文脈ディレクトリシステム

A Context Directory System

花川 賢治*1
Kenji Hanakawa

*1大阪府立工業高等専門学校
Osaka Prefectural College of Technology

To organize information on PCs more effectively, three actions are required: construct a structure when retrieving it, manage data of information resources and non-information entities together, have clear semantics of hierarchical structures. The context directory system is a information organizer connected to a database which contains metadata and data of non-information entities, such as human, time, place and more. When a user issue a query in a Prolog based logical language, the system construct a hierarchical structure from the query result. The hierarchical structure consists of context directories which has own context on which a query depends. The author introduces an implementation and query examples of the system.

1. はじめに

記憶装置の大容量化に伴い，利用者が管理するコンピュータのファイルの数が非常に多くなり，目的のファイルを探し出すことが困難になってきている．この問題を解決するために，従来から階層的組織化の手法とファイル検索の手法が用いられてきた．UNIX のファイルシステムに代表される階層的組織化の手法では，複雑な分類構造が表現でき，ディレクトリ間を移動し複数のファイルのグループを参照することができる．しかし，階層構造の作成には大きな労力がかかるため，一度作成されると永続的に利用され，参照時の要求に応じて最適な構造に変化させることは難しい．一方，ファイル検索による方法では，あらかじめファイルの属性のインデックスを作成しておき，参照時に，検索条件を指定してファイルを探し出す．この方法では，状況に応じて自由に検索条件が設定できるが，検索結果は一つのファイルのグループで，条件に合致したファイルの数が多すぎるときに，そこから必要なファイルを見つけるのが困難になる．互いに相補的な性質を持つ階層的組織化とファイル検索を統合する仮想ディレクトリ [Apple 04] も試みられたが，このような異質なツールの結合は複雑さをもたらし，成功していない．

本研究では，現状の情報の組織化ツールが持つ問題点を，情報の組織化が情報の格納時点に行われる，情報資源とそれ以外の実体が区別される，階層構造の意味論が明確でない，の 3 点であると考え，それらを解決することで，最初から検索と組織化が一体であるような文脈ディレクトリシステムを提案する．

このシステムは以下に示すような特徴を持つ．

- データベースに Prolog の事実を用いる
- 多様な実体についてデータベース化する
- 問い合わせ言語に Prolog の質問を用いる
- 問い合わせの構造は階層構造に反映される
- 階層構造のノードはデータベースの事実に対応する
- 階層構造の中間ノードと葉の区別は固定していない
- 問い合わせはカレントディレクトリのパスに依存する

連絡先: 花川 賢治，大阪府立工業高等専門学校，大阪府 寝屋川市幸町 26-12, hanakawa@ipc.osaka-pct.ac.jp

発表者は，本研究に先行して，文書の部品から利用者の要求に応じて構造化文書を構成する動的文書の研究を行った [Hanakawa 99, Hanakawa 01]．この研究において，構造化文書の自動構成のために，文脈に基づく文書構造の意味論と利用者が文書の構造を指定するのに論理式を用いることを提案した [Hanajawa 04]．事実から文脈ディレクトリを構成する処理と，文書部品から構造化文書を構成する処理は基本的に同じであり，本研究は動的文書システムの研究の延長線上にある．大きな違いは，移動先のノードに依存して問い合わせを行う機能が，動的文書システムでは存在しなかったが，文脈ディレクトリシステムでは重要な要素となっている点である．

本発表では，現状の情報の組織化における問題点を指摘し，それらを解決するための基本的な方針を述べ，データベースの問い合わせ結果からディレクトリ構造を構成する方法について説明し，文脈ディレクトリシステムの実装，動作例を紹介する．

2. 情報の組織化を行う上での問題点と解決方針

多くの情報の組織化ツールに共通する 3 つの問題点とそれらに対する抜本的な解決方針を以下に述べる．

情報の組織化が情報の格納時に行われる

情報の組織化の構造は情報を利用するときの条件に依存し多様であるので，情報の格納時点で組織化を行うことは不合理である．たとえば，UNIX のファイルシステムでは，ファイルを格納する時点で，ディレクトリの階層構造を決定し，ファイルをどこかのディレクトリに置かなければならない．そのため，ユーザは未来の参照状況を予測する負担と，実際に参照するときには必ずしも最適ではない構造を使用することを強いられる．

