接尾辞配列に基づいたRDFデータのための索引手法

的野晃整[†] 天笠俊之[†]
吉川正俊[††] 植村俊亮[†]

Semantic Web は次世代 Web としてその動向に大きな期待が寄せられている．Semantic Web におけるメタデータは，一般的に RDF で記述されることが多く，今後は RDF データが大量に作成されることが予想される．このため，RDF データを効果的に検索することのできる手法の開発が重要となる．本論文では，RDF データが有向グラフ構造であることに着目し，有向グラフ上の経路式に基づいた接尾辞配列を提案する．提案手法では，まず RDF データと RDF スキーマデータからスキーマとリソース間の関連に応じた 4 種類の部分グラフを抽出する．次に，それぞれの部分グラフから経路式を抽出し，最後に，それらから接尾辞配列を構築する．我々の手法を用いることで，経路式が問合せとして与えられた場合，RDF グラフから一致する経路式を効果的に発見することができる．また，実験によって本手法の性能評価を行う．

An Indexing Scheme for RDF and RDF Schema Based on Suffix Array

Akiyoshi Matono,[†] Toshiyuki Amagasa,[†]
Masatoshi Yoshikawa[††] and Shunsuke Uemura[†]

The Semantic Web is expected as a candidate for next generation of the Web. The metadata in the Semantic Web are commonly described in RDF, and it is anticipated that the quantity of RDF data will increase. For this reason, the demand for efficient querying RDF data will also increase. In this paper, we focus on that the structure of RDF data is directed graph and we propose suffix arrays for directed graph based on the path expressions. We first extract four kinds of DAGs from the RDF data and RDF schematic data according to relationship between schema and instance, and then we extract path expressions from each sub-graph. And finally we create four kinds of suffix arrays as indexing data for the RDF data and the RDF schema data. Our proposed indices make it possible to retrieve matching paths from RDF graph efficiently, when a path as query is given. And we will evaluate its performance in a series of experiments.

1. はじめに

Semantic Web[32] は，次世代 Web としてその動向に大きな期待がよせられている．Semantic Web では，ネットワーク上の資源に対するメタデータが豊富に存在する．このため，人と計算機，計算機と計算機の間でのコミュニケーションをより知的に行うことができる．現在の Web と Semantic Web との根本的な違いは，メタデータの質と量であるということができる．現在の Web のメタデータは，単純な情報しか含んでおらず，その量も少ないため，高度な処理を行うには不十分である．一方，Semantic Web では，あらゆる資源間の関係を表現した複雑なメタデータが存在するため，意味的な検索や推論，演繹といった知的な処理を提供することが期待できる．

Semantic Web におけるメタデータは，一般的に RDF（Resource Description Framework）[29] によって記述される．RDF はメタデータの記述と処理のための基礎となる枠組みで，その仕様では RDF の文法とデータモデルを定めている．RDF データモデルは，2 つの資源間の関係を表現するステートメントと呼ばれる基本単位によって構成されている．ステートメントは，主語と述語，目的語の 3 つの部分で構成されており，主語は資源を，目的語は資源あるいは文字列を表現し，述語はそれらの間の関係を表現する．ステートメントの集合によって，有向グラフ構造の情報を表現することができる．一方，RDF の文法では，

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†† 名古屋大学情報連携基盤センター
Information Technology Center, Nagoya University

```
<rdf:RDF xmlns:my="http://sample/#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description about="http://sample/egg">
      <my:grow rdf:resource="http://sample/chicken">
   </rdf:Description>
   <rdf:Description about="http://sample/chicken">
      <my:lay rdf:resource="http://sample/egg">
   </rdf:Description>
</rdf:RDF>
```

図 1 ニワトリと卵の関係を示した RDF/XML 文書（平坦）
Fig. 1 An RDF/XML represented relation between egg and chicken (flat).

```
<rdf:RDF xmlns:my="http://sample/#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description about="http://sample/egg">
      <my:grow>
         <rdf:Description about="http://sample/chicken">
            <my:lay rdf:resource="http://sample/egg">
         </rdf:Description>
      </my:grow>
   </rdf:Description>
</rdf:RDF>
```

図 2 ニワトリと卵の関係を示した RDF/XML 文書（入れ子）
Fig. 2 An RDF/XML represented relation between egg and chicken (nest).

XML（Extensible Markup Language）[30] が採用されている．また，RDF データの妥当性を保つために，RDF Schema [31] が提案されている．RDF Schema は，RDF のスキーマ定義言語で，資源のクラスや資源間のプロパティなどを定義することができる．RDF データは RDF Schema によって定義されたクラスやプロパティの実体を表現しているということができる．

今後，Semantic Web の普及にともない，RDF によって記述されたメタデータの質が向上するとともに，その量も増加することが予想される．このため，我々は，大量の RDF データを高速に処理できる RDF データベースの実現が重要であると考えている．RDF データベースを実現する 1 つの素朴な手法として，XML データベースを利用することが考えられる．しかしながら，この手法には大きな問題がある．それは，RDF データを XML 文法で記述した場合，同じ意味を表す複数の表現が存在することである．たとえば，**図 1** と**図 2** は，ニワトリと卵の関係を示した同一の RDF データである．この問題によって，XML 記述のレベルでは，意図する問合せを一意に表現できないという問題が発生する．

