データベースシステム 116－7
（1998. 7. 8）

WWWシステムとDBMSの相互利用／相互接続による応用支援環境について

Interconnecting WWW applications and DBMSs

小島 功
Isao KOJIMA

電総研情報アーキテクチャ部
ネットワーク情報ベースラボ
Networked Information Base Research Group
Computer Science Division, Electrotechnical Laboratory
http://www.etl.go.jp/etl/infobase/nib/

概 要

WWW検索エンジンやプロキシサーバなどWWWの応用システムや、関係DBMSを相互に接続し、相互利用を実現する環境について述べる。この目的は、それぞれのソフトウェアにできるだけ変更を加えることなく、これらの機能を統合することで、より優れたHTMLデータの管理、検索環境を実現することである。

複数のシステム上にある重複データの一貫性を保持するなど、独立したそれぞれのシステム間にまたがって一貫した動作を行なう必要があるため、ECA (Event-Condition-Action) 規則 [6]による制御を行なう点に特徴がある。また、相互接続を実現するためにシステム間を統合するインターフェイスを設ける代わりに、データベースの集合処理や、HTTPベースのアクセスなど、各システムの持つインターフェイスを相互に支援し、異なるインターフェイスを用いる場合の連携動作を規則記述で行なう。各システムはそれぞれのフロントエンドを持つ。

この環境により、複数の全文検索エンジンをデータベース的に統合するメタサーチ、全文検索をデータベースの索引として用いる、プロキシサーバとの接続／統合と言った応用が容易に支援できるようになる。応用例として全文検索エンジンとの相互接続を示し、評価する。

キーワード： WWW, 全文検索エンジン, プロキシサーバ,ECA,HTTP,SQL

Abstract

In this paper, interconnecting WWW application systems and DBMSs is discussed. The approach is to integrate the functions of these systems without any modifications. ECA(Event-Condition-Action) based rule processing is proposed to maintain the integrity of duplicated data between separate systems. Query modification rules and data transformation rules are presented. Since component systems have their own interface, both HTTP based access interface and set-oriented interface for database operations are supported, instead of providing a single integrated protocol.

Based on the architecture, several applications like meta-search engines and context retrieval over existing databases are supported. Integrating a WWW search engine and a relational DBMS is used as an example application. Some performance evaluations are also shown.

Keywords: WWW, Search engine, Proxy server, ECA,HTTP,SQL

1 まえがき

WWW の爆発的な普及 [4]に伴って、HTML データが増加の一途をたどっている。イントラネット的な応用からプロキシサーバ、検索エンジンのような大規模なデータ収集／管理に至るまで、HTML データへの検索機能や更新管理の機能への要求は非常に高くなっており、データベースシステム (DBMS) の機能を使った管理が有効と考えられる。

全文検索エンジンやプロキシサーバ等の WWW 関連システムの多くは、独自のファイル構造を持ち [3]、その上での検索／更新などの処理を行なっている。従って、応用分野や利用法を拡大するにつれ、機能的な問題が発生する。

例えば、専用の全文検索エンジンを HTML ファイル管理／検索のツールとして汎用的に用いる場合、次のような問題が考えられる [1]。

- **更新管理：** HTML データをロボットで定期的に巡回して集めるものは、索引の構成がバッチ的／静的で、更新に対する効率的な支援に欠けるものがある。特にイントラ的な応用では、頻繁な更新の効率的支援は不可欠である。
- **機能拡張：** 利用者が複数の検索エンジンを使い分ける動機の一つに、その検索／マッチング方式の差がある。そのため、汎用の枠組みで検索方法を区別／切替えたり利用者処理を組み入れられれば、利便性は上がる。

