2005－DBS－137（Ⅱ）（40）
2005／7／14

時系列データベースとストリーム処理の統合のためのデータモデルの提案

山田真一[†]，　渡辺陽介[†]，　北川博之[†,††]

概要

近年，ストリームという最新の情報を逐次オンラインで提供する情報源が増加し，ストリームデータ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として，過去に到着したストリームの履歴データを参照する問合せ，新規到着データに対して処理を繰り返し行う連続的問合せ，過去データと新規到着データを統合利用する問合せなどが挙げられる．本研究は既存の RDBMS と，時系列データの蓄積および分析に特化した時系列データベース，連続的な問合せ処理を提供するストリーム処理エンジンを組み合わせ，これらの問合せ要求を実現するためのシステムを構築することを目的とする．本稿が提案するデータモデルは，ストリームの特性を考慮し，ストリームデータを特殊なリレーションとして扱っており，従来のリレーショナル演算に加えて，大量のストリームデータの収集や活用に適した演算を提供している．本稿では，提案データモデルについて述べると共に，それらを実現するためのシステムアーキテクチャについて述べる．

A Data Model for Integrating Time Series Databases and Stream Processing

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

Abstract

Today, the amount of data delivered as data streams is increasing, and query processing of data streams has become important. There are various types of requirements, such as archiving and querying streams as historical data, processing continuous queries over incoming data, and integrating historical data and incoming data. Our goal is to develop a system to meet these requirements by combining a stream processing engine, RDBMSs and time-series databases. In general, these component-systems have quite different characteristics. In this paper, we propose a new data model for their integration. It models data streams as special relations which consist of current part and historical part. It provides operations to invoke many functions supported by time-series databases. We also explain an architecture of our prototype system.

1　はじめに

ネットワークの発達・普及に伴って，時々刻々と変化する事象に関する最新の情報を逐次オンラインで配信するストリーム型情報源が増加している．ストリームの例としては，天気予報，ニュース，株価情報，センサデータのような実世界情報などが挙げられる．ストリームは一般的に情報を次々と提供するため，短時間で大量のデータが提供される．また，過去の情報は時間の経過に伴って重要度が低下する可能性があるので，ストリームで提供される情報の処理には即時性が重要視されることが多い．このことから，ストリームデータに対する処理は高速であることが要求される．

ストリームに対する問合せ要求としては，「過去の履歴データを日付毎にまとめて取得したい」といった過去に到着したストリームの履歴データを参照する問合せ要求，「センサがある値以上のデータを配信

† 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
†† 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

してきたときに通知してほしい」といった新規到着データに対して繰り返し処理を行う連続的問合せ要求,「最近のデータの変化パターンと類似したパターンの履歴データがほしい」といった過去データと新規到着データを統合利用する問合せ要求が想定される．本研究では，これら 3 種類の要求を実現するシステムの構築を目指している．これらの要求を実現するためには，到着データを逐次処理する機能の他に，大量のデータを蓄積する機能やストリームデータ同士の類似度の計算などを行う機能が必要となる．

ストリームデータを扱うシステムとしてストリーム処理エンジン [1, 4, 5, 6, 7, 8] がある．ストリーム処理エンジンは，新規に到着したストリームデータに対する連続的な問合せ処理を行うことができるが，大量の履歴データを扱うには適していないため，蓄積処理や検索処理は RDBMS などの外部のシステムと連携しなければならない．しかし，RDBMS が提供する SQL 標準の機能だけでは，収集したストリームデータを活用するためには不十分で，例えば時系列データ同士の類似度の計算などを行うことは難しい．そこで，RDBMS の他に時系列データベース [9] を用いることが考えられる．時系列データベースは，時間と共に変化する数値データを蓄積し，それらを分析するための様々な機能を提供しており，また，それらの計算に特化しているため，高いパフォーマンスを発揮する．ただし，時系列データベースは数値以外のデータを扱うことには適しておらず，RDBMS の代替品として用いることはできず，ストリームからの新規データを逐次処理する機能も持たない．

本研究では，RDBMS，時系列データベース，ストリーム処理エンジンを統合活用することで，ストリームに対する各種処理要求を実現する．ストリーム処理エンジンとしては，我々の研究グループで開発を行っているストリーム統合システム StreamSpinner[1] を用いる．利用者にとって扱いやすいシステムを実現するためには，これら性質の異なるデータベースを統合的に扱うデータモデルが必要となる．本稿では，そのためのデータモデルを提案する．提案データモデルでは，ストリームを特殊なリレーションとして扱い，従来のリレーショナル演算を使用可能とする．また，時系列データを扱うためのデータ型とそれらに対する演算を新たに導入することで，ストリームデータの収集や類似度の計算などを実現している．