この問題を取り除くには，構造化が弱い，すなわち構成要素が小さく独立性が高いデータベースに情報を格納し，参照時にそのデータベースの情報とユーザの指定に基づき階層構造を構築する方法が有効である．構造化が弱いデータベースとして RDB が候補になるが，本研究では，さらに構造化が弱い Prolog の事実を用いる．

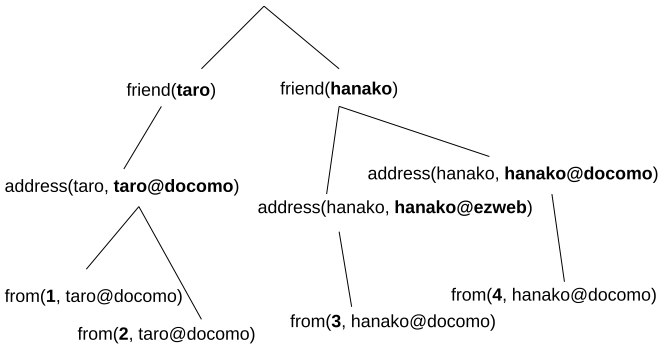

図 1: ノードが事実の階層構造

情報資源（記憶装置上にあるもの）とそれ以外の実体（記憶装置外にあるもの）が区別される

多くのツールにおいて，検索と組織化の対象が情報資源に限定され，データベース化の対象も情報資源の属性情報に限定される．この傾向はメタデータという言葉の普及が示すように，近年顕著になっている．しかし，情報資源とそれ以外の実体には，記憶装置上に存在するかしないかの違いしかなく，両者を区別することは不自然である．たとえば，メールのメッセージファイルは情報資源の一種であり，人間はそれ以外の実体であると考えるのが一般的であるが，メーラのサマリと住所録が似ているように，情報の処理において両者には大きな差は存在しない．両者を区別しないことで，単純で高機能な問い合わせシステムが実現できる．たとえば電子メールファイルを整理するときに，「友人の誕生日にその友人から送られた電子メール」を検索したり，カレンダやアドレス帳とリンクして電子メールを分類することが容易になると考えられる．

しかし，多くのツールは両者の区別を前提としている．検索エンジンなどの検索言語がキーワードを基本要素としているのは，検索対象の種類が限定されるため，検索入力を構成する要素が検索に適合する情報資源に対し真になる 1 引数の述語になり，省略してキーワードで表記可能だからである．また，階層構造のファイルシステムにおいては，情報資源を葉（ファイル）に対応させ，それを格納するディレクトリと明確に区別している．

本研究では，情報資源とそれ以外の実体を区別せず統一的に扱うために，変数と 2 個以上の引数を許す述語による問い合わせ言語と，内部ノードと葉の区別を固定しない階層構造のモデル化を導入する．

階層構造の意味論が明確でない

階層構造は，ファイルシステム，構造化文書，階層メニューなどコンピュータシステムで非常に広範に使われ，多くの人間が日常的に利用しているにもかかわらず，その意味論が明確に示されることは少ない．階層構造に関係したコンピュータの自動処理を可能にするには，人間の直感になるべく近い，単純で明確な意味論を定義する必要がある．

本研究では，階層構造におけるノードを事実に対応させ，ノード間のリンクは特定の述語論理式に含まれる論理積演算子に対応させる．階層構造の意味論に基づきディレクトリ階層を構成する研究として，Semantic File System [Gifford 91] と Logic File System [Padioleau 03] があるが，それらとは，リンクを論理積に対応づける点は共通するが，ノードの解釈が異なっている．

図 1 に階層構造の例を示す．この構造は 9 個のノードが 3 階層に配置されている．ノードは 3 種類の述語を用いて表現した事実に対応する．friend/1，address/2，from/2 は，それぞれ，友人であること，人と E-mail アドレスの関係，メッセージ ID と E-mail の送信元アドレスの関係を表現する．この図の構造は以下の述語論理式と対応している。

```
friend(F), address(F,A), from(M,A).
```

