データベースシステム 114－9
（1998. 1. 19）

HTMLファイルリンク構造管理システム
— Javaでの実装 —

小林 伸行　　北川 文夫
岡山理科大学大学院理学研究科　岡山理科大学理学部

インターネットの普及により，WWWを用いて情報が容易に提供できるようになった．WWWのデータ単位であるページはHTMLで書かれたテキストファイルである．このページは，さらに他のページへのアンカーを持つことができるので，サーバ上のページは複雑にリンクしている．そのためページを修正，削除した場合，他のページからのリンクに関して不整合が起こる．本稿では，このような不整合を減らすためにリンクの情報を管理し，構造を把握しやすくするHTMLファイルリンク構造管理システムを提案し，このシステムの概要とJavaでの実装状況を示す．

Management System for HTML-file Link Structure
— Implement by Java —

Nobuyuki KOBAYASHI† Fumio KITAGAWA‡
†Graduate School of Science, Okayama University of Science
‡Faculty of Science, Okayama University of Science

Today, anyone can offer his own information easily by using WWW, because Internet spreaded over the world wide and is growing up now. A 'page' is a unit of WWW, and it is a text file written by HTML. Any pages are able to have anchors for linking other pages. So the linking structure on a server will be too complex. Then, it may occur inconsistencies in structure of links when some existing pages are deleted or updated. In this paper, we propose a management system for linking structure. By using this system, we can manage a structure of links and be easy to understand the total WWW structure on a server. We also show the system implemented by Java.

1 はじめに

コンピュータ環境の発展に伴い，インターネットが急速に普及し，WWW(World Wide Web)を用いて文字・音声・画像といった様々な情報を容易に発信あるいは受信を行えるようになってきた．これはプラットホーム独立の情報ブラウザが普及したことやデータをHTML(Hyper Text Markup Language)という簡易な指定で行えることによる．このHTMLで記述されたファイル（以下HTMLファイルとする.）のリンクは他のページへのアンカーをHTMLファイル内部に持つことで行っている．このリンクの指定方法は容易であるが，相手先がHTMLファイルを更新・削除した場合について，リンクに対するアクセスの保証は行われていない．そのためにHTMLファイルの更新あるいは削除が行われた場合に，他のページからのリンクに関して不整合が生じる．しかしHTMLファイルを作成したユーザは常にページをチェックしているわけではないので，この不整合に対して全く気付かない事もしばしばである．

一方，WWWサーバを管理する上でも，各ページは個別に管理作成されているため，サーバ内の構造を把握することは困難である．従って，WWWサーバの管理者にとって，サーバ内のリンク構造を把握することが切望されている．更に作成されたリンク構造からページのタイトルやリンクのインデックスなどを見ることでページの内容を推測することも可能である．

また個人のホームページにおいて，リンク情報にある程度論理的な構造が存在しても，物理的には1つのディレクトリに数十から数百のページが存在するものもある．こういったページの管理は非常に困難であり，構造を明らかにするツールが必要である．

そこで本報告では現在見られるような不整合を少なくし，サーバ内の情報を管理しやすくするために，WWWサーバごとにリンク構造を管理するシステムの提案・作成を行った．このシステムは様々なプラットフォームで動作可能なJavaを用いて作成した．

2 基本的なリンク構造

WWWサーバを管理するためのリンク構造を表わすため，HTMLファイル相互の基本的な構造を作成する．近年，リンク構造のような半構造データに対するデータモデルが盛んに研究されている[1]．我々はこのリンク構造に関して木構造適することを以前に明らかにした[2]．この木構造を基本構造に用いて，HTMLファイルのリンクと対応させている．

またHTMLファイルのリンク先がリンク元のページ以外存在しない場合（図1のページ4, 5, 6），これらの2つのHTMLファイルは同一ノードにすることで同一レベルのページを1つのまとまった単位と見ることができる．この時リンク元のページ（図1のページ1, 2），をノードの代表となるページとして扱うことにする．次に，この木構造のrootから最短の経路となるノードとノードを繋ぐリンク（図1の実線の矢印）をメインルートと呼ぶことにする．またメインルートに対応しないリンクに関しては付加的なリンク（図1の破線の矢印）として取り扱うことにし，各ノードにおける詳細情報

