技術レポート

データストリーム管理システム

—DSMS（Data Stream Management System）—

株式会社カティエント　技術研究部

森下 民平

1．はじめに

本稿では、近年研究開発が進み、商用製品も出揃いつつあるデータストリーム管理システム（Data Stream Management System、以下「DSMS」と記す）*1の概要を紹介する。

近年、当社の技術研究部は、ユビキタス時代の到来に向けた技術の研究開発に焦点を当ててきた。ここで言うユビキタスとは、Xerox PARCの故Mark Weiserが提唱したユビキタス・コンピューティング（Ubiquitous Computing）と呼ばれるビジョンのことを指す。いわく、コンピュータは、環境の中に多数埋め込まれ、環境の中に溶け込み、人間からは意識されることも見えることもないままに、人間の日常生活を支援するようにならなければならない[27]。あらゆるモノや環境の中にセンサーやコンピュータ、アクチュエータが埋め込まれ、人間の活動をさりげなく支援する、というユビキタス時代の環境では、膨大な数のノードが、常にリアルタイムに処理されるべきデータを流し続けることになるだろう。我々は、ユビキタス時代におけるこのような環境、つまり、膨大なノードが集まり、リアルタイム処理要求の高いデータが常に大量に流れ続ける環境において、DSMSはきわめて有望なデータ処理技術であると考え、研究を行ってきた。

筆者は、DSMS研究をリードしてきたStanford大学InfoLabに、2003年から2005年の2年間にわたり客員研究員として在籍し、プロジェクトを率いるJennifer Widom教授から、DSMSの理論ならびに設計について、直接指導を仰ぐ機会を得た。現在、筆者らは、技術研究部門内のDSMS専任グループにて、DSMSの研究開発に従事している。本稿では、こうしたDSMSに関する経験をもとに、DSMSとは何か、必要とされるようになった背景とアプリケーション、その技術的な課題は何か、といった点を明らかにしたい。

2．DSMS登場の背景

データベース管理システム（DataBase Management System、以下「DBMS」と記す）は、企業情報システムにとって、最も重要で必要不可欠なコンポーネントとして定着している。また、企業情報システムにおけるさまざまな要請に応じて機能拡張が行われてきた。

しかし近年、DBMSの処理モデル、すなわちデータベースに蓄積したデータに対して都度クエリーを発行し検索結果を得る、という処理モデルでは対処することの困難な、新しい種類のアプリケーションが必要とされるようになってきた。新しい種類のアプリケーションとは、下記のように、リアルタイムに流れ続ける一過性のデータを、データベースに格納・蓄積してからではなく、即座に処理するタイプのアプリケーションである[7, 15, 23]。

- RFID（Radio Frequency Identification）タグからのデータにより、リアルタイムに人・物の位置把握やプロセスモニタリングを行うRFID アプリケーション
- 屋内外に張り巡らせたセンサーから、各種環境データを読み取り反応するセンサーネットワーク・アプリケーション
- 株価や外国為替レートなどの変動や傾向をリアルタイムにモニタリングし、ときには即座に売買を行う金融アプリケーション

＊1）文献によっては、ストリーム処理エンジン（Stream Processing Engine、SPE）と呼ぶ場合もある。

●ネットワーク上のパケット解析により異常検知を行うセキュリティ・アプリケーション
●HTTPパケットまたはWebサーバーログをモニターすることによりWebアプリケーションユーザーのリアルタイム行動分析ならびに統計処理を行うクリックストリーム分析アプリケーション
●オンラインオークション

これらのアプリケーションで扱うデータの流れ、すなわちデータストリームには、次のような特徴がある。

●データは継続的に到着し、常に流れ続けている。またその量は理論的に無限である。
●データの到着頻度は、ときに極めて高い。
●データの到着頻度や分布は時間と共に変化し、ときに予測ができない。
●データの流れが複数存在し、相互の関係を考慮した処理が必要な場合がある。

DSMSは、データストリームを効率的に管理するための仕組みとして、データ工学分野では、特に2000年以降、活発に研究開発が行われてきた。現在は、米国を中心に複数の商用製品が登場し、実用化が進行しつつある段階である。