3. 文脈ディレクトリ

Prolog の質問の解をパスに対応させることにより，全解探索の結果から階層構造を生成することができる．質問における副質問の順序は利用者が自由に決定でき，生成される階層の順序に反映される．利用者は質問を編集して再質問し，結果の階層構造を参照することを対話的に繰り返すことができる．その際，質問の右端に副質問の追加を行うと，葉であったものが中間ノードに変わる．中間ノードと葉の区別を固定せず，一度構成された木の葉に部分木を追加可能であることは，従来の階層構造のファイルシステムにはない特徴である．

その階層構造をファイルシステムに保存してナビゲーションの対象とした場合，あるディレクトリに移動して質問を行うことの意味について考える．ある質問をして，その結果に基づきディレクトリ階層が作られた後，その階層内のあるディレクトリに移動して別の質問をしたとすると，両方の質問の論理積が全体の質問であるとみなすことができる．実際に後者の質問に影響を与えるのは，前者の質問の変数への代入であるので，その変数への代入をノードに対応づけておけばよい．このような変数の代入の集合のことを「文脈」，それぞれが文脈を持つ階層構造のノードを「文脈ディレクトリ」と呼ぶ．

文脈ディレクトリが持つ文脈を利用することにより，質問を短くしたり，質問の再利用性を高めることができる．従来のファイルシステムでも，利用者は関心の絞り込みとディレクトリの移動を対応づけていたと思われるが，実際に，ディレクトリの移動によって変化するのは，そこに置かれているサブディレクトリとファイルだけであり，一般的には自動的に動作環境が変わることはなかった．文脈ディレクトリにより，移動先のディレクトリに依存して質問内容が変化させるという機能をファイルシステムに加わえることができる．

4. 実装

実際の複雑な実装を短く記述するのは困難であるので，その基本的なアイデアのみを図 1 のディレクトリ階層を生成する以下のコマンドラインの例で示す．

```
1  mkdir -p ‘echo ’
2     consult(’db.prolog’).
3     friend(F),
4     address(F,A),
5     from(M,A),
6     writeln(friend(F)/
7             address(F,A)/
8             from(M,A)),
9     fail.
10  ’ | pl‘
```

このコマンドラインでは，`echo` で質問を出力し，パイプを経由して Prolog インタプリタ pl に読み込ませている．質問の中の consult は事実のデータベースである `db.prolog` というファイルをロードし，writeln は照合した事実を / でつないだものを出力し，それを mkdir の引数に与えている．このコマ

ンドラインの簡単さは，Prolog のインタプリタの機能が，質問からディレクトリ階層の生成するのに非常に有効に働くことを示している．

この Prolog の機能を応用するという考え方に基づき，プロトタイプシステムを Prolog のメタインタプリタとして実装した．このシステムは，質問の結果から階層構造を深さ優先探索順に表示する機能と，文脈ディレクトリの階層構造をファイルシステムに書き込む機能を持つ．

質問の形式は，以下の表に示す演算子でリテラルを接続した論理式である．/のみが本システムの拡張である．その他の演算子の意味は標準的な Prolog と同じである．なお，演算子の優先順位は，/ ; の順である．

演算子	意味
/	論理積を意味し，階層を生成する．
,	論理積を意味するが階層を作らない．
;	論理和を意味し，枝分かれを生成する．

ノードはデータベースの事実に対応するが，事実の Prolog 表記をそのままノード名に用いると参照時にわかりにくいだけでなく，ファイルシステムに保存した場合のディレクトリ名としては扱いづらい．そのため，フォーマットを指定して Prolog 表記からノード名に変換する機能も実装した．

文脈 (質問結果に含まれる変数への代入の集合) とディレクトリとの対応づけには様々な手法が考えられるが，プロトタイプシステムでは，ディレクトリ名に変数名と代入値のペアのリストを含ませる簡単な方法を採用した．

5. 例

本節では，電子メールメッセージの分類に文脈ディレクトリシステムを応用することで多様で複雑な分類が可能になることを簡単な実験で示す．

この実験で用いた電子メールと人と日付のデータベースモデルを ER 図で表すと図 2 のようになる．

`message_id` は電子メールごとに付けられたユニークば整数である．`name` は個人を識別するユニークな名前で氏名やニックネームなどである．`relation` は利用者からその人をみたときの関係で，友人，仕事仲間，親戚などである．この ER 図から Prolog の述語を以下のように定義した．

