSによって挙動が異なる．このようなことから，一般的に書込レートを一意に設定することは難しい．そこで本研究では，単位時間当たりにどのくらいのタプルをDBに書き込むことができるかを，サービスを提供する前に実際に計測を行うことで書込レートの推定を行う．

一般に書き込む1タプルあたりのデータ量が増加すると，書込レートが低下していくと考えられる．本研究では，タプルのデータ量に応じた書込レートを推定する．書込レートは以下の式で推定を行う．

$$\lambda_S \simeq rate\_estimate(tuple\_size(SS_q)) \tag{5}$$

SS_q は問合せ q を処理した結果出力されるタプルのスキーマである．*tuple_size*() は，SS_q から1タプル当たりのデータ量を計算する関数である．また，*rate_estimate*() は，1タプル当たりのデータ量から書込レートを推定する関数である．以下で書込レートの推定方法について説明する．

推定するためのサンプルとして，データ量の異なるn種類のタプルで書込レートを計測する．そのサンプルの各点から直線近似することで他のデータ量の場合の書込レートの推定を行う．この計測については，実際のDBMSを利用して予備実験を行った．実験結果については5.2で述べる．

実際のシステムの運用を考えると，DBに書込めるのに書込めないと判断される場合よりも，DBに書込めないのに書込めると判断される場合の方が深刻である．判定処理の精度をより高めるためには，より多くの点で計測を行うことが必要となる．また，計測を行っていない点の書込レートを推定する場合には，対象としているデータ量よりも多いデータ量の書込レートを使用することで，より問合せ処理の安全性を高められる．

4 問合せ処理の判定例

図2に示す問合せの判定例について説明する．図5に，図2の問合せをシステムで処理するための処理木を示す．ここでは，ガスタービンには45個のセンサが付属しており，DBには5個のスイッチの状態が記録してあるとする．また，Turbineストリームの到着レートは $\lambda_{Turbine} = 20$ であるとする．Switchesリレーションについては，データが到着しないので

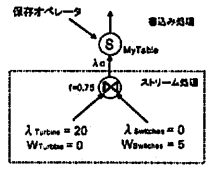

図 5: 処理木の例

$\lambda_{Switches} = 0$ であり，5 個のスイッチの状態が記録されているので $W_{switches} = 5$ となる．さらに，Turbine ストリームと Switches リレーションの結合演算の選択率は 0.75 であるとする．

まず，ストリームの処理可能の判定を行う．DB 内のデータは，あらかじめ DB からメモリ上に読みだされていて，読み出しコストが無視できるものとする．ストリーム処理可能の判定時には，結合演算における合計の処理時間 τ_o と出力レート λ_q を計算する．1 タプル当たりの結合処理にかかる処理時間が 0.5ms($C_{\bowtie} = 0.0005$) の場合について考えると，単位時間当たりにかかる処理時間は (4) 式から $\tau_o = 0.0005 * (20+0) = 0.01$．よって $\tau_o < 1$ であるので (3) 式によりこの問合せによるストリーム処理は処理可能である．一方，1 タプル当たり 60ms($C_{\bowtie} = 0.06$) の処理時間がかかってしまうときには，単位時間当たりに $\tau_o = 0.06 * (20+0) = 1.2$ の処理時間がかかる．この場合，$\tau_o > 1$ となってしまい，このまま処理し続けるといずれは破たんしてしまう．1 タプル当たりの結合処理にかかる時間はストリーム処理エンジンから取得する．また，問合せの出力レート λ_q を計算する．ここでは，(3) 式から λ_q は $\lambda_q = 0.75(20*5+0*0) = 75$ が求められる．

次に，書込処理可能の判定を行う．まず，問合せ q から出力タプルのスキーマ情報 SS_q を取得する．図 2 の問合せでは，$MyTable(s1,...,s45,sw1,...,sw5)$ というスキーマ情報が取得できる．次に，スキーマ情報から書込レート λ_S を推定する．*tuple_size* 関数によって与えられたスキーマ情報 $MyTable(...)$ から 1 タプル当たりのデータ量を計算する．すべての属性のデータ型が 1Byte の数値型であるとすると 1 タプル当たりのデータ量は $tuple_size(MyTable(...)) = 1*50 = 50$ のように計算される．*rate_estimate* 関数によって書込レートを推定し，λ_q と比較することにより処理可能かどうかを判定する．もし，推定書込レート $rate\_estimate(50) = 150$ だったときは，(2) 式により書込処理可能と判定する．