3．DSMSのアーキテクチャ

まずDSMSとその周辺システムとを概観した図1を示す。

入力データストリームは、各種のアプリケーションに応じたデータの発生元から、HTTPパケット、ネットワークパケット、TCP/IPソケットなど、さまざまな手段によりDSMSに到着する。入力データストリームは、入力アダプタによりDSMSエンジン向けの表現形式に変換される。DSMSエンジン内部で処理されたデータは、出力アダプタにより監視を行うダッシュボードアプリケーションや他システムに渡され、データベースやファイルなどに保管する必要がある処理結果についても出力アダプタにより記録される。つまり、通常DSMSエンジン外部とのデータのやり取りは、各々の入出力形式に対応した各種のアダプタ*2によって行われる。

DSMSはデータストリームの即時処理に特化しているため、自身ではデータ保管のためのデータベース機能を持たない場合がある。この場合、マスタデータや保管されたトランザクションデータとの突合わせを行うためには、外部データベースとの結合処理（ジョイン）を行う必要がある。頻繁に到着するストリーム中のデータに対し、都度データベースに参照問い合わせを発行するのは極めて非効率的であるため、データベース中のデータは、事前にキャッシュとしてDSMSエンジンに取り込まれる。

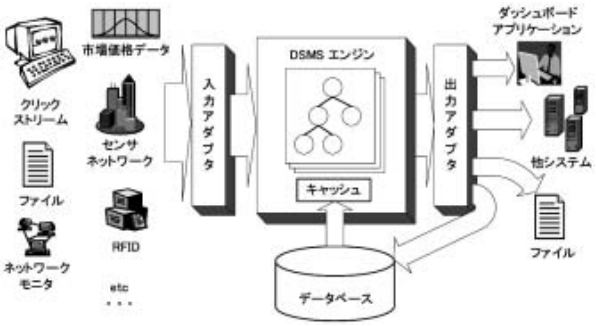

図1　DSMSの概観

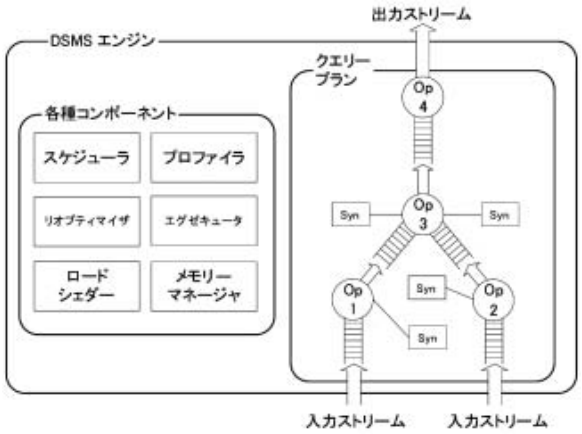

図2　DSMSエンジンのアーキテクチャ

次に、DSMSエンジン内部のアーキテクチャを図2に示す。

DSMSによるデータストリーム処理は、4.1節で述べるように、継続的クエリー（Continuous Query）による処理が中心となる。継続的クエリーは、多くの場合、図2に示したように、エンジン内部では、演算の実行順序とフローを表すクエリープランとして表現され実行される。図2のクエリープランの例では、DSMSエンジンは2つの入力ストリームを受け取り、ウィンドウ演算子（図ではOp1とOp2）により、まずウィンドウ処理が行われる（ウィンドウについては、4.2節で述べる）。ウィンドウ処理が行われたおのおのの入力データストリームは、ジョイン演算子（図中Op3）で結合され、選択演算を行う選択演算子（Op4）により結果が出力ストリームとして生成される。なお、図2の中で、○印で示された各演算子下方のハシゴ状の部分は、キューを表している。また、Op4を除く各演算子に連結されSynで示される四角形は、シナプシス（Synopsis＝概要）と呼ばれ、演算に必要な中間データを保持している。DSMSエンジン内部には、設計により相違はあるものの、おおむね図2に示した各種コンポーネントがある。スケジューラが次に実行すべき演算子を決定し、エグゼキュー