以下では，まず次節でストリームデータの利用例

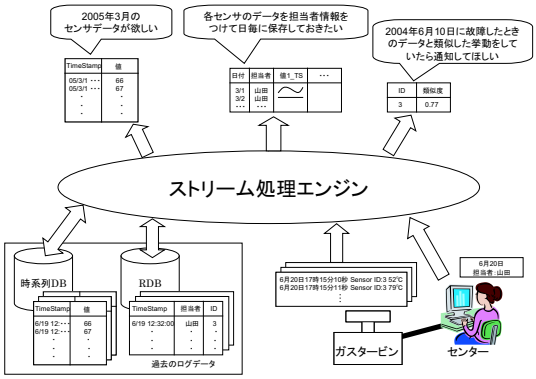


図 1: ガスタービンのモニタリング

について述べる．3 節で我々が提案するデータモデルについて述べた後，4 節でそのデータモデル上で行う演算について述べる．また，5 節でデータモデルを利用するためのシステムについて述べ，6 節で関連研究を紹介する．最後に結論として，まとめと今後の課題について述べる．

2　ストリームデータの利用例

本節では，逐次配信されるストリームに対する具体的な要求例として各種センサを利用して，ガスタービンのモニタリングをしている場合を考える (図 1)．ここでは温度センサーがガスタービンに取り付けられており，毎秒データが配信されるものとする．また，ガスタービンの状態を監視している監視担当者が存在し，その情報が送られてくるものとする．

このような状況ではまず，温度センサから得られる値をログとして保存するという管理要求が考えられる．例えば,「各センサのデータを担当者情報をつけて日毎に保存しておきたい」という要求である．この要求を実現するためには，センサから得られるデータとガスタービンを監視している担当者の情報を統合して，その情報を担当者毎，日付毎にまとめることが必要となる．

また，別の要求として故障予知が考えられる．例えば部品の劣化などでタービンが故障するような問題が考えられる．そこで，故障する前に部品の交換などを行いたいと考えている場合,「タービンが過去に故障した時のデータと類似した挙動をしていたら通知してほしい」という要求が考えられる．このような要求を実現するためには，予め故障したときのデータをまとめておき，最近のセンサデータと故障

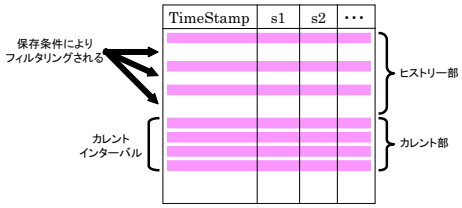


図 2: ストリーム

時のデータを使って，時系列データ同士の類似度を定期的に計算することが必要である．

本研究では，ストリームに対するこれらの処理を問合せによって実現できるデータモデルを提案する．

3　データモデル

本節では，我々の提案するデータモデルについて述べる．3.1 で提案データモデル上で扱うデータ型について述べ，3.2 で各情報源のモデル化について述べる．

3.1　データ型

本研究では，数値型，文字列型，TimeStamp 型，TimeSeries 型の４つのデータ型を使用する．数値型では整数や実数を扱い，TimeStamp 型では日付や時間などを扱う．TimeSeries 型は TimeStamp 型の値と TimeSeries 型以外のデータ型の値の対の配列で構成されるオブジェクトである．TimeStamp 型の値を t，TimeSeries 型以外のあるデータ型の値を v とすると TimeSeries 型 TS は以下の式で表現される．

$$TS = \{(t_1, v_1), (t_2, v_2), \cdots, (t_n, v_n)\}$$

ただし，$i < j$ のとき $t_i < t_j$ であり，$i \neq j$ のとき $t_i \neq t_j$ であるとする．また，すべての v_i は同じデータ型を持つとする．

3.2　各情報源のモデル化

提案データモデルでは，情報源は通常のリレーションまたはストリームで表される．本研究では，ストリームを TimeStamp 型の属性を持つ特殊なリレーションとしてモデル化する．ストリームでは情報源から配信される 1 配信単位をそのリレーションの 1 タプルとして扱い，そのタプルの到着時刻が TimeStamp 型の属性にセットされるものとする．ストリームは