これらの要求は、DBMS の持つ更新／共有機能や応用プログラムとのインターフェイス等で支援しやすい機能である。

一方、従来の DBMS にも、WWW 応用を支援する環境としてはいくつか問題がある。同様に、専用の検索エンジンと比較すると次のような問題がある。

- **テキスト管理：** like 文等のデータベースのテキスト検索は、特に大規模なデータ環境下での性能が悪い。また、検索自体の機能も限られる。索引は特定のテキスト形式でしか扱えず、word や acrobat 等応用固有のテキスト形式に対する検索は十分でない。
- **高度で対話性の高い検索：** SQL を代表とする検索言語では、あいまい性を含んだ検索や、辞書を使った検索などが直接支援できない。また、絞り込みなど対話性の高い検索も十分支援できない。

これらは現在の商用検索エンジンが最初から有している特徴と考えられ、全文検索の専用システムである検索エンジンと、汎用性の高い DBMS 技術のより良い接続／融合体系を考えることで、相互の問題を補完できる可能性が高い。例えば、DBMS のシステムの内部的な索引として全文検索エンジンの索引構造を用いることができれば、テキスト型に対する検索性能は大幅に向上する。

このように、DBMS の機能と WWW 関連システムの機能とのより良い統合を実現できれば、複数の全文検索エンジンの結果をデータベース処理で統合したり、プロキシーサーバ間の更新波及やデータの一貫性保持を DBMS の更新機能で支援したり、HTML データベースに対する問い合わせ環境の実現など、大量のデータが収集される WWW データ処理に対して優れた環境ができると考えられる。従って本稿では、全文検索エンジンやプロキシサーバなどの WWW 応用システムと SQL を基本とした関係データベースの相互接続環境の実現方法について述べる。

2 相互接続環境へのアプローチ

DBMS にテキスト管理や検索の機能を含めたり、データ型の拡張を行なって特定データ形式に対する索引を実現したり、あるいは WWW システムに対するインターフェイスを設けるなど、さまざまな研究が存在する。しかし、これには次のような問題が考えられる。

- 全文検索エンジンなどのシステムが既に独立したソフトウェアとして存在するので、これらと同等の機能をデータベースシステム側で開発／組み込むのは開発コストが大きい。
- 特に WWW では、個々のシステムの機能的発展／拡張が大きく、はじめから統合的、包括的なシステムを構築するのが難しい。
- 機能的な統合を行なう場合、例えば DBMS に抽象データ型／プラグイン的な拡張を行なう [5]ような方法では、外部関数的な扱いになる。問い合わせ最適化機構へ反映できる部分が少ないので、性能的な利点が小さい。

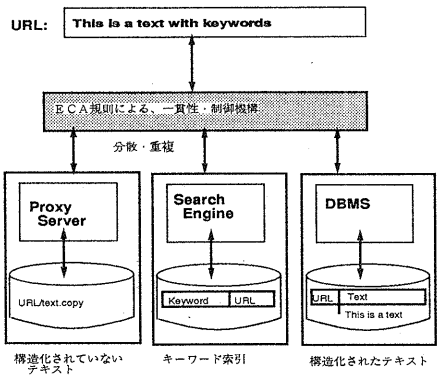

図1. アプローチの概要

従って、コスト／利益比を考えた時、既存のシステム／ソフトウェアを可能な限りそのまま使い、これらの相互接続と連携動作による統合的環境を実現することの意義が大きい。この実現のために本稿では、次のような問題を扱った。

- **データや処理の一貫性保持：** 例えば、検索エンジンとDBMSとを接続して、DBMSのデータに全文検索を行なうと考える。この場合、DBMS内のデータと、検索エンジンの索引と言う、別のシステムの持つデータを同一のものとして扱う必要がある。データベースへの更新は検索エンジン側にも波及されねばならない。プロキシサーバ内の更新についても同様である。
- **インターフェイスの機能的統合：** 個々のシステムは利用のための独自のインターフェイスを持つ。そのため、相互に利用可能にするには、例えばデータベース内のデータを検索エンジンのソフトウェアが直接アクセスできるようにしたり、検索エンジンの結果データを、DB処理に使えるよう、相互にアクセス可能なインターフェイス／プロトコルを設定する必要がある。