*2）アダプタは、製品や研究によってラッパーなど別の呼称で呼ばれることもある。

タが演算を実行する。プロファイラは流れるデータのサンプリングを行い、動的にクエリープランを最適化するための統計情報を収集し、リオプティマイザがクエリープランを常に最適化し続ける（クエリープランの動的な最適化については、4.3節で述べる）。ロードシェダーは、ストリーム流量などの負荷が高い場合、システム全体が停止あるいはクラッシュするのではなく、許容される誤差範囲内で近似解を算出できるよう入力データの一部を捨て、負荷調整を行う。メモリーマネジャーは、キューやシナプシスなどへのメモリー割り当てを行う。

4．データストリーム処理における課題とDSMSによる解法

では、データストリームを処理する際には、何が課題となり、DSMSではそれをどのように解決しているのだろうか。これまで発表された膨大な研究成果のすべてを紹介することはできないが、以下ではその一端を紹介する。

4.1　継続的クエリー（Continuous Query）

まず最初の課題は、データストリーム処理アプリケーションの典型的な要件、すなわち、あらかじめユーザーがデータの抽出条件や集約関数による加工条件を登録しておき、出力すべきデータが現れた場合、結果を取得する、という要件をいかに実現するか、という点である。たとえば、株式市場の株価データストリームにおいて、最近5分間のすべての株価の平均値を常に出力し続ける、という簡単な例を考える。ただこれだけの処理でも、DBMSを用いて実装するのは面倒で、率直に表現することができない。伝統的なDBMSの処理モデルは、図3に示すように、データはデータベース内に蓄積され、ユーザーは、必要な検索をその時点限りのアドホッククエリーとして発行し、その時点の結果のみが結果として返される、というものである。

これに対して、システムにあらかじめ登録され、継続的にシステム内部で動作するクエリーのことを継続的クエリー（Continuous Query）と呼ぶ。継続的クエリーの概念は、1992年にTerryら[25]によって発表されたが、現在主流となっているほとんどのDBMSは、継続的クエリーを直接サポートしていない。そのため、前述の株価平均算出のような簡単な例であっても記述は容易ではない。一方DSMSでは、継続的クエリーはもっとも重要な技術要素であり、効果的なネイティブサポートを行うことを目標に研究開発が進められてきた。DSMSにおける継続的クエリーの処理モデルイメージを図4に示す。

たとえば、前述の最近5分間のすべての株価の平均値を常に出力し続けるという例は、代表的なDSMSの学術研究プロトタイプの1つであるStanford大学のSTREAM[20, 2]では、継続的クエリー言語CQL[4]を用いて、リスト1のように記述できる。

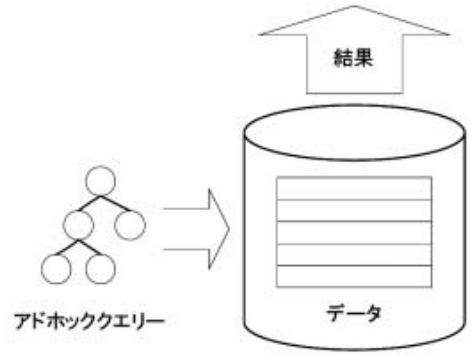

図3　DBMSの処理モデルイメージ（アドホッククエリー）

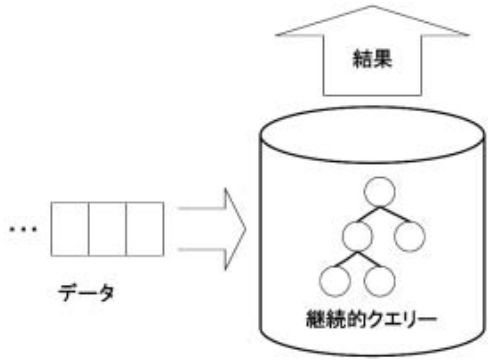

図4　DSMSの処理モデルイメージ（継続的クエリー）

リスト1　継続的クエリーの例

```
SELECT AVG (S. Price)
FROM StockStream [RANGE 5 MINUTES] AS S
```

