半構造データのための対応データモデルの提案と問い合わせ言語の設計

横尾　徳保†　　重松　保弘†

† 九州工業大学大学院工学研究科　〒 804–8550 福岡県北九州市戸畑区仙水町 1–1
E-mail: †{yokoo,shige}@cs.comp.kyutech.ac.jp

あらまし　現実世界を分析するとその分析結果はグラフ構造で表現されることも多く，またスキーマが静的に決定できないことも少なくない．このようなときリレーショナルデータモデルを適用すると，データの表現が複雑になる場合やデータに不整合が生じる場合がある．また，人間の知識は意味ネットワークのようなグラフ構造で表現されることもあることから，グラフ構造を持つデータをそのままモデル化／記号化することで，概念理解との親和性やデータの整合性が向上すると考えられる．本稿では，グラフ構造を持つデータをそのまま記号化することができ，またスキーマの動的な変化に柔軟に対処することができるデータモデルとして対応データモデルを提案し，操作言語の記述例から表現力や記述性について議論するとともに，問い合わせ言語の設計について述べる．対応データモデルでは，データをラベル付き対応（ラベル，始点，終点の 3 つ組の集合）としてとらえる．データの構造記述自体は非常に簡素であるが柔軟性および明瞭性の高いモデルである．対応データモデルでは，リレーショナルデータモデルのように対応代数を定義しており，体系的なデータの操作を実現した．

キーワード　半構造データ，ラベル付き対応，データモデル，データ操作言語

A Proposal of Correspondence Data Model for Semi-structured Data and a Design of Query Language

Noriyasu YOKOO† and Yasuhiro SHIGEMATSU†

† Graduate School of Engineering, Kyushu Institute of Technology
Sensui–cho 1–1, tobata–ku, Kitakyushu–shi, Fukuoka, 804–8550 Japan
E-mail: †{yokoo,shige}@cs.comp.kyutech.ac.jp

Abstract　In this paper, we propose a Correspondence Data Model for semi-structured data which has the network data structure and whose schema is not fixed, and also discuss the requirements of its query language. The Correspondence Data Model makes schema flexible, therefore the proposed data model has good compatibility with the semi-structured data. In the proposed data model, data were represented with the labeled correspondence which consisted of the set of three-tuples (a label, an initial vertex, and a terminal vertex), and data was operated with the correspondence algebra which was designed as the data manipulation language such as the relational one in the relational data model.

Key words　Semi-structured Data, Labeled Correspondence, Data Model, Data Manipulation Language

1. はじめに

現実世界を分析するとその分析結果が複雑なグラフ構造を持ち，リレーショナルデータモデルではその表現が困難な場合がある．また，リレーショナルデータモデルはスキーマが固定されてしまうことから，いわゆる “かたいデータ” を取り扱うことになり，スキーマが頻繁に変化するようないわゆる “やわらかいデータ” の取り扱いには適さない．近年，この “やわらかいデータ” に対する要求が増してきており，半構造データと呼ばれる “schema-less” というキーワードで表現されるようなデータのための研究が盛んに行われている [1].

一方，筆者らはこれまで，アルゴリズムの自然なプログラム化を目的として，集合指向言語 SOL の設計ならびに開発を行ってきた [2]〜[9]．SOL と類似した集合指向の言語としては SETL [10] や ISETL [11] が知られている．SOL は構文規則に集合，写像／対応，述語論理の概念を取り入れたプログラミング言語であり，プログラム実行中に写像／対応を動的に生成することができるなどの特徴を持つ．このような特徴から SQL のホスト言語としての応用 [3] やフローグラフのインターバル解析への応用 [4] などにおいてその有効性が確認されている．ま

た，SOL が提供するラベル付き写像／対応を利用することで，マルチメディアデータを含めた，スキーマに縛られないデータの構造化を行うプログラムを容易に記述できることが示されており [9]，グラフ構造を持つ半構造データを効率的に操作できることが期待できる．

そこで，SOL において半構造データの体系的な取り扱いを実現するデータモデルとして，グラフ構造を持つデータをそのまま記号化することができ，またスキーマの動的な変化に柔軟に対処することができる対応データモデルを提案する．対応データモデルでは現実世界の構造記述にラベル付き対応を用いる．ラベル付き対応とは，1 対多の対応関係にラベルを付けたものであり，実質的にラベル付き有向グラフで現実世界の構造記述を行うことと等価である．また，対応データモデルではリレーショナルデータモデルと同じようにデータ操作言語を定義しており，柔軟で簡便なデータの操作を体系的に実現している．なお，対応データモデルは SOL に特化したデータモデルではないため，幅広い応用が期待できる．本稿では，対応データモデルの概要について述べた後，データ操作言語の記述例からその表現力や記述性について議論するとともに，問い合わせ言語の設計について述べる．