表 2: 実験環境

OS	Windows XP Professional SP2
CPU	Pentium(R) D 3.00GHz
メモリ	2.00GB
DBMS1	MySQL 5.0
DBMS2	PostgreSQL 8.1
Java	J2SE 5.0

5 予備実験

本節では，本研究で行った予備実験について述べる．本予備実験の主な目的は，本稿で提案した書込レートの推定手法の確認である．本研究で行った実験はいずれも表 2 に示す実験環境で行った．

5.1 実験 1

提案手法で用いている書込レートは，単位時間当たりに平均的にどれだけのデータ量を書き込めるかという指標である．しかし，一般には DBMS にはキャッシュの機能があるため，テーブルに挿入したデータをすぐにディスクへ書き込むことはない．キャッシュが埋まるまでの間は高速にデータを挿入することができ，キャッシュが埋まった段階でディスクへのアクセスが発生する．そのため，瞬間的に DB へ書き込めるデータ量には，ばらつきがあると考えられる．書込時間が極端にばらつくようでは，書込可能判定に平均指標である書込レートを用いることが意味を持たない可能性がある．そこで実験 1 では，実際の DBMS を用いてタプルの挿入処理にかかる時間とそのばらつきを調査した．実験には，DBMS として MySQL[7] と PostgreSQL[8] を利用した．MySQL では内部エンジンとして InnoDB を用いた．プログラム言語には Java を用い，それぞれ JDBC ドライバを通して 5 分間連続して書き続け，1 タプル書込む毎にかかる時間を計測した．DBMS には特別なチューニングは一切行わず，1 タプル書込む毎にコミットするようにした．書き込みには，INT 型属性を 1 つだけもつテーブルを使用し，挿入するタプルは毎回乱数を用いて生成した．

図6に実験結果を示す．上側のグラフがMySQLによる結果で，下側のグラフがPostgreSQLによる結果である．それぞれの横軸は実行時間を表し，縦軸は書込時間を表している．PostgreSQLの場合は，1つのグラフで記述することができなかったので，1部のデータだけを抜粋している．PostgreSQLでは，途中に1点だけ他と比べると高い点があるものの，ほぼ平均的に同じ時間で書込処理が完了していることが分かる．またMySQLにおいても，書き込みに時間がかかる部分がほぼ周期的に表れているが，平均的にみると同じ時間で書き込みできているとみなすことができる．これらよりMySQLとPostgreSQLにおいては，平均的な書込レートを用いて，書込可能判定を行うことが妥当なアプローチであることが確認された．

5.2 実験2

実験2では，1タプルあたりのデータ量と，書込レートの関係が実際のDBMSでどのような関係になっているかを調査した．テーブルの属性数を1から1000まで変化させ，それぞれの場合について，実験1と同様のプログラムで5分間連続で書き続けたときの毎秒の書込レートを求めた．ここでは，属性数1〜100の間はすべての属性数で測定し，100〜1000の間では50属性刻みで測定した．

実験結果を図7に示す．上の2つのグラフがMySQLの実験結果で，下の2つのグラフがPostgreSQLの実験結果である．また，左側のグラフが1〜100属性まで1属性ずつ増加させたグラフであり，右側のグラフが50〜1000属性まで50属性ずつ増加させたグラフである．実際の書込レートは，データ量に対して書込可能なタプル数を計測するが，ここでは属性数を軸に取って計測を行っている．MySQLの場合は100属性付近まではほぼ一定のように見えるが，1,000属性までスケールを大きくして観察すると単調減少傾向が伺える．PostgreSQLはさらに単調減少傾向が強く，100属性までのグラフを見ても，100属性時の書込レートは1属性時の書込レートの半分ほどになっている．本実験により，少なくともMySQLとPostgreSQLに関しては，数点のサンプルを取るだけで1タプルあたりのデータ量と書込レートの関係を把握することが可能であることが確認できた．

6 関連研究

本研究に関連する研究について述べる．まず，ストリーム処理における到着レートを考慮したコストモデルを提案している研究として[3]がある．[3]ではさらに提案したコストモデルに基づいて，ストリーム処理ができないと判断したときには，処理データに対してサンプリングを行う方法を提案している．本研究で提案する判定手法は，[3]のコストモデルをベースとしてDBに対する書込処理判定に利用できるように拡張を行っている．今後の展開として，書込めないと判断したときに書込処理を実現する方法の1つとして，ストリーム処理に対して[3]で提案しているようなサンプリング手法を適用させて，データベースに書き込むべき量を減らすことも1つの解決策として考えられる．

