社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DBS－141 (14)
2007－GN－ 62 (14)
2007／1／26

# 対応データベースのためのデータベース言語の設計

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.ecs.kyutech.ac.jp

**あらまし** 半構造データはスキーマの制約が緩やかなため，従来の構造化データと比べて柔軟性に富んでいる．しかし，そこには計算機に“自動で”処理させるという暗黙の要件があり，必ずしもその柔軟性が十分に活かしきれない場面が存在すると考えられる．例えば，小学校の課外授業における情報収集や旅行日記などを考えた場合，余計な枠組みにとらわれず“簡単”かつ“自由”に「どんどんデータを収集できる」ことが望まれる．本研究では，このような要求を満足する対応データモデルに基づくデータベースのためのデータベース言語 GRQL（GRaph Query Language）を新しく設計した．対応データモデルは概念的にはラベル付き有向グラフと等価であり，GRQL はグラフ構造をもつデータを容易にデータベース化できるように設計してある．本稿では，GRQL の設計について述べるとともに，GRQL の特徴的な select 文の記述例を示しながら，その構文と機能について詳しく説明する．また，代表的な問合せ言語と比較することで GRQL の特徴を明らかにする．

**キーワード** 半構造データ，データベース言語，ラベル付き対応，ラベル付き有向グラフ

# Design of a Database Language for Correspondence Database

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.ecs.kyutech.ac.jp

**Abstract** In this paper, we propose a database language based on the Correspondence Data Model for semi-structured data, which we had proposed, and also discuss the requirements of the database 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 triple (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, Database Language, Labeled Correspondence, Labeled Directed Graph

## 1. はじめに

近年，“schema-less”というキーワードで表現される半構造データ [1], [2] が注目されている．半構造データはスキーマの制約が緩やかなため，その構造が不規則であったり既知ではないなどの特徴をもっており，従来の構造化データと比べて柔軟性に富んでいる．半構造データに関しては，セマンティック Web や情報配信に関連してさまざまな研究が行われているが，その根幹には計算機に「自動で処理させたい」あるいは「効率的に処理させたい」という暗黙の要求 (願望) がある．これに対して，余計な制約にとらわれず“簡単”かつ“自由”に「どんどんデータを収集できる」ことが望まれる場合もある．例えば，小学校の課外授業における情報収集や旅行日記などが挙げられる．もちろん，このような用途においても蓄積したデータを取り出せないような無秩序なデータになっては困るが，このような用途では柔軟性を最優先させた利用者主導のデータ収集を目的としたシステムが効果的であると考える．

そこで，筆者らはこのような用途を想定してデータモデルおよびデータベース言語に関する研究を行ってきた [3]〜[5]．現実世界においては，セマンティックネットワークのようにグラフとして構造化できるデータが多数存在しており，またこのようなグラフ表現は人間の概念理解との相性が良いことから，対応データモデルを提案している．対応データモデルの構造自体は単純な 3 つ組であるが，任意の 3 つ組を自由に追加できる自由の高さと，そのようにしてできた複数のグラフから簡単な検索ができるように配慮されている．

本研究では，対応データモデルに基づくデータベースのためのデータベース言語 GRQL（GRaph Query Language）を新しく設計した．対応データモデルは，グラフ構造をもつデータを概念理解そのままに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である [3]. GRQL はこの特性を継承しており，グラフ構造をもつデータを容易にデータベース化可能である．本稿では，まず対応データモデルの概要について述べ，対応代数について解説する．次に，データベース言語 GRQL について述べる．ここでは，select 文の典型的な記述例を示しながら，その構文と機能について説明する．また，代表的な問合せ言語と比較することで GRQL の特徴を明らかにする．