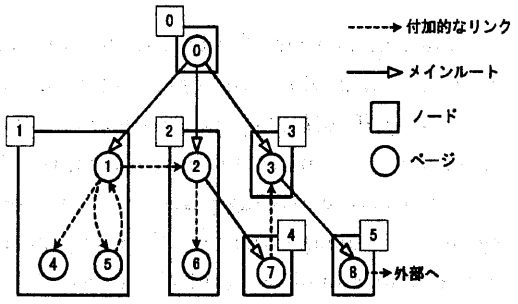


図 1: 木構造の例

とする．外部のサーバへのリンクに関しても同様である．

現在開発中のシステムでは，ユーザから指定された URL を木構造の root として，HTML ファイルのアンカーを再帰的に解析し，リンクの情報を取得する．リンクの情報をすべて取得した後，root から各ノードへの最短経路を探索し，メインルートである木構造を決定している．

3 テーブル定義

HTML ファイルのリンク構造を管理するために以下 5 つのようなテーブルを作成する．

- path ... 各ページの root からの絶対パス名を管理する．
- page ... 各ページのタイトルや作成日付等を管理する．
- link ... 指定サーバ内の参照関係を管理する．
- node ... ノードはページの集合であり，その関係を管理する．
- alias ... 別名で参照されるページを管理する．

以下では path, page, link, node, alias といった主なテーブルについて，順次説明する．

- path テーブルの定義
  表 1で示した path テーブルは対象となる WWW サーバ内のファイルのパス名を管理するためのテーブルである．ID フィールドはページ固有の番号でユニークに割りあてる (以後ページ ID と書く)．Status フィールドはページの探索の要求の有無を登録する，Path フィールドは URL のファイルパスである．ただしページ中の href で指定された URL がディレクトリ名の場合，パスの最後に "/" がない場合，"index.html" や "default.htm" を指定した場合などは，表 5の alias テーブルにこのファイルのパス名を登録し，path テーブルには "/ディレクトリ名/" を登録する．

フィールド名	型
ID	int
Status	int
Path	char(128)

表 1: path テーブル

- page テーブルの定義
  表2で示される page テーブルは各 HTML ファイル固有の情報を管理するためのテーブルである．代表的なフィールドを説明すると，ID フィールドはページ ID を表し，path テーブルの ID フィールドと対応している．Type フィールドはこのページの種類を表し，"text/html" の場合にはファイルを読み込み時に，リンク情報を解析する．ModifyTime フィールドは最終更新日時，SaveTime フィールドは情報取得日時を表す．これら 2 つはページの再読み込み，あるいはリンク先のファイルの再チェックなどの判断に用いる．Title フィールドはこのページのタイトルを

フィールド名	型
ID	int
Protocol	char(8)
Type	char(40)
Encoding	char(16)
Length	int
SaveTime	char(40)
ModifyTime	char(40)
Title	char(128)

表 2: page テーブル

登録する.

- link テーブルの定義
  表 3で示した link テーブルはリンクに関する情報を管理するためのテーブルである. FromID フィールドはリンク元のページ ID を表す. 同様に ToID フィールドはリンク先のページ ID を表す. ただし外部のサーバへのリンクは−1 の値を登録し, Path フィールドに URL を登録する. Flag フィールドはメインルートなどのリンクの種類の情報を表す. Comment フィールドはこのリンクに関するコメントを登録する.

フィールド名	型
FromID	int
ToID	int
Flag	int
Path	char(128)
Comment	char(128)

表 3: link テーブル

- node テーブルの定義
  表 4で示される node テーブルはノードとページ ID の対応を管理するためのテーブルである. NodeID フィールドは上述のページ ID の場合と同様にノード固有の番号でユニークに割りあてる (以後ノード ID と書く). また各ノードの代表となるページを表す Main フィールドがある.

フィールド名	型
NodeID	int
PageID	int
Main	int

表 4: node テーブル

- alias テーブルの定義
  表 5で示される alias テーブルは 2 つ以上の異なったパスで参照される同一ページを管理するためのテーブルである. ID フィールドはページ ID を表し, Path フィールドは同じページ ID への他のファイルパスを登録する.

フィールド名	型
ID	int
Path	char(128)

表 5: alias テーブル

4 HTML ファイルリンク構造管理システム

HTML ファイルリンク構造管理システムでは次の 3 つの動作を行う. (1) リンク構造の探索, (2) メインルートとノードの決定, (3) 構造の表示の 3 つであり, 手順を順次説明する.

