データベース・システム 78－7
（1990．7．19）

リンク定義言語を有するハイパーテキストシステム:TextLink-III

平山伸一[1,2]　西川記史[1]　難波克己[2]　田中克己[1]

[1]神戸大学 工学部計測工学科

[2](株)ケーシーエス ソフトウェア技術部

本稿では、ハイパーテキストにおけるリンクをデータベース質問の対として仮想的に設定するためのリンク定義言語を有するハイパーテキストシステムであるTextLink-IIIについて述べる。本システム開発の目的は、ハイパーテキストのリンク情報とノード･オブジェクト情報との独立性を高めることであり、これに基づいて開発したリンク定義言語により、(1)多様な視点からのリンクの設定･切り替え、(2)データベーススキーマを用いた組織的･集合的なリンクの設定、(3)オブジェクトへのインクリメンタルな属性追加やスキーマ進化を意識したリンクの設定などを可能としている。

TextLink-III: A Hypertext-System has a Link Definition Language.

Shinichi Hirayama[1,2]　Norifumi Nishikawa[1]　Katsumi Namba[2]　Katsumi Tanaka[1]

[1]Department of Instrumentation Engineering , Kobe University

Rokkodai, Nada-Ku,Kobe 657, Japan

[2]Software Engineering Division , KCS Corporation

64, Naniwa-Cho, Chuo-Ku , Kobe 650 , Japan

In this paper, we describe our hypertext system TextLink-III that has a link definition language. By the link definition language, we can virtually define a hypertext-link as a pair of database queries. The major objective of developing TextLink-III is to increase the independence between hypertext-links and node-objects. The link definition language is developed for this purpose, and by this, TextLink-III system can offer some facilities to (1)define / change hypertext-links from several user's viewpoints, to (2) define hypertext-links over database schema in a systematic and set oriented manner,and to (3) deal with schema evolution and object updates.

1 まえがき

ハイパーテキストシステムは、高機能ワークステーションの普及などを背景として、最近特に注目されているデータ管理手法であり、様々なシステムが開発、実用化されてきている。今後さらに、コンピュータネットワークの普及などとともに、オフィス情報システムやソフトウェア開発環境などといった、新しいアプリケーションの分野への適用が期待されているが、これらのアプリケーションが扱うデータには以下の特徴があり、何らかの形でデータベース的機能を実現する必要がある。

- 取り扱うデータが大量である。
- データの更新が頻繁に発生する。
- 複数の利用者がデータを共有する。

また、ハイパーテキストシステムの中にはデータベース的な機能を持ったものもいくつか開発されており、特に、ブラウン大学のIntermediaは主にリンク情報を格納・管理するために関係型DBMSであるIngressを用いており、Hypertextと汎用DMBSの融合という観点から興味深い。しかし、このような形でのHypertextと汎用DMBSの融合では依然次のような問題が存在する。

- テキストオブジェクト間のリンクが直接的・固定的であり、テキストオブジェクトの内容が頻繁に変化する場合は、リンクの更新が煩雑・複雑になる。
- 関係データベースやオブジェクト指向データベースが提供している強力な集合型データ操作言語を利用できるものはない。

我々はTextLinkの前バージョンTextLink-IIにて、1つめの問題に対しては、リンク先のオブジェクト群が固定的にリンクされるのではなく、データベース質問という形で表現される、実行可能リンク(executable link)を実現して対処し、2つめの問題に対しては、基本的な集合型操作言語をSmalltalk-80上にて実現し対処した。

Textlink-IIIでは、実行可能リンクという概念をさらに一般化して、テキストオブジェクトのクラス階層上に仮想リンク機能を実現するためにリンク定義言語(Link Definition Language---以下LDL)を用いている。

2 TextLink-IIIシステムの概要

本節では、我々が開発したTextLink-IIIシステムの概要について、TextLink-IIIのシステムの機能構成と、TextLink-IIIにおけるデータベーススキーマ・オブジェクトの更新の取扱いを中心に述べる。

