統合インデックスプラットフォーム アーキテクチャの提案

日髙東潮　寺本純司　須賀啓敏　小林伸幸　星隆司　櫻井紀彦

日本電信電話株式会社　NTT サイバースペース研究所

近年では実サービスにおいて利用されるデータソースとして、RDB に代表されるデータベースシステムに加え、Web 上のデータなど多様なものが存在する。これらデータの多様化は今後さらに進み、加えてデータの運用も独立したさまざまな形態のものが発生することが考えられる。それらの利用技術として、最近の Web アプリケーションに対するマッシュアップ技術などに見られるような、これらを連携・利用する仕組みがますます必要となる。

本論文では、独立した異種情報源に対する統合検索方法についてのアーキテクチャを提案する。また、アーキテクチャ検討において特に解決が困難な、異種情報間の結合処理（JOIN）について高速に処理するための考察を述べる。

INDEX Platform Architecture that Specializes in Integrating Heterogeneous Information Sources

Toshio Hitaka, Junji Teramoto, Yoshiharu Suga, Nobuyuki Kobayashi,
Takashi Hoshi, Norihiko Sakurai
NTT Cyber Space Laboratories, NTT Corporation

In recent years, as the data sources on real services, various kinds of web data sources could be seen in addition to the database system, for example, RDB, and diversification of data source is proceeding. Therefore, the mechanism of integrating data sources is needed more and more so that it is shown in the "mashup" technology as a recent Web application.

In this paper, we discuss the architecture of the integrated retrieval method to an independent heterogeneous information sources, moreover focus on the policy of process fast join processing.

1．はじめに

今日、インターネットの社会への浸透が著しく、中でも Web サービスは単純な情報案内サービスに留まらず、電子商取引を始めとする商用システムとしての利用も十分認知されている。このような社会背景から、情報爆発時代に向けた新しい IT 基盤技術の研究が提唱されており[1]、近年では Web2.0 サービスの普及に伴い、Web サービス単体での利用に加え、複数の Web サービスを連携させたマッシュアップ等の複数情報源の統合技術に対するニーズが向上してきている。この情報統合のための技術としては、Data Warehouse のように１つのスキーマへ完全にデータ統合するものから、Mediator のようにスキーマ統合による論理的なデータ統合のみを与えるもの[2,3,4]まで様々なものがこれまで提案されてきた。

しかし、情報源の提供がこれまで企業主体であったのに対し、今後はより個人に主体が移っていくことが想像され、その場合には完全なデータ統合はおろか、Mediator で必要なスキーマの統合さえも非常に困難になることが予想される。そのため、これまでとは異なったデータ統合のためのアプローチが必要になる。

本論文では Web サービスの中でも検索サービスにフォーカスし、サービスに求められるデータ統合の形態を整理した上で、その整理に適応したアーキテクチャを提案し、その中で特に性能に対

する技術的課題について検討する。

2．情報統合サービスのためのアーキテクチャ

複数の情報源を統合し検索サービスを提供するためのアーキテクチャとして、クエリの内容から情報源とその組み合わせ方法、アプリケーションに対するデータの開放範囲を決定するための制御レイヤと、情報源の物理的な所在管理、並びに統合処理を高速に行うレイヤといった、2つのレイヤが必要になる。ここでは前者を“ビュー制御部”、後者を“統合インデックス機能部”と呼ぶことにする。

・ビュー制御部：
アプリケーションからのクエリ要求に対し、どの情報源を組み合わせ、検索結果をアプリケーションに対してどのように見せるのかを制御する。
本機能部については、以下2点の考慮が必要である。
(a)アプリケーションへの出力制限制御
(b)データの整形制御
ただし、本論では様々な情報源の形態が存在することを考えると(b)を共通的なPFとして考えた場合に問題が爆発するため、(a)のみをビュー制御部としての機能と位置づけ、整形については個別のアプリケーションで整理する、というスタンスを取る。また、(a)については情報源単位での出力制限や、検索条件にデータを制限するための条件を暗黙で付加するなどの方策が考えられるが、この制御部については本論とは別途論じる。

