# World Wide Web ラッパーの問合せ処理における Navigator の利用とその評価

加藤数則[†] 森嶋厚行[††] 北川博之[††]

近年，コンピュータネットワークを通じた異種分散情報源の利用が容易になるにしたがい，それらの統合利用が重要な課題となっている．我々は，WWW，リレーショナルデータベース，構造化文書リポジトリを対象とした異種分散情報源統合利用環境の研究開発を行っている．本統合利用環境は，ラッパーと呼ぶソフトウエアモジュールを通じて各情報源の操作を行う．本稿では，WWW データを扱う Web ラッパーの設計と開発について述べる．特に，Web ラッパーにおける問合せ処理に焦点をあてる．一般に，Web ラッパー自身がすべての WWW ページの処理を行うと，WWW ページなどの転送データ量をはじめとした転送コストが膨大なものとなる．そこで，本稿で述べる問合せ処理方式では，ラッパー処理の一部を担当するエージェントである「Navigator」をコンピュータネットワーク上に分散配置することにより，転送コストを削減し，問合せ処理の効率化を図る．実験により，本方式が，データ転送量，処理時間を削減できることが確認できた．

# Design and Evaluation of a Navigator-based Query Processing Scheme for the World Wide Web Wrapper

KAZUNORI KATOH,[†] ATSUYUKI MORISHIMA[††]
and HIROYUKI KITAGAWA[††]

Integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments. We are developing an information integration environment for the World Wide Web, relational databases, and structured document repositories. In this environment, manipulation of the information sources is performed through software modules called wrappers. In this paper, we describe design and development of the Web wrapper. In general, Web page manipulation may cause very large data transfer cost if all the necessary pages are transferred to the Web wrapper. The proposed Web wrapper architecture uses distributed agents, named navigators. They cooperatively take part of the wrapper's functions at Web server sites, to reduce the cost of Web page transfer. Experimental results show the effectiveness of this approach.

## 1. はじめに

近年，コンピュータネットワークの発展に伴い，分散して存在する各種情報源の利用が容易になり，それらの統合利用が重要な課題となっている[2)7)16)]．特に，World Wide Web （以下 WWW ）の普及により，WWW とデータベース等の各種情報源の統合利用への要求が高まっている．

そこで我々は，WWW，リレーショナルデータベース，構造化文書リポジトリを対象とした統合利用環境の研究開発を行っており[8)11)12)13)17)]，異種情報源統合データモデル WebNR/SD を提案している[11)]．本環境では，WWW ページや構造化文書リポジトリ中の文書が，SGML，XML，HTML 等で記述されていることを想定している．図 1 に本環境のアーキテクチャを示す．この図のように，メディエータ[20)]，ラッパー[18)]と呼ばれるソフトウェアモジュールを用いて統合を行う．このアーキテクチャは，異種情報源の統合への有力なアプローチとして，多くの統合システムで用いられている[3)4)16)18)]．

統合処理の流れは次の通りである．まず，ラッパーが各種情報源を，統合データモデル WebNR/SD に変換する．メディエータはそれらに基づき，ユーザに統合スキーマを提供する．ユーザが視覚的操作系を通じて操作要求をメディエータに与えると，メディエータはそれを

† キヤノン株式会社
Canon Inc.
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

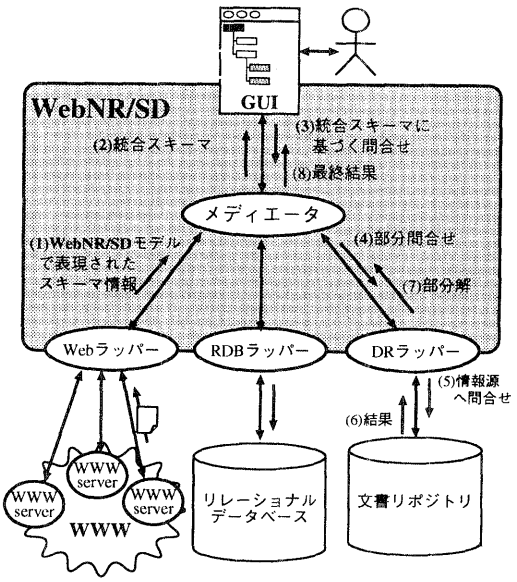



図1 統合利用環境
Fig. 1 Integration environment

分解し，各ラッパーに担当部分を振り分ける．各ラッパーは，それぞれの担当部分を各情報源が理解可能な命令に翻訳し，情報源に投入する．また，ラッパーは，各情報源から受け取った結果をWebNR/SDのデータ表現に変換して，メディエータに転送する．次に，メディエータが各ラッパーからの結果を基に最終結果を作成する．最後に，その結果は視覚的操作系を通じてユーザに示される．

本稿では，本統合利用環境におけるWebラッパーの設計と開発について述べる．Webラッパーは，複数のWWWサーバと通信を行いながら，WWW中のデータに関する演算処理を行うモジュールである．Webラッパーが行う重要な処理の一つに**Navigational Query**の処理がある．Navigational Queryとは，WWWに関する本質的な演算であるNavigation[1)14)]に基づく問合せであり，正規表現や述語として与えられたリンクパターンに沿って，起点のページからハイパーテキストリンクをたどり，特定のパスやページを見つけるという問合せである．多くのWWW問合せ言語がこのNavigational Queryをサポートしている[9)10)15)]．

通常，ラッパーは，各情報源の問合せ処理能力を活用するよう実装される場合が多い[18)]．しかし，Webラッパーの場合は，対象情報源であるWWW（とそのプロトコルであるHTTP）が，基本的にページのフェッチ機構しか提供しないため，Webラッパー自身が各種WWW操作の処理を行う必要がある．例えば，Navigational Queryの処理では，Webラッパーが各

| A | B | | E |
|---|---|---|---|
| | C | D | |
| X | 1 | <a href="http://..">... | <article> |
| | 2 | ... | <title>Web... |
| Y | 3 | ... | ... |
| | 4 | ... | |

図2 WebNR/SDにおけるリレーション構造の例
Fig. 2 Sample relation in WebNR/SD

WWWページを参照することによって，ハイパーテキストリンクをたどる必要がある．しかし，処理に必要な全てのページを，それぞれのWWWサーバからWebラッパーに転送すると，WWWページの転送量は一般に膨大なものとなる．そこで，この転送量を削減し問合せ処理の効率化を図るために，本稿では，Navigatorと呼ぶ分散エージェントを用いた問合せ処理を提案する．我々の知る限り，WWWにおけるNavigational Queryの効率的な処理方式の提案や実験報告は，これまで他の研究では行われていない．

本稿は以下のように構成されている．第2章で，統合データモデルWebNR/SDを説明する．第3章では，まず，Webラッパーの機能概要を示し，次に，Navigatorを用いた問合せ処理について説明する．第4章では，実験環境と実験結果を示し，考察を行う．第5章で，本稿のまとめと今後の課題について述べる．

## 2. WebNR/SD

本章では，統合データモデルWebNR/SDについて，WWWの操作に関連する事項を中心に説明する．より詳細な説明は11)にある．WebNR/SDにおけるデータ構造は，入れ子型リレーションにXML，SGML，HTML文書等の構造化文書を扱うための抽象データ型である「**構造化文書型**」（**SD型**）を導入したものである．また，XML，HTML文書などに現れるリンク構造を扱うため，**Hlink型**と呼ぶデータ型を導入している．