2.1 システムの機能構成

TextLink-IIIの機能構成はFig.2.1のようになっており、大きく以下の3つのモードに分かれている。

- クラス設計モード
- ブラウジングモード
- オーサリングモード

ブラウジングモードとオーサリングモードについては一般のハイパーテキストシステムのそれと同様の位置づけであるが、次章にて詳しく述べるようにリンク定義言語を用いていることが大きな特徴である。

また、TextLink-IIIをオブジェクト指向データベースのユーザインタフェイスとしてみた場合、データベースおよびデータベースオブジェクトの作成・更新機能は必須の機能である。我々のTextLink-IIIでは、クラス階層によるスキーマを用いて文書オブジェクトを取り扱う。すなわち、文書オブジェクトはあるクラスのインスタンスであり、クラスは継承機能をもつ階層をなす。このために、クラス設計モードが用意されており、クラススキーマの生成や更新、文書オブジェクトの生成や所属クラスの変更を行うことができる。

クラス設計モードにおけるデータベーススキーマ・オブジェクトの更新について次節に述べる。

2.2 データベーススキーマ・オブジェクトの更新

TextLink-III		
Class Design	Browsing	Authoring
・Make Subclass ・Remove Class ・Change SuperClass ・New TextObject ・Move TextObject ・Remove TextObject	・Open TextObject ・Text Atribute ・Open LDL ・Open Reverse LDL	・Set LDL ・Modify LDL ・Save LDL ・New-LDL-Dictionary ・Remove LDL-Dictionary

Fig.2.1「TextLink-IIIの機能構成」

データベーススキーマの更新とはオブジェクトの他のクラスへの移動およびクラス階層の変更・更新のことを指す。この分野には多くの方法・問題点が存在するが、ここではTextLink-IIIで採用、実現した機能について述べる。

2.2.1 データベーススキーマの更新

(1) Make Subclass

これは既存のデータベースに新たにクラスを作成する機能である。これは新しいクラスに属するオブジェクトをデータベースに追加する場合や、オブジェクトの具体化を行うための前段階として新たにクラスを作成する必要がある場合に使用される。ここでは新しいクラスをあるクラスのサブクラスとして作成していく方法をとった。

(2) Remove Subclass

これは既存のクラスを削除する機能である。削除されたクラスのインスタンスとダイレクトサブクラスは、それぞれ削除されたクラスのダイレクトスーパクラスのインスタンスとダイレクトサブクラスになる。

(3) Change Superclass

あるクラスのスーパークラスを他のクラスへ変更する。これは、オブジェクトの具体化の進行とともにスキーマの更新が要求される時に使われる。

2.2.2 オブジェクトの更新

(1) New Text-Object

データベースに新たにオブジェクトを追加する機能である。新たに追加されるオブジェクトは既存のクラスのインスタンスとなるが、適当なクラスが存在しない場合には"Make Subclass"を行ったのちにオブジェクトを追加する。また、クラスの属性に対して、属性値を入力するためのダイアログが表示される。このダイアロクに属性値を入力することによって、属性値を持つテキストオブジェクトを生成する。

オブジェクトの追加には、(a)新規にエディタを用いて作成する方法と、(b)既に(他のエディタやワードプロセッサにて作成され)存在するテキスト・ファイルを取り込む方法とがある。

(2) Move TextObject

あるオブジェクトを他のクラスに移動させる機能である。他のクラスに移動した時(一般的には、オブジェクトの具体化に伴うサブクラスへの移動が想定されるが)、属性値の追加が必要な場合にはシステムが追加を要求する。