2. ラベル付き対応

対応データモデルでは現実世界の構造記述にラベル付き対応を用いる．まず，ラベル付き対応の概念について説明する．

2.1 対応とラベル付き対応

A と B をある集合とする．ある規則 f によって，A の元 a に対してそれぞれ 1 つずつ B の部分集合 $f(a)$ が定められる（$a \neq a'$ に対して，$f(a)$ と $f(a')$ に同じ要素が含まれてもよい）とき，その規則 f のことを A から B への対応と呼び，A の元 a に対して定まる B の部分集合 $f(a)$ を f による a の像と呼ぶ．また，f が A から B への対応であることを式 1 のように記述する．対応 f におけるすべての像が B の元であるとき，f は写像と呼ばれる．

$$f : A \to B \tag{1}$$

この対応 f にラベル集合 L を与え，ラベル $l(\in L)$ 毎に異なる対応 $f\{l\}$ として扱えるようにしたものをラベル付き対応と呼び，式 2 ように記述する．

$$f\{L\} : A \to B \tag{2}$$

2.2 ラベル付き対応とラベル付き有向グラフ

ラベル付き対応は，式 2 における A, B, L をそれぞれラベル付き有向グラフの始点の集合，終点の集合，ラベル付き矢（以降，ラベル矢）の集合ととらえることで，ラベル付き有向グラフに変換することができ，また逆にラベル付き有向グラフをラベル付き対応に変換することもできる．ラベル付き対応の例として，ラベル付き対応 “K 高校のある生徒の家族” を図 1 に示す．ここでは概念的にラベル付き有向グラフ表現を用いる．

なお，以降はラベルのない対応は扱わないため，“ラベル付き対応” のことを “対応”，“ラベル付き有向グラフ” のことを “有向グラフ” のように “ラベル付き” は省略することにする．

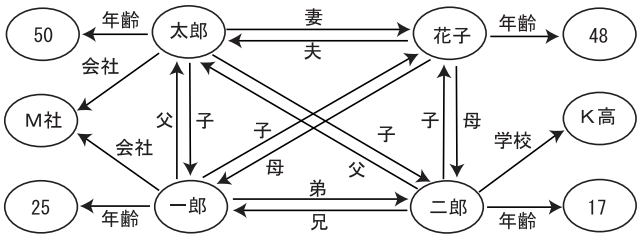

図 1　ラベル付き対応 “K 高校のある生徒の家族”

Fig. 1　A Labeled Corespondance on “The family of a K highschool student”

3. 対応データモデル

対応データモデルでは現実世界を対応の集合とみなし，その対応に対するデータ操作を対応代数で行う．本節では，対応による構造記述，対応代数，および対応の追加，削除，修正操作について述べる．

3.1 対応による構造記述

前述したように，対応はそれと等価な有向グラフとして表現でき，逆に有向グラフを対応として表現することができる．したがって，現実世界を対応で表現するときは有向グラフの形で表現すればよいことになる．対応データモデルでは現実世界の構造を有向グラフとしてとらえ，その構造を対応を用いて表現するため次の特徴がある．

（1） ノード毎に独自のラベル矢を持つことができる．

（2） 各ノードは同一ラベルのラベル矢を複数持つことができ，また任意のノードに向かう同一ラベルのラベル矢が複数存在することも許される．

（3） グラフ構造を概念理解そのままに構造化できる．

(1) の特徴により，例えばある学校の生徒のデータベースを作成するとき，生徒毎に異なったラベルを定義することが可能になる．また (2) の特徴により，1 対多の関係をそのまま表現することができ，例えば “趣味” のような複数の事柄が該当するような場合も問題が生じない．また，このとき任意のノードに向かうラベル矢にも制約がかからないため，実質的に多対多の関係を簡便に記述できる．さらに (3) の特徴から，現実世界の概念理解をそのまま対応で表現できるため，自然で柔軟なデータの構造化が可能となり，同時にデータの整合性の保持が容易になる．

なお，現在のところデータ自身がデータの型情報を持つものとする．ノードに関してはデータの型は特に限定せず任意のデータ型のデータを持つことができるものとし，ラベルに関しては文字列型に限定するものとする．また，データの一意性はユーザが保証するものとする．

3.2 対応におけるスキーマ

対応データモデルでは現実世界を対応で表現する．そのために前述したようにノード毎に固有のラベルを持たせることが可能である．つまり，同種のノードでもノード毎に固有のラベルを持たせることができる．しかし，仮に新しいラベルを追加できない（予め登録されたラベルしか利用できない）としてしまうと柔軟性が損なわれてしまう．

例えば，図 1 において “二郎が柔道をしていて三段である” ことが明らかになった場合，その事実を即座に追加できると非常に効果的であるが，そのためにはデータベース設計時には予期できないラベルの追加が必要になる．このため対応のスキーマ（どのノードがどのようなラベルを持つか等の対応の枠組）を固定してしまうとこのような操作は不可能になる．そこで，対応データモデルにおいてはスキーマグラフのような概念は導入せず，現実世界の構造の変化に柔軟に対処できるようにした．

3.3 データ操作言語

対応データベースでは，現実世界の情報を対応を用いてモデル化／記号化し，計算機に取り込む．このデータベースに対し，任意の対応，ノードまたはラベルを取り出したり，新規データの追加，あるいは不要になったデータの削除を行うために，対応に対するデータ操作言語が必要となる．そのデータ操作言語として対応代数を定義した．対応代数においては，表 1 に示す対応演算を規定した．以降，対応演算をその演算結果から 2 種類に分類して説明する．