**図2**にWebNR/SDにおけるリレーション構造の例を示す．属性$A$と$C$のドメインはそれぞれ文字列型と整数型である．また，属性$E$のドメインはSD型，属性$D$のドメインはHlink型である．

WebNR/SDでは，通常の入れ子型リレーショナル代数演算子とSD型に付随する文書検索関数群に加え，WebNR/SD固有の演算子である**コンバータ**を用いたデータ操作が可能である．コンバータは，構造化文書と入れ子型リレーション構造の動的かつ部分的な相互変換を実現する．

さらに，WebNR/SDは，WWWデータの操作を

```
memo     = seq(prolog, body)
prolog   = seq(from, opt(homepage), to)
to       = or(faxno, email)
homepage = hlink
body     = rep(para)
```

```
<memo><prolog><from>Lee</from>
<homepage href="http://T.com/lee/home.xml">
His home page</homepage>
<to><faxno>...</faxno></to></prolog>
<body><para> ...</para>
<para> ...</para></body></memo>
```

図3 SD値の例
Fig. 3 Sample SD value

実現する演算子群を用意している．**Navigate**演算子は，Navigational Queryを実現する．**Import**演算子は，WWWページの内容をSD型の値としてリレーションに取り込む．逆に，**Export**演算子は，リレーション内に格納されたSD型の値をもとに新たなWWWページを作成する．

これらの演算子群を組み合わせることにより，WWW，データベース，文書リポジトリに対する統合的な問合せや，再構成操作を実現する．

### 2.1 SD型 / Hlink型

**SD型**の値（**SD値**）は，文書構造を表すDTDと，そのDTDに従ったタグ付きテキストから構成される（図**3**）☆．テキスト中で同じ名前のタグで区切られた部分を**要素**と呼ぶ．各要素は列構造（**seq**），繰返し構造（**rep**），ハイパーテキストリンク構造（**hlink**）等の内部構造を持つ．hlink構造を持つ要素を特に**リンク要素**と呼ぶ．リンク要素は，要素属性hrefを持ち，内部構造（副要素）を持たない．例えば，図3の"`<homepage href="http://T.com/lee/home.xml"> His home page </homepage>`"はリンク要素である．

**Hlink**型はSD型の下位型である．具体的には，リンク要素ただ一つからなるSD値がHlink型の値として定義される．Hlink型の値を**Hlink値**と呼ぶ．

### 2.2 WWWデータ操作のための演算子

**Export/Import**

Export（**E**）は，リレーション中のSD値の内容を持つWWWページをWWW中に作成し，そのページを参照するHlink値を得る．逆に，Import（**I**）は，リレーション中のHlink値が参照するWWW中のページをSD値としてリレーション中に取り込む．以下の実行結果を図**4**に示す．

$$r_2 := \mathbf{E}_{B,U,L,G}(r_1) \quad (1)$$

$$r_1 := \mathbf{I}_{B,U,L,G}(r_2) \quad (2)$$

$\mathbf{E}_{B,U,L,G}(r_1)$は，属性$U$で与えられるURLを持つWWWページを，実際のWWW上に生成する．WWWページの内容は，$r_1$の属性$B$のSD値に対応する．結果リレーション$r_2$の属性$B$には，生成されたWWWページを参照し，かつ$r_1$の属性$L$の文字列値を内部に持つHlink値が格納される．$\mathbf{I}_{B,U,L,G}(r_2)$は逆演算を行う．結果リレーション$r_1$の属性$B$には，$r_2$の属性$B$のHlink値が参照するWWWページの内容を持つSD値が格納される．

$r_1$:

| A | B | U | L | G |
|---|---|---|---|---|
| 1 | ⟨ article = seq(title,body), "<article> <title>Web..."⟩ | http://K.com/i.xml | K corp. | a |

$r_2$:

| A | B |
|---|---|
| 1 | ⟨ a = **hlink**, "<a href="http://K.com/i.xml"> K corp. </a>" ⟩ |

図4 Exportと Importの例
Fig. 4 Examples of Export and Import operations

**Navigate**

Navigate（**N**）は，Navigational Queryを実現する演算子である．具体的には，パラメータで指定された**パス正規表現**（以後，**パス式**と呼ぶ）に基づいてWWWのリンクをたどり，条件を満たすパス集合を求める．パス式において，ハイパーテキストリンクは$\rightarrow$（同一のWWWサーバ上のページ間のリンク）もしくは$\Rightarrow$（異なるWWWサーバ上のページ間のリンク）で表現する．また，WWWページは，文字列（ラベルと呼ぶ）もしくはピリオドで表現する．例えば，パス式$B \rightarrow . \Rightarrow C \rightarrow D$は，次の条件を満たす各パスの集合を表す．すなわち，あるWWWサーバ$X$上のページ$B$から始まり，同一サーバ上のページ，異なるサーバ$Y$（$\neq X$）上のページ$C$を経て，サーバ$Y$上のページ$D$で終了するようなパスである．

パス式には，選択や繰返し構造も記述可能である．例えば，パス式$A(\rightarrow .)* \rightarrow B \mid A \Rightarrow B$で表されるパス集合に含まれるパスとは，(1) ページ$A$で始まり，一つ以上のローカルリンクを経てページ$B$へと到達するパス，もしくは，(2) ページ$A$で始まって，一つのグローバルリンクを経てページ$B$へと到達するパス，のいずれかである．また，"$*/n_1..n_2/$"が略記法として利用可能である．例えば，$A(\rightarrow .)*/2..3/ \rightarrow B$は，$A \rightarrow . \rightarrow . \rightarrow B \mid A \rightarrow . \rightarrow . \rightarrow . \rightarrow B$の略記である．

パス式では，ハイパーテキストリンク構造の指定の他にも，WWWページが，指定した語を含むという語包

☆ 適正形式のXML文書やHTML文書に対応するSD値の場合には，DTDがNULLになる．

```
path_reg_expr ::= label p_expr {'|' label p_expr}
p_expr        ::= link page [ '[' condition ']' ]
                | p_expr { p_expr }
                | p_expr { '|' p_expr }
                | p_expr '*' ['/'num'..'num'/']
                | '(' p_expr ')'
link          ::= '->' | '=>'
page          ::= label | '.'
```

図5 パス正規表現の構文規則
Fig. 5 Path regular expression syntax

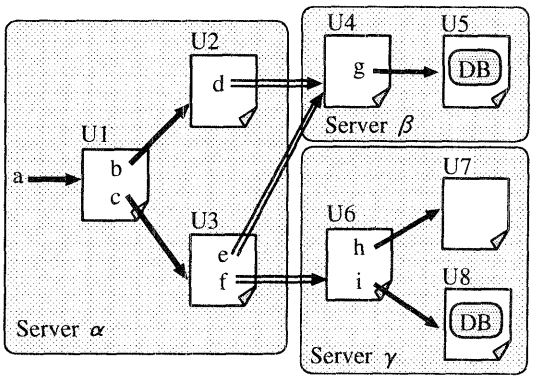



図6 ハイパーテキストリンク構造の例
Fig. 6 Example hyper-text link structure

含条件を記述可能である☆1．語包含条件は，各ラベルもしくはピリオドの直後に記述する．図5にパス式の構文規則を示す☆2．

ハイパーテキストリンク構造が図6のようになっていると仮定する．この図で$Ui$はURL，$\alpha$，$\beta$，$\gamma$はWWWサーバ名，小文字のアルファベットはリンク要素を表す．このとき，式(3)の結果を図7に示す．

$$r_4 := \mathbf{N}_{A(\rightarrow.)*\rightarrow.\Rightarrow B(\rightarrow.)*\rightarrow C[''DB''],D}(r_3) \quad (3)$$