ここは、次のようなアプローチを考える。概要を図1に示す。

1. **ECA規則による、システムの連携：**
   2つのシステムの間を接続した場合、更新や検索と言った事象に応じて様々な条件を調べ、その結果に応じて処理をする必要がある。例えば、データの更新があった時、もしそのフィールドにWWW索引があれば、索引の更新を行なうような場合である。このため、著者らはこのような動作をECA規則により記述し、規則の動作によって2つのシステムの間の一貫性を保持することとする。規則形式でシステムを接続するのは自然でプログラミングに有利と考えられる。また、ECAには並行記述やトランザクション記述も含まれるので、独立性の高いシステムの連携動作の記述には有益であると考えられる。
2. **複数のインターフェイスの支援：**
   複数のシステムを統合する方法として、統一的なプロトコル／インターフェイスを設定し、各システムのインターフェイスをそれに合わせて変換するフロントエンドを構築すると言うのが一般的である。
   しかし、ここでは次のような問題がある。
   (a) **データベースの規定するスキーマや構造と合わない:** データベースのモデリングによって統合することができるが、プロキシサーバや全文索引のデータは、大きな構造的複雑さを必要としない。一方、リンクのような構造やデータごとのアクセス手法は、データベースでの集合的管理が難しい。
   (b) **要素ソフトウェア独自のインターフェイスは、機能的に統合できない:** 例えば、全文検索エンジンの収集するデータは、WWWロボットにURLを通知することで行なわれる。つまり、WWWロボットがアクセスできるプロトコルがデータベース側になければ、全文検索の機能とデータを共有できない。

従って本稿では、特に統一的なモデル／インターフェイスを考えることはせず、相互接続のためのプロトコルを複数設定し、それぞれのインターフェイスを設けることとする。全文検索エンジンとデータベースの例で説明する。概要を図2に示す。

(a) **データベースのデータをHTTPベースでアクセス可能にするインターフェイス** データベース側に、HTTPベースで仮想的にテキストと見せるインターフェイスを構築する。また、構造を持ってアクセスする必要がある時は、メタタグを用いたフィールドの値の組に変換する。

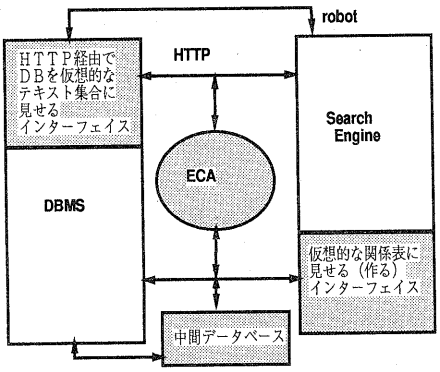

図2　相互接続の概要

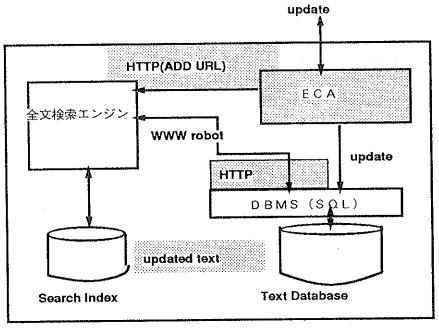

図3　更新の索引への波及

(b) **検索エンジンなど、WWWのシステムへの入出力を、データベース処理できるインターフェイス結果のテキストなどの集合は、URLをキーとした関係データベースの組として扱う。**索引エンジンも同様に、単語とURLの組として扱う。

この2つのインターフェイスの操作を規則で関連づけることで、例えば問い合わせ内の単語からHTTPベースで検索エンジンにデータを渡し、結果集合を関係の形でもらうと言うアクセス手法を取ることができる。

3 ECA規則による処理

利用者が複数のシステムの存在を意識することなく処理を行なうためには、規則による処理の変換が必要となる。ここでは、更新を複数のシステムに波及させる場合と、一つの処理を各システムに変換、分担する例を示す。