なお，対応 $f\{L\} : A \to B$ をラベル $l \in L$ と始点 $a \in A$ および終点 $b \in B$ の組の集合と考え，式 3 のように定義する．また，対応における 1 組の始点，終点，ラベルのことを対応要素，対応要素の集合のことを対応と呼ぶこととし，対応データベースは対応の集合からなるものとする．また，便宜上説明には有向グラフの用語を用いる．

$$f\{L\} : A \to B = \{(l,a,b) \mid f\{l\}(a) = b, l \in L, a \in A, b \in B\} \tag{3}$$

3.3.1 演算結果が対応である演算

（1） 対応和演算

対応 $f\{L1\}$ と $g\{L2\}$ の和 $f\{L1\} \cup g\{L2\}$ とは，$f\{L1\}$ と $g\{L2\}$ に含まれる対応要素を併せ持つ対応を生成する演算であり，式 4 のように定義する．

$$f\{L1\} \cup f\{L2\} = \{(l,a,b) \mid (l,a,b) \in f\{L1\} \vee (l,a,b) \in g\{L2\}\} \tag{4}$$

（2） 対応差演算

対応 $f\{L1\}$ と $g\{L2\}$ の差 $f\{L1\} - g\{L2\}$ とは,$f\{L1\}$ の対応要素から $g\{L2\}$ の対応要素を取り除いた対応を生成する演算であり，式 5 のように定義する．ただし，始点と終点が共に等しくてもラベルが異なっていればその対応要素は除外の対象から外される．

$$f\{L1\} - f\{L2\} = \{(l,a,b) \mid (l,a,b) \in f\{L1\} \wedge (l,a,b) \notin g\{L2\}\} \tag{5}$$

（3） 対応共通演算

対応共通演算とは，2 つの対応 $f\{L1\}$ と $g\{L2\}$ において共通の対応要素で構成される対応を生成する演算であり，式 6 のように定義する．

$$f\{L1\} \cap f\{L2\} = \{(l,a,b) \mid (l,a,b) \in f\{L1\} \wedge (l,a,b) \in g\{L2\}\} \tag{6}$$

（4） 選択演算

選択演算とは，対応から始点または終点が条件を満たす対応を生成する演算であり，始点に対して条件を課す domain 選択と終点に対して条件を課す range 選択の 2 種類の演算がある．ただし，使用する条件は真偽が常に定まるものでなければならない．domain 選択演算および range 選択演算はそれぞれ式 7，式 8 のように定義する．なお，式 7，式 8 に示す定義において θ は 2 項関係演算子であり，C は対応の始点または終点との比較対象である．

$$f[C\theta\{L\}] = \{(l,a,b) \mid (l,a,b) \in f\{L\} \wedge a\theta C\} \tag{7}$$

$$f[\{L\}\theta C] = \{(l,a,b) \mid (l,a,b) \in f\{L\} \wedge b\theta C\} \tag{8}$$

選択演算は，対応からある特定の対応を生成する演算であり，これは対応データベースに対するデータ操作を行う場合に最も基本となる操作である．また，ラベルに対する選択演算は定義していないが，対応 $f\{L1\}$ の定義から $L1'(\subset L1)$ を与えることで，特定のラベルからなる対応を生成することができる．

3.3.2 演算結果が対応（要素）の構成要素である演算

ここに分類される演算の結果は，対応の構成要素である始点の集合，終点の集合，またはラベルの集合である．また，これら対応の構成要素に対しては，通常の集合と同様に和，差，共通集合演算が利用できるものとする．

（1） domain 演算

domain 演算とは，対応 $f\{L\}$ における対応要素の始点の集合を取り出す演算であり，$dom[f\{L\}]$ と記述し，式 9 のように定義する．domain 演算は，有向グラフの視点で考えると，ラベル矢が出ているノードを取り出す演算であるといえる．

$$dom[f\{L\}] = \{a \mid \wedge(l,a,b) \in f\{L\}\} \tag{9}$$

（2） range 演算

range 演算とは，domain 演算とは逆に対応 $f\{L\}$ における対応要素の終点の集合を取り出す演算であり，$ran[f\{L\}]$ と記述し，式 10 のように定義する．有向グラフの視点で考えると，range 演算も domain 演算と同様にノードを取り出しているが，ラベル矢が指しているノードを取り出すという点で異なる．

$$ran[f\{L\}] = \{b \mid \wedge(l,a,b) \in f\{L\}\} \tag{10}$$

（3） label 演算

label 演算とは，対応 $f\{L\}$ における対応要素のラベルの集合を取り出す演算である．この演算には任意の始点から出ているラベル矢のラベル集合を求める domain ラベル演算とそれとは

表 1 対 応 演 算

Table 1 The correspondence operations

演算結果	分類	演算名	説明
対応	二項演算	対応和演算	対応（要素）の和集合を求める
		対応差演算	対応（要素）の差集合を求める
		対応共通演算	対応（要素）の共通集合を求める
	単項演算	domain 選択演算	始点の条件を満たす対応を求める
		range 選択演算	終点の条件を満たす対応を求める
		label 選択演算	ラベルの条件を満たす対応を求める
構成要素	–	domain 演算	対応から始点の集合を取得する
		range 演算	対応から終点の集合を取得する
		label 演算	対応からラベルの集合を取得する