パス式中の各ラベルは，WWWページとリレーション属性を対応づけている．Navigate演算子の第2パラメータ（ここでは$D$）は，結果リレーションの属性$D$が，パス式が表すパス集合を格納することを表す．

図7からも分かる通り，次のような結果が得られる．

(1) 各ラベルに対応する属性には，各ページの内容ではなく，そのページを参照するHlink値が格納される．

(2) ピリオドで表現されるWWWページを参照するHlink値は，属性$D$の副リレーションには格納されない．

Navigate演算子におけるパス式は，WebSQL[15]のパス正規表現と本質的に同じものである☆3．

$r_3$:

| ID | A |
|---|---|
| 1 | ⟨ a=hlink, "<a href="U1">a</a>" ⟩ |

$r_4$:

| ID | A | D | |
|---|---|---|---|
| | | B | C |
| 1 | ⟨ a=hlink, "<a href="U1">a</a>" ⟩ | ⟨ a=hlink, "<a href="U4">d</a>" ⟩ | ⟨ a=hlink, "<a href="U5">g</a>" ⟩ |
| | | ⟨ a=hlink, "<a href="U4">e</a>" ⟩ | ⟨ a=hlink, "<a href="U5">g</a>" ⟩ |
| | | ⟨ a=hlink, "<a href="U6">f</a>" ⟩ | ⟨ a=hlink, "<a href="U8">i</a>" ⟩ |

図7 Navigate演算の例
Fig. 7 Example of Navigate operation

## 3. Webラッパー

### 3.1 機能概要

本統合利用環境では，Webラッパーは，メディエータと同じマシン上に一つだけ存在し，複数のWWWサーバとの通信を行う．Webラッパーの機能は以下の通りである．

(1) 統合スキーマにおけるWWWのリレーションビューの提供：本統合利用環境における統合スキーマでは，WWWはHlink型の属性を持つ単項リレーションの集まりとしてモデル化される．例えば，WWWブラウザのブックマーク等が単項リレーションに対応づけられる．これらのリレーション中のHlink値は，Import，Navigate演算などで必要なWWWページを得るための起点ページとして，用いられる．

(2) Export，Import，Navigate演算子の処理：メディエータからの要求に従い，これらの演算子を実行する☆4．

以下では，Navigate演算の処理に焦点をおいて説明

☆1 さらに，選択条件としてリージョン代数式を用いた指定が可能であるが，本稿では省略する．

☆2 パス式は，この構文に従う他にも，次の条件を満たす必要がある．すなわち，パス式が受理可能な全てのパスについて，パス式に現れる全てのラベルが束縛されなければならない．例えば，パス式 $A \rightarrow B \mid A \rightarrow C$ は，この条件を満たしていない．

☆3 WebSQLでは，内部リンク（同じページ内のリンク）を区別している一方，ページの内容に関する選択条件の指定は許していないという違いがある．しかし，これらの違いは本論文の議論には影響しない．

☆4 メディエータ内で実体化されたリレーションに対して，これらの演算が適用された場合には，リレーションのタプル毎にWebラッパーの関数が呼び出され，処理が行われる．

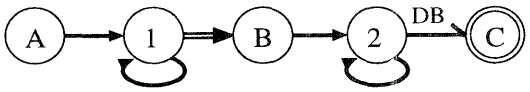



図8 パス式 $A(\rightarrow .)* \rightarrow . \Rightarrow B(\rightarrow .)* \rightarrow C[''DB'']$ により作成されたオートマトン
Fig. 8 Automaton corresponding to $A(\rightarrow .)* \rightarrow . \Rightarrow B(\rightarrow .)* \rightarrow C[''DB'']$

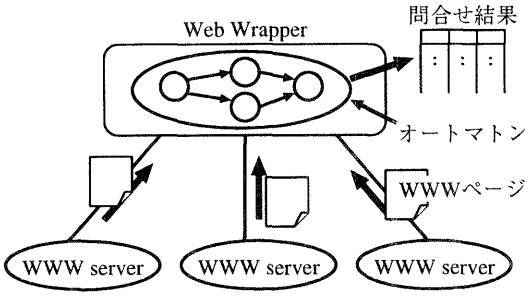



図9 集中型処理方式
Fig. 9 Centralized Query Processing Scheme

を行う．

Navigate演算の最も基本的な処理方式は，問合せ処理の全てをWebラッパー自身で行うというものである．本稿では，この方式を集中型処理方式と呼ぶ．まず，集中型処理方式とその問題点を示し，続いて，Navigatorを用いた処理方式について説明する．

### 3.2 Navigate演算の集中型処理方式

この方式では，まず，パス式に基づいてWebラッパー内にオートマトンを作成する．その後，WWWサーバからWebラッパーに転送されたWWWページ群を解析しながら，そのオートマトンに受理されるパスの集合を求める．

図8は，パス式 $A(\rightarrow .)* \rightarrow . \Rightarrow B(\rightarrow .)* \rightarrow C[''DB'']$ により作成されたオートマトンである．各状態には，文字列もしくは数値が結びつけられている．文字列は，パス式中のラベルに対応する．状態遷移は(1) ローカルリンク，(2) グローバルリンク，(3) 語の包含という3種類の条件によって行われる（それぞれ $\rightarrow$，$\Rightarrow$，$\overset{term}{\rightharpoonup}$ によって表される）．これらの解釈は，パス式における場合と同じである．

集中型処理方式においては，条件を満たす可能性のあるパス上の全てのWWWページを，WWWサーバからWebラッパーに転送する必要がある（図9）．この方式の問題点は，探索対象となる全てのページの内容をWebラッパーに転送するため，ページ転送コストが膨大となり効率的ではないことである．

### 3.3 Navigatorを用いた処理方式

ページ転送コストの問題に対処するため，分散エージェントを用いた処理方式を提案する．このエージェントを**Navigator**と呼ぶ．Navigatorは，各WWWサーバマシン上に分散配置され，自分が配置されたWWWサーバの範囲において（部分的な）パスの判定を行い部分解リレーションを生成し，それをWebラッパーに転送する．Webラッパーは，それら部分解を結合し，最終的なパスの集合を作成する（図**10**）．

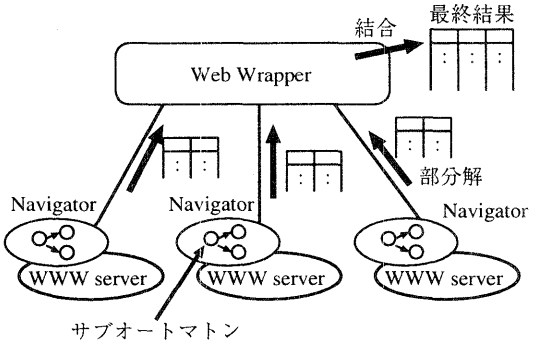



図10 Navigatorの利用
Fig. 10 Utilization of navigators

集中型処理方式と異なり，Webラッパーへは，結果のパス集合の部分解のみを転送すればよい．したがって，情報の転送コストが削減される．

Navigatorを用いた処理方式を実現するためには，Webラッパーは，パス式より作成されたオートマトンを，複数のサブオートマトンに分割する必要がある．各サブオートマトンは，それぞれ特定のWWWサーバの範囲内での部分解を求めるためのものである．これらは，各Navigatorに組み込まれる．