(3) Remove TextObject

データベースから既存のオブジェクトを削除する。

2.2.3 ビジュアル・インタフェイス

スキーマの生成や更新に関して、特別な言語を用いなくてもよいようにビジュアル・インタフェイスを用いている。Fig.2.2にあるように、データベースのスキーマ

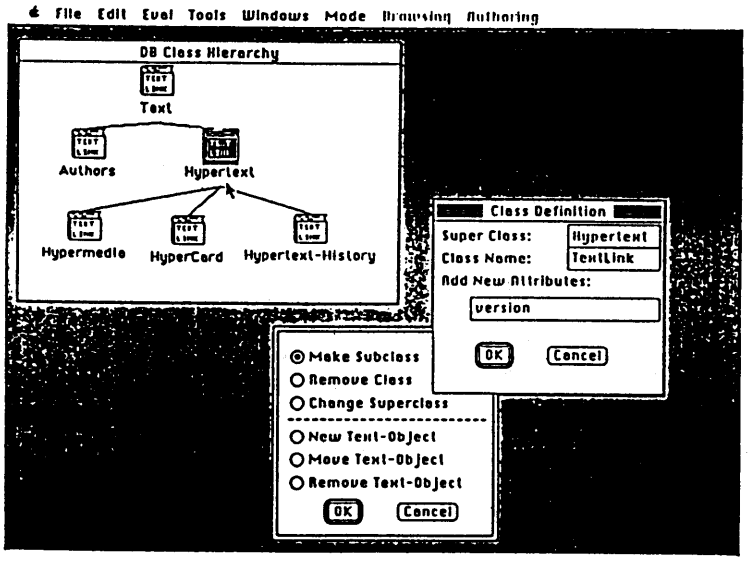

Fig.2.2「スキーマ更新の例」

を表現するグラフをもとに、ダイアログ・ボックスを通してシステムからの応答要求に答えることにより行われる。

Fig.3.1の例は、クラス"Hypertext"のサブクラスに"TextLink"というクラスの追加を行っているところであり、属性"Version"を定義している例である。ユーザはまず、"DB Class Hierarcy"グラフの中から"Hypetext"をマウスでクリックする。システムはそのクラスに対するオペレーションを要求してくるので、"Make Subclass"を選択する。システムはクラス定義に関する要求をしてくるので、ユーザは必要項目の入力を行う。

3 リンク定義言語 (Link Definition Language--LDL)

我々はTextLink-IIIにおける参照リンクを、テキスト・セグメントに固定させる(固定リンク)かわりに、リンク定義言語という特別な言語で記述される仮想リンクにより実現している。この方法は従来の固定リンクに対し、実行速度の点では劣るが、複数のユーザや視点ごとのリンク設定や、リンクの組織的・集合的設定を可能にし、オブジェクトの更新やスキーマ進化へ柔軟に対応することができる。

3.1 複数のユーザや視点ごとのリンク設定

現状のハイパーテキストシステムのほとんどは、一人のユーザがパーソナルな情報を管理するような場で利用される。しかし、今後の適用分野としては、オフィス情報システムやソフトウェア開発環境といった、複数のユーザがデータを共用するような分野が考えられる。

複数人のユーザでハイパーテキストシステムを利用しようとする場合には、リンクの設定が一通りでは不都合な場合がある。例えば、オフィス情報システムへの適用を考えた場合、一般にオフィスには様々な職種の人が存在し、その職種によって、参照リンクもことなってくるはずである。この問題に対しては、従来の固定リンクでも、リンクの属性等を利用して対処することができるが、リンクの数が膨大であったり、データの更新が頻繁に発生する場合には、リンクの更新が非常に煩雑なものになってしまう。

我々はTextLink-IIIにおいて、仮想リンクを適用し、データベースとは独立にリンク・ディクショナリに保持して、データベースを利用する際にはこの(リンク定義言語で記述された)仮想リンクを通してデータベースを見るようにすることにより、ユーザから見ればあたかもデータベースに直接リンクが設定してあるかのように見せかける方法を採用した。この概念図をFig.3.1に示す。

3.2 リンクの組織的・集合的設定

ハイパーテキストシステムにおいてオーサリングするときに大量のテキストに対して1つずつリンクを設定することは非常に手間のかかる作業である。一方、リンクを設定しようとする場合、リンク先のオブジェクトは何