**関連研究**

半構造データのためのデータモデルとしては，OEM(Object Exchange Model) [6], [7] や EGM(Edge-labeled Graph Model) [8], [9] がよく知られている [10]．例えば，OEM に基づいたデータベースシステムの問合せ言語である Lorel においては，オブジェクトの型に依存しない比較やワイルドカード等の導入により厳格な型付けが緩和され，また正規表現とパス表現の導入により宣言的な問合せができるようになったことから，半構造データに柔軟に対応できる．このようなデータモデルに基づいたオブジェクト指向データベースや XML データベース [11] の登場で半構造データの取り扱いが可能になったが，その基本的な構造は木構造であり，グラフ構造のデータの取り扱いには最適でない．最近は XML(Extensible Markup Language) [12] が普及し，その問合せ言語 XQuery [13] も利用されるようになった．XML は，文書やデータの意味や構造を記述するためのマークアップ言語の一つであり，自己記述の特性をもつ半構造データの表現やデータの交換を目的として広く利用されている．ただし，XML もその構造は階層的な木構造がベースとなっており，前述の問題を内包している．

また，対応データモデルと類似したデータモデルとしては RDF(Resource Description Framework) [14] が挙げられ，その問合せ言語 SPARQL [15] も標準化されている．RDF は，Web 上のデータを自動かつ効率的に計算機で処理させるためのメタデータの記述を目的としている．リソースの関係をリソース，プロパティおよびプロパティの値の 3 つの要素で表現する構造自体はシンプルな 3 つ組モデルである．しかし，処理の自動化を主たる目的としていることから，意味や定義の曖昧性をなくすためにリソースには URI 参照による名前付けが必要となり，リテラルの使用にも制限がかかる．プロパティの値にも URI 参照によるリソースの指定もできるが，名前空間の制約を強く受けることになる．基本的に利用者が RDF ボキャブラリを知っておく必要があり，システムの柔軟性を要求する本研究の目的とは合致しない．また，3 つ組モデルとしては，この他にも松永による情報の “三つ組” モデル [16] などもあるが，こちらは，それぞれの情報本体ごとに名前と諸属性を 3 つ組で表現している点で対応データモデルと異なるが，データを「どんどんため込んでさっと出す」というコンセプトは対応データモデルと共通するものがある．

## 2. 対応データモデル

対応データモデルにおいては，現実世界を対応の集合とみなして記号化する．また，その対応に対するデータ操作を対応代数で行う．ここでは，まずラベル付き対応の概念について簡単な説明と用語の定義を行い，対応による構造記述，データ操作言語，およびその記述例について述べる．

### 2.1 対応とラベル付き対応

集合論において，$A$ と $B$ を集合としたとき規則 $f$ 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 $f(a)$ が定められるとき，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 [17]．一般に，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．この対応 $f$ に対して新たにラベル集合 $L$ を付与し，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ぶ．これを式 1 のように表現する．

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

式 1 における $L$，$A$，$B$ はそれぞれラベル付き有向グラフのラベル付き矢の集合，始点の集合，および終点の集合ととらえることができ，ラベル付き対応はラベル付き有向グラフと等価である．

なお，記述を簡略化するために，本稿では “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 “グラフ” と略記することにする．

### 2.2 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る．例えば図 1, 図 2 が対応の具体的な例になる．これらの図から，(1) ノード毎に独自のラベル矢をもつことができる，(2) 各ノードは同一名のラベル矢を複数もつことがができ，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，ことが容易にわかる．RDF グラフにおいても同じような特徴があるといえるが，対応データモデルにおいては，その利用目的から名前空間の概念がなく，またノードには自由に名前を付けることができるようになっている点が異なる．対応データモデルのこのような特徴は利用者に名前の一意性を委ねることになるので，厳密な語彙の管理は放棄することになるが，任意の 3 つ組を自由に追加していくことができる非常に柔軟性の高いモデルである．

### 2.3 データ操作言語

対応データベースでは，対応を用いて現実世界の情報をモデル化／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．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 2 のように定義する．また，対応における 1 組の始点，終点，ラ

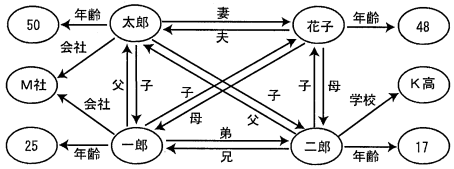


図 1　対応 “K 高校のある生徒の家族”

Fig. 1　A correspondence on “the family of a K high school student”.