**オートマトンの分割法**

パス式全体を表すオートマトンを $X$，分割の結果得られるサブオートマトンを $X_i$ とそれぞれ表す．また，$I$，$F$ を次のように定義する．

- $I \equiv \{IS(X)\} \cup GS(X)$．
  ここで，(1) $IS(X)$ は，$X$ の初期状態，(2) $GS(X)$ は，$X$ 内でグローバルリンク（$\Rightarrow$）の遷移先となる状態の集合，である．$X$ を図8のオートマトンとすると，$IS(X) = A$，$GS(X) = \{B\}$，$I = \{A, B\}$ となる．$I$ 中の各状態は，分割後の各サブオートマトンの初期状態として用いられる．
- $F \equiv GS(X) \cup AS(X)$．
  ここで，$AS(X)$ は，$X$ の受理状態の集合である．$X$ を図8のオートマトンとすると，$AS(X) = \{C\}$，$F = \{B, C\}$ となる．$F$ 中の各状態は，分割後のオートマトンの受理状態として用いられる．

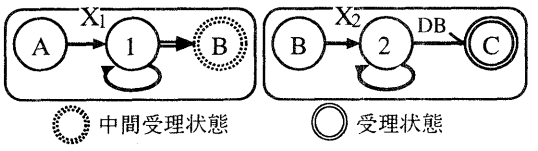



図11 分割されたオートマトン
Fig. 11 Decomposed automatons

このとき，各 $X_i$ はそれぞれ次の条件を共に満たすオートマトンである．

(1) 初期状態（$IS(X_i)$ と表記）が $I$ 中のいずれかの状態である．

(2) $X$ 中の $IS(X_i)$ から到達可能な状態のうち，$F$ 中の状態を越えないで到達可能な全ての状態を持つ（すなわち，$X_i$ の受理状態は，必ず $F$ に含まれることになる）．

**図11** は，図8のオートマトンを，分割した結果である．$X_1$ と $X_2$ の二つのサブオートマトンが得られる．$X_i$ 内の受理状態が $AS(X)$ に属さないとき，その受理状態を**中間受理状態**（intermediate accepting state）と呼ぶ．また，$IS(X)$ をもつオートマトンを**ルートオートマトン**（root automaton）と呼ぶ．

**Navigatorの配置，処理手順**

Navigatorは次の手順で動的に配置される．まず，Webラッパーは，Navigate演算の実行指示を受けると，パス式からオートマトンを作成し，サブオートマトンに分割する．次に，ルートオートマトンを持つNavigator（N1とする）を生成する．さらに，そのN1を，Navigate演算の起点のページ（起点のHlink値が指すページ）を管理するWWWサーバ（$\omega$ とする）と同じマシンに配置する．

Navigator N1は，起点のHlink値から探索を始める☆．探索中，N1のオートマトンに受理されるパス（すなわち，WWWサーバ $\omega$ の範囲のパス）を発見すると，まず，それを部分解リレーションに格納する．次に，発見されたパスの受理状態の種類によって，次の2つのうち，いずれかの処理を行う．

**(A)** パスの受理状態が「中間」受理状態（$s$ とする）の場合：これは，$\omega$ とは異なるWWWサーバ（$\psi$ とする）をさらに探索する必要があることを示している．なぜなら，中間受理状態に対する遷移は，必ずグローバルリンクによって生じるからである．この場合，Navigator N1は，(1) ただちにこのことをWebラッパーに報告する．そして，(2) さらにサーバ $\omega$ 内の他のパスの探索を続行する．一方，Webラッパーは，Navigator N1からの報告に基づいて $s$ を初期状態とするサブオートマトンを持つ別のNavigatorを生成し，WWWサーバ $\psi$ と同じマシンに配置する．この時，生成しようとするNavigatorと同じオートマトンを持つものが既にそのマシンに存在する場合は，新たなNavigatorは配置せず，そのNavigatorを再利用する．続いて，その新しい（もしくは再利用された）Navigator（N2とする）が，探索の起点となるHlink値をWebラッパーから受け取り，Navigator N1と同様の処理を行う．この時，Navigator N1は，すでに他のパスの探索を始めているので，Navigator N1とN2が並列処理を行うことになる．

**(B)** パスの受理状態が，中間でない真の受理状態の場合：これは，全体のパス式を満たすパスが見つかったことを示す．この場合，Navigator N1は，単にwebサーバ $\omega$ 内の他のパスの探索を続行する．それ以外の処理は行なわない．

以上の処理が終了するのは，全てのNavigatorについて，たどるべきハイパーテキストリンクがなくなった時である．具体的な処理は次の通りである．まず，各Navigatorは，探索過程 において，たどるべきハイパーテキストリンクがなくなると，Webラッパーに対して，メッセージ “waiting” を送信する．これは，そのNavigatorが，「現在は 探索すべきパスがないため，待機状態である」ことを表す．ただし，待機状態のNavigatorに対して，Navigator N2の説明で行ったように，Webラッパーから新たにたどるべきパスの起点のHlink値が送られてくる可能性がある．その場合，Webラッパーは，そのNavigatorが再び “waiting” を送信するまで，そのNavigatorの待機状態は解除されたとみなす．Webラッパーは，全てのNavigatorが待機状態であることを確認して初めて，Navigate演算の修了処理を行う．

以上の処理過程において，各Navigatorの内部では，サブオートマトン中の全ての遷移に対して，どのWWWページ間でその遷移が生じたのかという情報を記録しておく☆☆．例えば，図11のサブオートマトン $X_1$ を持つNavigatorでは，$A \rightarrow 1, 1 \rightarrow 1, 1 \Rightarrow B$ という3つの遷移に対して，その情報を保持している．そして，新たな遷移が生じるたびに，その記録との比較を行う．もし，以前に全く同じページの組合せで同じ遷移が生じて

☆ 本稿で提案する処理手順は，各Navigatorが担当するWWWサーバ内の探索手順とは独立である．WWWサーバ内の探索は，例えば深さ優先探索等でよい．ただし，処理時間は，この探索手順に影響される．

☆☆ 具体的には，リンク要素が記録される．

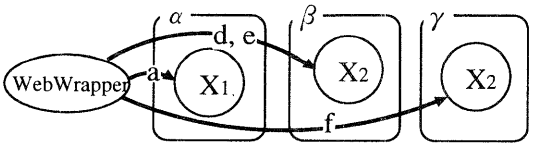



図12 Navigator の生成
Fig. 12 Creation of navigators

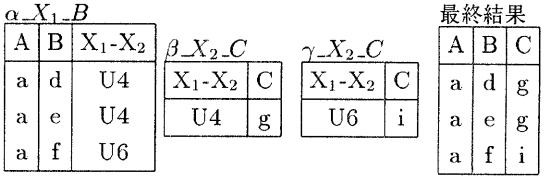



図13 Navigator により作成される表と最終結果
Fig. 13 Relations managed by navigators, and the final result

いた場合は，そのパスについては，そこから先の探索を打ち切る．これにより，WWW のリンク構造にサイクルが存在するような場合でも，静的 Web[14] に対しては，この手順は必ず終了することが保証される☆．