3.1 データベースの更新を索引に反映するための規則

検索エンジンの索引構造があるので、更新に伴い関係上の元データとの一貫性を保持する必要が生じる。利用者は索引の存在を仮定しないで問い合わせるため、更新に伴う一貫性保持処理を規則として定義する。例えば、

```
Event: update text
Condition: if there exist search index.
Action: send updated URL(in this case,
        text and the location) to the search
        engine.
CM:decoupled.(the update is not required
        to be reflected immediately)
```

のようなものである。URLを送る処理は、更新されたURL（次章で後述、IDとして付加される）をWWW検索エンジンに通知して、その部分を更新するようなトリガとして働く（図3)。実際には、検索エンジンがこのURLを参照してロボットがHTTP経由で該当データへのアクセスを行ない、索引の更新を行なう。

ここでの特徴は、2つの独立性の高いソフトを結び付けている以上、トラブル波及の防止など、相互の処理の結合関係は留意する必要がある。ここでは、そのためにECA規則の結合モードを用いている。上の例は、ロボット自体がアクセス先でトラブルを起こす可能性があるため、索引更新のトランザクション自体は独立して処理され、エラーはDBMS側には影響しない。

3.2 処理の変換／分担

SQL問い合わせを検索エンジンを併用する検索への変換を示す。問い合わせの変換は概念的には、

```
Event: select text
Condition: if there exist search index.
Action: execute the modified query
        using the search index
        and return the result.
CM: immediate
```

の形式[†1]で書かれる。実際には、Actionで書かれる問い合わせの変換には、いくつかの問題がある。

- **集合結果の演算**: 一般に問い合わせは、

```
select * from MYDB
        where text xlike("STRING")
```

と;属性の個々の値に適用される外部関数的な表現で書かれる。ところが、単語による検索は入力が1単語（ないし列）で出力が集合である。組み込み関数と考えると、関数の多くは1入力1出力で、集約関数も多入力1出力である。このため、索引の検索を通常のDB関数として扱う場合、1入力1出力の形式に揃える必要がある。このため、Actionで呼ばれる外部関数を実現する形式は

```
define function xlike(keyword,table)
  /* キーワード検索のプログラムと
        結果の関係への格納 */
  returns(tablename)
```

とし、検索エンジンの結果（集合）はいったん表に作成される。戻り値は結果のtablename 1つ[†2]であり、これを用いて結合など他のSQL処理を行なう必要がある。Actionで行なわれる問い合わせ変換は、集合処理を仮定した問い合わせを、関係を戻す形式と明示的な関数呼び出しに変換し、一種の副問い合わせとして処理する。

```
select * from (select * from
     TABLE xlike("MYDB" "STRING"))
```

IN句を用いた場合については本稿では省略する。処理の概要を図4に示す。

- **ランキングの扱い**： 検索エンジンはマッチングの度合を示すランキングを戻し、結果はランキングでソートされている。先のように関数処理の結果は一旦関係に構成されるので、ランキングの属性を作成する。これにより結果に対してSQL側からランクに関する条件の評価が可能となるが、条件

†1 本稿では、ECAの各要素をSQL表現として定義／処理するフロントエンドを仮定している。文献[10]。

†2 illustra等では戻り値として集合を返すことができるが、このために動的にカーソルが生成されるので、処理としては同じと考えられる

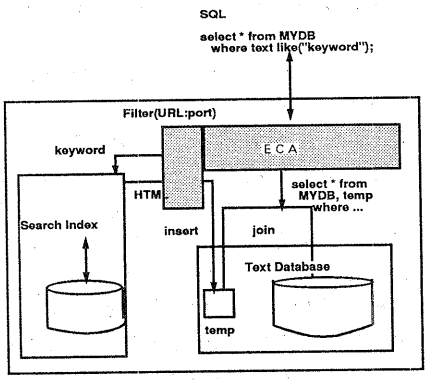

図4 データベース処理と索引検索の結合

をDBMS側で見る限り、検索エンジンから戻った全てのデータを一々処理する無駄がある。従って、ランキングの処理できる検索エンジンでは、

```
define function xlike(keyword,table,
                ranking_cond,op)
returns(tablename)
```