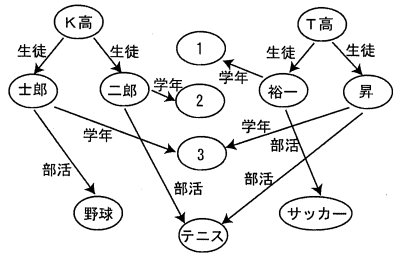


図 2　対応 “K 高校と T 高校の生徒の部活動状況”

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

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．

$$f\{L\} : A \to B =$$
$$\{(l,a,b) \mid f\{l\}(a) = b, l \in L, a \in A, b \in B\} \quad (2)$$

### 2.3.1　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もつ対応を生成する演算であり，式 3 のように定義する．

$$f\{L1\} \cup f\{L2\} = \{(l,a,b)$$
$$\mid (l,a,b) \in f\{L1\} \vee (l,a,b) \in g\{L2\}\} \quad (3)$$

（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 4 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．

$$f\{L1\} - f\{L2\} = \{(l,a,b)$$
$$\mid (l,a,b) \in f\{L1\} \wedge (l,a,b) \notin g\{L2\}\} \quad (4)$$

（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 5 のように定義する．

$$f\{L1\} \cap f\{L2\} = \{(l,a,b)$$
$$\mid (l,a,b) \in f\{L1\} \wedge (l,a,b) \in g\{L2\}\} \quad (5)$$

（4）　選択演算

選択演算とは，対応から始点，終点またはラベルが条件を満たす対応を生成する演算であり，domain 選択，range 選択，ラベル選択演算をそれぞれ式 6，式 7，式 8 のように定義する．なお，これらの定義において $\theta$ 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．

$$f[C\theta\{L\}] = \{(l,a,b) \mid (l,a,b) \in f\{L\} \wedge a\theta C\} \quad (6)$$
$$f[\{L\}\theta C] = \{(l,a,b) \mid (l,a,b) \in f\{L\} \wedge b\theta C\} \quad (7)$$
$$f[\{L'\}] = \{(l',a,b) \mid (l',a,b) \in f\{L\} \wedge l'\theta C\} \quad (8)$$

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．

### 2.3.2　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．

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

（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．

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

（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合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．

### 2.4 データ検索

データ検索を行うにはデータベースに対して問合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，問合せ文を記述する．ここでは対応演算による問合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合せ文を記述することになる．記述上の留意点を 3 つ挙げておく．(1) 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．(2)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．(3)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 1，図 2 の対応を例として取り上げ，これらの対応から "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る" 問合せを生成していく．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．

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

（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，問合せ "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る" 問合せ文は式 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，条件を満たす問合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．

### 2.5 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．

**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．

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

**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．

$$D\{L\} \leftarrow D\{L\} - Del\{L_{del}\} \quad (19)$$

（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．

$$A \leftarrow A - \{a\} \quad (20)$$

$$B \leftarrow B - \{b\} \quad (21)$$

（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．

$$L \leftarrow L - \{l\} \quad (22)$$

**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．

$$D\{L\} \leftarrow D\{L\} \cup M\{L_m\} - Mod\{L_{mod}\} \quad (23)$$

## 3. データベース言語 GRQL の設計

### 3.1 対応データモデルと対応データベース

対応データモデルに基づくデータベースを対応データベースと呼ぶ．ここでは，新しく設計した対応データベースのためのデータベース言語 GRQL（GRaph Query Language）について述べる．GRQL は対応データベースに対し，任意のグラフやノードまたはラベルを取り出したり，追加あるい削除などの操作を行うデータベース言語である．ここでは言語設計について述べた後，対応完備性について説明する．また，GRQL の特徴である select 文について具体例を示しながら詳しく説明する．最後に，SPARQL との比較を通して，GRQL の構文について議論する．