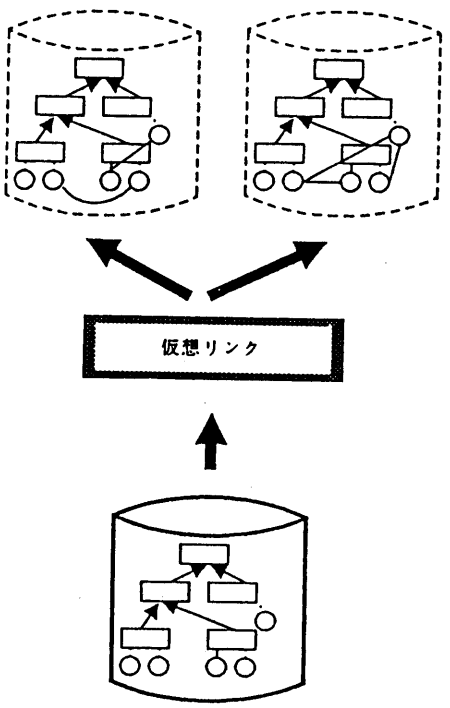

Fig.3.1「仮想リンク」

かある特徴を持っており、ユーザはそれを手掛かりとしてリンクを設定しているわけであり、オブジェクトがその特徴を満たせば自動的にリンクを設定してくれることを望むはずである。

前節において仮想リンクを用いた複数のユーザや視点ごとのリンク設定について述べたが、我々の仮想リンクは、リンク定義言語(Link Definition Language--LDL)によって記述されており、以下LDLの各記述部について述べる。

3.2.1 LDLの各記述部

一般的にリンクはあるオブジェクトから別のオブジェクトに対して設定するものと見ることができるが、このとき、リンク元のオブジェクトおよびリンク先のオブジェクトをそれぞれソース・オブジェクト、ターゲット・オブジェクトと呼ぶことにする。我々のTextLink-IIIでは、LDLにてソース・オブジェクト、ターゲット・オブジェクトの特徴を定義することにより、リンクを組織的・集合的に設定する。従って、LDLにはソース・オブジェクトの定義部とターゲット・オブジェクトの定義部とを用意している。また、リンク自身の情報を定義するためにリンク情報記述部も用意しており、次節以降にそれぞれの記述部について述べる。

3.2.1.1 リンク情報記述部

LDLの参照、更新の手掛かりにするために、リンク名

やリンクに関するドキュメントを記述する。

(a)Link-Name
リンク名を記述する。
(b)Document
LDLの修正及び削除の手掛かりになるように、リンクに関するコメントを任意に記述する。

3.2.1.2 ソース記述部

リンク元を特定する記述部である。クラスフィールド(From)、属性フィールド(Where)、キーワードフィールド(Select)の3つの部分に分けられる。

(a)From
ここには、リンクがどのクラスのテキストオブジェクトから張られているかをクラス階層によって、チェックするためのクラス及びクラス階層の演算式が記述される。

(b)Where
ここには、リンク元のテキストオブジェクトを属性値によって特定するために属性値の条件式が記述される。

(c)Select
ここには、実際にリンクが張られるキーワードの文字列が記述される。

3.2.1.3 ターゲット記述部

リンク先を特定する記述部である。クラスフィールド(From)、属性フィールド(Where)の2つの部分に分かれる。

(a)from
ここには、リンクがどのクラスのテキストオブジェクトに張られているかをクラス階層によって、チェックするためのクラス及びクラス階層の演算式が記述される。

(b)Where
ここには、リンク先のテキストオブジェクトを属性値によって特定するために属性値の条件式が記述される。

3.2.1.4 条件記述

本説では実際に記述される条件式の記述様式について説明する。

●クラスフィールド

クラスフィールドにはクラスを用いた演算式を記述する。このフィールドに記述された演算式の結果得られるクラス集合に属する全オブジェクトが次の属性フィールドに記述される演算式の対象になる。クラスフィールドの演算式はマウスを用いたビジュアルな操作で定義することができる。

"Class A" : "Class A"のみのオブジェクトを要素とする集合を表す。
"Class A*": "Class A"とその全サブクラスのオブジェクトを含む集合を表す。