RDF データベースを実現する他の方法として，関係データベースを利用する手法があげられる [2),19),33)]．これらの手法は，RDF データを細かく分割し，それぞれを関係表に格納する手法である．RDFSuite [2)] は，RDF のための問合せ言語 RQL（RDF Query Language）[13)] をサポートした RDF データベースで，RDF スキーマデータに依存して，関係スキーマを決定する点が特徴である．また，Jena [19)] は，RDF データの関係データベースへの格納や検索を行うことができる Java API で，RDF 問合せ言語 RDQL（RDF Data Query Language）[11)] を実装している．このように，いくつかの RDF データベースが提案されているが，そのほとんどは，ステートメントを平坦に単一のテーブルに列挙する手法を採用しており，問合せ処理を行うためには，多くの結合演算を必要とする．このため大量のデータに対しては実用的であるとはいえない．

本論文では，RDF データをより効果的に検索するための索引手法を提案する．本手法は，RDF データが有向グラフ構造であることに着目した，RDF グラフ上の経路式に基づくの接尾辞配列 [18),37)] である．まず，RDF グラフからクラス，プロパティ，リソース間の関係を表現している 4 種類の部分グラフを抽出する．具体的には，クラス間の関係，インスタンス間の関係，クラスの継承関係，およびプロパティの継承関係を表す 4 種類の部分グラフである☆．その後，それぞれの部分グラフからすべての経路式を抽出し，その経路式から接尾辞配列を生成する．この手法によって，経路式が問合せとして与えられた場合，RDF グラフ中の一致する経路式を効果的に発見することができる．

我々は本手法を実装し，その性能を実験によって評価した．実験は，本手法の索引を用いた場合と用いない場合の処理速度を比較した．この結果，我々の手法は RDFSuite のみを用いた場合と比較して，処理速度の面でより効果的であることを確認した．また，我々の手法は問合せの経路式の長さに対するスケーラビリティを備えていることも確認した．

本論文の構成を示す．2 章では，例を用いて RDF と RDF Schema の概要について述べる．3 章では，提案する効果的な RDF データ検索のための索引手法について述べる．特に，非巡回有向グラフのための接尾辞配列の定義や RDF グラフから抽出する 4 種類の部分グラフ，それらの部分グラフから経路式を抽出するアルゴリズムやその文法について述べる．また，4 章では，提案手法を実装し，実験の結果からその性能の評価を行う．5 章で関連した研究について議論し，最

☆ 前提条件として，インスタンス間の関係を表現したグラフの構造を，非巡回有向グラフに限定する．現実に流通している RDF データは非巡回有向グラフであることが多く，この制限でも本手法の適用範囲にそれほど制限を受けないことは 2.2 節で考察する．

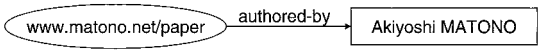

図 3 単純なステートメントの例
Fig. 3 A simple statement example.

後に 6 章でまとめる．

2. RDF の概要

RDF（Resource Description Framework）[29] は，メタデータ記述と処理のための基礎となる枠組みで，RDF を利用することによって，資源の意味的な発見やメタデータの互換性の提供，計算機に理解可能な情報の記述といったさまざまな応用が期待できる．

2.1 RDF の仕様

RDF の仕様[29] では，メタデータを表すための RDF データモデルとメタデータの記述と移植のための文法を提供している．また，RDF Schema[31] は，RDF データのスキーマ情報を記述するための仕様である．

RDF を用いることで資源に対してメタデータを付加することができる．資源とは，URI（Uniform Resource Identifier）[3] を用いることで表現できるものすべてを指し，本論文では，主に Web 上に存在する資源を対象としている．RDF は，資源間の二項関係を表現することができるステートメント（*rdf:Statement*）と呼ばれる基本単位によって構成されている．たとえば，ステートメントを用いることで，“This paper is authored by Akiyoshi MATONO.” という情報を表現することができる．ステートメントは，文としての形式をとっており，主語（*rdf:subject*, “This paper”）と述語（*rdf:predicate*, “is authored by”），目的語（*rdf:object*, “Akiyoshi MATONO”）という 3 つの部分によって構成されている（**図 3**）．そのため，ステートメントはトリプルと呼ばれることがある．ステートメントの集合を用いることで，複雑な構造の情報を表現することが可能になり，その構造は，ステートメントの主語と目的語を頂点とし，述語が有向辺に対応した有向グラフ構造の情報となる．

RDF の交換と記述のためにいくつかの規格が提案されている．その 1 つに XML を用いた文法[29] がある．RDF データの構造は，有向グラフであるため，XML データの構造とは異なる．そのため，RDF/XML 文法で表現した場合，1 つの RDF データを表現しても，ステートメントを平坦に列挙する方法や，入れ子にする方法など，複数の記述方法が存在する．

RDF Schema[31] は，RDF データのためのスキーマを定義するための仕様で，資源のタイプ（*rdf:type*）となるクラス（*rdfs:Class*）の定義や，資源間の関係のタイプとなるプロパティ（*rdf:Property*）の定義ができる．さらに，プロパティの範囲（*rdfs:range*）と定義域（*rdfs:domain*），クラスやプロパティの継承（*rdfs:subClassOf*, *rdfs:subPropertyOf*）を定義することができる．RDF の方針では，URI で記述できる資源はすべて資源クラス（*rdfs:Resource*）のサブクラスとして扱うため，前述したイタリック体で記述した *rdf* と *rdfs* の接頭辞を持つオブジェクトはすべて資源である．また，一般的に大文字から始まるものはクラスクラス（*rdfs:Class*）のタイプで，小文字で始まるものはプロパティクラス（*rdf:Property*）のタイプである．スキーマの定義も RDF のステートメントによって記述される．本論文では，*rdfs:subClassOf* および *rdfs:subPropertyOf*, *rdf:type* を述語とするステートメントの関係を継承関係と呼び，そのほかの述語で構成されるステートメントの関係を述語関係と呼ぶ．