### 3.2 言語設計

対応データベースはグラフの集合であり，GRQL にはデータベース言語の基本機能に加えて，特にグラフ操作を簡便に記述できることが要求される．このことから基本的な機能として，新しくグラフを生成する機能やグラフを削除する機能，既存のグラフを読み込む機能や書き出す機能，リンク（ラベル，始点，終点の 3 つ組み）を追加／削除する機能などが必要である．また，複数の利用者がグラフに対して並列にアクセスする場合の相互排除を実現するロック／アンロック機能，およびグラフの一覧表示機能などが必要になる．またグラフを操作する機能として，グラフを結合する機能，ノードやラベルを指定した検索機能，ノードをたどる機能などが必要である．設計した GRQL 文とその機能一覧を表 2 に示す．これらの文はそれぞれ独立に解釈/実行されコマンドのような振る舞いをする．また，リレーショナルデータベースの操作言語として SQL が広く普及していることから，利用者の利便性を考慮して GRQL の構文は SQL に類似した構造にした．GRQL の構文規則の詳細については付録 1. に示す．GRQL の簡単な使用例を次に示す．ここでは新しく newgraph1，newgraph2 の 2 つのグラフを作成して，newgraph1 に「"Taro"の"age"が"20"である」というデータを追加している．次に select 文を用いた照会結果を set 文を用いて newgraph2 に代入している．

**GRQL の簡単な使用例**

```
CREATE newgraph1;
CREATE newgraph2;
ADD ["Taro","age","20"] TO newgraph1;
SET
  SELECT GRAPH FROM newgraph1
  WHERE LABEL = "age" TO newgraph2 ;
```

### 3.3 対応完備性

データベース言語が対応演算をすべて実現できる場合，これを対応完備と呼ぶことにする．select 文の記述例からもわかるように，from 句で複数のグラフを指定することで式 3 の対応和演算の機能が実現されている．また，where 句で LABEL，SOURCE，または DEST に対して条件を付与することで，それぞれ式 8，式 6，式 7 の選択演算が実現されている．これに加えて select 句おい SOURCE，DEST または LABEL を指定することで，それぞれ，式 9，式 10，式 11 および式 12 の機能が実現されている．また，記述の具体例は割愛するが対応差演算は delete 文を使用することで実現できる．このように表 1 の対応演算は GRQL ですべて記述でき，GRQL は対応完備であるといえる．

### 3.4 select 文

実際に取材して収集した北九州市門司港の観光情報の一部を図 3 に示す．これは graph1，graph2 および graph3 からなる対応データベースの例である．この例ではそれぞれのグラフは連結グラフになっているが，対応データベースにおけるグラフは，ラベル，始点，および終点 3 つ組の集合であることから連結グラフである必要はない．ここでは，このデータを元に select 文を使った記述例を示しながらその機能をみていく．図 4 が select 文に関連した構文規則の抜粋である．

まず，select 文の機能を概説する．なお，説明の簡単化のために，図 4 における <selectS> の規則中の FROM <graphNameList> の部分と WHERE <expression> の部分を SQL に倣いそれぞれ from 句，where 句と呼ぶことにする．

select 文は，データの照会を行う文であり，SELECT に続けて取得する対象を指定する．図 4 の構文規則をたどるとわかるように，ここでは予約語である GRAPH（グラフ），SOURCE（定義域集合），DEST（値域集合），または LABEL（ラベル集合）を指定できる．from 句では照会対象とするグラフを指定する．from 句で指定した複数のグラフは UNION 結合で結合されることにより，グラフの結合という所望の結果が得られる．条件を課す場合は where 句で指定する．where 句においては select 文の入れ子記述が何段でも可能であり複雑な問合せを行うことができる．

**Query1: 単一グラフに対する単純な問合せ**

```
SELECT GRAPH FROM graph2;
```

これは select 文における最も単純な問合せの例であり，from 句で指定した graph2 から LABEL 要素を取得するものである．結

表 2 GRQL 文の機能一覧

Table 2 The list of GRQL statements and their functions.