各 Navigator の部分解リレーションは，この遷移記録情報を利用して作成される．部分解リレーションの具体的な例は次の処理例に示す．

処理例

処理の流れを具体例に沿って説明する．$\mathbf{N}_{A(\rightarrow.)*\rightarrow.\Rightarrow B(\rightarrow.)*\rightarrow C[''DB''],D}(r_3)$ の処理を例にとる（$r_3$ は図7のものを利用する）．この時，分割して得られるサブオートマトンは，図11のサブオートマトン $X_1$ と $X_2$ である．$r_3$ の場合，起点となる Hlink 値は，URL が $U1$ の WWW ページを参照している．WWW ページのリンク構造は図6のようになっているものと仮定する．

**図12** は，この例における，Navigator の生成と，起点となる Hlink 値の受渡しを示したものである．図中の円は Navigator を示す．また，その中の $X_1$，$X_2$ は Navigator が持つサブオートマトンを示す．

具体的には，次のように処理が行われる．まず，Web ラッパーがルートオートマトン $X_1$ を持つ Navigator を生成し，WWW サーバ $\alpha$ と同じ計算機に配置する．その Navigator は，WWW サーバ $\alpha$ の範囲内で，Hlink 値 $a$ から始まるパスを探索する．その結果，Hlink 値 $d$，$e$，$f$ が指す先のパスを，オートマトン $X_2$ によって探索しなければならないことがわかる．ここで，$d$ と $e$ は WWW サーバ $\beta$ 内の WWW ページを参照しており，$f$ は WWW サーバ $\gamma$ 内の WWW ページを参照している．Web ラッパーはサブオートマトン $X_2$ を持つ Navigator を WWW サーバ $\beta$, $\gamma$ の両方に割り当て，それぞれの起点となる Hlink 値を渡す．これらの Navigator は並列に動作し，それぞれがさらにパスを探索する．

先に触れた通り，各 Navigator は，遷移の記録情報を基に，部分解リレーションを作成する．これは，単一の WWW サーバの範囲に関するサブオートマトンで受理されたパスを格納するためのリレーションである．このリレーションは，各 Navigator が持つサブオートマトンの各受理状態毎に作成される．したがって，一般には各 Navigator は複数の部分解リレーションを持つ．**図13** は上の例で作成されるリレーションである．例えば，リレーション $\alpha\_X_1\_B$ は，WWW サーバ $\alpha$ に割り当てられた，オートマトン $X_1$ を持つ Navigator により管理されており，中間受理状態 $B$ に達したパスの集合を格納する．他のリレーション（$\beta\_X_2\_C$ と $\gamma\_X_2\_C$）も同様である．

これらのリレーションの属性名には次の2種類がある．(1) パス式中のラベルに対応するもの．(2) 結合属性（この例では $X_1$-$X_2$）．(1) の属性はラベル付けされた状態への遷移を引き起こした Hlink 値を格納する．(2) の結合属性は，それがリレーションの右端にある場合には，中間受理状態への遷移を引き起こした Hlink 値が指すページの URL を格納する．一方，リレーションの左端にある場合は，Navigator がパスを調べるために利用した起点のページの URL を格納する．最終結果（全体のパス式を満たすパスの集合）は，これらのリレーションを結合することにより求められる．

## 4. 実　　験

Navigate 演算の集中型処理方式と，Navigator を用いた処理方式を，データ転送量と処理時間の観点から比較するための実験を行った．Web ラッパーおよび Navigator は Java オブジェクトとして実装した．各 Navigator は，担当する Web サーバ内を深さ優先で探索するよう実装した．Java 仮想機械は JDK 1.1.5 を用いた．また，WWW サーバとして Apache 1.2.6 を使用した．WWW サーバと，Web ラッパーもしくは Navigator の間の通信は HTTP に基づいて行った．一方，Web ラッパーと Navigator 間の通信は HORB 1.3.b1[5)6)] を利用した．

☆ 現実の WWW は動的 Web であるので，この手順が終了しない可能性がある．そのような場合，時間や資源消費の制限などを指定し，処理を途中で打ち切る必要がある．

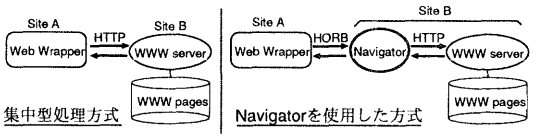



図 14 実験で使用した方式
Fig. 14 Experiment schemes

### 4.1 実験 A：単一の WWW サーバ中のデータを対象とした場合

#### 4.1.1 実験方法

単一の WWW サーバ中の WWW ページを検索対象として，2つの処理方式の比較を行った（**図 14**）．データ転送量は，集中型処理方式の場合は，WWW サーバと Web ラッパーの間で送られたパケットを計測した．Navigator を利用した方式の場合は，Navigator と Web ラッパーの間で送られたパケットを計測した．また，処理時間は Web ラッパーが Navigate 処理を始めてから終了するまでの時間を計測した．

検索対象の WWW ページのリンク構造は，**図 15** に示すように，3段の木構造をしている．ルートページの下に中間ノードページがあり，各中間ノードページはそれぞれ 49 個のリーフページへのリンクを持つ．末端のリーフページはリンクを持たない．ルートページから中間ノードページへのリンクの個数を変えることにより，検索対象となるページ数を 100，200，400，800，1200，1600，2000 と変えた 7 種類のデータを用意した☆．各ページサイズは 8Kbyte である．

上記データに対する Navigate 処理を行う際のパス式としては，次のものを使用した．

$$A(\rightarrow .)* \rightarrow B$$

このパス式では，起点のページ A からローカルリンクだけをたどることによって，到達できる全てのページへのパスの集合が，検索結果となる．

実験には，神奈川工科大学と，筑波大学に配置された2台の計算機（C1，C2）を使用した（**図 16**）．

C1，C2 のメモリ量の違いによる影響を考慮し，図 14 での役割をそれぞれ次のように割り当てた場合について，同じ実験を行った．

**(a)** SiteA：C1（神奈川工科大），SiteB：C2（筑波大学）

**(b)** SiteA：C2（筑波大学），SiteB：C1（神奈川工科大）

処理時間に関する実験結果の値は，これら2つの場合の平均である．

☆ ルートページ以外のページが，検索対象ページとなる．

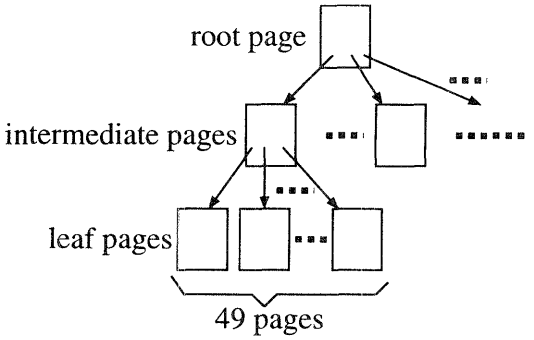



図 15 実験 A：実験データ（WWW ページ）の構造
Fig. 15 Experiment A: Structure of experiment data

| C1（神奈川工科大） | C2（筑波大） |
|---|---|
| **CPU** UltraSPARC 143MHz<br>**メモリ** 32MB<br>**OS** Solaris 2.6 | **CPU** UltraSPARC 143MHz<br>**メモリ** 64MB<br>**OS** Solaris 2.6 |