またDSMSは、継続的クエリーの処理に各種の最適化を施している。そのため、伝統的なDBMSを用いたトリガーやストアドプロシージャによる代替的な継続的クエリーの実現と比較すると、8倍から26倍以上高速に動作するとの実証結果[22]が報告されている。

4.2　ウィンドウ

データストリーム処理における2点目の課題は、理論的に無限であるデータストリームをどのように扱えばよいか、という点である。Arasuら[3]は、継続的クエリーを分析し、どのようなクエリーが無限のメモリーを必要とするかを明らかにした。有限のメモリーで処理を行うもっとも単純な方法は、ウィンドウの利用による一種の近似計算を行う方法である[7]。ウィンドウとは、無限長のデータストリームを論理的に有限の長さに区切るものである。もっと

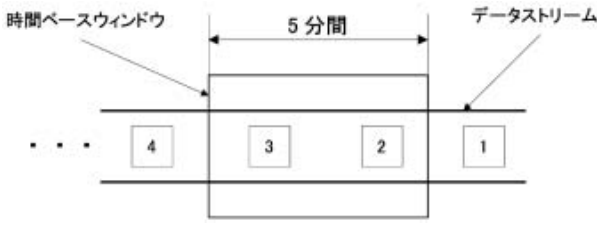

図5　時間ベースウィンドウ

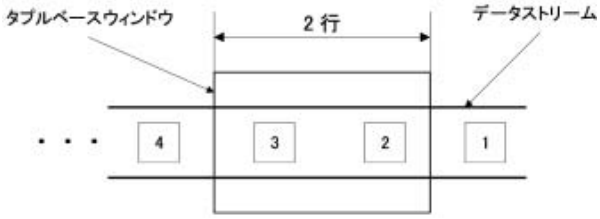

図6　タプルベースウィンドウ

リスト2　タプルベースウィンドウの記述例

```
SELECT AVG(S. Price)
FROM StockStream [RANGE 2 ROWS] AS S
```

も基本的なウィンドウは、指定時間内に含まれるデータを処理対象とする時間ベースウィンドウ（図5）と、指定タプル数（行数）を保持するタプルベースウィンドウ（図6）である。

タプルベースウィンドウの記述例は、リスト2のとおりである。

これまでに紹介した時間ベースウィンドウとタプルベースウィンドウは、ウィンドウの更新のきっかけによる分類だったが、さらに端点の動き方による分類と、更新間隔による分類とがある[15]。

端点の動き方による分類としては、スライディングウィンドウ（Sliding Window）とランドマークウィンドウ（Landmark Window）がある。たとえば直近の5分間の入力データを対象とするのは、時間ベースのスライディングウィンドウだが、午前9時以降のすべてのデータを対象とするウィンドウは、午前9時時点でウィンドウの片方の端点が固定されたランドマークウィンドウである。

更新間隔による分類としては、ジャンピングウィンドウ（Jumping Window）と宙返りウィンドウ（Tumbling Window）がある。

更新間隔が、1行ごとあるいは1単位時間ごとではなく、複数行あるいは複数単位時間ごとであるものをジャンピングウィンドウと呼ぶ。宙返りウィンドウはジャンピングウィンドウの特殊なケースで、更新前にウィンドウ内に含まれていたデータが更新後にはすべて後続の新規データに置き換わるウィンドウのことを言う。

