1D5-5

単純に層化された文脈木によるデータベースとファイルシステムの統合

Integrating Databases and File Systems with Simply Layered Context Trees

花川 賢治[*1]
Kenji Hanakawa

[*1]大阪府立大学工業高等専門学校
Osaka Prefecture University College of Technology

Techniques using a simply layered context tree for integration of searching and organizing files are proposed. A simply layered context tree has a sequence of layers consisting of nodes which represent variable bindings extracted from a Prolog database answer. A node of a context tree has a context. In other words, interpreting of a question depends on the current node. A user specifies an ordering of layers using a sequence of variable names, which is translated into a conjunction of predicates used as a Prolog question. A context tree is visualized using nested panes according to user's specification.

1. はじめに

まず最初に，この研究でのデータベースとファイルシステムの意味について整理しておきたい．

日常的に PC に格納されるファイルを効果的に組織化するためには，多様なファイル以外の実体についての情報が必要である．たとえば，4 月が誕生日の友人から来たメール，ある人が出席した会議の議事録，大阪市内のレストランで食事をしたときに撮った写真などのファイルを組織化するためには，メール，議事録，写真だけでなく，友人，組織，レストランの情報が必要になる．ここでのデータベースは，ファイルに直接的な情報だけでなく，ファイルに関係したファイル以外の実体についての情報も格納する．最近よく話題になるファイルの直接的な属性であるメタデータと比較すると，ここでのデータベースは，それよりも広い範囲の情報を格納する．

ファイルシステムの長所は二つある．一つは，記憶装置の容量の拡大に伴い莫大になったファイルを整理するのに階層構造の組織化を活用している点である．もう一つは，ユーザが自由に構造を決定できる点である．GUI，Web ページ，クラスタリングによる自動分類など階層を利用した情報システムは非常に多くあるが，ほとんどのシステムで，エンドユーザは自由に構造を決定することができない．ただし，現状の多くのファイルシステムは，フォルダの生成を手作業で行うので，階層構造の構築に非常に労力がかかるという欠点がある．

本研究で提案する手法では，階層構造は，ユーザが作成するのではなく，ユーザの質問とデータベースから取り出した情報に基づいて，システムが自動生成する．この手法を使えば，ユーザの労力が減るだけでなく，多様な要求に応じた最適な階層構造を得ることができる．一方，データベースの構築はユーザにとって新たに発生する作業である．しかし，現状においても実際には多くのユーザは同様の作業を行っているのである．ハードディスクのフォルダの名前と構造には，ファイルに関係した豊富な情報が反映されている．ただし，その情報は人間は読み取れても，コンピュータは読み取ることができない．

先行研究 [花川 2004] から継続して本研究でも中心的な役割をする，三つのアイデアについて以下に述べる．

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

ファイルとそれ以外の実体を等しく扱う ファイル以外の実体もデータベースの範囲に含めると，ファイルとそれ以外の実体を区別せず同等に扱う方が合理的である．そうすると，従来のファイルシステムのフォルダとファイルの区別は意味がなくなる．つまり，ファイルが必ずしも葉とは限らず，逆にそれ以外の実体が中間ノードとは限らない．また，あるノードが中間に置かれるか葉になるかは流動的である．具体例をあげると，ある月のカレンダーを表示させると，日付は，年–月–週–曜日の下の最下位層に置かれるが，ユーザが質問を追加して，日付のセルに会議のスケジュールメモ，メールメッセージを置くと，日付は葉から中間ノードに変わる．さらに，それらの下の層に会議の出席者，メールの添付ファイルを置くと，スケジュールメモとメールメッセージも葉から中間ノードに変わる．

質問の論理構造を検索結果の組織化に反映させる データベースとファイルシステムの統合とは，検索と組織化の統合であり，検索結果をユーザの指示にしたがって階層構造に配置することである．その配置の指示の表現手段を独立して設けるのではなく，検索のときの質問の構造を反映させる手法を用いる．基本的には，質問は複数の副質問の論理積とし，各副質問の問い合わせ結果を一つの層に置き換え，上から下に階層を構成する．この方法では，検索と組織化の方法を同時に一つの記述で表現できるので，ユーザの入力を減らすことができる．