上記の2種類のクラス表現を用いたオブジェクトの集合の表示と下記の3つの関数を使って、クラスフィールドの演算式を記述する。

(& U V): アンドを表す。すなわち、2つの集合U、Vの共通要素を持つ集合を返す。

(%union U V): オアを表す。すなわち、2つの集合U,Vの要素を持つ集合を返す。この時、要素の重複を許さない。

(%Remove U V): 集合Uの要素から、集合Vの要素を除いた要素を持つ集合を返す。

Fig,3.2にソース記述部におけるクラスフィールドの定義例を示す。ここでは"TextLink*"が定義されている。すなわち、クラス"TextLink"か、または、そのサブクラスに属するテキストオブジェクトが、ここで記述するリンク元の選択対象となる。

●属性フィールド

属性フィールドにはテキストオブジェクトの属性の条件を記述する。このフィールドではクラスフィールドによって、選択されたすべてのテキストオブジェクトを演算対象とする。
このフィールドでは以下のような比較演算式が使われ、これらの条件はand(アンド)を用いて接続可能である。

(1) (@= 属性名 Key
{name,contents,num,size,member} Value)

① Keyが'nameの時、テキストオブジェクトの属性名の属性値の文字列がValueの文字列と等しいか比較する。
② Keyが'contentsの時、テキストオブジェクトの属性名の属性値の文字列が副文字列としてValueの文字列をもつか、テストする。
③ Keyが'numの時、テキストオブジェクトの属性名の属性値の数値がValueの数値と等しいかを評価する。
④ Keyが'sizeの時、テキストオブジェクトの属性名の属性値の要素数がValueの数値と等しいか比較する。
⑤ Keyが'memberの時、テキストオブジェクトの属性名の属性値がValueの文字列を含むかを評価する。

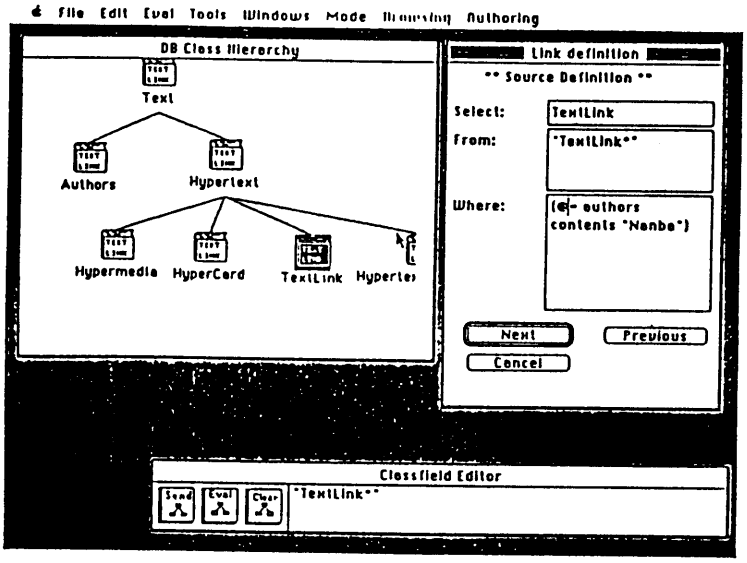

Fig.3.2「クラスフィールドの定義例」

(2) (@<= 属性名 Key(num,size) Value)
(@>= 属性名 Key(num,size) Value)
(@< 属性名 Key(num,size) Value)
(@> 属性名 Key(num,size) Value)

① keyがnumの時、テキストオブジェクトの属性名の属性値の数値とValueの数値との大小関係を比較する。
② Keyがsizeの時、テキストオブジェクトの属性名の属性値の要素数とValueの数値との大小関係を比較する。

属性フィールドの条件の例をいくつかあげる。

○著者が"Tanaka'さんである文献(テキストオブジェクトの属性authorsの値が"Tanaka"である。)
(@= authors name "Tanaka")
○ タイトルに"HYPERTEXT"という語句が含まれる文献(テキストオブジェクトの属性titleの値が副文字列として"HYPERTEXT"という文字列を持つ。)
(@= title contents "HYPERTEXT")
○ 著者のひとりに'Nishikawa'さんが含まれている文献(テキストオブジェクトの属性authorsの値に"Nishikawa"が含まれる。)
(@= authors member "Nishikawa")
○ 著者の人数が3人以上の文献(テキストオブジェクトの属性authorsの属性値の要素数が3以上である。)
(@>= authors size 3)
○ 1980年より前に書かれた文献(テキストオブジェクトの属性yearの属性値が1980より小さい。)
(@< year num 1980)