4.3　アダプティブクエリー

データストリーム処理における3点目の課題は、データの分布が前もってわからないという条件下で、どのようにクエリーを最適化するかというものである。例として、株式市場の歩み値データをモニタリングし、A社の株価が1万円以上で取引された場合、その回数と出来高を一定時間単位で算出するという継続的クエリーを考える。この場合、株式市場の歩み値データから、まずA社株の取引データを抽出し、さらに抽出されたデータが1万円以上であれば集計の対象とする、というクエリープランと、先に1万円以上で売買された株のデータを抽出し、抽出されたデータからさらにA社のデータを抽出し集計対象とする、というクエリープランとが考えられる。基本的には、選択率（selectivity）の小さい条件、つまり、抽出される結果が入力に対して少ない条件を先に適用したほうが低いコストでクエリーを処理することができる。市場全体の取引のうち、A社株の売買が少ない場合には、先にA社株のデータを抽出するクエリープランのほうが低いコストで算出できるだろう。しかし、A社関連のニュースにより株式市場がA社株売買の注文に占められ、かつほとんどが1万円未満の株価での売買だった場合には、先に株価が1万円以上のデータを抽出した方がクエリープランのコストが低くなることもあるだろう。このように、時間と共に変化するデータの分布に応じ、クエリープランを動的に最適化することのできるクエリーをアダプティブクエリー（Adaptive Query）と呼ぶ。たとえば、Stanford STREAMにおけるアダプティブクエリーでは、プロファイラがフィルタ条件に合致しなかった最近分のデータを一定の割合でプロファイリングし、複数フィルタに対して合致したか否かの真偽値を持つことにより選択率を見積もり、この見積もりに応じてフィルタ条件の適用順序を変更するA-Greedyと呼ぶアルゴリズムを提案している[9]。

DSMSにおけるアダプティブクエリーは、Stanford STREAMのようにクエリーオプティマイザが最適なクエリープランを選択するタイプのものと、UC BerkeleyのTelegraphCQプロジェクト[11]におけるCACQ[19]やPSoup[12]のように、タプルルータが入力されたデータをどのオペレータにルーティングするかをタプルごとに決定するタイプのものがあり、Babuら[8]は、前者をCQベースシステム、後者をルーティングベースシステムと呼んでいる。アーキテクチャの節（3節）で示したDSMSエンジンのアーキテクチャ図（図2）は、CQベースシステムを表している。

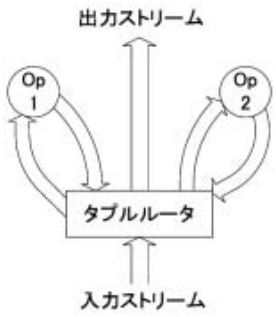

図7　ルーティングベースシステム

ルーティングベースシステムの概要を図7に示す。

ルーティングベースシステムでは、タプルルータはタプル（データ）ごとに次にどの演算子にタプルを送れば良いかを決定し、演算子にタプルを送る。各演算子はタプルを処理し、条件に合致しているもののみをタプルルータに結果として返す。タプルルータは、必要な演算子にタプルをルーティングし終え、処理結果が残っていれば、処理結果を出力ストリームとして出力する。

ルーティングベースシステムにおけるルーティングポリシーは、バックプレッシャ（Back Pressure）やチケットによるルーティング経路の最適化により決定される[6,19]。バックプレッシャは、各演算子のキューサイズを制限することにより、より処理の早い演算子に優先的にタプルがルーティングされるようにする方式である。チケットを用いる手法では、タプルルータはタプルを入力として受け付けた演算子にチケットを発行し、処理結果を戻した演算子からはチケットを取り戻す。こうすることにより、選択率の大きい演算子、すなわち入力に対して出力の多い演算子はチケットを多く持つようになる。タプルルータが、チケットの少ない演算子に優先的にタプルをルーティングすることにより、より選択率の小さな演算子による処理が早い段階で行われるようになり、全体のスループットを向上させることができる。

4.4　パターンマッチング

アプリケーションによっては、複数のイベントがどのようなパターンで出現しているかが主要な関心事となる場合がある。例として、クリックストリーム分析による行動ターゲティング（Behavior Targeting）アプリケーションを考えよう。このアプリケーションは、Webサイトのリアルタイムなページアクセス情報、すなわちクリックストリームをモニタリングし、ユーザーの行動特性に応じ有益である見込みの高い広告を提供する。たとえば、沖縄観光に関するページを閲覧し、さらに那覇のホテル情報ページを閲覧したユーザーには、那覇を拠点とする観光タクシーの割引広告ページへのリンクを提供する、といったものである。またRFIDなどを利用した物流プロセスモニタリング・アプリケーションでは、製品を詰めた箱が30分以内にゲートA、ゲートBを順に通過し、しかしゲートCは通っていない、といった状態を検知したいという要望があるだろう。一部のDSMSには、このようなイベントの出現パターンを検知するための仕組みが備わっている。この物流プロセスモニタリングでのパターンマッチを、商用DSMS製品Coral8のCCL（Continuous Computation Language）により記述した例をリスト3に示す。