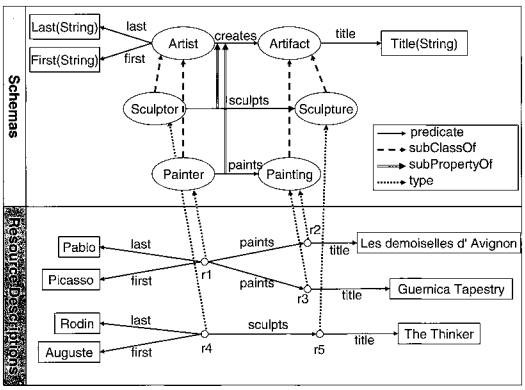

図 4 RDF と RDF Schema を用いた複雑な RDF グラフの例
Fig. 4 A comlex example using RDF and RDF Schema.

RDF と RDF Schema を用いたやや複雑な RDF グラフの例を**図 4** に示す．図の上部は，RDF Schema を用いて定義された RDF スキーマデータを示すもので，クラスとプロパティやそれらの継承，プロパティの範囲と定義域を示している．たとえば，*creates* プロパティは定義域として *Artist* クラス，範囲として *Artifact* クラスを持つ．また，*Sculptor* クラスは，*Artist* クラスのサブクラスである．図の下部は，RDF スキーマデータ中で定義されたクラスやプロパティのインスタンスである資源と，それらの関係を示した RDF データである．*r1* や *r2* は資源で，クラスのインスタンスである．すなわち，たとえば *r1* は *last* と *first*, *paints* という *Artist* クラスと *Painter* クラスから継承した 3 つのプロパティを持つ．また，“Pablo” や “Picasso” などの文字列は，RDF Schema のリテラル（*rdfs:Literal*）クラスのインスタンスである．

2.2 RDF データの特徴

本節では，現在流通している RDF データの特徴について考察する．RDF データは，述語（有向辺）によって構造化されている．この述語には，さまざまな意味が与えられており，大きく分けて 2 種類に分類することができる．ここでは仮に，そのステートメントを経由すると発散するものを発散述語，そのステートメントで終端する述語，あるいはそのステートメントを経由した後は収束に向かうものを終端述語と呼ぶことにする．終端述語は，目的語がリテラルであるもの，あるいは主語と目的語が包含関係にあるものが一般的で，主語を直接的に修飾するために用いられる．たとえば，“名前” や “所有” などがある．発散述語は，主語と目的語が同一レベルのクラスに属するもので，主語と目的語との関連を示すために用いられる．たとえば，“恋人関係” や “関連研究” などがある．これらの分類は，RDF データの構造に大きくかかわっており，発散述語の割合が増加するにつれて複雑さが加速度的に増していく．

現在，Web 上に普及している RDF 文書からいくつか代表的なものをあげる．RSS[26] や FOAF[7]，Dublin Core[9] などは，Web 上に実際に存在する画像やページなどの資源のメタデータを付加することを目的とした RDF である．また，Wordnet[21] や，OPD（Open Directory Project）[23]，Gene Ontology[10] は，語彙やゲノムなどの資源を体系化するための RDF データである．両者とも 1 つの RDF 文書自体では，終端述語が多く使われており，発散述語はあまり用いられていない．これは，1 つの RDF 文書内で，大量の発散述語を用いて閉じた RDF グラフを構成することは困難であることがその要因であると考えられる．前者の特徴は，1 つの RDF 文書で 1 つの資源のメタデータを表現しているため，一般的にデータサイズが小さく，それぞれの構造は単純である．しかしながら，複数の RDF 文書によって構成される RDF グラフは，発散述語によって他の関連する RDF データへのリンクを表現しているため，非常に複雑で巨大な RDF データになる．一方，後者は，1 つの RDF 文書内で多数の資源間の関係を体系化しているため，一般的にデータサイズは大きく，前者の 1 つの RDF 文書で構成される RDF グラフに比べて複雑な構造をしている．しかしながら，前述したように，1 つの RDF 文書内では，終端述語が多く使われるため，発散述語を多用した前者の複数の RDF 文書で構成される RDF グラフに比べると単純な構造をしている．実際，上記の 3 つの RDF グラフの構造は，木あるいは非巡回有向グラフの構造で，巡回を含まない．このように，現在の RDF データには，1 つの RDF 文書内で閉じている巨大なグラフと複数の RDF 文書で構成される開いたグラフの 2 種類が存在していると考えられる．

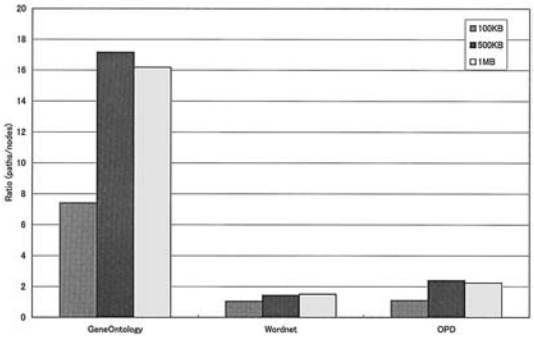

図 5 RDF グラフの頂点数と経路数の割合
Fig. 5 The ratio between paths and nodes.