逆に任意の終点へ向かうラベル矢のラベル集合を求める range ラベル演算の 2 種類がある．式 11，式 12 のように定義し，この演算の引数では始点，終点の 1 つのノードだけでなく複数のノード（始点，終点の部分集合）を指定する．なお，N はノードの集合を表す．

- domain ラベル演算

$$lab[N = f\{L\}] = \{l \mid (l, a, b) \in f\{L\} \wedge a \in N\} \quad (11)$$

- range ラベル演算

$$lab[f\{L\} = N] = \{l \mid (l, a, b) \in f\{L\} \wedge b \in N\} \quad (12)$$

ラベル演算はラベルを取り出すことができるため，あるノードともう 1 つのノード間のラベル，つまり 2 ノード間の関係を求める問い合わせを記述することができる．また，ラベル演算は始点または終点を指定し対応からラベルを取り出す演算であるから，有向グラフの視点で考えた場合，有向グラフのラベルを取り出す演算であるといえる．

3.4 データ検索

データ検索とは所望の情報をデータベースから取り出す操作である．これを行うためにはデータベースに対して問い合わせ文を与える必要がある．対応データモデルでは対応演算を用いて，問い合わせ文を記述する．ここでは対応演算による問い合わせ文の記述例を示す．

対応演算の演算結果として得られるものは，対応，始点の集合，終点の集合，ラベルの集合であり，これらの演算を任意に組み合わせて問い合わせ文を記述することになるが，記述例における注意事項を 3 つ挙げる．

- 対応の和，差，共通演算，選択演算の演算結果，および $f\{L\}$ は対応を表す．
- domian 演算，range 演算の演算結果は対応の始点または終点の集合を表し，ラベル演算の演算結果はラベルの集合を表す．
- domain 選択演算において $f[C = \{l\}]$ のように “=” で比較を行う場合は $f\{l\}[C]$ と略記する．つまり，この $f\{l\}[C]$ は C の像を表すのではなく対応 $f[C = \{l\}]$ を表す．

図 2 に対応 “K 高校と T 高校の生徒の部活動状況” を示す．ここでは，この対応 “K 高校と T 高校の生徒の部活動状況” と図 1 に示した対応 “ある K 高校の生徒の家族” から “昇と同種の部活動を行っている K 高校の生徒の父と母を求めよ” という問い合わせを生成していく．以降，対応 “ある K 高校の生徒の家族” を対応 “家族”，対応 “K 高校と T 高校の生徒の部活動状況” を対応 “部活動状況” と呼ぶことにする．

（1） 昇の部活を特定するために対応 “部活動状況”{ 部活 }[昇] の range をとる．

$$ran[\text{“部活動状況”}\{\ \text{部活}\ \}[\text{昇}]] \quad (13)$$

（2） K 高校の生徒を見つけるために対応 “部活動状況”{ 生徒 }[K 高] の range をとる．

$$ran[\text{“部活動状況”}\{\ \text{生徒}\ \}[K\ \text{高}]] \quad (14)$$

（3） 昇と同種の部活をしている K 高校の生徒を特定するために対応 “部活動状況”{ 部活 }[式 14]（K 高校の生徒とその部活を示す対応）から，その終点が式 13（昇の部活）と等しいものを選択し，その domain をとる．

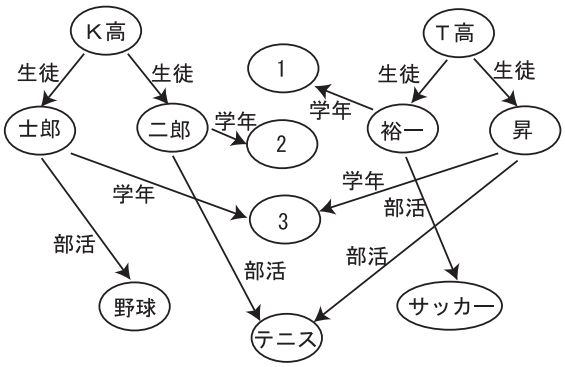

図 2　対応 “K 高校と T 高校の生徒の部活動状況”

Fig. 2　A correspondence on “The club activity of K highschool students and T highschool students”

$$dom[(\text{“部活動状況”}[\text{式 14}])[\{\ \text{部活}\ \} = \text{式 13}]] \quad (15)$$

（4） 昇と同種の部活をしている K 高校の生徒が式 15 と特定でき，対応 “家族” から “家族”{ 父, 母 }[式 15] を取り出す．

$$\text{“家族”}\{\ \text{父, 母}\ \}[\text{式 15}] \quad (16)$$

したがって，問い合わせ “昇と同種の部活動を行っている K 高校の生徒の父と母を求めよ” を示す問い合わせ文は式 17 のようになる．

$$\begin{aligned}&\text{“家族”}\{\ \text{父, 母}\ \}[\\ &dom[(\text{“部活動状況”}[ran[\text{“部活動状況”}\{\ \text{生徒}\ \}[K\ \text{高}]]])\\ &[\{\ \text{部活}\ \} = ran[\text{“部活動状況”}\{\ \text{部活}\ \}[\text{昇}]]]]]\end{aligned} \quad (17)$$