という形式の関数を実現し、条件を検索エンジン側に送る方が効率的である。一方、問い合わせの変換は、その条件文を元のSQLから切り離して行なう。

4 データベースとWWWシステムの相互アクセス

4.1 データベースへのHTTPアクセス

HTTPによるデータベースのアクセスを可能にするために、次のようなCGIに基づくインターフェイスを設ける。このために、付加データ構造としてURLによるIDをテキスト型に設ける。これは、一般のWWWシステムが普通URLでしか組を直接アクセスできないため、仮想的にデータベースの内容をテキスト集合のように見せる必要があるからである。

- **関係表のデータのID（URL）リストを戻すインターフェイス**： これは、ディレクトリのindexの代わりをするプログラムで、実質はselect URL from MYDB 等の単純なリスト出力であるが、HTML形式を出力する必要があるためヘッダが付加される

のと、アンカーをアクセス可能にするために、リストの各項目を

```
<a href='' (URL)''></a>
```

で挟んだ形式で出力する。

- URL を ID（キー）とし、テキストデータを検索するインターフェイス: これは、実際にキーを URL として、検索された関係表のテキスト部分を戻すプログラムである。実際には、このデータを得た検索エンジンが索引を構築する。索引は単語と URL（つまり、関係表のキー属性）の組で構築されるので、このように関係表の ID を URL の形式にして HTTP アクセス可能にする必要がある。†3

ロボットに代表される WWW システムは、最初に ID リストを得るスクリプトを実行し、次に順番にそのリスト中の URL をアクセス（実際にはプログラムが実行され）して、テキストデータを得る。URL と言う付加データ構造はシステムの都合によりできたので、これを維持するための ECA 規則を加える。例えば、

```
Event: insert TABLE
Condition:
Action: append URL field
            and insert tuple
Coupling:immediate
```

（action 部分を含め概要のみ）である。同様に、スキーマ作成段階で URL の属性を生成する規則を設ける。このような規則の定義によって、付加属性の管理を利用者の負担を押えて実現できると考えられる。URL は、

```
http://cite/database/table/bin?ID
```

のような形式で作成し、アクセススクリプトに対して ID 番号を送って個々のデータが得られる形式とする。ここで、テキストデータ型の詳細は検索エンジンが判別し、DBMS 側では構造に関わらない。

†3 関係データベースのアクセスを別のプロトコルによる URL 表現、例えば HTTP でなく eca-sql://data/base/text と言う方法が可能であるが、一般性を失うのでここでは用いない。但し、関係 DB の標準機能と接続可能とするため、jdbc などの標準インターフェイスを用いることは可能

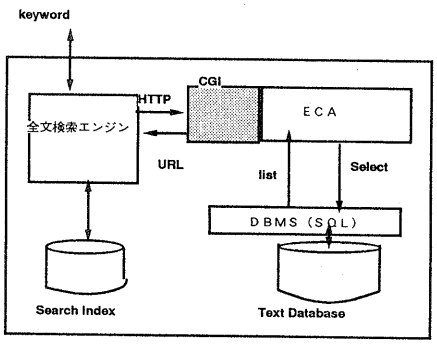

図 5 索引エンジンからの DB の検索

4.2 WWW システムへのデーターベースアクセス

WWW システムのインターフェイスは一般に HTML/HTTP ベースであるが、これらのデータをデータベース管理するには、仮想的なデータベースとして見る必要がある。既に述べたように、全文検索エンジンではキーワードとそれを含むテキストへの URL の組からなる表と考えられる。さて、この場合次の 2 つの利用法がある。

(a) 外部データベースとして見る場合: インターネットの情報とイントラのそれと合わせてデータベースと考えるような場合、外部にある全文検索エンジンとその指すデータは、外部のデータベースと考えられる。