| 文 | 機能 |
|---|---|
| create | グラフの作成 |
| remove | グラフの削除 |
| rename | グラフ名の変更 |
| open | グラフデータのディスクからの読み込み |
| close | グラフデータのディスクへの書き出し |
| list | グラフの一覧表示 |
| lock | グラフに対するロック |
| unlock | グラフに対するアンロック |
| add | グラフおよび対応要素の追加 |
| delete | グラフおよび対応要素の削除 |
| replace | ラベル名およびノード名を変更 |
| temp | 作業用グラフの作成 |
| set | グラフの代入 |
| fix | デフォルトのグラフ名の設定 |
| select | グラフデータの照会 |

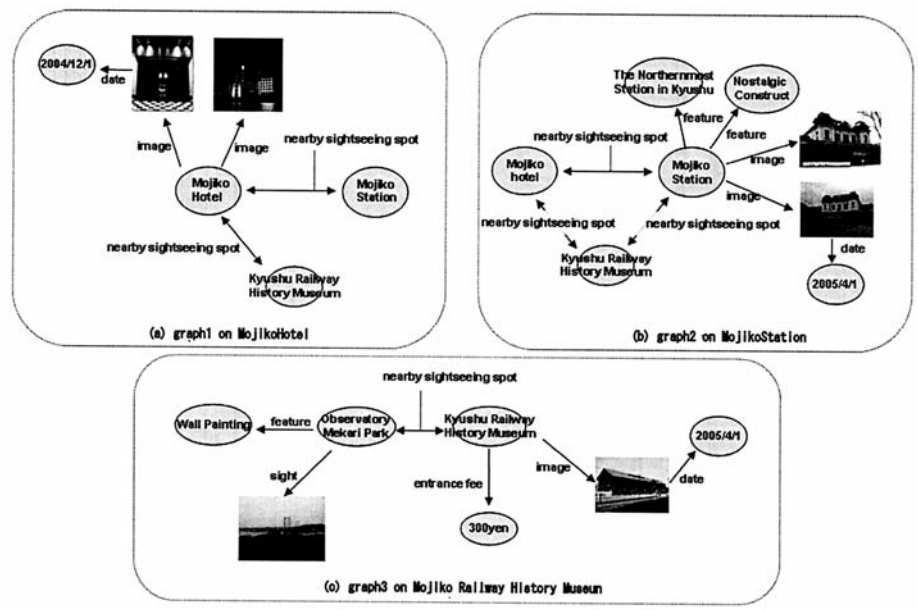


図 3　門司港の観光情報のグラフ例

Fig. 3　Example graphs on the sightseeing information in Mojiko.

```
<selectS>::=
  SELECT <elementConstatnt1> FROM <graphNameList>
    [WHERE <expression>]
<graphNameList> ::= <graphName>[,<graphName>]*
<expression> ::= <term>[{+|-|UNION| OR}<term>]*
<term> ::= <factor>[{*|/||NTERSECT|AND}<factor>]*
<factor>::=<element>[{>|<|>=|<=|=|IN}<element>]
<element>::=<constant>|<elemetConstant2>|<set>|
 (<selectS>)|(<expression>)|NOT<element>|
<functionName>(<expression>[,expression>]* )
<set> ::="{"[<constant>[,<constant> ]* ] "}"
<constant>::=[-]<unsignedNumber>|<string>
<elementConstant1> ::= GRAPH|SOURCE|DEST|LABEL
<elementConstant2> ::= SOURCE|DEST|LABEL|NODE
<tripleList> ::= <triple> [,<triple> ]*
```

図 4　select 文関連の構文規則（拡張 BNF）

Fig. 4　A part of the GRQL syntax related to the select statement in extended BNF.