表 1: 演算の一覧表

演算名	役割
選択	リレーショナルモデルの基本演算
射影	
直積	
結合	
T-CAST	タプルの集合からTimeSeries型を持つタプルに変換する
R-CAST	TimeSeries型を持つタプルからタプルの集合に変換する
追加	利用者が作成したテーブルに対して
削除	タプルの追加・削除を行う
MERGE	TimeSeries型を持つタプルをテーブルに追加するときに TimeSeries型の要素同士を統合する
EVAL	関数を利用するための演算

カレント部と**ヒストリー部**から構成される（図 2）．カレント部はストリームのうち**カレントインターバル**の期間に含まれるタプル集合のことを指している．カレントインターバルとは，配信された全てのストリームデータを保持する期間のことを言い，その幅は時間幅で定義される．例えば，「3ヶ月間はきちんとログを保存しておきたい」という要求があるとき，カレントインターバルは 3ヶ月となる．また，ヒストリー部はカレントインターバル以前の期間に含まれるタプル集合のことを指す．ただし，ヒストリー部は**保存条件**によってフィルタリングされている．保存条件とはカレントインターバルの範囲よりも古くなったデータを保持しておく際の基準となるものである．例えば，ストリームがセンサなど等間隔で現在のデータを通知する情報源のときには，前回到着したデータと現在のデータとでは変化量が少ない場合が多いと考えられるので，「直前のデータから値が変化したデータのみ保持する」や「配信されたデータのうち 80%だけサンプリングして保持する」などの保存条件が考えられる．

一般に到着した全てのデータを保持すると，ディスク容量が足りなくなってしまうので，過去に到着した不要なデータを保存条件を用いてフィルタリングすることは必要である．しかし，到着したばかりのデータなどは問合せにおける重要度が高いと考えられるので，カレントインターバルの期間は全てのタプルが保存される．保存条件やカレントインターバルは，サーバの性能やディスクの容量と密接に関係していると考えられるので，管理者が裁量で決めるものとする．

4　提案データモデルにおける演算

本節では，提案データモデル上で行う演算について述べる．本研究で扱う演算を表 1 に示す．

4.1 問合せの実行と演算の評価

本データモデルでは，通常の問合せと連続的問合せの 2 種類の問合せを想定している．通常の問合せとは，問合せを与えた時に一度だけ処理される問合せのことである．また，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問合せ処理を実行し，前回実行時の処理結果からの差分の結果を生成することを繰り返し行う問合せである．

連続的問合せでは，演算と共にマスタ情報源とウインドウを指定する．マスタ情報源とは，連続的問合せにおける問合せ実行のきっかけを与える情報源のことである．例えば，マスタ情報源がセンサデータを配信するストリームであるときには，センサデータがシステムに到着する度に問合せ処理が行われる．マスタ情報源には通常のストリームの他に，クロックストリームを指定することもできる．クロックストリームとは，定期的にアラームを送信するような仮想的な情報源のことである．クロックストリームをマスタ情報源に指定することで，定期的な時間に処理を行う連続的問合せが可能となる．ウインドウとは，演算の適用対象となるタプルを選択するための時間幅のことを言う．ストリームでは逐次タプルが到着するため，過去に到着した全てのデータを演算の処理対象とした場合，非常に大量のタプルを処理しなければならない．そこでウインドウを用いて演算の適用対象を限定することで処理の効率化を図る．

提案データモデルは，リレーションをベースとしているためリレーショナルモデルの基本演算である，選択・射影・直積・結合演算をそのまま用いる．本研究で導入した演算については後述する．ストリームに対して演算を適用する場合は，カレント部，ヒストリー部を明示的に指定して用いるものとする．

要求例 4.1 *2* 節のガスタービンのモニタリングの例で,「各センサのデータに担当者情報をつけて提供する」連続的問合せについて考える．図 *3* にこのときの処理木を示す．センサのデータが毎秒 *Turbine* ストリームで配信されており，また監視担当者の情報が *Monitor* ストリームによって配信されているとする．*Turbine* ストリームと *Monitor* ストリームのスキーマ定義は図 *3* の左側に示した通りである．*T* は *TimeStamp* 型であり，*Id* と *V* は数値型であるとする．また，*Name* は文字列型であるとする．*V* にはセンサから得られた値が入り，*Id* にはセンサの *ID* が入る．今は簡単化のため *Id* は全部 *1* であるとする．図 *3* で