図 16 実験環境
Fig. 16 Experiment environment

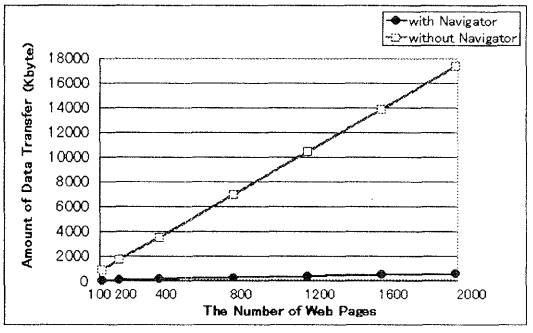



図 17 実験 A: データ転送量
Fig. 17 Experiment A: Data transfer cost

#### 4.1.2 実験結果

**(1) データ転送量**

結果を**図 17** に示す．Navigator を利用することにより，データ転送量を大幅に削減できることが確認できた．データ転送量は，Navigator を利用した場合には，集中型処理方式の場合の約 3.4 %になった．

**(2) 処理時間**

ネットワークのトラフィック状況の変化を考慮し，平日昼間と深夜の両者に対して測定を行った．

**平日昼間**

検索ページ数に関わらず，Navigator を用いた方が，常に処理時間が短い．処理時間の短縮率の平均は 30 %となった（**図 18**）．

**深夜**

深夜における短縮率の平均は 22 %であり，平日昼間

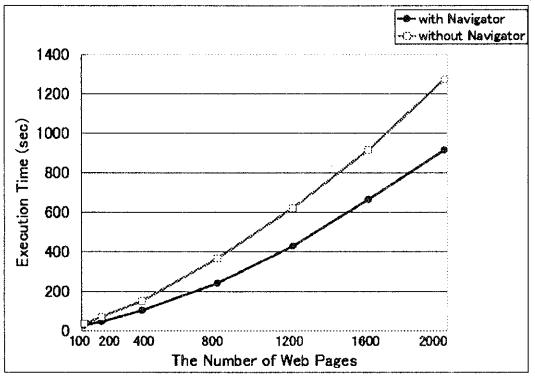



図 18 実験 A: 処理時間 (平日昼間)
Fig. 18 Experiment A: Execution time (in the daytime on weekday)

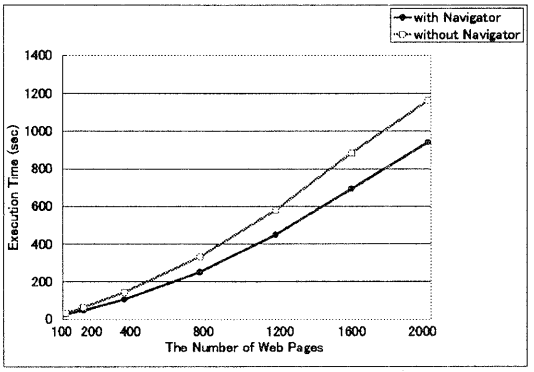



図 19 実験 A: 処理時間 (深夜)
Fig. 19 Experiment A: Execution time (at midnight on weekday)

の 30 %よりも小さくなっている (**図 19**). この理由は, 実験を行った時刻が深夜でありネットワークのロードが小さかったため, 集中型処理方式のデータ転送時間が短縮された一方で, Navigator を用いた場合の処理時間は, ネットワークのトラフィック状況にほとんど影響されなかったためであると考えられる.

**昼夜の平均**

**図 20** は, 図 18, 図 19 の平均をとったものである. 短縮率の平均は 26 %となった.

### 4.2 実験 B: 単一の WWW サーバ中のデータを対象とし, 検索ページ数を少なくした場合

#### 4.2.1 実験方法

単一 WWW サーバ中の WWW ページを検索対象とし, 検索ページ数を少なくした場合の処理時間を計測した.

実験対象の WWW ページのリンク構造を**図 21** に示す. 総ページ数が 1, 3, 5, 10 の 4 種類のデータを用意した. 実験環境は実験 A と同じである. また, パス式

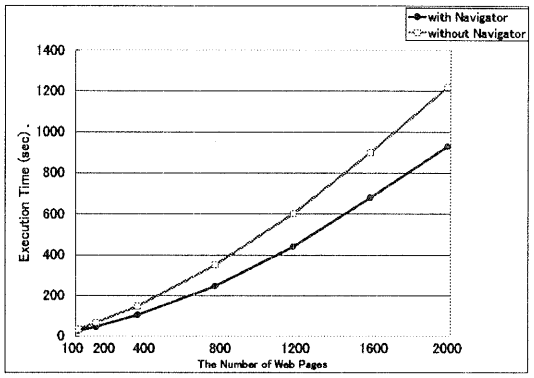



図 20 実験 A: 処理時間 (昼夜平均)
Fig. 20 Experiment A: Average of daytime and midnight results

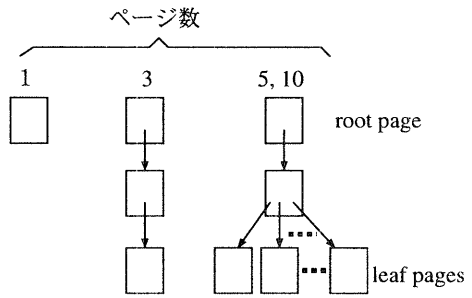



図 21 実験 B: 実験データ (WWW ページ) の構造
Fig. 21 Experiment B: Structure of experiment data

も同じものを使用する.

#### 4.2.2 実験結果

**図 22** に実験結果を示す. 実験結果は昼夜の処理時間の平均を示している. 検索ページ数が 5 ページ以下の場合には, 集中型処理方式の方が処理時間が短くなっている. これは, Navigator の生成や配置にかかる時間が検索ページ数に関わらず一定であるため, 検索ページ数が少ない時には, ページ転送時間の削減が Navigator の生成や配置による処理時間の増加よりも小さくなることによると考えられる.

### 4.3 実験 C: 複数の WWW サーバ中のデータを対象とした場合

#### 4.3.1 実験方法

複数の WWW サーバを検索対象とした場合の処理時間についても計測を行った.

検索対象の WWW ページのリンク構造を**図 23** に示す. まず, 実験 A のデータと同じ 3 段の木構造を持つ総ページ数 1001 のデータを 3 セット用意し, それらを異なる計算機上の WWW サーバに配置した (Server B, C, D). さらに, もう一つの WWW サーバ (Server A) を用意し, その WWW サーバ上に Navigate 演算

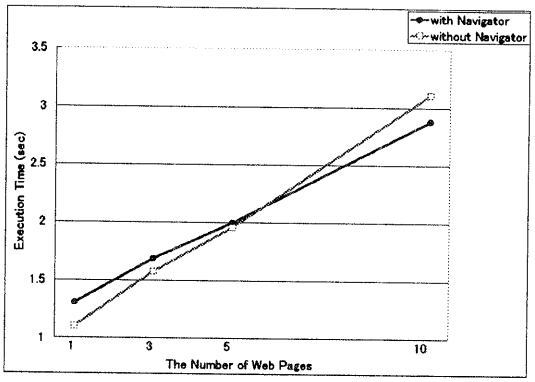



図 22 実験 B: 処理時間
Fig. 22 Experiment B: Execution time

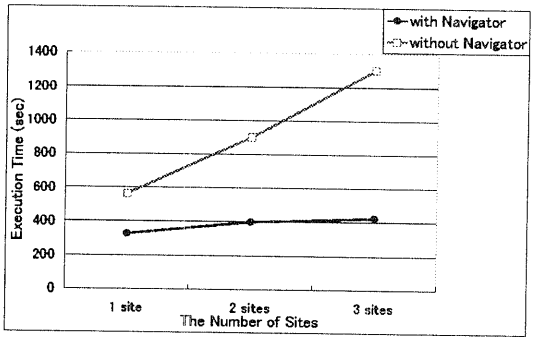



図 24 実験 C: 処理時間 (平日昼間)
Fig. 24 Experiment C: Execution time (in the daytime on weekday)