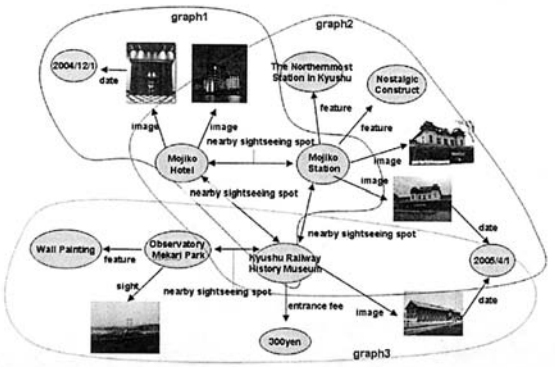


図 5　門司港の観光情報のグラフ例を結合したグラフ

Fig. 5　Union join of graphs on the example graphs on the sightseeing information in Mojiko.

果としては graph2 そのものが得られる．

**Query2: 複数グラフに対する単純な問合せ**

```
SELECT GRAPH
FROM graph1, graph2, graph3;
```

これは複数の結合したグラフから問合せを行う例であり，from 句で指定した graph1,graph2 および graph3 を結合したグラフ（図 5）から，GRAPH 要素を取得するものである．結果としては図 5 のグラフそのものが得られる．

**Query3: 条件を付与した単純な問合せ**

```
SELECT GRAPH
FROM graph2
WHERE SOURCE = "Mojiko Station" AND
    LABEL= "nearby sightseeing spot";
```

これは，特定の条件を課して問合せを行う例であり，where 句で SOURCE= “Mojiko Station” であり，かつ LABEL= “nearby sightseeing spot” という条件を課して graph2 から GRAPH 要素を取得している．この問合せの結果は図 6 のグラフになる．

**Query4: select 文の入れ子による問合せ**

```
SELECT GRAPH
```

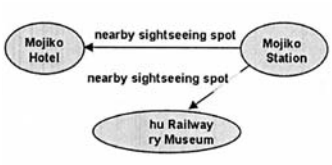


図 6　Query3 の問合せ結果
Fig. 6　The result of Query3.

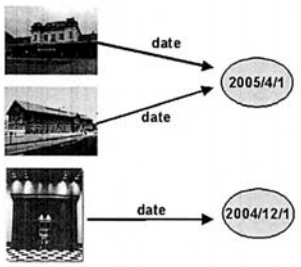


図 7　Query4 の問合せ結果
Fig. 7　The result of Query4.


図 8　Query5 の問合せ結果
Fig. 8　The result of Query5.

```
FROM graph1, graph2, graph3
WHERE SOURCE IN (
   SELECT DEST
   FROM graph1, graph2, graph3
   WHERE LABEL = "image");
```

これは，select 文の入れ子記述による問合せの例である．IN 演算子は集合の要素に含まれるかどうかを判定する演算子であり，IN 以降の select 文で LABEL が image である DEST 要素を取得して，IN 演算を用いてその DEST 要素を SOURCE とする GRAPH を取得する．結果として，graph1，graph2，graph3 から補足データをもつ “image” データのグラフを取得する例になる．問合せの結果は図 7 のグラフになる．

**Query5: 関数使用した問合せ使用**

```
SELECT GRAPH
FROM graph1, graph2, graph3
    WHERE DEST = MINELM (SELECT DEST
        FROM graph1, graph2, graph3
        WHERE LABEL = "entrance fee");
```

これは，関数を使った問合せの例であり，最小値の要素を取得する MINELM 関数を使って，“entrance fee” が最も安い観光地とその入場料を GRAPH として取得する．問合せの結果は図 8 のグラフになる．

### 3.5　SPARQL との比較

対応データベースにおける問合せでは，複数のグラフに対して指定した条件を満たす部分グラフ，ノード，ラベルを照会する操作が基本とであり，任意のノードやラベルに着目して，グラフを順方向あるいは逆方向にたどるイメージで照会を行うことになる．まず，前述の Query4 を例に SPARQL を用いて記述し，GRQL の記述と比較する．