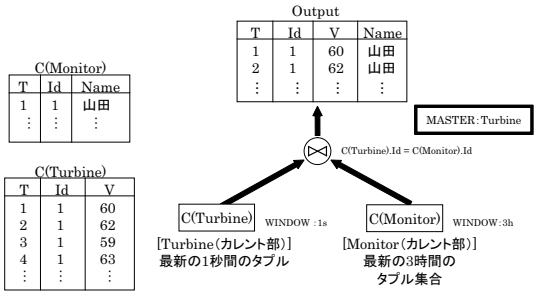


図 3: 要求例 4.1 における処理木

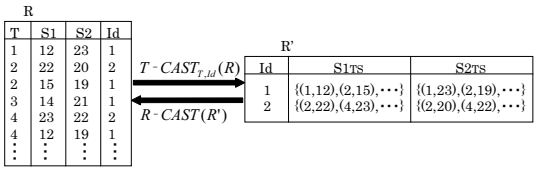


図 4: T-CAST，R-CAST 演算

はマスタ情報源は *Turbine* ストリームで，*Turbine* ストリームのウインドウは *1* 秒間である．また，監視担当者は頻繁に変わるとは考えられないので，*Monitor* のウインドウは *3* 時間とした．図 *3* の問合せの場合，*Turbine* ストリームが配信される毎に問合せ処理が実行されて，それぞれのストリームが *Turbine* ストリームから提供される *Id* と *Monitor* ストリームから提供される *Id* で結合されたテーブルが出力される．

4.2 T-CAST 演算，R-CAST 演算

提案データモデルで新たに導入した T-CAST 演算，R-CAST 演算について説明する．

T-CAST 演算はタプルの集合から TimeSeries 型を持つタプルに変換する演算である．ただし，TimeSeries 型の生成に必要な TimeStamp 型の属性を 1 つ指定する必要がある．また，TimeSeries 型を作成するためにグループ化するキーを指定する．これは，同じキーの値を持つタプル集合をまとめて TimeSeries 型を生成するためである．キーは複数指定可能である．図 4 の例では Id をキーとして，S1 と S2 をそれぞれ TimeSeries 型に変換している．

R-CAST 演算は TimeSeries 型を持つタプルから元のタプルの集合に逆変換する演算である．TimeSeries 型の要素に対して R-CAST 演算を適用すると，TimeStamp 型と数値型から構成されるスキーマを持つリレーションに変換される．T-CAST，R-CAST 演算を導入することで，ストリームをそのままのリレーション

表 2: 関数の分類

入力	出力	関数の例
TimeSeries × TimeSeries	TimeSeries	sim（類似度の計算） correlation （相関の計算） など
TimeSeries	TimeSeries	percentMove（値の変化率） nearestInteger（切り上げ・切り下げ） など
TimeSeries	Real	average（平均の計算） standardDeviation（標準偏差の計算）など

として操作したいという要求と，ある程度まとまった時系列データとして扱いたいという要求を実現可能である．また，グループ化を行うので，ログデータとして保存する場合に意味のあるまとまりとして管理することが容易となる．

4.3 EVAL 演算

時系列データベースには，時系列データ集合に対する様々な計算を行う関数が用意されている．提案データモデル上では，それらの関数を利用するための演算である EVAL 演算を用いる．

EVAL 演算は与えられたリレーションから，属性を 1 つ追加したリレーションを作り出す演算である．追加された属性には，各タプルを関数で評価した結果が入り，結果のデータ型は利用する関数によって決まる．関数の引数は TimeSeries 型の属性を用いる．本研究では関数としては表 2 のようなものを想定している．

要求例 4.2 *2* 節のガスタービンの例で「タービンが故障した時のデータと類似した挙動をしていたら通知する」場合の処理木を図 *5* に示す．図 *5* ではマスタ情報源は *Turbine* ストリームで，ウインドウは *30* 分である．処理木の左側では，*Turbine* ストリームのカレント部 *C*(*Turbine*) のタプル集合から，*T-CAST* 演算により *TimeSeries* 型を持つリレーションに変換している．ここでは，*TimeStamp* 型の属性として *T* を，グループ化のキーとして *Id* を *T-CAST* に与えている．ヒストリー部の *H*(*Turbine*) ストリームから，選択演算によって故障したときのタプル集合を取り出し，そのタプル集合にも同様の処理を行い *TimeSeries* 型を持つリレーションに変換する．次に，変換した 2 つの *TimeSeries* 型を持つリレーションを結合し，*EVAL* 演算によって *sim()* 関数を実行し結果を得る．出力結果は図 *5* の上の *Output* のようになる．