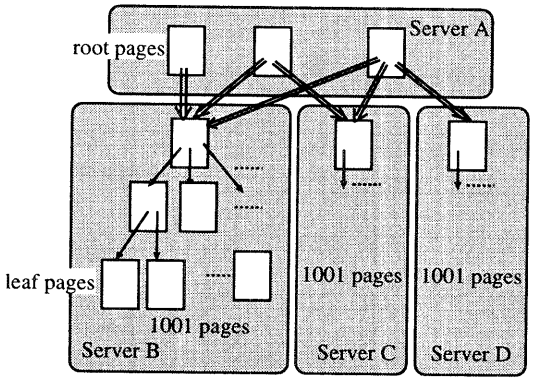



図 23 実験 C: 実験データ (WWW ページ) の構造
Fig. 23 Experiment C: Structure of experiment data

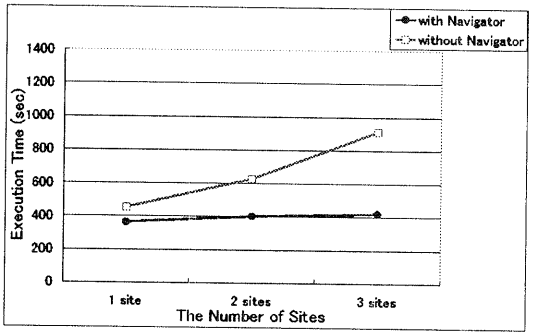



図 25 実験 C: 処理時間 (深夜)
Fig. 25 Experiment C: Execution time (at midnight on weekday )

の起点となるルートページを 3 つ用意した．これらの 3 つのルートページからは先の 3 セットのページ群のそれぞれ，1 つ，2 つ，3 つに対してリンクを張る．したがって，それらのルートページから検索可能なページ数（ルートページを除いた数）は，それぞれ 1001，2002，3003 となる．

実験には，神奈川工科大学で 1 台（C1），筑波大学で 4 台（C2，C3，C4，C5）の計算機を使用した．C3，C4，C5 は，いずれも CPU が UltraSparc 167MHz，メモリが 128Mbyte のマシンである．筑波大学の計算機には，それぞれ一つずつ，合計 4 つの WWW サーバを起動し，WWW データを配置した（Server A，B，C，D を，それぞれ C2，C3，C4，C5 で起動）．一方，神奈川工科大学の計算機上には Web ラッパーを配置した．

Navigate 処理を行う際のパス式としては，以下のものを使用した．

$$A \Rightarrow .(\rightarrow .)* \rightarrow B$$

このパス式では起点のページ A からグローバルリンクを一つたどり，さらに一つ以上のローカルリンクをたどって到達できる全てのページへのパスの集合が検索結果となる．

### 4.3.2 実験結果

**処理時間**

実験 A，B と同様に平日昼間と深夜に実験を行った．

平日昼間の場合，処理時間の短縮率は，平均で 55 %，最大で 77 %（3 サイトの場合）となった（**図 24**）．

深夜の場合は，処理時間の短縮率は，平均で 42 %，最大で 55 %となった（**図 25**）．

**図 26** は，平日昼間と深夜の平均をとったものである．処理時間の短縮率は，平均で 49 %，最大で 62 %となった．

実験 A の場合と同様に，集中型処理方式の処理時間はネットワークのトラフィックの状況に大きく影響されるが，Navigator を用いた場合には，ネットワークのトラフィックの状況にはほとんど影響を受けないことが分かる．

また，実験データが複数の WWW サーバ（サイト）にまたがっている場合には，Navigator が複数の計算機上に分散配置されるため，それらの Navigator 間で

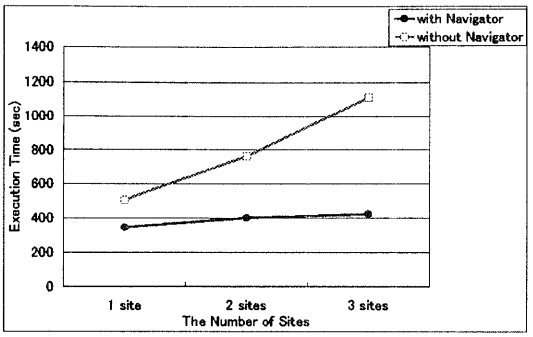



**図 26** 実験 C: 処理時間 (昼夜平均)
Fig. 26 Experiment C: Average of daytime and midnight results

は並列処理が行われる．このため，単一の WWW サーバ (1 サイト) を対象とした場合に比べ，処理時間の短縮率が大幅に向上している．ただし，WWW サイト数が増えるに従い，Navigator を用いた方式の処理時間もやや増大する傾向が確認された．理由としては，次のようなことが考えられる．

(1) Web ラッパーでは各 Navigator からの部分解リレーションを結合し最終結果を作成しているが，サイトが増えると一般に結合のためのタプル比較回数が増えるため，部分解リレーションの結合の処理時間が増大する．なお，現在の実装では，部分結果の結合処理はメモリ上での単純な入れ子ループ結合で行われている．したがって，タプル数 $M$ のリレーション $R$ とタプル数 $N$ のリレーション $S$ の結合は $M \times N$ のオーダの処理時間を要する．実験 C において結合処理にかかっている時間を計測すると，1 サイトで 4.4 秒，2 サイトで 17.4 秒，3 サイトで 37.6 秒であった．

(2) 各 Navigator はオートマトンに受理されるパスを見つけると，それを Web ラッパーに報告するが，このパスの報告をするための Web ラッパーのメソッドは，同期をとるため排他処理を行っている．このため，Navigator が複数存在する場合には，その報告を待たされる場合がある．

(1) については改善の余地がある．その理由は，Navigator によって作成される部分解リレーションが次の性質を持つからである．すなわち，部分解リレーションの最も左の属性 (結合属性になる) には，その Navigator が割り当てられている WWW サーバのページを指す URL 以外は現れない．したがって，例えば，部分解リレーションの結合 $R \bowtie_{A=B} S$ を最適化して，$\sigma_{LinkTo(A,S)}(R) \bowtie_{A=B} S$ とする事が可能である．ここで，$LinkTo(A,S)$ は，$A$ の値が $S$ の WWW サーバが管理するページへのリンク要素である時に成立する述語である．特に複数の WWW サーバを対象とした Navigational Query の場合には，$\sigma_{LinkTo(A,S)}(R)$ のタプル数が，$R$ のタプル数 $M$ よりも大幅に小さくなる可能性がある．