4.1 リンク情報の取得

ユーザより入力された URL を root とする. そして実際の WWW サーバにアクセスし, リンクの情報を探索していくことを考える. 最初に指定された URL に対応するパスを path テーブルから検索し Status の値を未取得の状態にして, ページ探索を以下のアルゴリズムで行う.

アルゴリズム 1

1. path テーブルから未取得の状態のパスを ID 順に 1 つ*取得
2. page テーブルに更新が必要であればステップ 5へ

*現在は 1 つしか取得しないが, 今後複数のページを同時に作業できるように拡張する予定である.

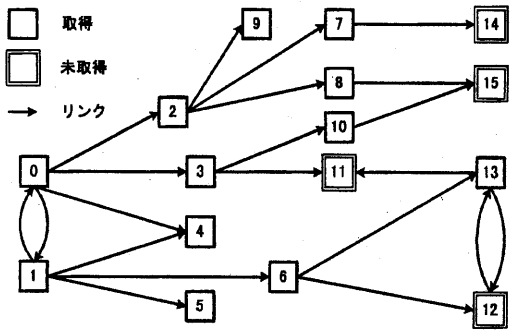


図 2: リンク情報の取得例

3. リンク先のパスの再検査が必要なければステップ 8へ
4. path テーブルから再検査の必要なパスを検索し，Status の値を未取得状態に変更後，ステップ 8へ
5. ネットワークを通じてページを取得
6. ページを解析し，新しいリンクを link テーブルに登録し，リンク先のパスを未取得状態に変更，あるいは新規に追加
7. ページの情報を page テーブルに更新
8. path テーブルの Status の値を取得状態に変更
9. path テーブルに未取得の状態のページが無ければ終了
10. ステップ 1へ

以上のアルゴリズムにより，ページ ID を 13 まで実行したものが，図 2である．

4.2 メインルートとノードの決定

次にメインルートとノードの決定を行う．この動作のための前提条件として，リンクの情報がすべて取得してあることが必要になる．すなわち，path テーブルに未取得状態にあるページが存在しないことが必要である．ここではメインの処理である 2 つのアルゴリズムを紹介する．

まず同一ノードの決定は link テーブルを用いて行い，そのアルゴリズムを以下に示す．

アルゴリズム 2（ノードの探索と決定）

1. ToID フィールドの値をを昇順に 1 つページ ID を取得，これを p1 とする
2. p1 へのリンクの FromID の値が 2 種類以上あればステップ 6へ
3. p1 へのリンクの FromID の値を p2 とする
4. p1 から p2 以外へのリンクが存在すれば，ステップ 6へ
5. p1 から p2 へのリンクの Flag の値をノード内のリンクとしてデータベースに登録
6. ToID の値を全て取得した場合，終了
7. ステップ 1へ

ノードが決定された後，メインルートの決定を行う．この処理も link テーブルを用いて行い，アルゴリズムは以下のようになる．

アルゴリズム 3（メインルートの探索）

1. FromID の値が root のページ ID であるリンクの Flag の値をメインルート候補の状態にする
2. ToID の値が root ページ ID であるリンクの Flag を付加的なリンクとしてデータベースに登録
3. Flag の値がメインルート候補の FromID の値を 1 つ取得し，m1 とする

4. ToID の値が m1 であるリンクの Flag を付加的なリンクとしてデータベースに登録

5. FromID の値が m1 である ToID の値を 1 つ取得し，m2 とする

6. ToID の値が m2 であるリンクの Flag を付加的なリンクとしてデータベースに登録

7. m1 から m2 へのリンクを，次のレベルのメインルート候補状態にする

8. FromID の値が m1 であるリンクが他に存在していれば，ステップ 5へ

9. メインルート候補が他に存在すれば，ステップ 3へ

10. 次のレベルのメインルート候補が存在しなければ終了

11. 次のレベルのメインルート候補を現在のメインルート候補にする

12. ステップ 3へ

ここでの処理が終了することで HTML ファイルのリンク構造に関するデータの作成が終了する．

4.3 構造の表示

最後に実際に表示する画面について説明する．メインの画面は 3 つ存在する．まず全体のノードの構造を示す画面を表示する．次に各ノードの詳細な情報を見るための画面を示し，最後に現在エラーとなっているリンクを表示する．これら 3 つの画面を表示することで，ユーザから指定されたサーバにおける HTML ファイルのリンク構造を視覚的に捕らえることができ構造の把握がし易くなる．