ここでは，条件を満たす問い合わせ文の一記述例を示したが，対応の和や差を用いることで様々なビューでの対応を求めることも可能である．

3.5 データ操作

対応データモデルではデータ追加，削除，修正は対応要素単位で行うためそれぞれ対応追加，対応削除，対応修正と呼ぶ．ここではこれらのデータ操作について述べる．

3.5.1 対応追加

対応に対して新たな対応（対応要素）を追加することを対応追加という．有向グラフの視点で考えると，新たなラベルもしくは新たなノードとラベルを追加する操作である．この操作は，単純に追加する対応を加えることで実現できる．既存の対応を $D\{L\}$，追加する対応を $Add\{L_{add}\}$ とすると，対応追加は対応演算を用いて次のように記述できる．

$$D\{L\} \leftarrow D\{L\} \cup Add\{L_{add}\} \quad (18)$$

3.5.2 対応削除

対応から任意の対応（対応要素）を削除することを対応削除と呼ぶ．有向グラフの視点で考えると，典型的にはラベル矢を削除する操作となる．対応データモデルでは対応そのものだけではなく対応の構成要素である始点の集合，終点の集合，ラベ

ルの集合に対しても和，差，共通集合の集合演算を許しているため，対応削除を行う方法は次の 3 種類がある．

（1） 対応の削除

既存の対応 $D\{L\}$ から対応 $Del\{L_{del}\}$ を削除するとすると，対応の削除は次のように記述できる．

$$D\{L\} \leftarrow D\{L\} - Del\{L_{del}\} \tag{19}$$

（2） 始点の集合，終点の集合の要素の削除

既存の対応 $D\{L\}$（始集合 A，終集合 B）から始点 a，終点 b を削除するとすると，それぞれ次のように記述できる．

$$A \leftarrow A - \{a\} \tag{20}$$

$$B \leftarrow B - \{b\} \tag{21}$$

（3） ラベル集合の要素の削除

既存の対応 $D\{L\}$（始集合 A，終集合 B）からラベル l を削除するとすると，次のように記述できる．

$$L \leftarrow L - \{l\} \tag{22}$$

3.5.3 対応修正

対応の始点の集合，終点の集合，ラベルの集合の任意の要素の値を他の値に修正することを対応修正と呼ぶ．有向グラフの視点で考えると，ノードもしくはラベルの値を修正する操作に相当する．対応演算では始点の集合や終点の集合，ラベルの集合の要素を単独で修正できる演算がないため，この操作を行うためには複数の演算を組み合わせる必要がある．具体的には修正対応を既存の対応に加え，さらに被修正対応を取り除く．この操作から分かるように対応修正とは対応追加と対応削除の組み合わせで実現できる．既存の対応を $D\{L\}$，その部分対応である被修正部分対応を $Mod\{L_{mod}\}$，修正部分対応を $M\{L_m\}$ とすると対応の修正は式 23 のように記述できる．

$$D\{L\} \leftarrow D\{L\} \cup M\{L_m\} - Mod\{L_{mod}\} \tag{23}$$

4. 他のデータモデルとの比較

最も代表的なデータモデルであるリレーショナルデータモデルと比較しながら，対応データモデルを評価する．比較する項目としては次の 4 つが挙げられる．

- データ表現力
- データ操作言語
- データ操作
- データ更新時異状

ここではデータ表現力とデータ更新時異状の 2 項目について比較を行う．比較の方法としては現実世界 “K 高校の生徒の特技と家族” をそれぞれ 1 リレーション，1 対応でモデル化しそれらを比較するという手法をとる．図 3 および図 4 に各モデリング結果を示す．

4.1 データ表現力

対応データモデルとリレーショナルデータモデルを現実世界の表現力という観点から比較する．

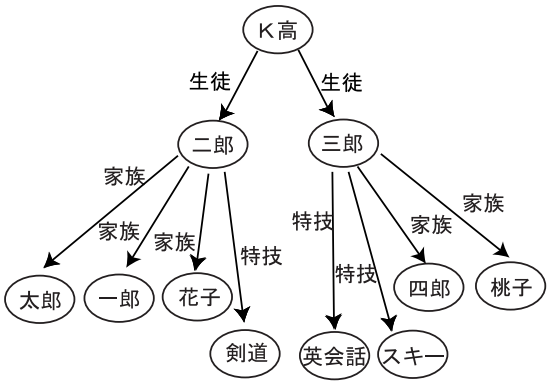

図 3 リレーション “K 高校の生徒の特技と家族”

Fig. 3 A relation on “The family and ability of a K highschool student”

生徒	特技	家族
二郎	剣道	太郎
二郎	剣道	花子
二郎	剣道	一郎
三郎	英会話	四郎
三郎	英会話	桃子
三郎	スキー	四郎
三郎	スキー	桃子

図 4 対応 “K 高校の生徒の特技と家族”