(b) 内部の索引として見る場合： イントラ的な応用で、DBMS そのもののデータに対して全文検索索引を立てたい場合、データベースの実体は DBMS にある。

ここでは、原則的にデータはすべて別のデータ扱いで、先のように、更新の同期をとる ECA 規則を設定することでデータの同一性を実現する。一方で、例えばプロキシサーバの中にあるデータそのものは、HTTP でしかアクセスできない。この場合には、関係データベースの個々の組のアクセスが、関数呼びだしを含んだ HTTP によるデータアクセスに書き変えられる。

5 実装／実験環境と比較

以上のような環境を、サーチエンジンとしてultraseek [2]データベースとして、SQL表現に基づくECA規則を処理できるプロトタイプECA-SQL [10]を用いた実装を行なっている。ECA-SQLのバックエンドには、illustraを用いているが、可能な限りSQL92の機能分だけを用い、拡張機能はECA-SQLの機能で実現するよう考慮されている。ultraseekとの接続と、cgiなどwebのインターフェイスはperlで書かれ、ECA-SQLから呼ばれるスクリプトとして実現される。

実験評価用として、メーリングリストのメールデータを用いた。1件が数キロバイト程度のテキストである。いくつかのサイズに分割し、約6800件／25メガバイド程度から、4万件、150メガ程度のものまで、簡単な性能評価を行なった。httpdはapache, 計算機はss10主記憶32メガバイトである。ネットワーク負荷や最適化機構の影響をさけるため、1000から10000回の処理の平均をとったが、他の理由でのばらつきが大きいためか、詳細な比較とはなっていない。簡単な概要を図6に示すが、簡単には次のようなことが言える。

- テキスト検索は、索引を持たない単純なスキャンよりは速い場合が多い。
- 全文検索エンジンを用いた方が、データ量の増大に対する影響が少ない。なお、このうち純粋に全文検索に用いられる部分は、HTTP経由のアクセスを入れて1.8秒前後。後はDB処理のオーバヘッドである。従って、結果となる答えが多い場合、この数字は悪化する。
- 索引（実験したillustraはテキスト索引型を持たないので、別の索引検索で代行した）よりは遅い可能性が高い。

実現技術による性能への影響の問題もあり、この数字の信頼性は高くないが、概してイントラ的なデータサイズにおいては有効な場合が多いと考えられる。

他の研究と比較すると、次のような特徴が考えられる。

- DataBlade等のプラグイン拡張機能 [7]：本稿では、外部のソフトウェアをそのまま使っている。また、DataBladeでは型固有の索引(dtree)を作れるが、内部化するためHTTPのアクセス等に対応しにくく、索引管理をシステム自身で行なう必要がある。
- 外部索引との結合：専用検索システム [8]とDBMSの結合で、HTTPベースのアクセスを仮定しているものは少ない。また、ECAの枠組みで接続／問い合わせ変換を統一的に行なうものが少ない。

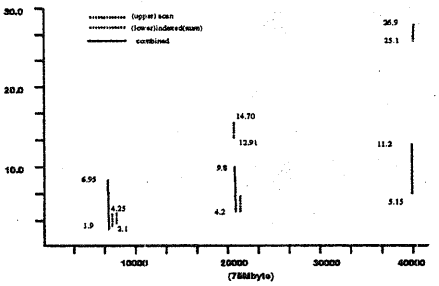

図6 応答時間の比較

ECA規則による問い合わせ変換と、HTTPを意識したデータ構造を用いることで、大規模な全文検索エンジンを変更することなくDBMSと組み合わせて応用を容易に構築できると考えられる。ここではECAを処理できるプロトタイプを用いたが、これを一般の関係DBMSに対するプラグインとして実現する作業も行なっている。

6 応　用

ここでは主として全文検索エンジンとの相互接続について述べたが、一般にこのアーキテクチャに基づいて、次のような応用が考えられている。

メタサーチ： 既に述べたように、WWWシステム側からの戻り値集合は関係集合として扱われる。従って、例えば複数の全文検索エンジンに問い合わせを発行し、その結果を集合演算すれば、メタサーチの実現が可能となる。例えば先の例で、

```
select ID from INFOSEEK
    where text xlike (''keyword'')
```