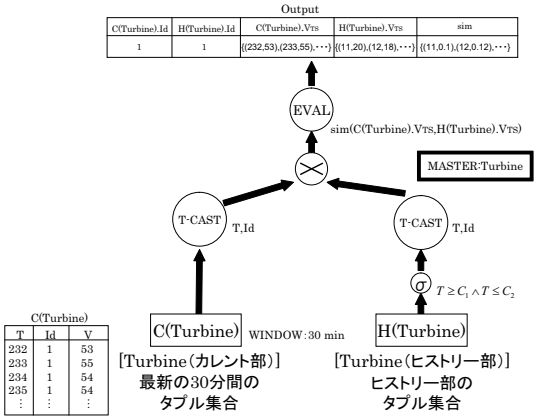


図 5: 要求例 4.2 における処理木

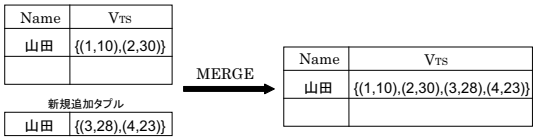


図 6: MERGE 演算

4.4 ストリーム上のビューとテーブル

本研究では，問合せによってストリームに対するビューを作成することも想定する．ビューを定義することで，利用者は独自のスキーマでデータを参照することが可能である．ストリーム上のビューは実体であるストリームの影響を受け，データの新規到着やカレントインターバルの移動，保存条件によりビューの中身は絶えず変化する．

また，利用者からの問合せによって，利用者専用のテーブルを作成・操作することも許可する．これは，問合せによって利用者自身が加工したデータを必要に応じて保存するためである．利用者が作成したテーブルに対しては，タプルの追加と削除を行う演算の他，MERGE 演算を導入する．MERGE 演算は TimeSeries 型を持つリレーションにタプルを追加するための演算で，通常の追加演算との違いは，TimeSeries 型属性以外の属性の要素が全て同じであるタプルが存在するならば，追加するタプルの TimeSeries 型の要素を，既にあるタプルの TimeSeries 型の要素に統合する点である．MERGE 演算の例を図 6 に示す．図 6 では，Name でグループ化して作成された TimeSeries 型を持つテーブルがあり，そのテーブルに同じ名前「山田」を持ったタプルを MERGE 演算を使って追加する例である．

連続的問合せによって T-CAST 演算を行うと，連

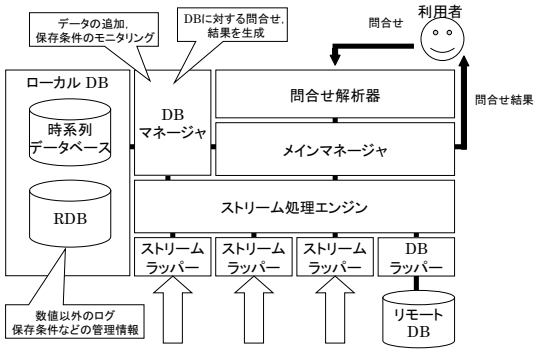


図 7: Harmonica システム

続的問合せは前回実行時からの差分の生成を繰り返すものなので，複数のタプルで同じキーでグループ化された TimeSeries 型をもつタプルが現れる可能性がある．そのような状況で，タプルを追加するときに MERGE 演算を行うことで，1 つのタプルにまとめることが可能である．

5 プロトタイプシステム Harmonica

本節では，提案データモデルを実現するためのシステムについて述べる．

5.1 アーキテクチャ

我々は提案データモデルを利用したシステムとして，図 7 に示すようなアーキテクチャを持った Harmonica というシステムの構築を進めている．Harmonica システムは，問合せ解析器，メインマネージャ，ストリーム処理エンジン，DB マネージャ，ローカル DB から成っている．また，ローカル DB には時系列データベースと RDB が存在する．