・統合インデックス機能部：
各種情報源の構造管理、所在管理を行い、複雑な問い合わせに対して高速な検索機能を提供する。
本機能部については、以下2点の考慮が必要である。
(a)さまざまな形態を持つ情報源の構造、所在を一元的に管理するためのフレーム
(b)複数の情報源を対象とした検索クエリを高速に処理するためのエンジン
インデックス管理として考えた場合、(a)が扱うデータは文字列、数値を基本とする方法があり、これについてはRDB等での既存の方法が適用可能である。一方で、(b)については分散した情報源に跨ったクエリ処理のために特に性能の問題を考慮しておく必要がある。

図1に、全体のレイヤ構成を示す。

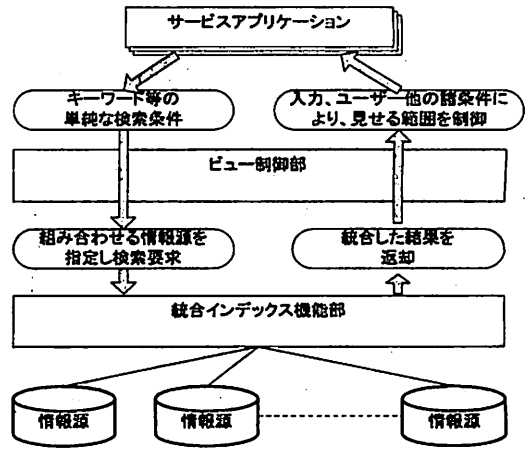

図1：全体のレイヤ構成図

本論ではこのうち統合インデックス機能部について、情報源統合クエリの分析と、問題点の明確化を3章、4章で行い、それを踏まえたアーキテクチャの提案を5章で行う。

3．データの統合処理に対する整理

各情報源は様々な形態を有しているが、検索サービスを提供する上で、以下のような構造データの集合と単純化して考えることができる。

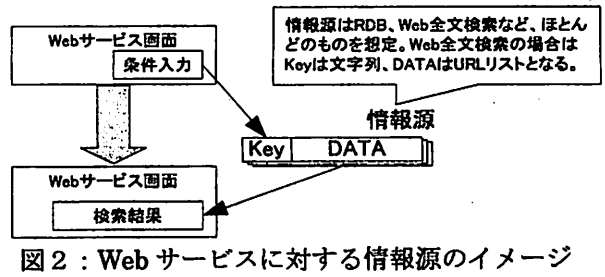

図2：Webサービスに対する情報源のイメージ

これを踏まえ、情報源の統合について主に2つの形態にここでは整理する。

①連結
情報源A,Bがある時、まず情報源Aに対して検索を行い、検索結果を用いて新たな検索キーに加工し、情報源Bの検索を行う。

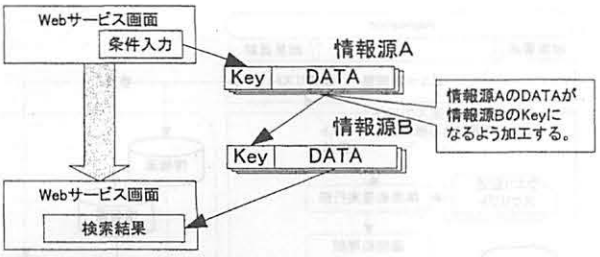

図３：情報源の連結のイメージ

連結処理については、情報源からの返却値により次の検索を実施するため、情報源に応じた返却値のキーへの加工など、情報源個別の実装が必要である。また、直列的な処理であるため、性能は個別の情報源の検索性能にそのまま依存し、情報源Aからの検索結果が多い場合は、情報源Aからの転送処理、ならびにそのデータを使っての情報源Bとの連結処理のすべてが性能面での問題となる。なお、インターネット上の情報源 A、B を連結する場合は上記の性能的な問題から、情報 A からの出力を足切りして B の入力に加工するなどの対処が一般的に見られる。