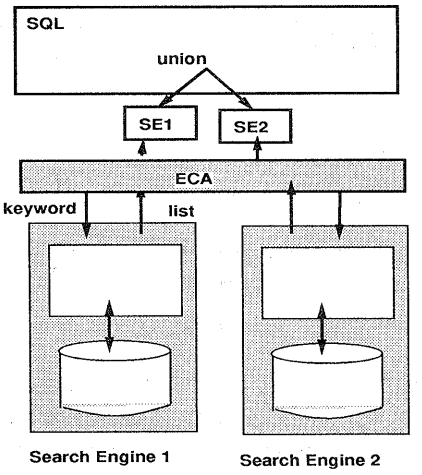

図7　複数の検索エンジンの接続

```
union all
select ID from MITAKE
     where text xlike (''keyword'')
order by 1;
```

は、xlike関数の実体が全文検索エンジンでのキーワードサーチで、かつINFOSEEK、MITAKEと言うデータベース名が外部にあるサーチエンジンのデータを指すよう変換される。従って、2つのサーチの結果を合わせ、重複を抜くことで、一種のメタサーチが実現7できる。

プロキシサーバとの接続：

全文検索エンジンDBMSを接続した場合でも、多くの全文検索エンジンの持つデータベースそのものはWWWロボットによって収集するしか方法がないので、ここでは単にその取り込み先のＵＲＬを更新に応じて通知することしかできない。一方で、ロボットによるデータ収集はインターネットへの負荷が高い。ここでは、DeleGate [11]などのプロキシと組み合わせることを考える。例えばプロキシの段階でのデータ収集と、ここで示すようなDBMSによる検索環境を実現することで、次のような特徴が生じる。

- **ロボットを使わないデータ収集：** ロボットによる積極的なデータ収集は訪れたサイトで問題を起こすことがあり、ネットワークの負荷も大きい。変更がなければそのアクセスは無駄となる。従って、プロキシなどと接続し、そのサイトを通るデータをついでに集めるような、負荷の小さい収集方法を取る。
- **更新変更のコストの安い索引：** 索引構成がバッチ的に行なわれる検索エンジンは、小規模な変更や頻繁な更新に耐えない。そこで、今まで述べたようなECA規則による更新管理を導入することで、更新に対して柔軟で、かつ利用者プログラミングの余地の大きい環境を作る。

ロボットによる動的なデータ収集の一方で、更新コストの大きい索引を維持するより、中継中に集めたりプロキシのキャッシュ等 [12]と統合するなど静的な収集を行なう代わりに、DBMSによる汎用の環境を提供することで更新コストや更新単位を小さくすることができる。

謝辞: 日頃から有益な御議論をいただく情報アーキテクチャ部の諸氏に深謝します。なお、本研究はネットワーク情報ベースラボ（構成：小島功、佐藤 豊、瀬河 浩司）の研究の一環として行なわれている。

参考文献

[1] 小島：”WWW全文検索エンジンとDBMSの相互接続環境について”、データ工学ワークショップ、1998.03.
[2] http://software.infoseek.comr
[3] 丹波、硯、”Alta Vistaにおける大規模検索” ADBS,pp19-26,1996.12
[4] The 6th WWW Conference Proceedings,1997.04.
[5] illustra User's Guide. informix corp.
[6] J.Widom and S.Ceri,”Active Database Systems :Rules and Triggers in Database Systems”, Academic Press, 1995.
[7] illustra Developer's Kit Manuals. informix corp.
[8] Readings in Information Retrieval, Morgan-Kaufmann, 1997.
[9] 小島：”アクティブデータベースシステム” 電総研研究速報 TR95-31,1995.
[10] 小島：” ECAアーキテクチャを支援するためのSQLサーバの拡張” 情報処理学会研究報告,1996.01
[11] http://delegate.etl.go.jp/
[12] http://squid.nlanr.net/Squid/