色	ノードの特徴
白	このノードへの参照が多い
緑	同一サーバ内へのリンクが多い
黄	ノードを構成するページの数が多い
青	同一ページへのリンクが多い
シアン	外部のサーバへのリンクが多い

表 6: ボタンの背景色の例

さらに全体のノードの構造を表示する画面は，ボタンの背景色（表 6）を変更することで各ノードの特徴を示し．より視覚的に把握し易くする．

各ノードの詳細情報を表示する画面には，このノードにあるページの詳細情報を表示し，各リンクのタイトルやリンク先のノードを表示する．また，ページのタイトルやリンクのコメントなどの詳細情報を表示することで，このノードの内容を把握することもできる．

リンクのエラーを表示する画面は，同一サーバ内へのリンクのエラーと外部のサーバへのリンクのエラーがある．まず同一サーバ内のリンクのエラーは，現在サーバ内に存在しなくなったページのパス名を表示し，さらにそのページへのリンクがあるノードも表示する．また外部のサーバへのリンクのエラーは，現段階では外部のサーバに対してファイルの有無までは検査していない．従って，ここでは外部のサーバのアドレスが存在しない場合に，そのリンクの情報を表示する．リンクのエラーを修正する場合，この画面を確認することで容易に理解できる．

5 システム構成

現在，本システムのシステム構成は図 3に示すように，ユーザインタフェースと SQL サーバおよびデータベースで構成されている．

本システムは主に次の 3 つの動作を行うものとする．HTML ファイルのリンク構造を表

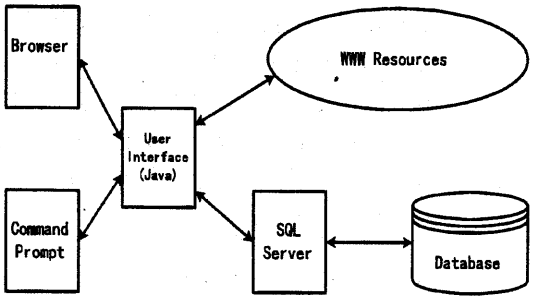


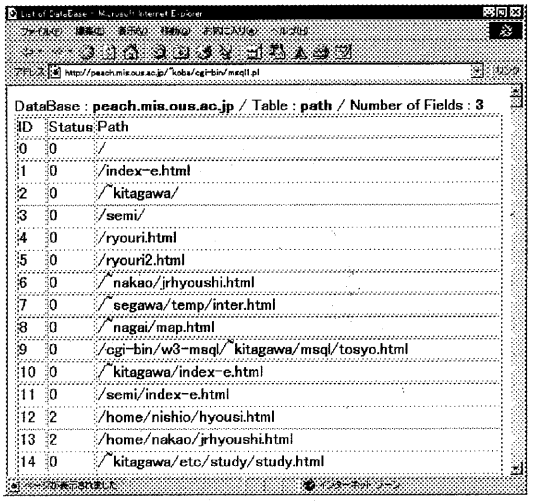
図 3: システム構成

示するためにリンクの情報が必要となる．そのため WWW リソースに存在する WWW サーバに対してリンクの情報を取得し，その情報を SQL サーバを通してデータベースに登録する．次にデータベースに登録したリンクの情報を解析し，ノードやメインルートを決定しデータベースを更新する．その結果，解析されたノードやメインルートをすばやく表示できる．最後にデータベース上にある解析後のデータを元にリンク構造を表示し，リンクエラーや各ページにおけるリンクの数なども合わせて表示することで，HTML ファイルのリンク構造の把握がしやすくなる．また本システムはブラウザ上のアプレットとしてだけでなく，Java のアプリケーションとして動作し，入力ダイアログを使ってアドレスを指定できる．

6 システムの実行例

ここでは前述のシステムについて，Java を用いて作成し，実際にこのシステムを実行して取得したデータとリンク構造の出力画面をそれぞれ図 4, 5に示す．

本システムは開発段階にあり，多くの問題点が残されているが，その主要なものとしては，現在のところ次の 5 点が考えられる．



DataBase : peach.mis.ous.ac.jp / Table : path / Number of Fields : 3

ID	Status	Path
0	0	/
1	0	/index-e.html
2	0	/~kitagawa/
3	0	/semi/
4	0	/ryouri.html
5	0	/ryouri2.html
6	0	/~nakao/jrhyoushi.html
7	0	/~segawa/temp/inter.html
8	0	/~nagai/map.html
9	0	/cgi-bin/w3-msql/~kitagawa/msql/tosyo.html
10	0	/~kitagawa/index-e.html
11	0	/semi/index-e.html
12	2	/home/nishio/hyousi.html
13	2	/home/nakao/jrhyoushi.html
14	0	/~kitagawa/etc/study/study.html