②結合（JOIN）

情報源 A,B の内部構造を踏まえてそれぞれのキーを指定し結合（JOIN）処理を行い検索する。

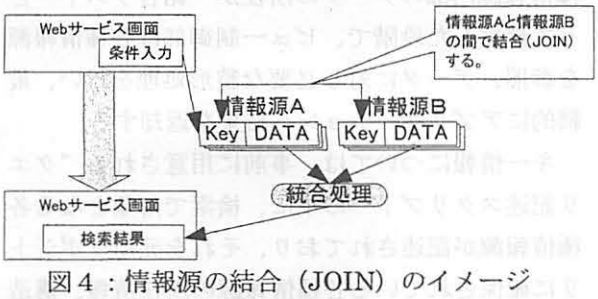

図４：情報源の結合（JOIN）のイメージ

情報源 A、B の内容を相互参照しながら結合処理を行うため、Key 間の結合対象が多い場合は、情報源A,B間で大量のKey参照処理が必要になる。そのため、インターネット上の情報源 A、B を結合する場合は、結合対象データが非常に多いと実サービスとしては非現実的なレスポンス性能しか得られず、また各情報源に対しても過大な検索負荷をかけることになる。なお、連結処理の場合は足切り処理による処理の簡略化が可能であるが、結合処理前にデータの足切りを行うと結合結果に大きく影響を及ぼすケースがあるため、簡略化によるレスポンス性能改善は困難である。

これらのことから、現在マッシュアップサービスとして見られるものとしては連結処理相当である Yahoo Pipes や、Ajax による個別サービス API を結合させた Web アプリなどに限定される。

4．結合処理における問題点について

前節で述べた情報源間の結合処理について、より詳細に分析するため、サービス例を挙げて検討する。図５は観光情報サービスを想定したもので、地名を指定し、花見の名勝地と、その周辺情報をセットで提供するものである。

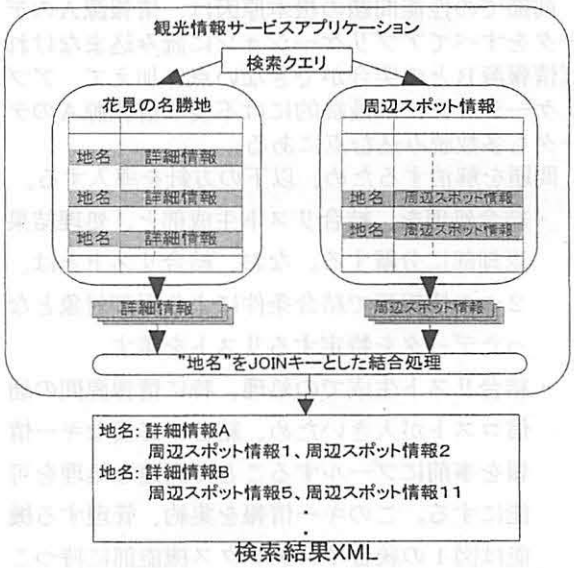

図５：統合検索によるサービス例とデータフロー

この例をネットワーク経由で結合処理する場合、結合処理部（アプリケーション）に一旦一方の情報源のデータをすべて読み込み、それをベースにもう一方の情報源に対しループジョインを行うため、情報源内のデータ量が多い場合、膨大なデータを通信しなくてはならず、Web サービスに必要な性能を得ることは非常に難しい。

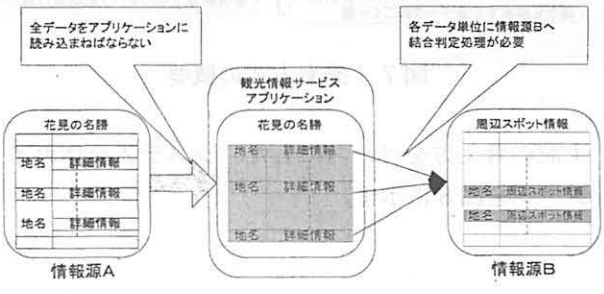

図６：ネットワーク経由での結合処理

結合処理に限らず、すべての検索処理は基本的に以下の２つのステージで構成される。

a.与えられた条件のデータ位置を特定する。

b.特定されたデータを返却(Fetch、整形)する。

前述の例で問題となるのは、情報源Ａと情報源Ｂとを突合させ、最終的に必要となるデータを確定する、上記ステージのうちａである。そのため、前述した結合処理での性能を改善するためには、上記a.を高速化する方式が必要となる。