ノードに文脈を持たせる ここでの文脈とは，システムが生成した階層構造上でカレントノードを設定して，さらに次の質問を行ったとき，質問の解釈を，カレントノードのパスに依存させることである．この機能により，対話的に質問を行う場合に，段階的に必要な情報を絞り込むことができる．文脈の利用は，ファイルマネージャの対話処理だけでなく，アプリケーションソフトウェア間の連携にも利用できる．カレントノードのパスは，大部分の OS では，相対パスの起点の意味でしかないが，文脈を利用したシステムでは，環境変数のように汎用的にプログラムの動作を変えるのに使用される．たとえば，カレンダ上で，ある日付を選択して，メーラを起動したときに，その日付のメールのみがメーラのサマリに表示される．このとき，選択された年，月，日がカレントノードのパスに相当し，その情報が文脈としてカレンダからメイラに渡される．

自由な組織化のためには，問い合わせ言語が双方向性を持つことが要求される．Prolog がその要件を満たすので，本研究では Prolog を採用する．Prolog は，述語と論理記号だけを用いた非常に単純な言語であるが，それでもエンドユーザにとっては習得は困難である．エンドユーザによる自由な組織化のためには，Prolog で書かかなくてよいユーザインターフェースが必要である．

そこで，文脈木の構造に単純な層化という制約を与え，変数の名前を並べるだけの入力方式を提案する．単純な層化とは，一つの層に一つの変数だけを対応させた層化である．エンドユーザが変数の並びを入力すると，あらかじめ定義された述語と変数の関係に基づき，そこからシステムは述語の並びを自動生成する．

単純な層化は，階層構造の視覚化においても有効である．多くのファイルシステムにも GUI が備えられていて，フォルダを木の節や入れ子の箱で表現することができる．しかし，文脈木システムでは，フォルダとファイルの区別がなくなり，階層構造が多様化するので，より自由な視覚化をユーザが選択する必要性がある．単純に層化されていると，ユーザはそれぞれの層に対して視覚化の方法を指定するだけでよいので，非常に簡潔な視覚化の定義が可能になる．

以降，メールの分類のシナリオを考察し，文脈木による表現と単純な層化について述べ，Prolog の質問の自動生成と階層構造のユーザ定義の手法を提案する．

2. 電子メールの分類のシナリオ

この節では，日常的な情報が格納されたデータベースを使ってユーザが「最近友人から来た電子メールを調べる」ときのシナリオについて考察する．

大部分のメーラでは，このような要求を達成するのに，フォルダを作成し，振り分け条件に，メールヘッダの From:のアドレスが友人のアドレスであることを設定する．その方法は，ファイルであるメールメッセージとそれを格納するフォルダを明確に区別している．ここでは，それとは対照的に，メールメッセージ，個人，メールアドレス，日付，標題は，個体として同等に扱う．そのことにより，多くのメーラよりも，ユーザにとって自由度が高い情報の検索と組織化が実行可能である．

データベースは以下の機能を持つと仮定する．

- データベースには日常的な情報 (住所録· カレンダ· メールサマリ) が格納されている．
- データベースに質問を行うと，全ての答えが返される．
- ユーザは返された答えから関心のある一つを選ぶことができる．
- 答えを選択して次の質問を行うと，データベースは選択に関係した答えを返す．

作業の手順の一例を以下に示す．

- 友人の個人名を質問する (A-Q)
- 一人の友人 (たとえば ichiro) を選択する (A-S)
- [その友人の] メールアドレスを質問する (B-Q)
- 一つのメールアドレスを選択する (B-S)
- 曜日を質問する (C-Q)
- 一つの曜日 (たとえば日曜日) を選択する (C-S)
- 今月の[選択した曜日の] 日付を質問する (D-Q)
- 一つの日付を選択する (D-S)
- [選択されたメールアドレスと日付の] メールメッセージ ID と表題を質問する (E-Q)
- 一つのメールメッセージを選択する (E-S)

括弧内の末尾が Q の項目は質問で，S の項目は選択である．質問の[ ] に囲まれた部分は文脈で，それより前の選択から得られる．ユーザは条件を加えて答えの範囲を狭くすることを望むと仮定すると，文脈は自明であることが多い．その場合は，文脈の記述は，ユーザが明示的に質問に入れなくても，システムが自動的に補うことは可能である．上の例では，すべての文脈のユーザ入力は省くことが可能である．