表 1 RDF グラフの統計
Table 1 The statistics of RDF graphs.

file size (KB)	100	500	1,000
	Gene Ontology		
#of Nodes	978	5,118	10,524
#of Arcs	1,335	7,606	15,462
#of Roots	67	127	517
Max in-degree	103	738	1,349
Max out-degree	42	703	703
Ave. degree	1.37	1.49	1.47
#of Paths	7,235	87,801	170,261
Max path length	8	8	10
Ave. path length	4.31	3.73	4.38
	Wordnet		
#of Nodes	1,150	5,680	10,859
#of Arcs	1,109	5,647	10,851
#of Roots	220	1,046	2130
Max in-degree	13	14	41
Max out-degree	11	20	38
Ave. degree	0.96	0.99	1
#of Paths	1,209	8,144	16,370
Max path length	6	10	11
Ave. path length	1.94	2.70	2.83
	OPD		
#of Nodes	1,140	4,393	8,929
#of Arcs	1,428	6,108	12,966
#of Roots	4	3	3
Max in-degree	14	65	153
Max out-degree	100	271	434
Ave. degree	1.25	1.39	1.45
#of Paths	1,252	10,524	19,945
Max path length	11	15	15
Ave. path length	6.14	9.09	8.62

図 5 に Wordnet と OPD，Gene Ontology のデータサイズが増加したときの頂点数 v と経路数 p の割合 v/p を示した．また，**表 1** に RDF グラフの統計を示す．データは，それぞれの RDF 文書の先頭から，

サイズが 100 KB，500 KB と 1 MB となるよう作成した．図 5 では，Gene Ontology が他の 2 つと比べ，頂点の増加にともなって経路式の増加の割合が大きいことが分かる．しかしながら，ファイルサイズが 1 MB になると，その増加の割合が低くなっていることが確認できる．なお，これらの統計は Gene Ontology や OPD がスキーマを持たないため，インスタンスのみの RDF データでの値である．

3. 接尾辞配列を用いた RDF データ検索

3.1 問題提起

RDF データとそれに対する問合せは，RDF グラフの中から問合せ式に相当する部分グラフを発見する処理と考えることができる．これは，XML データの検索と類似している．これまで XML データ検索において経路式を用いることで，効果的な処理を実現した研究が多数存在する[12),35),36)]．Yamamoto ら[35)] は，XML データから抽出された経路式に基づいた接尾辞配列を提案している．これによって，XML 木に含まれる経路式の発見を効果的に行うことができる．

しかしながら，RDF データの構造と XML データの構造の違いによって，上記の手法はそのまま RDF データに適用することはできない．これは以下の理由による．1) RDF データは巡回を含んだ有向グラフ構造であるが，XML データは木構造である．そのため，RDF グラフからすべての経路式を抽出する処理は自明ではない．2) RDF は，RDF Schema によって記述されたスキーマ情報が付随している必要がある．これは，Semantic Web のレイヤ☆にも示されるように，RDF データを推論などの高度な処理で利用するにはスキーマ情報が必要になる．一方，XML データでは，スキーマ情報は必須ではなく，実際，前述した XML の経路に基づく手法ではスキーマ情報を利用していない．3) RDF グラフでは，頂点だけでなく有向辺にもラベルがある．これに対し，XML 木は要素や属性を頂点として扱い，有向辺はラベルを持たない．

3.2 用語の定義

本節では，提案手法を説明するのに先立って，本論文で使用する用語の定義を行う．

定義 1（有向グラフ） 有向グラフ G は，頂点と呼ばれる要素からなる非空有限集合 V と，V の 2 要素を順序つきで接続する有向辺と呼ばれる要素からなる有限族☆☆ A からなる．V は G の頂点集合，A は G の有向辺集合と呼ばれる． □

定義 2（隣接および接続） 任意の有向グラフ G の 2 つの頂点 v と w において，頂点 v から頂点 w へ向かう有向辺 $a_{v\to w}$，あるいは，頂点 w から頂点 v へ向かう有向辺 $a_{w\to v}$ が A に存在しているとき，

i) 有向辺にラベルを持つ場合，v と w は隣接しているといい，頂点 v および w は有向辺 $a_{v\to w}$ あるいは $a_{w\to v}$ に接続しているという．
ii) 有向辺にラベルを持たない場合，v と w は接続しているという．

また，任意の頂点あるいは有向辺 x と y が接続するとき，$x\to y$ と表現する． □

定義 3（頂点経路） 任意の有向グラフ G が与えられたとき，G の長さ m の頂点経路とは，

$$v_0\to v_1\to v_2\to\cdots\to v_m$$

の形をした頂点の有限列をいう $(0\leq i\leq m)$．このとき，v_i と v_{i+1} は隣接している． □

定義 4（混合経路） 任意の有向グラフ G が与えられたとき，G の長さ m の混合経路とは，

$$v_0\to a_{v_0\to v_1}\to v_1\to a_{v_1\to v_2}\to\cdots\to a_{v_{m-1}\to v_m}\to v_m$$