５．提案するアーキテクチャ

前節での性能問題の根本原因は、情報源Ａのデータをすべてアプリケーションに読み込まなければ情報源Ｂとの突合ができない点に加えて、アプリケーションには最終的には不要な情報源Ａのデータも多数読み込む点にある。

問題を解消するため、以下の方針を導入する。

- 結合処理を、結合リスト生成部と、処理結果返却部に分離する。なお、結合リストとは、２つの情報源で結合条件により返却対象となったデータを特定するリストを差す。
- 結合リスト生成での処理、特に情報源間の通信コストが大きいため、結合に必要なキー情報を事前にプールすることで高速な処理を可能にする。このキー情報を集約、管理する機能は図１の統合インデックス機能部に持つことを想定する。

これらの方針を導入した概要を図に示す。

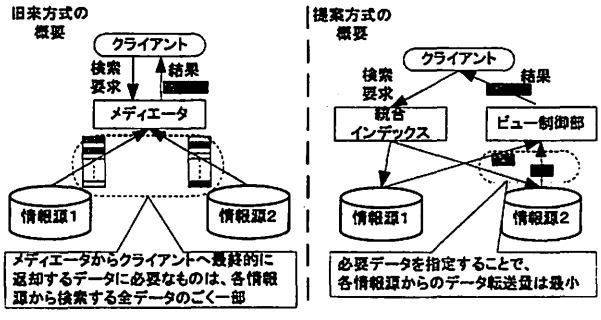

図７：提案方式の概要

上記の考え方をベースにした、システム全体のイメージを図８に示す。

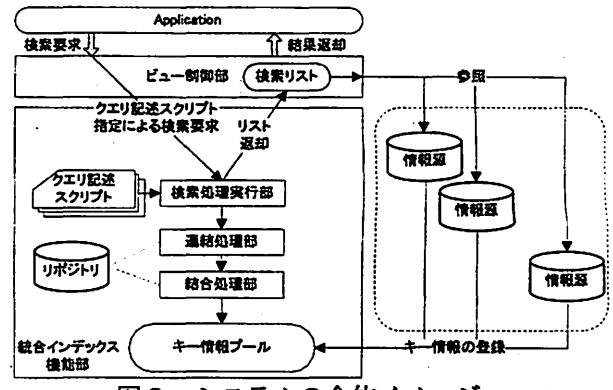

図８：システムの全体イメージ

事前に統合インデックス機能部内のクエリ記述スクリプトに検索で使用する情報源の組み合わせを記述、用意しておく。検索時にはアプリケーションから単純キーワード指定等による検索クエリを与えられたビュー制御部が、統合インデックス機能部に複数用意された“クエリ記述スクリプト”を選択し、必要な検索用パラメタを指定して検索を実行する。“クエリ記述スクリプト”には情報源の連携方式として連結と結合を記載できるようにしておき、その内容に応じて連結処理部と結合処理部を呼び出し実行する。検索の実行においては、各種情報源の構造、内部リソース管理を行う“リポジトリ”を参照しながら行う。これにより、各種情報源内部のデータの所在が“結合リスト”として確定した段階で、ビュー制御部は各種情報源を参照、データに対し必要な整形処理を行い、最終的にアプリケーションへ結果を返却する。

キー情報については、事前に用意される“クエリ記述スクリプト”の中に、検索で対象となる各種情報源が記述されており、それを元にリポジトリに確保されている各種情報源の所在情報、構造情報を参照し、“クエリ記述スクリプト”で使用するキー情報を“キー情報プール”に登録してもらうことで、内部にキー情報をすべて確保する。