- day を除く実体集合について，主キーの値を引数とする 1 引数の述語を定義する
  `email/1`, `address/1`, `person/1`
- 関連集合について，隣接する実体集合の主キーの値を引数とする 2 引数の述語を定義する
  `date/2`, `from/2`, `to/2`, `address_name/2`, `birthday/2`
- 属性について，属性名を述語名，第 1 引数を実体名，第 2 引数を値とする 2 引数の述語を定義する
  `subject/2`, `relation/2`, ...
- 属性のいくつかについて，値の候補を引数とする 1 引数の述語を定義する
  `month/1`, `day_of_week/1`, `relation/1`

事実のデータベースは簡単な perl スクリプトを用いて電子メールのヘッダから作成した．

以下にプロトタイプシステムの動作例を示す．ただし，紙面のスペースを考慮しコマンドの出力は示さない．`q` は質問結果をディスプレイに表示するコマンド，`m` は質問結果からディレクトリ階層を生成するコマンドである．`-d` はこれらのコマンドが出力する深さの制限を指定するオプションである．

1. 2008 年の 4 月の電子メールを日曜日を除いた曜日ごとに分類する．

```
q 'day_of_week(Dow),not(Dow=sun)/
   Y=2008,M=apr,
   day_of_week([Y,M,Dom],Dow),
   date(Mes,[Y,M,Dom]).'
```

2. 2008 年の 4 月の 10 日から 15 日に家族からきた電子メールを日と人の階層で分類する．

```
q 'between(10,15,Dom)/
   date(Mes,[2008,apr,Dom]),
   relation(N,family)/
   address_name(A,N),
   from(Mes,A).'
```

3. 電子メールを 12 月 24 日と 25 日で分け，さらに 2006 年と 2007 年で分ける．

```
q '(Dom=24;Dom=25)/
   (Year=2006;Year=2007)/
    date(Mes,[Year,dec,Dom]).'
```

4. アドレス帳 (名前，関係，誕生日，電子メールアドレス (繰り返しあり) を表示する

```
q 'person(N),relation(N,R),
   birthday(N,[Y1,M,D])/
   address_name(A,N).'
```

5. 電子メールのサマリ (メッセージ ID，日付，From アドレス，題目) を表示する．

```
q 'email(Mes),date(Mes,Date),
   from(Mes,A),subject(Mes,S).'
```

6. 友人が自分の誕生日に送ったメールを友人ごとに分ける．

```
q 'relation(N, 友人),
   birthday(N,[Y1,M,D])/
   date(Mes,[Y2,M,D]),
   address_name(A,N),
   from(Mes,A).'
```

7. 2008 年 1 月 1 日のメールの送り主の一覧を関係に分類して表示する．

```
q -d2 'relation(R)/
       relation(N,R)/
       address_name(A,N),
       from(Mes,A),
       date(Mes,[2008,jan,1]).'
```

8. 友人達のディレクトリを作成し，その中の太郎のディレクトリに移動して，太郎から送られたメールの一覧をアドレスごとに分けて表示する．

```
m 'relation(F, 友人)'
ls
cd 'relation(太郎, 友人):F=太郎'
q 'address_name(A,F)/
   from(Mes,A).'
```

以上の例から，このシステムが簡単な質問で多様な個体に関する検索と階層構造の指定ができることとと，カレントディレクトリの文脈に依存させた質問ができることが確認できた．