Fig. 4 A correspondence on “The family and ability of a K highschool student”

図 4 から分かるようにリレーション “K 高校の生徒の特技と家族” ではある生徒が特技を複数持っている場合や家族が多い場合，二郎の特技が剣道であるという情報が 3 つのタップルに現れたり，三郎の家族に四郎がいるという情報が 2 つのタップルに現れたりと，冗長な表現になってしまう．これはリレーショナルデータモデルではリレーションは第 1 正規形でなければならないためであり，これ以上タップルを減らすと現実世界の情報が失われてしまう．これを改善するには正規化する必要がある．

一方，対応データモデルにおいては 1 対多の関係を簡便に記述できるため，特殊な操作を必要とせずに表現でき，冗長な情報も現れない．また，対応データモデルではノード毎に固有なラベルを持たせることが可能であるため，例えば生徒毎に任意の情報を持たせることができる．そのため，このようなデータに対しては対応データモデルの方がリレーショナルデータモデルより，現実世界をわかりやすく表現でき，適合性が高いと考えられる．

4.2 データ更新時異状

データ更新時異状とはデータベースの更新時に発生する異状のことをいう．データ更新操作はデータの追加，削除，修正の 3 種類があり，データ更新時異状もそれに対応し 3 種類ある．ここでは，これらのデータ更新時異状について対応データモデルとリレーショナルデータモデルの比較を行う．

（1） データ追加時異状

データ追加時異状はリレーショナルデータモデルではタップル挿入時異状と呼ばれる．リレーショナルデータモデルにおいて，

新たに “K 高校の生徒の五郎の特技がテニスである” という情報を追加する場合，タップル（五郎，テニス，ー）を挿入することになるが，このタップルは主キーである “家族” が空値（ー）であるためキー制約に抵触し，挿入することができない．これがタップル挿入時異状である．これを回避するにはリレーション “K 高校の生徒の特技と家族” に対し高次の正規化を行う必要がある．

一方，対応データモデルにおいてはデータの追加は対応の追加として行う．対応を構成する始点，終点，ラベルのいずれが欠けても対応にはならないため，始点や終点，ラベル単体での追加はできない．しかし，データベースに追加するデータは何らか意味を持っており，対応の最小単位である対応要素は常に構築できる．したがって，対応データモデルにおいてはデータ追加時異状が起きることはない．

（2） データ削除時異状

データ削除時異状はリレーショナルデータモデルではタップル削除時異状と呼ばれる．リレーション “K 高校の生徒の特技と家族” において “戸籍上，三郎と桃子が家族でなくなった” 場合，タップル（三郎，英会話，桃子）と（三郎，スキー，桃子）を削除することになる．しかし，これらを削除するとリレーション “K 高校の生徒の特技と家族” から三郎に関する情報が全て削除されてしまう．またそれを避けるためにタップルを（三郎，英会話，ー），（三郎，スキー，ー）と修正しようとしてもキー制約に触れてしまう．これがタップル削除時異状である．この異状を避けるにはタップル挿入時異状の場合と同じく高次の正規化を行う必要がある．

対応データベースにおけるデータ削除は特定の対応の削除である．そしてその方法として削除する対応を直接指定する方法，始点や終点もしくはラベルを削除することで間接的に対応を削除する方法がある．同様のデータ削除操作はそれぞれ次のように行うことになる．

- 対応の削除

対応 “K 高校の生徒の特技と家族” から対応要素（家族，三郎，桃子）の削除を行う．

- ノードの削除

対応 “K 高校の生徒の特技と家族” の始点集合，終点集合からノード “桃子” を削除することでノード “桃子” に関する全ての対応要素を削除する．

- ラベルの削除

対応 “K 高校の生徒の特技と家族” のラベル集合からラベル “家族” を削除することでラベル “家族” に関する全ての対応要素を削除する．

対応を直接指定する方法では該当する対応要素のみの削除であるため他の必要な情報まで削除してしまうことはない．一方，ラベルを削除する方法の場合，ラベル “家族” を削除するため他の生徒の家族の情報も削除してしまう．また，この例ではノードの削除でも特に異状は生じないが，ノードの削除においてもノードに関する全ての対応要素を削除するという性質上，他の必要な情報まで削除してしまう．しかし，ノードおよびラベルを削除する方法は関連のある対応要素を一括して削除できる利点としてとらえることもでき，用途に応じて使い分けることで有効に活用できるものと考えられる．

（3） データ修正時異状

リレーショナルデータベースではこの異状はタップル修正時異状と呼ばれる．リレーション “K 高校の生徒の特技と家族” において “二郎が特技は剣道でなく柔道であった” 場合，その修正はタップル（二郎，剣道，太郎）だけでなくタップル（二郎，剣道，花子），（二郎，剣道，一郎）にも波及してしまう．これがタップル修正時異状である．これを避けるためには高次の正規化が必要である．

一方，対応データモデルでは，対応要素（特技，二郎，剣道）を修正するだけでよい．また対応関係を複数修正する場合でもその対応要素をまとめて 1 つの対応といて扱うことができるためリレーショナルデータモデルのような異状は生じないと考えられる．