以上の処理の中で、特に性能的な問題となるのが“結合処理部”となる。問題の観点としては、主に以下の２点となる。

・レスポンス性能の要求

上記構成は、“各種情報源に対して検索結果を構成する上で必要なデータを特定する”ことを提供するものであり、サービスに必要な結果を整形して最終的に返却する処理がさらに必要になる。そのため、サービスに必要なレスポンス性能を得る

ためには、上記構成の提供する機能範囲については極めて高速なレスポンス性能が要求される。ここでは上記要求を解決するため、メモリにキー情報を配列として格納し、そこに範囲検索条件等も考慮し Tree 系インデックスを付与することでレスポンス性能を得ることを考える。

・動的構成と静的構成

結合処理においてレスポンス性能を向上させるには、RDB の場合は結合表を準備しておくなどの静的な構成方法により、クエリ要求が来た際の処理を可能な限り前倒ししておく、という方法が考えられる。しかし静的な構成の場合はあらゆる結合ケースを想定して事前に結合表を作ると組み合わせ爆発を起こしてしまうため、限定的に利用する組み合わせを決めてしまい、結合表を作るのが一般的である。想定するサービスでは、静的に用意された結合表が使えないケースが必ず発生してしまうため、動的な構成を想定したレスポンスを前提とした性能検証が必要である。

図８のうち、特に性能的に重要となる結合処理部について、レスポンス性能と結合組み合わせに対する柔軟性、スケールアップを考慮し、図９のように結合処理部とインデックス部との２段構成にする。

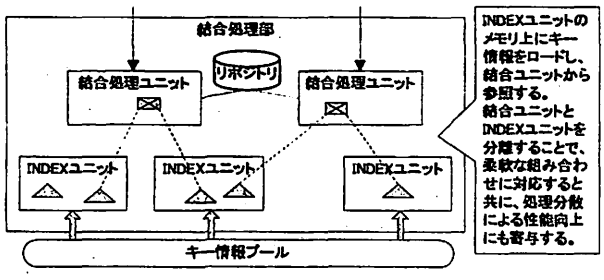

図９：結合処理部の概略

INDEX ユニットには、各種情報源のキー情報に対するインデックスをメモリ上で構成しておき、結合処理だけでなく完全一致、範囲一致などのさまざまな検索クエリに対して高速にデータを確定可能とする。

結合処理ユニットは、ビュー制御部からの検索要求を受け、リポジトリにアクセスすることで、結合対象の情報源に相当するキー情報を保持している INDEX ユニットを確定する。その上で、複数の INDEX ユニットを介して結合処理を行い、検索結果に必要なデータの確定処理を行う。

６．性能面に対する考察について

前節で提案したアーキテクチャの性能面に対する Feasibility 検証として、結合処理の性能予測を行う。

一般的にソート済の２つのリストを結合する場合には、ソートマージジョインが高速であるため、これをベースに２つのマシンのメモリ上リストをネットワークで結合する場合を想定する。実験としては下記条件でのネットワーク転送性能を実測した。

＜環境条件＞

下記マシンを２台用意。

CPU: Intel Xeon 5130 2GHz
Memory: 16GB
OS: RedHat EnterpriseLinux4
マシン間は GbEther で接続

＜測定方法＞

・ネットワーク通信の connect()と close()の時間は除外。send()のみを計測する。
・send()の実行監視では、確実に転送先までパケットが到達する時間の計測が困難なため、別途 tcpdump コマンドでの解析を実施。
・５回の試行による平均を算出。計測結果値の分散が激しい場合には、試行回数を増やして検証する。
（計測結果としては極めて平滑化しており、５回で計測を打ち切った）

＜結果＞

			転送データサイズ(byte)			
			256	2048	6000	9000
転送効率 MB/Sec	試行回数	1	28.845	141.241	207.792	285.714
		2	25.924	143.719	209.607	288.000
		3	26.947	143.719	208.696	289.157
		4	26.947	143.719	212.389	286.853
		5	25.924	143.719	209.607	286.853
	最大		28.845	143.719	212.389	289.157
	最小		25.924	141.241	207.792	285.714
	平均		26.606	143.719	209.303	287.235