前述の Query4 は，graph1，graph2，graph3 から補足データをもつ “image” データのグラフを取得する例であるが，ここでは foaf:Image ををもつ graph1.rdf，graph2.rdf，graph3.rdf を仮定する．このとき，この問合せは次のように記述できる．ただし，結果を RDF グラフとして取得したい場合は，CONSTRUCT を使用する．

**SPARQL による Query4 の記述例**

```
PRIFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?domain ?label ?range
FROM NAMED <graph1.rdf>
FROM NAMED <graph2.rdf>
FROM NAMED <graph3.rdf>
WHERE {?tmpdom foaf:Image ?domain .
       ?domain ?label ?range}
```

SPARQL では，変数を使いながら WHERE 句でリソース，プロパティとその値の 3 つ組の形式で条件を記述する．元になるグラフ構造が理解できていて，なおかつプロパティを指定しながら問合せを行うような場合は，クエリを作成する上でも理解が容易であるが，逆方向に関係をたどったり，この例のようにプロパティ自体が検索対象になるような問合わせになると，GRQL のような手続き的な問合せの方が理解しやすい．

表現力としては，GRQL においては select 文の入れ子を用いることで，また SPARQL においては条件になるトリプルを増やしたり OPTIONAL や FILTER キーワードを指定することで複雑な問合せを記述できるが，RDF グラフと違って語彙や構造に制約がかからないようなグラフが対象となる場合は GRQL が効果的に機能すると考えられる．

## 4.　おわりに

本稿では，特にグラフ構造のデータを考慮した対応データベースシステムのためのデータベース言語 GRQL の設計を行いその表現力を既存の問合せ言語と比較し GRQL の有効性を明らかにした．また，本稿では触れていないが，プロトタイプシステムは完成しており，所望の結果が得られることが確認できている．

今後の課題としては，操作対象がグラフ構造をもつことからデータの可視化が不可欠であり，効果的な GUI の開発が必要になる [18]〜[20]．また，スキーマの柔軟性によりデータベース化の制約は著しく軽減されるものの，収集したデータの照会や既存データの取り込みなどを考慮すると，あいまい検索や簡便なスキーママッピング [21] のサポートのような支援機能も必要である．また，本文では触れていないが XQuery などにあるようなパス表現の必要性も考えられるが，対応データベースにおいては，概ね検索対象とするグラフの構造が曖昧であること，またグラフをたどる際には順方向と逆方向の両方を考慮する必要があることから，単純なパス表現ではあまり効果が期待でき

ない．RDF データを対象とした検索グラフの自動生成に関する研究[22]などが行われているが，これと類似した試みを GUI ベースで検討している．

なお，本システムは，医療分野や教育分野のようなスキーマの柔軟性が必要になる分野への応用を想定している．特に，小学校の授業への利用を想定した応用システムを試作中である．授業で生徒に情報収集をさせる場合，あらかじめ決まった項目だけでなく，生徒の新しい発見をそのままデータベース化できることが望ましく，本システムはこの要求を満たす．また，現場での実際の利用を想定して，PDA などの携帯端末との連携，プレゼンテーションの雛型作成などの課題にも取り組んでいる．

## 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] 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).

[3] 横尾 徳保，重松 保弘：半構造データのための対応データモデルの提案と問合せ言語の設計，DEWS2003，ISSN 1347-4413, 8-B-04(2003).

[4] N.YOKOO, Y.KATSURA and Y.SHIGEMATSU：Design and Implementation of a Correspondence Database System，Proc. of the IASTED International Conference on INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS, pp. 230-235(2005).

[5] Y.KATSURA, N.YOKOO and Y.SHIGEMATSU：Design and Development of a Graphical Editor for Correspondence Database，Proc. of the IASTED International Conference on INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS, pp. 224-229(2005).

[6] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener: The Lorel Query Language for Semistructured Data, International Journal on Digital Libraries, 1(1), pp.68-88(1997).

[7] 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).

[8] 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(1996).