の形をした頂点と有向辺の有限列をいう $(0\leq i\leq m)$．このとき，v_i と v_{i+1} は $a_{v_i\to v_{i+1}}$ に接続している． □

定義 5（経路および始点，終点） 任意の有向グラフ G における経路とは，有向グラフにラベルを持つときは混合経路を示し，有向辺にラベルがないときは頂点経路を示す．また，経路の最初の要素を始点といい，最後の要素を終点という． □

3.3 部分グラフの抽出

提案手法では，まず，RDF グラフから 4 種類の部分グラフを抽出する．問題 3) で示したように，RDF グラフの有向辺には複数の意味がラベルとして割り当てられている．その有向辺を，スキーマの述語関係と，インスタンスの述語関係，クラスの継承関係，プロパティの継承関係に分類し，それぞれの分類に属する有向辺で構成される部分グラフを生成する．

インスタンス述語グラフ このグラフは，ステートメントの集合によって構成されるため，主語と目的語を頂点とし，述語を有向辺としたグラフである．複数の意味を持つ有向辺が混在したグラフとなる．また，その構造は非巡回有向グラフ構造に制限される．

スキーマ述語グラフ このグラフは，巡回を含む可能性がある．インスタンス述語グラフと同様に，ステー

☆ Layer Cake – 上部層は，下部層の文法と意味論を利用する（http://www.w3c.org/2002/Talks/04-sweb/slide12-0.html）

☆☆ 複数個の同じ要素が含まれていてもよい集まりを「族」と呼ぶ．たとえば {a,b,c} は集合であるが，(a,a,b,c,b,a) は族である．

トメントの集合によって構成されるため，主語と目的語を頂点とし，述語を有向辺としたグラフである．複数の意味を持つ有向辺が混在したグラフとなる．

クラス継承グラフ このグラフは，継承の関係を表現しているグラフであるため，もともと巡回を含まない非巡回有向グラフ構造である．グラフは，クラスを頂点とし，有向辺は継承を表現する関係で，ラベルを持たないグラフとなる．このグラフの葉はクラスのインスタンスとなることがある．このグラフの有向辺は，*rdfs:subClassOf* と *rdf:type* の 2 種類のみである．

プロパティ継承グラフ このグラフもクラス継承グラフと同様に，巡回を含まない非巡回有向グラフ構造である．また，プロパティを頂点とし，有向辺はラベルを持たないグラフとなる．このグラフの有向辺は，*rdfs:subPropertyOf* のみである．

このように RDF グラフから部分グラフを抽出することで，3.1 節の問題 1) であげた RDF グラフの巡回数を減少させることができる．つまり，RDF グラフの巡回には異なる意味の有向辺によって構成される巡回が含まれている．そのうち，上記の分類間をまたがるような有向辺で構成される巡回は，部分グラフに分解することで巡回ではなくなる．また，問題 2) はスキーマとインスタンスを異なる部分グラフに分割することで明らかに解決できる．さらに，問題 3) では，4 種類の部分グラフのうち，クラス継承グラフとプロパティ継承グラフの 2 種類の部分グラフは，その有向辺の意味を単一化することができる．

3.4 経 路 式

次に，経路式の文法を定義する．これによって，問題 3) に対処する．すなわち，スキーマ述語グラフとインスタンス述語グラフには，複数の意味の有向辺が混在しており，これらの混合経路を表現する経路式を定義し，部分グラフから経路を抽出するアルゴリズムを提案する．

図 6 は，EBNF（Extended Backus-Naur Form）を用いて定義した経路を表現するための文法である．この図で，`instancePath` はインスタンス述語グラフ，`schemaPath` はスキーマ述語グラフ，`classPath` はクラス継承グラフ，`propertyPath` はプロパティ継承グラフにそれぞれ対応している．クラス継承グラフとプロパティ継承グラフは単一の有向辺で構成されるため，その経路式は頂点経路であるのに対し，インスタンス述語グラフとスキーマ述語グラフの経路式は，混合経路である．経路式は，'>' を要素の区切り文字として使う．また，いくつかの特殊文字は，それぞれの要素を区別するために利用する．たとえば，'#' は，インスタンス述語グラフの経路式では資源を示す文字で，プロパティ継承グラフの経路式ではプロパティを示す文字である．もしこれらの特殊文字が要素のラベルに含まれていれば，XML の実体参照を用いて表現する．**図 7** は図 4 に示した RDF を 4 種類の部分グラフに分割し，それぞれの部分グラフから抽出した経路式の一覧である．

部分グラフが与えられたときに深さ優先順で要素を探索し，経路を抽出するアルゴリズムを**図 8** に示す．

```
instancePath  ::= ('#' URI-reference '>' '+' propName '>')*
                  ('#' URI-reference | '"' literal '"')
schemaPath    ::= ('#' typeName '>' '+' propName '>')*
                  '#' typeName
classPath     ::= ('#' typeName '>')*
                  ('$' URI-reference | '"' literal '"')
propertyPath  ::= ('#' propName '>')* '#' propName
typeName      ::= 参照29)
propName      ::= 参照29)
literal       ::= 参照29)
URI-reference ::= 参照29)
```

図 6 経路式の文法（EBNF 表現）
Fig. 6 Path expression syntax (EBNF).