リスト3　パターンマッチングの記述例

```
INSERT INTO AlertStream
SELECT A.BoxID,
      'Gate C skipped !' AS Message
FROM GateAstream AS A,
     GateBstream AS B,
     GateCstream AS C
MATCHING [30 MINUTES: A, B, !C]
ON A. BoxId = B. BoxId = C. BoxID;
```

さらにCornell大学とIndian Institute of Technologyの研究グループによるCayuga[14]のように、複数のイベントをまとめて単一のイベントのように扱える言語も提案されつつある。複合イベント処理（Complex Event Processing、CEP）[17]と呼ばれるこのようなイベント処理は、初期のDSMSには取り込まれてこなかったが、近年、Massachusetts大学Amherst校を中心とした研究グループによるSASE[28, 16]、Microsoft ResearchによるCEDR[10]など、複合イベント処理の可能なDSMSが多く提案されるようになってきている。

4.5　時刻の扱い

データストリーム処理をナイーブに実装すると、データのタイムスタンプの順序通りにデータが整列していない場合やデータの到着が遅れた場合、処理結果が不正になることがある。たとえば、オンラインオークションで、ある出品に対し直近5個の入札額の平均値を求めるとしよう。ここで、5番目の入札がネットワーク遅延など何らかの理由で遅れ、6番目の入札が先にシステムに到着すると、5番目の入札額によって正しく平均値が算出される前に、6番目の入札額が平均値算出に使用されてしまう。また別の例として、パターンマッチングの節（4.4節）で取り上げた例（リスト3）を思い返してみよう。ここで、ある箱がゲートA、Bを順に通過し、30分経過の直前にゲートCを通過したとする。しかし、ゲートCの通過イベントがシステムに到着したのが30分を過ぎてからだった場合、この継続的クエリーは間違ったアラートを発生させてしまうことになる。このような不整合は、データストリーム処理における

ウィンドウ演算や複数ウィンドウの結合演算など、いくつかの演算において発生する。

よって、データストリーム処理では、複数のストリームから到着するデータが、必ずしもタイムスタンプ通りの順序を保っていなかったり、到着が遅れたりする問題について対処する必要がある。

Tuckerら[26]は、通常のデータの中にパンクチュエーション（Punctuation＝句読点）と呼ぶ制御パケットを送り、処理を先に進めてよい合図とする方法を提案した。パンクチュエーションは、もともとはデータストリーム処理におけるいくつかの関係演算、たとえばGROUP BY演算では、特に工夫をしなければすべてのデータを読み込んだ後でなければ出力できないという出力のブロッキング問題に対する解として提案された。Brandeis/Brown/MIT各大学の研究グループによるAurora[1]では、データの順序性に敏感な各演算子に、順序が正しくなるまで待つ時間あるいはタプル数を表すスラック（Slack=緩み）と呼ぶパラメータを渡すことで問題を解決しようとしている。一方Srivastavaら[21]は、順序の乱れや到着の遅れは、演算子単位のパラメータではなくストリームの特性と考えるのが自然であるとし、さらに複数分散ストリームにおける到着時間のずれや順序の狂いが起こりうる点に着目した。彼らは、ハートビート（Heartbeat）と呼ぶ制御パケットの出力アルゴリズムにより、クエリーを正確に評価できることを示し、さらにこのアルゴリズムのパラメータとなるデータの遅延時間やストリーム間のデータ到着時間のずれをシステムが見積もる方法についても提案した。CEDR[10]では、ハートビートで導入されたアプリケーション時間とシステム時間の概念に加え、イベントの有効期間の概念を導入することで、現時点で有効なイベントをすべて出力する、といった要求にも応じられるようになっている。