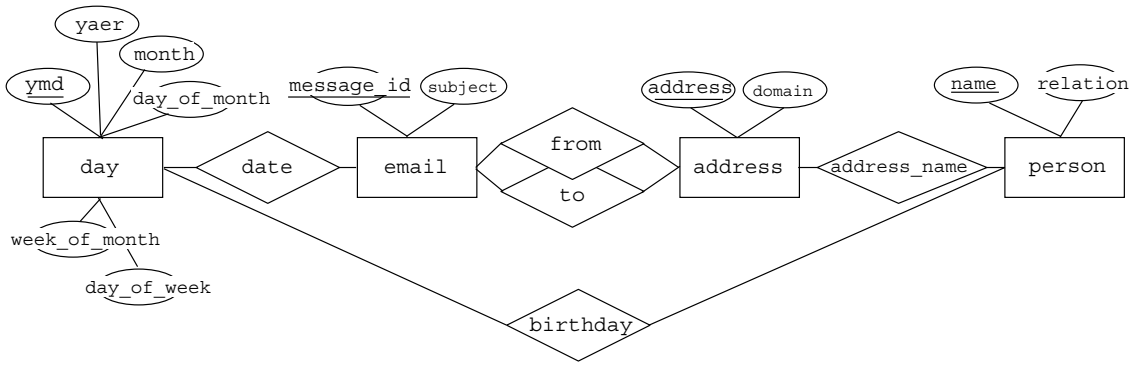

図 2: データベースモデル

6. 今後の課題

プロトタイプシステムは，基本的な機能を持つが実用性においては不十分である．実用化に向けて以下の課題がある．

1. 文脈の保存方法

   プロトタイプシステムでは，変数の代入である文脈を文字列表記に変換してディレクトリ名に含ませた．しかしこの方法ではディレクトリ名が長くなりすぎる問題があった．ディレクトリごとに文脈を記述した隠しファイルを置く方法や，文脈を格納できるようにファイルシステムを拡張する方法などが考えられる．

2. ファイル名とデータベースの自動生成

   従来のファイルシステムでは，同じファイル名でもファイルが置かれているディレクトリが異なれば別のファイルとみなされるが，文脈ディレクトリシステムでは，ファイルの名前はシステム内でユニークでなければならない．必然的にファイル名は長くなり命名に労力が伴うようになるので，ファイルの種類ごとの規則に従い自動的にファイル名を生成する機能が必要である．多くのファイルは，ヘッダなどに種類ごとに定まったフォーマットで属性情報を持ち，インターネット上のメタデータが利用可能な場合もあるので，事実のデータベースの生成は自動化が可能である．また，Prolog を基本とするシステム間では，述語名と引数の形式の標準化により，データベースの共有も可能である．

3. RDB の利用

   Prolog のデータベースではなく，広く普及している RDB を利用することは，利用範囲を広げるために重要である．問い合わせ言語は Prolog の質問とし，RDB をバックエンドに置き，Prolog の質問を SELECT 文に，結果のテーブルを事実の列に変換する方法が考えられる．

4. ユーザインターフェース

   キーボードで Prolog の質問を書くには，述語名と引数の並びを覚えておき，多数の文字をタイプする必要がある．この労力を軽減するためにコマンド入力と呼ばれる機能を開発している．コマンドは簡単な文字列あるが，カレントディレクトリの文脈に応じて設定された規則に従い質問に変換される．しかし，このようなコマンドラインインターフェースは一般的なユーザには好まれないので，GUI による質問の入力支援も必要である．階層構造に様々な変換を行ったグラフィカルな出力，質問入力と結果の参照とが有機的に結びついた対話インターフェースなども今後研究する必要がある．

7. おわりに

本発表では，情報資源以外の情報も含む事実を要素とするデータベースと Prolog の質問から，ディレクトリ階層を生成するシステムを提案した．このシステムで生成されたディレクトリは変数への代入の集合である文脈を持ち，質問の解釈は文脈に依存する．このシステムでは，質問の構造で生成する階層構造を指定することができ，文脈を利用することにより質問の簡潔性と再利用性を高めることができる．この能力を示す質問例の紹介も行った．

参考文献

[Hanakawa 99] Hanakawa, H., Takeda, H., and Nishida, T.: Construction of a Dynamic Document Using Context-Free Pieces. Proc. SEKE'99. pp. 212–216 (1999)

[Hanakawa 01] Hanakawa, K., Yoshikawa, M. and Uemura, S.: Social Writing and Individual Reading with Dynamic Documents. Proc. KES2001 pp. 1598–1608 (2001)

[Hanajawa 04] 花川賢治, 波多野賢治, 天笠俊之, 宮崎純, 植村俊亮: 文書構築のためのツールとしての Prolog, 第 3 回情報科学技術フォーラム (FIT2004) E-006 pp.123-124 (2004)

[Gifford 91] Gifford, D. K., Jouvelot, P., Sheldon, M. A., and O'Toole Jr, J. W.: Semantic File Systems, Proceedings of the 13th ACM Symposium on Operating Systems Principles, pp. 16–25 (1991)

[Padioleau 03] Padioleau, Y. and Ridoux, O.: A Logic File System, USENIX 2003 Annual Technical Conferencem, pp. 99–112 (2003)

[Apple 04] Apple Computer Inc.: Mac OS X Tiger: Spotlight, http://www.apple.com/macosx/, (2004)