## 5. おわりに

本稿では，WWW，リレーショナルデータベース，構造化文書リポジトリを対象とした異種情報源統合利用環境における Web ラッパーの設計について述べた．特に，Navigator と呼ぶ分散エージェントを用いた WWW ページ転送量の削減方式を提案し，実験結果の報告を行った．本実験環境では，HORB を用いて Navigator をリモート生成する機構を実装したが，他のエージェント生成機構を用いた場合にも同じ議論が成立する．すでに，我々は，Servlet[19] を用いた Navigator の生成機構も実装している．

実験結果により，Navigator を用いた問合せ処理方式が，データ転送量を大幅に削減し，また，多くの場合において，処理時間も短縮することを示した．さらに，複数の WWW サーバにまたがるデータを処理する場合には，複数の Navigator による並列処理が可能となり，処理時間を大幅に短縮できることを示した．

集中型処理方式と Navigator を用いた処理方式の処理時間の差は，ネットワークの負荷やページ数等によって大きく影響を受け，場合によっては集中型の方が速いこともある．したがって，今後の課題としては，ネットワークの混み具合などから Navigator の使用と不使用を動的に判断する仕組みの開発があげられる．

今回の実験により，複数の Navigator による並列処理が有効であることが分かった．これをさらに有効に活かすためには，各 Navigator 内のパス式評価順序の最適化手法を開発する必要がある．例えば，各 Navigator が，グローバルリンクを優先に探索すれば，並列性が増し，処理時間が向上できると考えられる．

また，今回の実験では比較的単純な構造をした WWW データを用いたが，より複雑なリンク構造を持つデータや，実データに対する実験評価も今後の課題である．

**謝辞** 評価実験にあたり御協力頂いた神奈川工科大学助手鈴木孝幸氏に深謝いたします．本研究の一部は，文部省科学研究費特定領域研究「高度データベース」(08244101)，基盤研究 (C) (09680321)，特別研究員奨励費，ならびに電気通信普及財団の助成による．

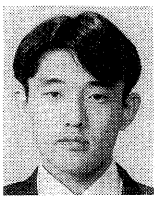
## 参考文献

1) S. Abiteboul and V. Vianu, "Queries and Computation on the Web", *Proc. 6th*

*International Conference on Data Theory (ICDT'97)*, pp 262–275, 1997.
2) A. K. Elmagarmid and C. Pu, (eds.), "Special Issue on Heterogeneous Databases", *ACM Computing Survey*, vol. 22, no. 3, 1990.
3) Mary F. Fernandez and Daniela Florescu and Jaewwoo Kang and Alon Y. Levy and Dan Suciu, "STRUDEL: A Web-site Management System", *Proc. SIGMOD Conf.*, pp. 549–552, Tucson, May 1997.
4) M. R. Genesereth, A. M. Keller, O. M. Duschka, "Infomaster: An Information Integration System", *Proc. SIGMOD Conf.*, pp. 539–542, 1997.
5) S. Hirano, "HORB Home Page", `http://ring.etl.go.jp/openlab/horb-j/WELCOME.HTM`.
6) S. Hirano, "HORB: Distributed Execution of Java Programs", *Worldwide Computing and It's Applications '97 (WWCA'97)*, pp. 29–42, Tsukuba Japan, Mar. 1997.
7) A. R. Hurson, M. W. Bright, and S. Pakzad, (eds.), "Multidatabase Systems: An Advanced Solution for Global Information Sharing", *IEEE Computer Society Press*, 1994.
8) K. Katoh, A. Morishima and H. Kitagawa, "Navigator-based Query Processing in the World Wide Web Wrapper", *Proc. 5th Intl. Conf. on Foundations of Data Organization '98 (FODO'98)* pp. 191–199, Kobe, Japan, Nov. 1998.
9) D. Konopnicki and O. Shmueli, "W3QS: A Query System for the World-Wide Web", *Proc. VLDB Conf.*, pp. 54–65, 1995.
10) L. Lakshmannan, F. Sadri, and I. Subramanian, "A Declarative Language for Querying and Restructuring the Web", *Proc. 6th Intl. Workshop on Research Issues in Data Eng. (RIDE'96)*, Feb. 1996.
11) A. Morishima and H. Kitagawa, "Integrated Querying and Restructuring of the World Wide Web and Databases", *Proc. Intl. Symp. on Digital Media Information Base (DMIB'97)*, pp. 261–271, Nara, Japan, Nov. 1997.
12) 森嶋厚行, 北川博之, "構造化文書とデータベースの統合利用のためのデータモデル NR/SD+ とその問合せ処理", 情報処理学会論文誌, Vol. 39, No. 4, pp. 954–967, Apr. 1998.
13) 森嶋厚行, 北川博之, "視覚的操作系による異種情報源統合利用支援", 電子情報通信学会論文誌 D-I, Vol. J28-D-I, No. 1, pp. 315–326, Jan. 1999.
14) A. O. Mendelzon and T. Milo, "Formal Models of Web Queries", *Proc. 16th ACM Symposium on Principles of Database Systems (PODS'97)*, pp. 134–143, 1997.
15) A. O. Mendelzon, G. A. Mihaila, and T. Milo, "Querying the World Wide Web", *Proc. Symposium on Parallel and Distributed Information Systems (PDIS'96)*, pp. 80–91, Dec. 1996.
16) Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources", *Proc. 11th DE Conf.*, pp. 251–260, Mar. 1995.
17) 根本 剛, 森嶋厚行, 北川博之, "異種情報源統合利用環境におけるメディエータの設計と開発", 電子情報通信学会技術研究報告 ( データ工学 ), Vol. 98, No. 42, pp. 39–46, May 1998.
18) Mary Tork Roth and Peter M. Schwarz, "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources", *Proc. VLDB Conf.*, pp. 266–275, Athens, Greece, 1997.
19) Sun Microsystems, Inc., "Servlets", `http://jserv.javasoft.com/products/java-server/servlets/index.html`.
20) G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computer*, pp. 38–49, Mar. 1992.

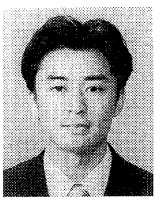
(平成11年3月20日受付)
(平成11年5月 6 日採録)

(担当編集委員 石川 博)



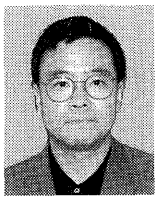
**加藤 数則**

1974 年生. 1997 年筑波大学第三学群工学システム学類卒業. 1999 年同大学大学院工学研究科修士課程修了. 同年キヤノン株式会社入社. 現在, 同社勤務.



**森嶋 厚行** (正会員)

1970 年生. 1993 年筑波大学第三学群情報学類卒業. 1998 年同大学大学院工学研究科修了. 博士 (工学). 現在, 日本学術振興会特別研究員 (研究従事機関: 筑波大学電子・情報工学系). 異種分散情報源統合利用方式, 半構造データ管理等に興味をもつ. ACM, IEEE-CS 各会員.



北川 博之（正会員）

1955 年生．1978 年東京大学理学部物理学科卒業．1980 年同大学大学院理学系研究科修士課程修了．日本電気（株）勤務の後，1988 年筑波大学電子・情報工学系講師．同助教授を経て，現在，筑波大学電子・情報工学系教授．理学博士（東京大学）．異種分散情報源統合，文書データベース，問合せ処理等に興味をもつ．著書「データベースシステム」（昭晃堂）「The Unnormalized Relational Data Model」（共著，Springer-Verlag）等．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE-CS 各会員．電子情報通信学会データ工学研究専門委員会委員長．