3.3 オブジェクトの更新への対応

LDLによるリンク先の決定はリンク参照時に行われる。このことは、データベースに頻繁にテキスト・オブジェクトが追加されるような場で有効であると考えられる。例えば、ソフトウェア開発における開発作業や保守作業などを行う場合、多くの(ある程度分類化された)ドキュメントを互いに参照しつつ作業を行っていく。このような場合にハイパーテキストシステムのリンクを利用して作業を行うことは有効であると考えられるが、一つの作業は大抵新たなドキュメントを生み出すことになる。このようにして生成されるドキュメントを、データベースに追加するたびにリンクを設定することは非常に困難な作業である。ソース記述によりリンク元を仮想化することにより、データベースに追加されたドキュメントからすぐに参照を行うことが可能になる。

3.4 スキーマ進化への対応

ここでのスキーマ進化とは、特に、将来的にオブジェクトの具体化が進み、データベーススキーマが詳細化されることを指す。LDLではクラスフィールドの条件記述にて、"Class A*"を記述することにより、このサブクラス方向へのスキーマ進化に対応することができる。この対応は前節で述べたオブジェクトの更新への対応と全く同じメカニズムにて対処できるが、ユーザはより慎重にクラスフィールドの条件記述を行わなければならない。例えば、Fig.3.3に示すように、リンク設定時に、クラスA、Bから構成されるスキーマが、将来、クラスA、B、C、Dから構成されるスキーマに進化するとする。リンク設定時にクラスA、Bの両方をクラスフィールドへ指定するための条件記述の方法には以下の4つがあり、スキーマ進化後は、LDL実行時にそれぞれ異なるクラス記述として展開される。

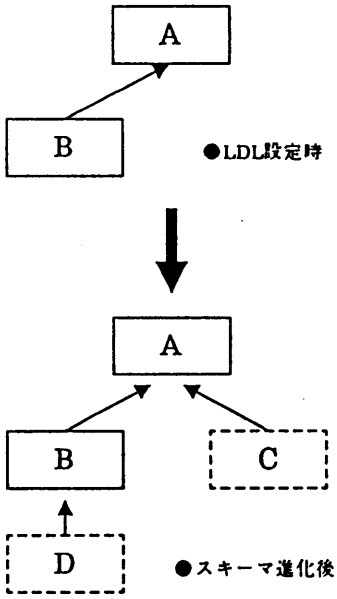

Fig.3.3「スキーマ進化とクラスフィールド記述」

(a) (%union A B)
{A,B}
(b) A*
{A,B,C,D}
(c) (%union AB*)
{A,B,D}
(d) (%union (%remove A* B*) B)
{A,B,C}

4 インプリメンテーション

TextLink-IIIのインプリメンテーションは、Apple社のパーソナル・コンピュータMacintosh II上で、Allegro Common Lispを用いて行われている。Macintoshを選んだ理由としては、(a)良好なユーザ・インタフェイスを実現したかったことと、(b)将来的にマルチメディアやネットワーク対応を行いたいことなどが主なものであり、また、Allegro Common Lispについては、(a)プロトタイプ・システムの開発を前提とした場合、迅速な開発が行えること、(b)MacintoshのToolboxのルーチンを全て利用できること(high levelまたはlow levelのシステムインタフェイスを用いて)などが主な理由である。なお、Allegro Common Lisp 上 で はPCLはPortable Common Loops---Common Lisp上の標準オブジェクト指向プログラミング言語CLOSの評価バージョンと考えてもよい)を用いている。Fig.4.1にTextLink-IIIの使用例を示す。

LDLは現在、リンク情報記述部、ソース記述部、ターゲット記述部を要素とするLISPのリスト構造によって表現されている。また各リンク・ディクショナリもLDLのリストを要素とするリスト構造である。LDLの定義や削除はユーザ・ディクショナリのリストにLDLのリストを追加や削除をすることによって行われる。クラスフィールド、属性フィールドの条件記述は、LISPのS式で記述され、リンクの参照時にそれらのS式が評価されLDLは実行される。