このようにリレーショナルデータモデルでは高次の正規化を行うことでデータ更新時異状を回避するのに対し，対応データモデルではそのような操作を行うことなくデータ更新時異状を回避できる．このことより，対応は現実世界を表現する上で，高次の正規化を行ったリレーションと同等の表現力があるといえる．

5. 問い合わせ言語の設計

対応データモデルの提案にともない，対応データモデルのための問い合わせ言語を設計している．ここでは，問い合わせ言語の設計方針と検討中の構文とコマンドを簡単に紹介する．

設計方針としては，次の 5 つが大きな柱として挙げられる．

（1） SQL ライクの言語設計

（2） XML Use Case への対応

（3） グラフ構造を意識した Path 表現の実現

（4） グラフを視覚的に操作できる GUI の実装

（5） 集合指向言語 SQL への組み込み実装

(1)〜(4) はそれぞれ “利用しやすさ”，“表現能力”，“記述性”，“操作性” に対する方針を示すものである．(1) は広く利用されている問い合わせ言語であることから参考にする意義は大きいと考えられ，また (2) への対応も幅広い応用を模索する上では必要不可であると考えられる．ただし，ある用途に特化した方が実用的なシステム構築につながるがることも考えられる．(3), (4) はグラフ構造のデータを扱うことから必要不可欠であると考えられる．XQuery [12] のような Path 表現を取り入れ，グラフを効果的に表現することが必要である．グラフ構造のデータを操作する上で視覚的な操作は不可欠であると考えられるが，コマンドによるデータ操作との連携が重要であると考える．(5) に関しては，実装に際して SQL の利点が活用できる反面，機能的に不十分な点もあり，SQL の言語仕様の拡張を含めた検討が必要である．

検討段階の基本的なコマンドと問い合わせ言語の構文の一部をそれぞれ表 2，図 5 に示す．問い合わせ言語の構文は拡張 BNF 記法を用いて表す．コマンドは複数の対応データベースから対象とする対応データベースの選択などを行う Main 状態，

表 2　コマンド一覧と説明

Table 2　The commands and its explanation

(a) Main 状態：

演算名	説明
create	対応データベースの作成
del	対応データベースの削除
connect	対応データベースへの接続，Command 状態へ移行
ls	存在する対応データベースの一覧表示

(b) Command 状態：

演算名	説明
set	変数に問い合わせ結果や演算結果の格納
in	対応の選択，Operation 状態へ移行
add	対応の追加
del	対応の削除
ls	存在する対応の一覧表示，対応名指定で対応要素の一覧表示

(c) Operation 状態：

演算名	説明
add	対応（要素）の追加
del	対応（要素）の削除
ls	存在する対応要素の一覧表示
readfile	対応のファイルからの読み込み
writefile	対応のファイルへの書き出し
Nodelist	ノードの一覧表示
Labellist	ラベルの一覧表示

(d) 共通：

演算名	説明
help	コマンドの一覧表示
quit	Operation 状態からの Command 状態への移行，Command 状態からの Main 状態への移行，main 状態におけるシステムの終了
END	システムの終了

選択した対応データベースに対する操作を行う Command 状態，選択された対応に対して操作を行う Operation 状態の 3 つの状態においてそれぞれコマンドを定義している．

5.1　問い合わせ言語の設計上の課題

紹介した問い合わせ言語の構文は，手続き的な構文になっているが，宣言的な記述も必要であると考えられる．特に，有向グラフとしてデータを操作することから，グラフの向きを考慮した問い合わせの記述や木構造にはないグラフ構造特有の問い合わせを宣言的に行えるようにする必要があると考えられる．また，ループや対応の入れ子などの表現，一貫性制約の記述などについても考慮する必要がある．また，グラフ構造を持つデータを扱うことから，GUI で問い合わせができることは有効であると考えられるが，CUI 環境においても簡便にデータを操作できるように，まずコマンドベースの問い合わせを充実させる．

```
<Query>::=
    <Correspondence>{<State>}
<State>::=
    <StateElement>( ,<StateElement> )* | ε
<StateElement>::=
    start:<Node> | path:<Condition> | end:<Node>
<Correspondence>::=
    <OneCorrespondence> ( ,<OneCorrespondence> )*
<OneCorrespondence>::=
    <CorrespondenceName>  | <Variable> | "{"<Query>"}"
<Node>::=
    ( not )? ( ( <Node> ( ( and | or ) <Node> )*
        | "("<Node>")" ) | <SimpleNode> )
<SimpleNode>::=
    <NodeName>
  | ( /<RegularExpression>/ | all | <NodeFunction> )
        ( ( > | >= | != | < | <= ) <Num> )?
<Condition>::=
    (not)? ( <Condition> ( ( and | or ) <Condition> )*
        | "("<SimpleCondition>")"
<SimpleCondition>::=
    (<LabelName> | /<RegularExpression> /)
        ( ( to | from ) <Node> )?
  | . ( /<RegularExpression>/ | LabelName )
        ( . (/<RegularExpression>/ | LabelName ) )*
```