上に示した例には，以下の任意に変更可能な要素があり，それらに変更を加えることにより，他の作業手順を生成することができる．

選択 選択は任意に行うことができる．選択を行わないと，複数のそれぞれの答えを文脈とする質問に変換される．たとえば，B-S の選択を除くと，E-Q の答えは ichiro の複数のアドレスについてのメールメッセージとなる．同様に C-S の選択を除くと，E-Q の答えは全曜日のメールメッセージとなる．

連結と分割 間に選択をはさまないときには，複数の質問を連結することが可能である．たとえば，A-Q と B-Q を連結すると，個人名とメールアドレスのペアの形式で答えが返される．同様に，C-Q と D-Q を連結すると，月，曜日，日付の組の答が返される．どの質問を連結するか，逆の言い方をすれば分割するのかは任意で，連結と分割の方法によって答えが変わることはない．

質問の順序 A から F までのすべての質問の並び順は任意である．質問の並び順により答えが変わることはない．たとえば，AB と CD を入れ替えた場合，送信元と日付のどちらを先に指定するかの違いだけである．しかし，状況によっては，順序を選べるのは便利である．A と B を入れ替えた場合，最初の B ですべてのアドレスが表示されて，それが多すぎるときに，A を使って友人だけに絞り込むような状況で有効である．

3. 文脈木による表現と単純な層化

前節に述べた例での個体と個体間の関係を図 1 に示す．図の長方形は個体集合で，エッジは個体間の関係を意味する．関係を Prolog の述語で表現し，Prolog の事実を格納することでデータベースを構築することができる．

前節の質問を Prolog で表現すると以下のようになる．

- $relation(Person, friend)$. (A-Q)
- $address(Person, Address)$. (B-Q)
- $day\_of\_week(Day\_of\_week)$. (C-Q)
- $calendar(2013, 'Jun', Day\_of\_week, Date)$. (D-Q)
- $from(Message, Address), date(Message, Date),$ $subject(Message, Subject)$. (E-Q)

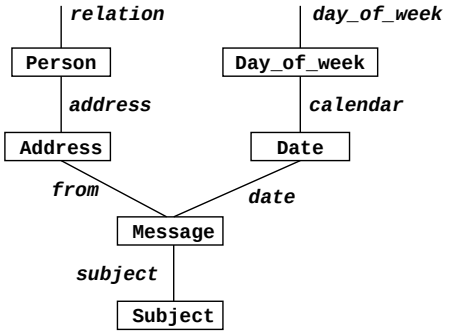

図 1: 電子メールの分類に関係した個体間の関係

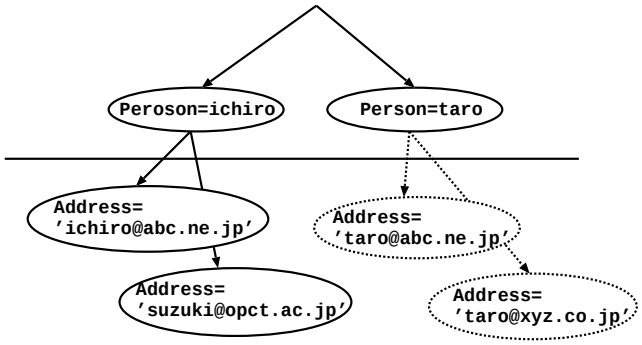

図 2: 質問の構造を反映した階層

Prolog では，質問の結果は変数と定数のバインドで表現される．Prolog インタプリタは，通常の対話的な使用では最初に見つかった答えだけを返すが，強制失敗によるバックトラック機能を利用すれば全解探索を実行させることができる．

基本的には，Prolog の述語には，あらかじめ引数が定数か変数の区別は設定されていない．たとえば，address の第 1 引数を定数またはバウンド変数，第 2 引数を変数にすると，ある個人のアドレスを質問することができ，逆に第 1 引数を変数，第 2 引数を定数またはバウンド変数にすると，あるアドレスの所有者を質問することができる．両方が変数の場合は，データベースに格納されている全ての個人名とメールアドレスのペアが出力される．ただし，引数が定数かバウンド変数であることを条件とする述語も存在する．

Prolog の答えから階層構造を構成するためには，ノードが何かが重要になる．ここでは，ノードは変数へのバインディングであると考え，ノードのラベルは”変数 = 定数”の書式で表現する．”変数 = 定数”には 2 重の意味があり，文脈の表現にも使われる．カレントノードのパスにあるノードのラベルを質問に含ませることにより，質問に文脈を加えることができる．そのとき = は unify 述語と解釈される．