図 4: データベースに登録された path テーブルの例

- **実行速度の向上**
  Java を用いることで，プラットフォームの依存はないが，データの取得，解析や表示など時間がかかる．データの取得は，スレッド化を行い，同時に複数のページを取得することで解決できると思われるが，解析や表示に関してはアルゴリズムの高速化を再検討する必要がある．

- **全体構造の表示画面の改善**
  表示するノードは 1 つのサーバに限定しているが，それでもノードの数が多く理解しにくい．全体の構造を表示するにあたり，表示方法などに工夫が必要となる．また各ノードの特徴を示す色分けについても，ノードの内容をさらに比較検討を行った後，参照される回数の多いノードに対する条件を変える必要性も認められる．

- **ページの解析方法の拡張**
  HTML ファイルのタグの記述の自由度が高いため，リンクが正しく取得されない

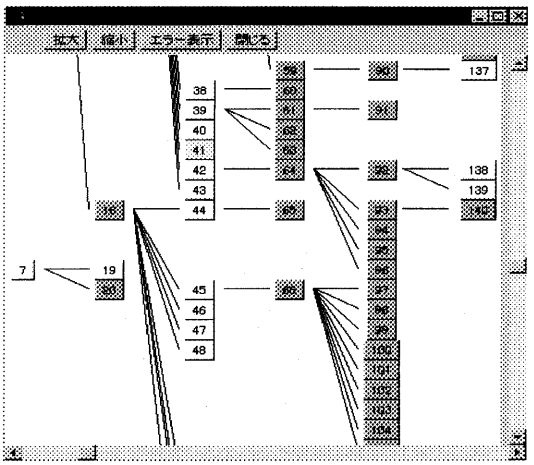


図 5: リンク構造の出力画面の例

場合があり，より正確なページにおけるリンクの解析を行うためパスの取得方法をもっと柔軟にする必要がある．同様にCGIのファイルについても誤ったリンクを取得する場合があるので，これについても考える必要がある．

- **外部のリンクエラーの改善**
  4.3章の構造の表示でも述べたが，外部のサーバに対してのリンクに関してはサーバの存在に対しての検査しか行っていない．このため外部へのリンクの情報には，ページが存在しない可能性がある．したがって今回提案した方法を外部へのリンクに対しても適用する必要がある．

- **システムの将来計画**

1. メインルートに従った物理ディレクトリの再配置
   HTMLを手で書いて順次追加していく方法を認めている以上，リンク構造を管理することは難しい．より作り手に管理し易い構造にするために，論理構造を物理構造に反映させる仕組みを提供することが必要である．物理的再配置と共に各ファイル内のタグ情報を書き直すことが必要である．XML(Extensible Markup Language)[3]を正式にサポートによる，外部リンクでの対応も考えられる．

2. XMLのスキーマ管理との関連
   本システムでは，サーバ内のHTML情報を管理するものなので，XMLのサーバスキーマ定義への対応が必要である．

7 終わりに

本報告ではHTMLファイルのリンク構造を把握するため，またリンクの情報の不整合を削減するために，HTMLファイルリンク構造管理システムを提案し，Javaで作成されたプロトタイプの状況について報告した．その結果，WWWサーバ内のリンク構造を表示することで視覚的に全体の構造を把握し易くなり，さらにはリンクのエラー表示画面で，リンクの不整合の削減が可能となった．今後，実行速度の向上，表示画面のさらなる改善，解析手法の拡張などを行い，ページや部品の管理，検索などを含めたシステムの再構築を行う必要がある．

参考文献

[1] 田島敬史: “半構造データに関する研究動向,” 情報処理シンポジウム論文集 Vol.97, No.11, pp.101-110, 1997.

[2] 小林伸行, 北川文夫: “HTMLデータの総合管理システム,” 電子情報通信学会技術研究報告, DE96-53, No176, pp.177-182, 1996.

[3] World Wide Web Consortium. Extensible markup language (XML). Proposed Recommendation, http://www.w3.org/TR/PR-xml, 1997.