インスタンス述語経路式

```
#r1>+paints>#r2>+title>"Les demoiselles d' Avignon"
#r1>+paints>#r3>+title>"Guernica Tapesty"
#r1>+last>"Pablo"
#r1>+first>"Picasso"
#r4>+sculpts>#r5>+title>"The Thinker"
#r4>+last>"Rodin"
#r4>+first>"Auguste"
```

スキーマ述語経路式

```
#Artist>+creates>#Artifact>+title>#Title
#Artist>+last>#Last
#Artist>+first>#First
#Sculptor>+sculpts>#Sculpture
#Painter>+paints>#Painting
```

クラス継承経路式

```
#Artist>#Painter>$r1
#Artifact>#Sculpture>$r2
#Artifact>#Painting>$r3
#Artist>#Sculptor>$r4
#Artifact>#Painting>$r5
#Last>"Pablo"
#Last>"Podine"
#First>"Picasso"
#First>"Auguste"
#Title>"Les demoiselles d' Avignon"
#Title>"Guernica Tapesty"
#Title>"The Thinker"
```

プロパティ継承経路式

```
#creates>#sculpts
#creates>#paints
```

図 7 図 4 から得られる経路式の一覧
Fig. 7 The list of path expressions extracted from Fig. 4.

```
/* triple を格納するため */
var stack : stack
/* 入次数が 0 である頂点のリスト*/
var roots1 : list of vertexes
/* roots1 に含まれる頂点を除いた頂点リストを
              入次数から出次数を引いた値で降順にソート */
var roots2 : list of vertexes
var roots := append ( roots1, root2 )
foreach start ( roots ) begin
  createPath ( start )
end
function searchGraph ( start : vertex ) : Void
var end : vertex
var arcs : set of arcs
var triple : tuple of ( vertex, arc, vertex )
begin
  roots.remove ( start )
  arcs := a set of arcs connected from start
  foreach arc ( arcs ) begin
    end := a vertex connected from arc
    roots.remove ( end )
    triple := ( start, arc, end )
    /* 経路式は同じステートメントを含まない */
    if ( stack = nil or triple ∉ stack.items ) then
      stack.push ( triple )
      searchGraph ( end )
      stack.pop ()
    end
  end
  /* 経路式の生成 */
  var path : concat ( path, stack[0].start )
  for (var i := 0; i < stack.length; i := i + 1) then
    path := concat( path, stack[i].arc, stack[i].end )
  end
end
```

図 8 経路式生成アルゴリズム
Fig. 8 An algorithm for extracting path expressions from directed graph.

このアルゴリズムは，すべての部分グラフに対して利用できる．このアルゴリズムによって，スキーマ述語グラフを除く部分グラフからすべての経路式を抽出することができる．スキーマ述語グラフは巡回を含むため，このアルゴリズムではすべての経路式を抽出できないが，問合せ処理の際に，問合せ経路の未発見な後半の部分経路を問合せとして再発行することによってこの問題は解決できる．詳細は，3.6 節で例をあげながら述べる．

3.5 接尾辞配列

最後に，有向グラフ構造のデータを対象とした接尾辞配列を定義し，その定義に沿って部分グラフを表現する経路式集合からそれぞれ接尾辞配列を生成する．この接尾辞配列は，通常の接尾辞配列を有向グラフに適用するように拡張したもので，直感的には，経路式中に出現する各要素を，通常の接尾辞配列における 1 文字として扱う手法である．通常の接尾辞配列は，テキストの全文検索のためのデータ構造で，具体的には，テキストからすべての接尾辞を抽出し，それらを辞書順に並べ替えた索引点の配列である．この配列に対して，二分探索を行うことで，任意の部分文字列を検索できる．接尾辞配列の特徴は，データサイズの小ささにある．すなわち，検索に必要なデータは，テキストデータ自体と索引点の配列のみである．

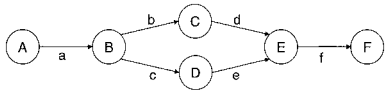

図 9 単純な有向グラフ
Fig. 9 A simple directed graph.

接尾辞配列を有向グラフに適用するには，複数の経路式の接尾辞を一意に指し示すための索引点に拡張を施す必要がある．そのため，我々は，2 次元の整数で構成される索引点を用いた．1 次元目は経路式の識別子，2 次元目はその接尾辞の識別子を表す．

以下に有向グラフのための接尾辞配列および，その定義に用いる頂点経路と混合経路の接尾辞を定義する．

定義 6（頂点経路の接尾辞） 有向グラフ G において，長さ m の頂点経路 $v_0 \to v_1 \to v_2 \to \cdots \to v_m$ の i 番目の接尾辞は，$v_i \to v_{i+1} \to v_{i+2} \to \cdots \to v_m$ である $(0 \leq i \leq m)$． □

定義 7（混合経路の接尾辞） 有向グラフ G における，次に示す長さ $2m+1$ の混合経路において，

$$v_0 \to a_{v_0 \to v_1} \to v_1 \to a_{v_1 \to v_2} \to \cdots \to a_{m-1} \to v_m$$

i) $2i$ 番目の接尾辞は，

$$v_i \to a_{v_i \to v_{i+1}} \to v_{i+1} \to a_{v_{i+1} \to v_{i+2}} \to \cdots \to a_{m-1} \to v_m$$

で $(0 \leq i \leq m)$，

ii) $2i+1$ 番目の接尾辞は，

$$a_{v_i \to v_{i+1}} \to v_{i+1} \to a_{v_{i+1} \to v_{i+2}} \to v_{i+2} \to \cdots \to a_{m-1} \to v_m$$