問合せ解析器は，利用者からの問合せを解析してメインマネージャに渡す．メインマネージャは，利用者からの問合せが連続的な処理でない場合，DB マネージャに結果を問い合わせる．連続的に問合せを処理しなければならない場合には，ストリーム処理エンジンに処理を依頼する．また，メインマネージャは利用者からの問合せによる結果を利用者に返す役割がある．ストリーム処理エンジンは，連続的な問合せの処理を行う．またストリーム処理エンジンは，ストリームとストリーム，ストリームとリモート DB など各種情報源の統合を行う．ストリーム処理エンジンと各種情報源の間にはラッパーが存在し，情報源で扱っているデータ形式を Harmonica 上で扱うデータ形式に変換する機能を持っている．また，ストリーム処理エンジンはストリームのヒストリー部を扱う必要があるときには DB マネージャと通信し，最新のデータと過去のデータの統合や，T-CAST 演算や R-CAST 演算，カレントインターバルの管理などを行う．

DB マネージャはローカル DB と通信し，ローカル DB と他のコンポーネントとのデータの受け渡しを行う．ローカル DB はストリームのヒストリー部に該当するデータを保存するために利用する．一般的にストリームデータは，数値型の要素や文字列の要素が組み合わさったデータで構成されているので，数値の扱いが得意な時系列データベースには数値型の属性値を保存し，それ以外の属性値は RDB に保存する．DB マネージャは，このようなデータ型の違いによる保存先の違いを他のモジュールに意識させることなく DB からデータを取得する．DB マネージャはまた，ヒストリー部のデータを保存条件に従って管理を行ったり，カレントインターバルを超えたデータをローカル DB に保存する作業を行う．

5.2 データモデルの実現方法

実装する際のストリームデータの扱いについて述べる (図 8).

まず，カレント部について述べる．カレントインターバルの範囲に含まれるデータはストリーム処理エンジンによってメモリ上に管理されている．カレントインターバルを超えたデータはストリーム処理エンジンから DB マネージャに渡される．DB マネージャはそのストリームに関連付けされている保存条件を元に，渡されたデータに対してフィルタリングを行い，ヒストリー部のデータとして数値型とそれ以外のデータに分割してローカル DB に保存する．分割したデータから元のデータを復元するための構成情報などは RDB に保存される．

また，ヒストリー部は Harmonica システム上では図 8 の左側に示すような分割された状態のリレーションのままでも使用できるように利用者に提供される．これは，実際にデータが格納されている状態を利用者が意識して問合せをした方が現実的には効率が良い場合が多いためである．ビューを用いることで，図

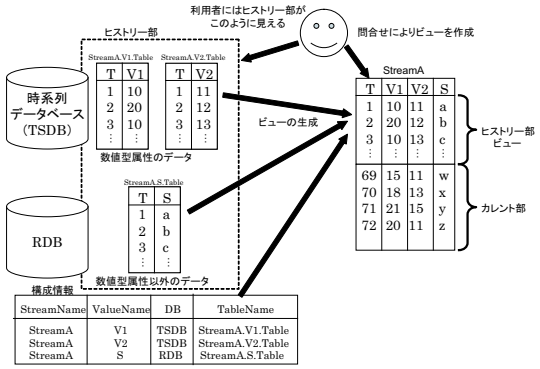


図 8: ストリームの実装

8 右側のようなリレーションとして操作することも可能である．

次に，TimeSeries 型の実装について述べる．TimeSeries 型の実体はローカル DB 内に置かれる．これは EVAL 演算の実行において，時系列データベースに関数の評価を行わせるためである．カレント部に含まれるタプルを TimeSeries 型に変換したときには，ストリーム処理エンジン上のデータを時系列データベースに一時的にコピーする．ストリーム処理エンジン上で TimeSeries 型はローカル DB 上への参照として扱われる．カレント部に含まれるタプルから作成された TimeSeries 型の実体は，必要が無くなった時に DB マネージャによって削除される．

5.3 問合せ実行の流れ

Harmonica システム上で問合せがどのように実行されるかを図 5 の問合せ例を用いて説明する．予め，図 5 の問合せがシステムに登録されているものとする．Turbine ストリームが到着する度に以下が行われる．

1. ストリーム処理エンジンがメモリ上の Turbine ストリームのカレント部からウインドウ分（30 分間）のデータを取り出し，T-CAST 演算によって TimeSeries 型に変換する．このとき，ストリーム処理エンジンは TimeSeries 型のデータの実体を時系列データベースに保存し，そのデータへの参照を得る．