階層構造は，質問の論理積を縦の関係，論理和を横の関係とみなすことで構築する．つまり，質問に出現するアンバウンド変数に対応した層を生成し，出現順に上から下に配置する．

図 2 は A-Q と B-Q を AND 演算子で結合した質問に対する階層である．A-Q により変数 Person に友人である ichiro と taro がバインドされた二つのノードから成る一つの層が生成される．さらにその下に B-Q により変数 Address に対応する層が生成される．

質問を分割して，A-Q で上の階層を生成した後，Person=ichiro のノードを選択し，B-Q を質問すると，システムは選択されたノードを質問に含めるので，質問は Person=ichiro,address(Peroson,Address) になり，Address の答えは ichiro のメールアドレスになる．その結果，図 2 の破線の部分を除いた階層構造が得られる．

データベースの情報を用いないで，ノードを生成することも可能である．たとえば，X=1，X=2，X=3 の 3 個のノードを生成する方法は 2 種類ある．一つは組み込み述語を使って，between(1,3,X) を呼び出す方法である．もう一つの方法は手作業で (X=1;X=2;X=3) という質問を入力する方法である．セミコロンは Prolog の OR 演算子なので，この質問は 3 個の答えを返す．このようなデータベースと無関係なノードの作成は，従来のファイルシステムの mkdir，新規フォルダと同様の機能である．

述語の論理積による Prolog の質問から階層構造を生成する場合には，必ず単純に層化される．単純でない層化は，たとえば，$(X = a; Y = b), (Z = c; W = d)$ のような，特殊な質問をした場合に限られる．この質問は上の層が X と Y，下の層が Z と W の変数が対応しており，単純な層化ではない．

4. 質問の自動生成

これまでは，質問から階層をどう構成するかを議論してきたが，この節では，逆に，階層が与えれたときに質問をどう構成するかを議論する．

質問の自動構成の前提となる条件を以下のように設定する．

個体集合名・変数名・層名を一体とする． 変数名は，ユーザにとって意味のある名前でなければならない．システムが自動生成する質問に含まれる変数名は，ユーザが入力した層の名前でもある．この制約により，複雑な質問は生成不可能な場合がある．たとえば「ichiro と同じ日にメールを送ってきた人」は Person に対応する変数が 2 個必要になるので，対応することはできない．

文脈を最大限に使って解をせまく絞る． ほとんどのユーザの望むのは，可能なだけ条件を絞って少ない個数の解を得ることであるので，システムは層の変数が関係する述語を可能な限り質問に加える．

個体集合間の関係は一位に決定可能である． 現実世界では，個体間の関係が複数存在することがある．たとえば，メールアドレスとメールメッセージの間には，From と To の関係がある．しかし，システムはどちらを選択すべきか判断できないので，一つだけを選択してシステムに登録する必要がある．

以降，階層の並びから述語の並びを構成するアルゴリズムについて述べる．3 個以上の引数の述語があっても，議論の本質は大きく変わらないので，ここでは，議論を単純にするために，述語は 1 引数か 2 引数であるとする．1 引数の述語は，要素となる個体を返し，述語名は変数名と一致する．ここではドメイン述語と呼ぶことにする．多くの変数はそれぞれのドメイン述語を持つが，要素数が無限か多すぎるためにドメイン述語を定義できない変数もある．2 引数の述語は，両引数が定数かバウンド変数であることが条件となる述語と，片方がアンバウンド変数でもよい述語がある．ここでは，それぞれをテスト述語，連想述語と呼ぶ．

基本的なアルゴリズムは，ユーザ入力の変数名の列から変数名を 1 個ずつ取り出し，述語に変換することを繰り返す．変換の過程は，選択肢生成と制約充足から構成され，最初に選択肢

表 1: 生成される述語の例

変数	選択肢生成	制約充足
Person	personドメイン述語	なし
Address	address 連想述語	なし
Message	from 連想述語	date 連想述語