である $(0 \leq i \leq m)$．

□

定義 8（有向グラフのための接尾辞配列） 有向グラフ G において，識別子 j を持つ経路の i 番目の接尾辞に対する索引点を $[j, i]$ とする．有向グラフ G の接尾辞配列とは，すべての経路の接尾辞を辞書順に並べ替え，重複部分を削除した索引点の配列である． □

図 9 に示した単純な有向グラフの例を用いて接尾辞配列を具体的に説明する．このグラフから得られる経路式は，混合経路である “A.a.B.b.C.d.E.f.F” と “A.a.B.c.D.e.E.f.F” の 2 つである．それらの経路に対し，**図 10** のように接尾辞に二次元の索引点を付加し，

	0	1	2	3	4	5	6	7	8
0	A	a	B	b	C	d	E	f	F
1	A	a	B	c	D	e	E	f	F

図 10 図 9 の混合経路式とその接尾辞に対する索引点
Fig. 10 The mixed path expressions and the indexing-points for its suffixes of Fig. 9.

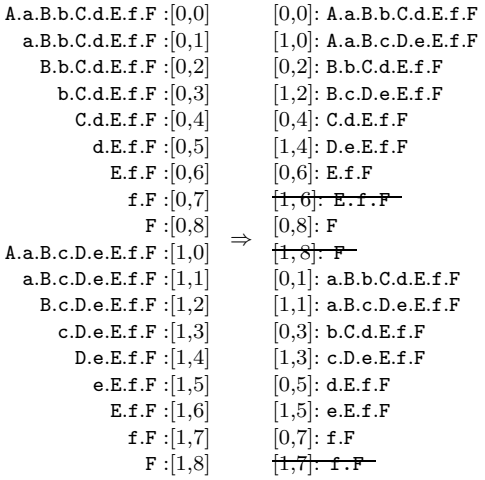

図 11 接尾辞のソートと重複要素の削除
Fig. 11 Sorting the suffixes and deletion of the redundancies.

図 11 のように辞書順に並べ替え，重複経路を削除する．結果，図 9 の接尾辞配列は [1,1] [2,1] [1,3] [2,3] [1,5] [2,5] [1,7] [1,9] [1,2] [2,2] [1,4] [2,4] [1,6] [2,6] [1,8] となる．また，ソートと重複要素の削除は，接尾辞配列を生成する過程で行う．つまり，索引点を格納するための空のリストを用意し，生成した接尾辞の索引点を辞書順に格納していき，その過程ですでに格納済みの接尾辞の接尾辞は格納しない．これによって，ソートと重複要素の削除を同時に行う．接尾辞の格納の際，その接尾辞が格納されるべき辞書順に正しい位置を検索するには，二分探索を用いて検索する．そのため，ソートおよび重複要素の削除における計算量は最悪で $\sum_{k=1}^{n} log(k-1)$ となる．

3.6 接尾辞配列を用いた問合せ

本節では，接尾辞配列を用いた問合せについて述べる．次に，有向グラフのための接尾辞配列を用いた問合せと解を定義する．

定義 9（接尾辞配列を用いた問合せと解） 有向グラフ $G = (V, A)$ に対する接尾辞配列を用いた問合せは，経路 p である．また，以下を満たすとき，その解は e_{m+1} である．

- p の要素はすべて $(e \in V \cup e \in A)$ で構成されている．

```
function Int binarySearch ( String graph,
                String query, Int max, Boolean flag ) {
  var Int upper := max
  var Int lower := 1
  var Int center := 0
  var Boolean bingo := false
  while ( lower <= upper ) {
    center := (Int) ( ( upper + lower ) / 2 )
    if ( getPartialPath ( graph, center ) == query ) {
      if ( flag ) lower := center + 1
      else        upper := center - 1
      bingo := true
    } else
    if ( getPartialPath ( graph, center ) > query ) {
      upper := center - 1
      bingo := false
    } else
    if ( getPartialPath ( graph, center ) < query ) {
      lower := center + 1
      bingo := false
    }
  }
  if ( not bingo ) {
    if ( flag ) center := center - 1
    else        center := center + 1
  }
  return center
}
```

図 12 二分探索による検索アルゴリズム
Fig. 12 An retrieval algorithm using binary search.

- e_m は終点である．
- $e_m \rightarrow e_{m+1}$

□

次に，問合せと RDF グラフに含まれる経路との一致に関する定義を行う．

定義 10（一致および前方一致） 経路 $p = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n$ と $q = r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_m$ において，$s_i = r_i$ $(0 \leq i \leq n)$ かつ $n \leq m$ のとき，p は q に一致するといい，$s_i = r_i$ $(0 \leq i \leq m)$ かつ $n > m$ のとき，p は q に前方一致するという． □

問合せ処理のアルゴリズムを**図 12** に示す．このアルゴリズムは，接尾辞配列に対して二分探索を行って目的の接尾辞を発見する手法である．解となる経路式は 1 つではなく，問合せ経路式に一致するすべての部分経路式を返す必要があるため，まず二分探索によって配列中の最初に一致した索引点と最後に一致した索引点（つまり配列中の一致した索引点の範囲）を求める．このため，計算コストは $O(2log(n))$ となる．このとき，n は配列の長さで，索引点を元に部分経路式を求めるコストは含まれていない．最後に得られた範囲から解となる部分経路式集合を求める．