データベーススキーマをビジュアルに表現し、スキーマの更新をビジュアル・インターフェイスを用いた簡単な操作で行うために、各クラスの情報(PCLによるクラス定義、クラススキーマのグラフを描画するためのデー

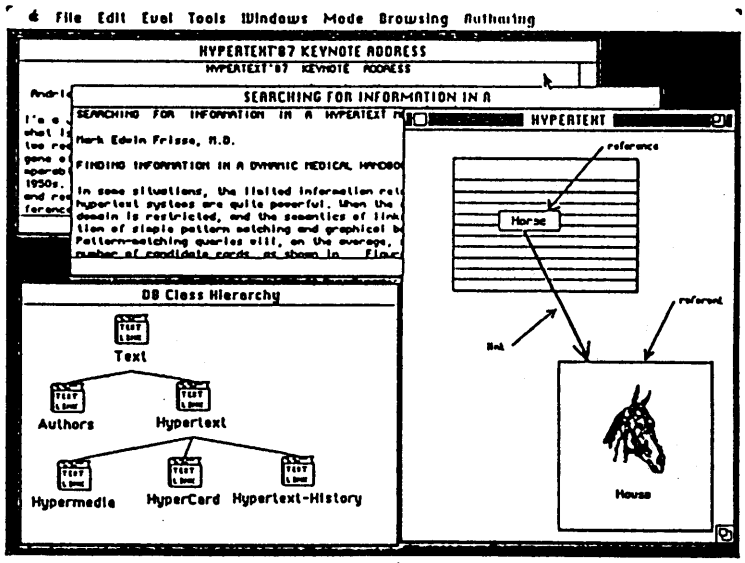

Fig.4.1「TextLink-IIIの使用例」

タなど)の管理を行うクラスノードをPCLのオブジェクトで実現している。TextLink-IIIの大部分の操作は、それぞれの操作に従って、クラスノードオブジェクトのメソッドを起動することによって実現している。このクラスノードオブジェクトが保持している情報を保存することによって、現在のデータベーススキーマの状態を保持している。

Textlink-IIIではデータベーススキーマや各文書オブジェクトのクラスの変更が頻繁に行われる。PCLではクラスの再定義に伴うインスタンスの構造の修正やインスタンスのクラスの変更に伴うインスタンスの構造の修正が容易であるため、テキストオブジェクトをクラススキーマとして定義されたPCLのクラスのインスタンスとしている。

5 まとめ

今回の研究において我々は、ハイパーテキストのリンク情報の更新とデータベースの更新との間の独立性を高める機能の開発を目的として、リンクを仮想的に設定するためのリンク定義言語を有するハイパーテキストシステムであるTextLink-IIIの設計および開発を行った。

ハイパーテキストシステムの今後の適用分野として、オフィス情報システムやソフトウェア開発環境など、データが大量かつ更新が頻繁に発生し、グループでの協調作業がなされる分野が考えられるが、我々のTextLinkのアプローチはこれらの分野にかなり有効であると考えられ、我々の今後の課題として、これらの分野への適用実験を考えている。

参考文献

(1) 田中、西川、「オブジェクト指向データベースのハイパーテキスト型インタフェイスTextLinkについて」、情報処理学会データベースシステム研究会技術報告、1989年7月

(2) Tanaka,K.andYoshikawa,M.,Towards an Integrated Environment for Editing,Viewing and Publishing Multimedia Database Objects, Proc. International Symposium on Computer '88,Kobe,Japan,pp.59-66,Oct. 1988

(3) Conklin,J., Hypertext: An Introduction and Survey,IEEE Computer Magazine,pp.17-41,Sept. 1987

(4) Banerjee,J., Kim,W. et al, Semantics and Implementation of Schema Evolution in Object-Oriented Databases, Proc. of ACM-SIGMOD International Conference, 1987.

(5) Kim,W. and Lochovsky,F.H. ed., Object-Oriented Concepts, Databases, and Applications,ACM Press, 1989.