図 5　問い合わせの言語の構文（抜粋）

Fig. 5　A part of the syntax of the query language

6. おわりに

本稿では，現実世界の自然な構造記述およびスキーマの変化に柔軟に対処可能な対応データモデルを提案した．また，体系的なデータ操作を実現する対応代数を定義し，その対応代数を用いて様々な問い合わせの記述が可能であることを示した．対応データモデルは現実世界を対応を用いてモデル化する．概念的には，ノード毎に固有のラベルを持つことができ，また 1 対多（多対 1 になることを妨げない）の関係が許されることから，スキーマに縛られない自由度の高いデータモデルであるといえる．また，グラフ構造で表現される現実世界の概念理解をそのまま記号化できることから，自然で理解し易いモデル化／記号化が可能である．さらにスキーマフリーであることから現実世界の変化に対し柔軟にそのデータ構造を変えることができる．また，正規化の必要がないことも利点であり，データ更新時異状が発生しにくいといいう特徴も持つ．

今回は，研究が導入段階であることもあり，リレーショナルデータモデルとの比較しか行っていないが，OEM(Object Exchange Model) [13], [14]，EGM(Edge-labeled Graph Model) [15], [16] などのデータモデルとの比較も必要であると考えられる．また，グラフ構造を持つ半構造データの研究に関しては GUI ベースの研究も多いことから，今後は問い合わせ言語の設計においても，GUI ベースの問い合わせ言語 [18]〜[20] の機能を参考にしながら，設計／実装を進めていく．また，実際の応用分野を検討し，具体例を用いて他の問い合わせ言語 [12], [17]〜[20] との比較評価を行う予定である．

文　　献

[1] 田島 敬史：半構造データのためのデータモデルと操作言語，情処論データベース，Vol. 40，No. SIG3(TOD1)，pp. 152-170，1999.

[2] 重松 保弘，吉見 康一，吉田 将：集合指向言語 SOL とその言語処理系の開発，情処論，Vol.30，Num.3，pp.357-365，1989.

[3] 重松 保弘，興那覇 誠，吉田 将：集合指向言語 SOL のデータベースへの応用，情処論，Vol.33， Num.8，pp.1041-1051，1992.

[4] 重松 保弘，吉田 将：集合指向言語 SOL の拡張とフローグラフのインターバル解析への応用，情処論，Vol.34，No.2，pp.229-238，1993.

[5] 迫江 義彦，重松 保弘：不定型とラベル付き写像記法の導入による集合指向言語 SOL の仕様拡張とその評価，情処論，Vol.38，No.8 pp.1662-1665，1997.

[6] N. Yokoo，and Y. Shigematsu：Development and Extension of Set Oriented Language (SOL), Proc. of the 2nd Int. Conf. on Parallel and Distributed Computing and Networks, pp.64-69, 1998.

[7] 横尾 徳保，重松 保弘：集合指向言語 SOL のマルチメディアデータへの対応，情処論，Vol.42，No.SIG7，pp.88，2001.

[8] N. Yokoo，and Y. Shigematsu：Development of Remote Execution Environments for SOL as Interactive Applications on the Web，Proc. of the IASTED Int. Conf. on Computers and Advanced Technology in Education, pp.328-333, 2002.

[9] 横尾 徳保，重松 保弘：集合指向言語 SOL のマルチメディアデータ型と手続きに関する拡張とその評価，情処論，Vol.43，No.6，pp.1930-1939，2002.

[10] J.T. Schwartz, R.B.K. Dewar，E. Dubinsky, and E. Schonberg: Programming with Sets–An Introduction to SETL，P.493，Springer-Verlag, New York, 1986.

[11] N. Baxter, E. Dubinsky, and G. Levin：Learning Discrete Mathematics with ISETL，p.416, Springer-Verlag, New York, 1988.

[12] XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/

[13] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom: Object exchange across heterogeneous information sources, Proc. of the IEEE Int. Conf. on Very Large Data Bases (VLDB), pp.132-141, 1994.

[14] R. Goldman, and J. Widom: Data Guides:Enabling Querey Formulation and Optimization in Semistructured Database, Proc of the 23rd VLDB Conf., pp.436-445, 1997.

[15] P. Buneman, S.B. Davidson, G. Hillebrand, and D. Suciu: A Query Language and Optimization Techniques for Unstructured Data, Proc. of ACM SIGMOD Conf. on Management of Data, pp.505-516, June 1996.

[16] P. Buneman, S.B. Davidson, M. Fernandes, and D. Suciu: Adding Structure to Unstructured Data, Proc. of Int. Conf. on Database Theory(ICDT), 1997.

[17] 宝珍 輝尚：グラフに基づくデータベースに対する集合指向の統合演算, 情処論，Vol. 35，Num. 3，pp. 444-452，1994.

[18] S. Flesca, F. Furfaro, and S. Greco: XGL: a Graphical Query Language for XML, Proc. of Int. Database Engineering and Applications Symposium, 2002.

[19] M.P. Consens, and A.O. Mendelzon: GraphLog: a Visual Formalism for Real Life Recursion, Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databse Systems, pp.404-416, 1990.

[20] T. Houchin: DUO: Graph-based Database Graphical Query Expression, Proc. 2nd FarEast Workshop on Future Database Systems, pp.286-295, 1992.