4.6　その他の課題

これまでに紹介した以外にも、データストリーム処理の課題としてさまざま研究が行われているのでごく簡単に触れておきたい。

- **近似計算**：無限のデータストリームを限られた資源で処理するため、あるいはバースト的に負荷が高くなってもシステムをクラッシュさせることなく一定精度の演算結果を提供するため、近似計算を行う手法が多数提案されている。
- **資源共有**：アーキテクチャの節（3節）で述べたように、クエリープラン内部では、シナプシスやキューなどのデータ構造が作られている。複数の継続的クエリーが同時に同じストリームやウィンドウを参照する場合、これらデータ構造を共有することが行われている[5]。
- **データマイニング・機械学習**：データストリームを対象としたデータマイニングや機械学習についての研究も盛んに行われている[29]。

5．将来の展望

Sweeney[24]は、全世界で販売されたハードディスク容量の合計を世界人口で割った値は、1983年には0.02MBだったが、1996年には28MB、2000年には472MBとなったとしている。またLymanら[18]は、2002年の1年間に人類が集積した情報量は5エクサバイト（5×10^{18}バイト）、米国議会図書館の書籍ライブラリ37,000個分に達し、世界人口1人当たりでは800MBになるとしている。また世界の情報量の増加率は、1999年から2002年にかけて毎年30%であったと報告した。この情報量の急激な増加は、まさに情報爆発といってよい。すでに、コンピュータシステムが扱うデータは、従来のように人間が計算機に打ち込んでいたものだけではなく、各種のデバイスやセンサーからの自動収集データ、さらにこれらが複合し合成加工されたデータにまで及んでいる。このようなデータは、ほぼリアルタイムに処理されることによって価値が生まれる一過性の高いデータの割合が大きいと考えられる。大量データをリアルタイムに処理することに長けるDSMSは、こうした情報爆発時代の1つの解として、今後ますます重要な位置を占めるようになるだろう。

またビジネスの現場では競争が激化し、より早いデータ分析や意思決定が求められるようになっている。従来、データが溜まるのを待ち、定期的に集計分析処理を行っていたものも、今後は、データ発生時点の即時処理により、常に現状の分析結果を把握できるシステムが求められるようになるのではないか。このようなシステムにおいても、DSMSはキーコンポーネントとしての役割を担うだろう。

現在DSMS製品は、Stanford STREAMの影響を受けたCoral8、Brandeis/Brown/MIT各大学合同プロジェクトAurora[1]の子孫であるStreamBase、UC BerkeleyのTelegraphCQをもとにしたAmalgamated Insight、GPLで公開されているEsperなど、多数登場しつつある。今後は、パターンマッチングの節（4.4節）で述べたように、複合イベント処理（CEP）にも向くような言語拡張と最適化が1つの方向性となるだろう。

6．終わりに

本稿では、DSMSの概要を、登場の背景と対象となるアプリケーション、DSMSのアーキテクチャ、データストリーム処理における課題とDSMSによる解法、そして将来の展望という形で紹介した。各商用製品の詳細については、稿を改めて紹介することとしたい。

現在当社の研究開発グループでは、複数の商用DSMS製品を評価し、アプリケーションに応じた最適なDSMS製品を提供できるように準備を進めている。またこれら製品に、特定のアプリケーションに特化した独自技術ならびにフレームワークを組み込み、さらなる付加価値をつけることを目標に研究開発を続けている。

参考文献

[1] Daniel J. Abadi, Don Carney, Uğur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, Vol. 12, No. 2, pp. 120-139, 2003.

[2] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. STREAM: The Stanford Stream Data Manager. *Stanford University Technical Report*, 2004. http://dbpubs.stanford.edu/pub/2004-20.

[3] Arvind Arasu, Brian Babcock, Shivnath Babu, Jon McAlister, and Jennifer Widom. Characterizing memory requirements for queries over continuous data streams. *ACM Trans. Database Syst.*, Vol. 29, No. 1, pp. 162-194, 2004.