2. ストリーム処理エンジンが Turbine ストリームのヒストリー部から故障した時間に該当するデータをローカル DB から DB マネージャを通して受け取り，受け取ったデータを T-CAST 演算によって TimeSeries 型に変換する．(1) と同様に，ストリーム処理エンジンは，変換したデータを時系列データベースに保存し，そのデータへの参照を得る．

3. TimeSeries 型を要素に持つ 2 つのリレーションに対して，ストリーム処理エンジンが直積演算を行いテーブルを生成する．

4. ストリーム処理エンジンは TimeSeries 型のデータへの参照を用いて，時系列データベースに類似度の計算を行わせる．ストリーム処理エンジンは，EVAL 演算により関数の計算結果を先ほど作ったテーブルと結合する．

6 関連研究

$\mathcal{SEQ}$[3] ではデータシーケンスを扱うためのデータモデルの提案をしている．データを順序付きのデータに対応させることでシーケンスデータを定義している．シーケンスデータに対しては，シフトさせながら処理を行うオフセット演算や，日付から週に変換するシーケンスズーム演算など時間を使った演算が豊富に考えられている．また，データシーケンスのサブシーケンスであるシーケンスグループについても考えられている．ただし，マスタ情報源やウインドウなどの連続的問合せ処理に必要な機能は考えられてはいない．また，ストリームで逐次配信される最新のデータと過去のデータとの統合処理についても考慮されていない．

KRAFT[2] では，データの鮮度と永続性の両立，類似検索，周期的監視という 3 つの要求に答えるセンサデータベースの設計を行っている．類似検索においては，センサ型というデータ型を導入してユークリッド距離やダイナミックタイムワーピング距離を計算する．センサ型はセンサデータが DB に到着した時間と値の組から構成されている．また，組み込み方式を採用しており，全ての機能を備えた単一のシステムとして実装されている．本研究では，ストリームデータに対するリレーションとしての操作要求と TimeSeries 型としての操作要求に柔軟に対応するために，独自の T-CAST・R-CAST 演算を導入した．また，本研究で想定しているシステムは分散ラッパー型である．分散ラッパー型だと既存の DB を使ってシステムを構築することができる．また，我々のアーキテクチャでは DB を分散配置することが容易である．

7 おわりに

本稿では，ストリームで逐次配信される最新のデータと過去のデータとの統合処理を実現するデータモデルを提案した．提案データモデル上で問合せを行うことで，利用者の要求に合ったログデータの管理や，リアルタイムの異常値検出などを支援する．また，提案データモデルを実現するプロトタイプシステム Harmonica のアーキテクチャについての考察を行った．今後の課題としては，システムの実装と，提案データモデルの有用性の実証が上げられる．

謝辞

本研究の一部は，日本学術振興会特別研究員奨励費 (15・330)，科学研究費補助金特定領域研究 (2)(#16016205)，基盤研究 (B)(#15300027) による．

参考文献

[1] 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．

[2] 川島英之，今井倫太，遠山元動，安西裕一郎．”センサデータベースシステム KRAFT の設計と実装 ”，情報処理学会論文誌，Vol.45，No.SIG14(TOD24)，pp.39-53，2004 年 12 月．

[3] S. Praveen, L. Miron, and R. Raghu. ”$\mathcal{SEQ}$:A Model for Sequence Databases ”, Proc. International Conference on Data Engineering, pp.232-239, March 1995.

[4] L. Liu, C. Pu, and W. Tang. ”Continual Queries for Internet Scale Event-Driven Information Delivery”, IEEE Trans. Knowledge and Data Engineering, vol.11, no.4, pp.610-628, 1999.

[5] J. Chen, D. J. DeWitt, and J. F. Naughton. ”Design and Evaluation of Alternative Selection Placement Strategies in Optimizing Continuous Queries”, Proc. International Conference on Data Engineering, pp.345-356, 2002.

[6] D. J. 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 Vol.12, No.2, pp.120-139, 2003.

[7] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. ”TelegraphCQ: Continuous Dataflow Processing for an Uncertain World”, Proc. Conference on Innovative Data Systems Research 2003.

[8] R. Motwani, J. Widom, A. Arasu, B. Babcock, S.Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. ”Query Processing, Resource Management, and Approximation in a Data Stream Management System”, Proc. Conference on Innovative Data Systems Research 2003.

[9] http://www.lim.com