表1：データサーズによる転送効率の変化

1回あたりのデータ転送サイズが6KB以下では、データ転送サイズによらず1転送回数あたりの処理時間はほぼ変化が見られなかった。このことから、ネットワーク通信における send()命令発行のオーバーヘッドを抑え、効率の良い通信を行うためには、1回あたりのデータ転送サイズを 9KB 以上の十分大きなサイズにする必要があり、その結果として最大約 280MB/Sec というデータ転送性能が得られることが分かった

単純なループジョインでは、1 回のデータ転送サイズは1レコード相当になるため、通信データを 6KB 以上にブロッキングを行うよう、アルゴリズムの改良が必要となる。

情報系システムで扱う DB を例に取ると、ほとんどのものが1テーブルあたり1億レコード以下で構成される。結合処理でその 1%が結合対象になると仮定し、1レコードのデータ量を 64byte 程度とした場合、ループジョインで発生するネットワーク転送量は最大で約 128Mbyte であり、0.5 秒程度の転送時間で処理可能と想定される。一般的に Web サービスに求められるレスポンス性能は1回の問い合わせに対して2秒以下であるものが多く、結合リストの生成をその 25%程度の時間で処理ができることから、本アーキテクチャで十分な Feasibility が得られると考えられる。

7．今後について

データの整形については、旧来より連邦 DB、並びにセマンティック Web などで論じられてきているが[5]、異種情報源はそれぞれ独自に設計、運用されていることが多く、多様なデータすべてに対して適用可能な整形手段の確立は極めて困難である。本論ではその点に関してはビュー制御部、もしくは個別のサービスアプリケーションで解決することを前提としているが、別途、プロトタイプを作成した上でビュー制御部の機能範囲を含めた考察が必要と考えている。

また、情報源の提供がより個人主体へ推移した場合、多数の情報源について、サービスアプリケーションがどの情報源を組み合わせ、それをどのような形式に加工（ビュー化）するのかを決定する仕組みに相当するビュー制御部内の機能が必要となる。今回は事前にアプリケーションで利用対象の情報源については“クエリ記述スクリプト”という形で記述し用意しておく方式を提案したが、個人が提供する情報源は極めて流動的であり、加えて情報源自体が個別に運用されていることから、動的に変化していく情報源の状態を踏まえて選択利用するための仕組みも必要になると考えている。これについても別途検討を行う予定である。

8．おわりに

将来ますます増加していくと思われる様々な異種情報源を統合利用する方法と、求められる性能、運用性、拡張性を満たすためのアーキテクチャを提案し、今回は特に動的な結合処理におけるレスポンス性能について考察を行った。

今後は、ビュー制御部の詳細化の上、システムのプロトタイプ化を進めることでその他必要な機能モジュールを明らかにし、その後プロトタイプでの性能検証、運用検証を行うと同時にサービスへの適用性についても検討を行う予定である。

参考文献

1) 喜連川 優 他,“研究領域「情報爆発時代に向けた新しい IT 基盤技術の研究」”, http://itkaken.ex.nii.ac.jp/i-explosion/

2) 池田 哲夫, 鈴木 源吾, 町原 宏毅, 安田 浩: 連邦データベースシステムにおけるスキーマ構築の一方式, 情報処理学会論文誌, Vol.40,No.SIG8(TOD4), p.29--40 (1999).

3) 宗像 浩一 他, “メディエータによる異種分散情報源の検索処理方式”, 1996 年電機情報通信学会総合大会, SD-1-3

4) 片山 薫 他, “分散メディエータ環境における索引情報のキャッシング手法の提案”, 情報処理学会 第 53 回全国大会講演, pp411-412

5) 岩爪 道昭, 白神 謙吾, 畑谷 和右, 武田 英明, 西田 豊明, "オントロジーに基づく広域ネットワークからの情報収集・分類・統合化", 情報処理学会論文誌, Vol.38, No.3, pp.606--615(1997).