本研究で開発を行っているHarmonicaに関連する研究としてAurora[4]がある．Auroraはストリームの新規到着データに対して問合せを実現するストリーム処理エンジンである．Auroraではストリーム処理が可能かどうかを判定し，困難であると判断したときには到着データに対して自動的にフィルタリングを行い，処理するデータ量を減らす機能を提供している．本研究で目指しているHarmonicaシステムは，ストリームの新規到着データに対する処理だけでなく，ストリームの履歴データを逐次DBに保存する機能を提供する．また，本研究ではストリームが処理可能の判定だけでなく，ストリームの処理結果が逐次DBに蓄積できるかも判定する．判定の結果処理できないと判断したときの処理については今後の課題である．

7 おわりに

本研究では，ストリーム処理エンジンとDBを統合してストリーム管理システムHarmonicaの開発を行っている．本稿では，Harmonica上で利用者の問合せが処理可能かどうかを判定するための機能について述べた．問合せ処理可能の判定には，ストリームの到着レートを考慮したコストモデルから出力レートを計算し，DBに書込可能かどうかを判定する．本稿ではまた，実際のDBMSで書込み判定を行えるかを確認するために予備実験を行ない，本稿で提案する判定手法で判定できるDBMSが存在することを示した．今後の課題としては，まず，到着レートの変

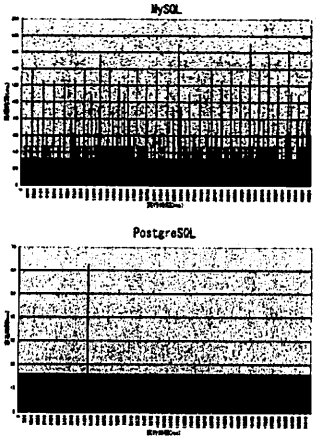

図 6: DBMS の書込特性

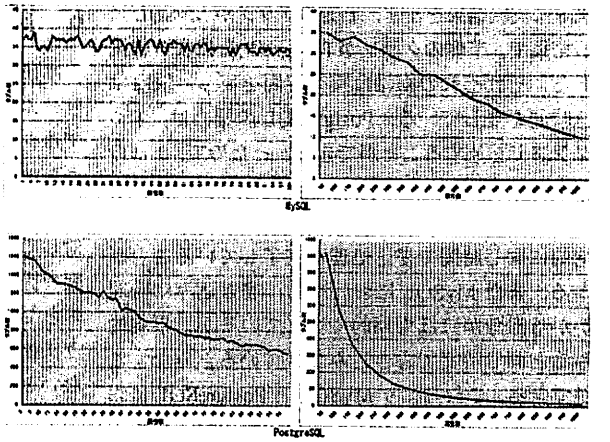

図 7: 書込レートの測定

化するストリームに対してコストモデルを拡張することが挙げられる．現在の評価手法では，ストリームは一定の到着頻度で配信されることを想定しているが，到着頻度が一定でないストリームデータも数多く存在する．より判定の精度を上昇させるためには，このような到着頻度の変化するストリームに対して考慮しなければならない．また他の課題としては，利用者に対して処理可能な問合せを提示する機能の実現が挙げられる．最低限保証してほしい精度を問合せに記述できるようにして，問合せ処理不可と判定したときに，その精度を元にサンプリングを行うなどの機能が必要と考えられる．

謝辞

本研究は，科学研究費補助金基盤研究（A）(#18200005)，科学技術振興機構 CREST「自律連合型基盤システムの構築」による．

参考文献

[1] 山田真一，渡辺陽介，北川博之．"ストリーム処理とデータベースを統合した実世界情報管理基盤"，データ工学ワークショップ（DEWS），2006 年 3 月．

[2] 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手法"，電子情報通信学会論文誌，Vol.J87-D-I,No.10, pp.873-886, 2004 年 10 月．

[3] Ahmed M. Ayad and Jeffrey F.Naughton. "Static Optimization of Conjunctive Queries with Sliding Windows Over Infinite Streams", Proc. of the ACM SIGMOD international conference on Management of data, pp.419-430, 2004.

[4] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul and S. Zdonik. "Aurora: A New Model and Architecture for Data Stream Management" VLDB Journal, (12)2:120-139, August 2003.

[5] S. Chandrasekaran et al. "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World", Proc. Conference on Innovative Data Systems Research 2003.

[6] R. Motwani et al. "Query Processing, Resource Management, and Approximation in a Data Stream Management System", Proc. Conference on Innovative Data Systems Research 2003.

[7] MySQL. http://www.mysql.com/

[8] PostgreSQL. http://www.postgresql.org/