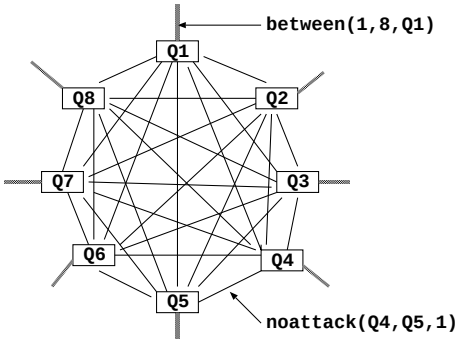

図 3: エイトクイーンの述語のグラフ

表 2: 視覚化ルール

上位レベル		個人名	アドレス	ID	標題
下位レベル	個人名	アドレス	ID	標題	日付
ルール 1		水平	垂直	水平	水平
ルール 2	垂直	垂直	垂直	–	–

図 4: 文脈木の視覚化

生成のための述語を選び，次に制約充足のための述語を選ぶ．

選択肢生成のための述語は，ドメイン述語か隣接するもう一方の引数がバウンドである連想述語である．該当する述語が存在しない場合は，この処理は失敗する．選択肢生成に必要な述語は一つだけである．制約充足のための述語は，隣接するもう一方の引数がバウンドである 2 引数の述語 (テスト述語でも連想述語でもよい) すべてである．

図 1 の電子メールのデータベースにおいて，Date が文脈として与えられていて，入力が Person，Address，Message の並びの場合，生成される述語は表 1 のようになる．

テスト述語を使った例としてエイトクイーンの問題を例示する．Q1 から Q8 は盤面の 8 列に対応した変数で，そこにバインドされる整数がクイーンの置かれる行を表すとする．図 3 はそれらの変数と 1 から 8 の整数を返すドメイン述語である between，二つのクイーンが対角線上にないと成功するテスト述語である noatack の関係を示す．ユーザが自由な順序で与えた変数の並びに対して述語を自動生成することができる．たとえば，Q1 と Q2 の後に Q3 を与えた場合，$between(1,8,Q3), noatacck(Q1,Q3,2), noattack(Q2,Q3,1)$ を生成する．

5. 文脈木の視覚化

ファイルマネージャでフォルダの階層構造を視覚化する方法はいくつかあり，ユーザは選択することができる．しかし，その指定をフォルダ単位に行うなど，効率の良い視覚化の指定はできない．それに対し，単純に層化された構造では，視覚化の指定は層単位に宣言的に行うことができる．ここでは，ノードを水平または垂直に配置する非常に簡単な例を示す．

画面上のウインドウが，他の要素を格納することができる長方形の領域であるペインとテキストを表示するためのラベルの 2 種類のオブジェクトから構成される場合に，部分木ごとの処理を以下に示す．なお，この手順は中間ノードの場合で，葉の場合は，1 で上位ペインのみ生成し，2 を行ったのち終了し，3 と 4 は行わない．

1. 根のレベルとその一つ下のレベルのために二つペインを生成し，ルール 1 に従いそれらを配置する．
2. 根のノード名のラベルを生成し，上位ペインに配置する．
3. 一つ下のレベルの部分木のためにペインを生成し (通常複数)，ルール 2 に従い下位ペインに配置する．
4. それぞれの部分木に対して再帰的にこの処理を呼び出す．

ルール 1 もルール 2 も水平，垂直の 2 種類のみとし，個人名，アドレス，ID，標題，日付の階層を視覚化する方法として，以下の表 2 の指示が与えられた場合，図 4 に示すようなウインドウが生成される．

この言語は，Treemap[Shneiderman 09] と木構造に従って再帰的に長方形を分割する点に少しの類似点があるが，長方形の大きさが重要でないこと，表示属性を層ごとに定義できることなどが異なる．この言語の特徴は，視覚化の仕様の簡潔な記述にあり，ルールを豊富にすることにより，アイコン，クロス表，散布図など多様な視覚化の定義が可能になる．

6. おわりに

PC 上のファイルの検索と組織化のための日常的な情報を格納したデータベースに対する検索結果をユーザの質問の構造に基づいて階層化した文脈木について述べた．また，質問を Prolog で書く方法に代わる，変数名の並びから Prolog の質問を自動生成する手法と，文脈木の視覚化を簡潔に記述する宣言的言語を提案した．

参考文献

[花川 2004] 花川賢治: 文脈ディレクトリシステム, 第 23 回人工知能学会全国大会,2009

[Shneiderman 09] Ben Shneiderman: Treemaps for space-constrained visualization of hierarchies, http://www.cs.umd.edu/hcil/treemap-history, 2009