[4] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal*, Vol. 15, No. 2, pp. 121-142, 2006.

[5] Arvind Arasu and Jennifer Widom. Resource sharing in continuous sliding-window aggregates. In *(e) Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB), Toronto, Canada, August 31 - September 3 2004*, pp. 336-347, 2004.

[6] Ron Avnur and Joseph M. Hellerstein. Eddies: continuously adaptive query processing. *In SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 261-272, New York, NY, USA, 2000. ACM Press.

[7] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMODSIGACT-SIGART symposium on Principles of database systems*, pp. 1-16, New York, NY, USA, 2002. ACM Press.

[8] Shivnath Babu and Pedro Bizarro. Adaptive query processing *in the looking glass. In Proceedings of the International Conference on Innovative Database Research (CIDR)*, pp. 20-31, 2005.

[9] Shivnath Babu, Rajeev Motwani, Kamesh Munagala, Itaru Nishizawa, and Jennifer Widom. Adaptive ordering of pipelined stream filters. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 407-418, New York, NY, USA, 2004. ACM Press.

[10] Roger S. Barga, Jonathan Goldstein, Mohamed H. Ali, and Mingsheng Hong. Consistent streaming through time: A vision for event stream processing. In CIDR 2007 [13], pp. 363-374.

[11] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel Madden, Vijayshankar Raman, Frederick Reiss, and Mehul A. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *CIDR*, 2003.

[12] Sirish Chandrasekaran and Michael J. Franklin. PSoup: a system for streaming queries over streaming data. *The VLDB Journal*, Vol. 12, pp. 140-156, 2003.

[13] *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*. www.crdrdb.org, 2007.

[14] Alan J. Demers, Johannes Gehrke, Biswanath Panda, Mirek Riedewald, Varun Sharma, and Walker M. White. Cayuga: A general purpose event monitoring system. In CIDR 2007 [13], pp. 412-422.

[15] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *SIGMOD Rec.*, Vol. 32, No. 2, pp. 5-14, 2003.

[16] Daniel Gyllstrom, Eugene Wu 0002, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. Sase: Complex event processing over streams (demo). In CIDR 2007 [13], pp. 407-411.

[17] David Luckham. *The Power of Events*. Pearson Education, Inc, Boston, MA, USA, 2002.

[18] Peter Lyman and Hal R. Varian. How Much Information 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003, 2003.

[19] Samuel Madden, Mehul Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously adaptive continuous queries over streams. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 49-60, New

York, NY, USA, 2002. ACM Press.

[20] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query Processing, Resource Management, and Approximation in a Data Stream Management System. *In Proceedings of the International Conference on Innovative Database Research (CIDR)*, pp. 245-256, 2003.

[21] Utkarsh Srivastava and Jennifer Widom. Flexible time management in data stream systems. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 263-274, New York, NY, USA, 2004. ACM Press.

[22] Michael Stonebraker, Chuck Bear, Uğur Çetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, and Stanley B. Zdonik. One Size Fits All? Part 2: Benchmarking Results. In CIDR 2007 [13], pp. 173-184.

[23] Stream Query Repository. http://infolab.stanford.edu/stream/sqr/.

[24] Latanya Sweeney. Information explosion. In L. Zayatz, P. Doyle, J. Theeuwes, and J. Lane, editors, *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*. Urban Institute, Washington, DC, 2001. http://privacy.cs.cmu.edu/people/sweeney/explosion.html.

[25] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. In *SIGMOD '92: Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, pp. 321-330, New York, NY, USA, 1992. ACM Press.

[26] Peter A. Tucker, David Maier, Tim Sheard,and Leonidas Fegaras. Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 3, pp. 555-568, 2003.

[27] Mark Weiser. The Computer for the 21st Century. *Scientific American*, Vol. 265, No. 3, pp. 94-104, Sep 1991.

[28] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 407-418, New York, NY, USA, 2006. ACM Press.

[29] 有村博紀、喜田拓也。データストリームのためのマイニング技術. 情報処理、Vol. 46, No. 1, pp. 4-11, 2005.