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

[10] Serge Abiteboul, Peter Buneman, Dan Suciu :Data on the Web, Morgan Kaufmann Pub.(1999).

[11] 野村 直之，川崎 洋治：XML データベースの種別と活用法上の留意点および問合せ言語への要求について，IPSJ-DD02034003 (2002)

[12] Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/REC-xml/ ,(2006).

[13] XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/ , (2006).

[14] Resource Description Framework (RDF):Concepts and Abstract Syntax, http://www.w3.org/TR/rdf-concepts/, (2004).

[15] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/ , (2006).

[16] 松永 義文：情報の "三つ組" モデルに基づく情報検索・蓄積ツール：情報箱，情報学基礎 40-6, pp.41-48(1995).

[17] 松永和夫：集合・位相入門，岩波書店 (1985).

[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.

[21] D.W Embley, L. Xu, and Y. Ding: Automatic Directed and Indirect Schema Mapping Experiences and Lessons Learned, SIGMOD, Vol.33, No.4, pp.14-19(2004).

[22] 飯塚京士，佐藤宏之，イコプラムディオノ，村上隆彦：RDF データを対象としたグラフ検索におけるクエリ生成方式の検討，人工知能学会研究会資料，SIG-SWO-A502-08，(2004).


## 付　録

### 1. GRQL の構文規則（拡張 BNF）

```
<GRQLstatement> ::= [<statement>] ;
<statement> ::=
  {<createS>|<removeS>|<renameS>|<openS>|<closeS>|
   <listS>|<lockS>|<unlockS>|<tempS>|<setS>|<fixS>|
   <replaceS>|<addS>|<deleteS>|<selectS>}
<createS> ::= CREATE <graphNameList>
<removeS> ::= DELETE <graphNameList>
<renameS> ::= RENAME <graphName> TO <graphName>
<openS> ::= OPEN <graphNameList>
<closeS> ::= CLOSE <graphNameList>
<listS> ::= LIST [ALL]
<lockS> ::= LOCK <graphNameList>
<unlockS> ::= UNLOCK <graphNameList>
<tempS> ::= TEMP <graphNameList>
<setS> ::= SET[<selectS>|<graphName>]TO<graphName>
<fixS> ::= FIX <graphName>
<replaceS>::=REPLACE[LABEL<labelName>WITH<labelName>
  |NODE<nodeName> WITH <nodeName>][IN <grahName>]
<addS> ::= ADD { <tripleList> | <graphNameList> }
  [TO <graphName>]
<deleteS> ::= DELETE { LABEL <labelNameList> |
  NODE <nodeNameList> |
<tripleS> | <graphNameList> } [ FROM <graphName> ]
<selectS>::=
  SELECT <elementConstatnt1> FROM <graphNameList>
    [WHERE <expression>]
<graphNameList> ::= <graphName>[,<graphName>]*
<expression> ::= <term>[{+|-|UNION| OR}<term>]*
<term> ::= <factor>[{*|/||NTERSECT|AND}<factor>]*
<factor>::=<element>[{>|<|>=|<=|=|IN}<element>]
<element>::=<constant>|<elemetConstant2>|<set>|
 (<selectS>)|(<expression>)|NOT<element>|
<functionName>(<expression>[,expression>]* )
<set> ::="["[<constant>[,<constant> ]* ] "]"
<constant>::=[-]<unsignedNumber>|<string>
<elementConstant1> ::= GRAPH|SOURCE|DEST|LABEL
<elementConstant2> ::= SOURCE|DEST|LABEL|NODE
<tripleList> ::= <triple> [,<triple> ]*
<triple>::="["<nodeName>,<labelName>,<nodeName> "]"
<graphNameList> ::= <graphName> [ ,  <GraphName> ]*
<labelNameList> ::= <lableName> [ ,  <lableName> ]*
<nodeNameList> ::= <nodeName> [ ,  <nodeName> ]*
<graphName>,<functionName> ::= <string>
<nodeName>,<labelName>::=<string>|[-]<unsignedConstant>
```