このアルゴリズムはすべての部分グラフの接尾辞配

列に適用できる．しかし，3.4 節で述べたように，スキーマ述語グラフは，巡回を含むために，すべての経路式を抽出できない．そのためこのアルゴリズムだけでは，完全な解を得ることができない．これには次のように対処することができる．まず，1) 問合せ経路式がこのアルゴリズム中で得られた部分経路式に対して一致ではなく，前方一致していた場合も一致したと見なす．すなわち，問合せ経路式 “a>b>c>d” は，部分経路式 “a>b” に前方一致しているとする．2) 問合せ経路式が，処理結果として得られた経路式集合に前方一致している場合は，得られた部分経路の最後の要素から始まる経路式（つまり，“b>c>d”）を新たな問合せ経路式として，再び問合せを発行する．1) と 2) を繰り返すことで巡回したグラフに対しても提案した索引を利用して検索を行うことが可能である．1 つの問合せに対して繰り返し処理を行わなければならないため効率は悪いが，スキーマ述語グラフは，一般的にサイズが小さいため実用上は問題がないと考える．

4. 性能評価

我々は，本手法を実装した．本章では，実験による本手法の性能の評価を行う．

4.1 実験環境

4.1.1 実験データ

本手法に適した RDF 文書は，経路式が長くかつ指数的に増加しない RDF 文書である．そのため，2.2 節であげた RDF 文書のうち，我々の手法に最も適していると考えられるのは OPD[23] であるが，OPD の RDF データは仕様に準拠しておらず，実験に用いることはできない．また，Wordnet[21] は経路式が短いため，我々の手法の有用性を示すことはできない．このため，Gene Ontology[10] を実験データとして採用した．Gene Ontology は，経路式の数が他の 2 つに比べ増加の割合が高く，接尾辞の数が増えるため，効果的な性能を期待できない．しかし，逆にいえば，Gene Ontology でも我々の手法の有効性を示すことができれば，他の RDF 文書でも有効であることが期待できる．

Gene Ontology は分子の生体機能に関してゲノムデータベースを統合的利用するために作成されたオンラインデータベースで，主に関連する分子間の関係を表現している RDF 文書と分子の生体機能に関して記述している RDF 文書の 2 種類あり，前者は後者のサブセットである．我々は前者を実験に採用し，データサイズを 500 KB にしたもので評価を行った．また，Gene Ontology には RDF スキーマは定義されていないため，XML の DTD を基に我々が作成した．

表 2 性能評価に用いた問合せとしての経路式
Table 2 Performance evaluation path expressions as queries.

	クラス継承グラフへの問合せ	
#1	`#Resource>#`	サブクラスの検索
#2	`#Resource>#Class>#Datatype>$`	子孫クラスの検索
	プロパティ継承グラフへの問合せ	
#3	`+seeAlso>#`	定義域の検索
	スキーマ述語グラフへの問合せ	
#4	`#Resource>+`	述語の検索
#5	`+dbxref>#`	目的語の検索
#6	`#Resource>+isDefinedBy>#Resource>+dbxref> #Resource>+member>#Resource>+seeAlso>#Resource>+`	長い経路式
	インスタンス述語グラフへの問合せ	
#7	`#GO:0003674>+n_associations>"`	ステートメント検索
#8	`#GO:0050646>+is_a>#GO:0050543>+is_a>#GO:0050542> +is_a>#GO:0005504>+is_a>#GO:0008289>+dbxref> #&empty;>+database_symbol>"`	長い経路式

4.1.2 問合せ経路式

表 2 に実験で用いた問合せとしての経路式を示す．つまり，この経路式に一致する部分経路式を Gene Ontology の RDF データから生成した接尾辞配列を用いて発見することを問合せ処理とする．なお，表 2 では，見やすくするためそれぞれの資源の名前空間を省略してあるが，本来の経路式では含まれる．これらの経路式を問合せとして用いることで，次のような意図を含んでいる．問合せ 1 と 2 はクラスの継承関係に関する問合せ，問合せ 3 はプロパティの継承関係に関する問合せ，問合せ 4 から 6 はスキーマの述語関係に関する問合せ，問合せ 7 と 8 はインスタンスの述語関係に関する問合せである．

4.1.3 実験方法

実験は，RDF データベースとして，Jena2[19] を用いた．Jena2 は RDF を扱うための Java API で，関係データベースによる管理をサポートしている．Jena2 の関係スキーマは基本的にステートメントをそのまま格納する構造である．

実験は，Jena2 と提案手法の処理速度を比較した．まず，Jena2 の API を利用して構文解析を行い，生成された RDF データから 4 種類の部分グラフの接尾辞配列を生成した．その後，表 2 に示した問合せ経路式を，生成した接尾辞配列から発見するまでに要する処理時間と，それと同義の問合せを RDQL[11] を用いて Jena2 に対して発行したときの処理時間を比較した．それぞれ 100 回問合せを発行し，それらの平均をとった．

また，実験には，Athlon 1.4 GHz の CPU と 1,024 MB のメモリを搭載した計算機を用い，OS は

表 3 経路式数と接尾辞数
Table 3 The number of path expressions and suffixes.

	経路式数	接尾辞数
クラス継承グラフ	203	608
プロパティ継承グラフ	25	26
スキーマ述語グラフ	6,530	16,322
インスタンス述語グラフ	87,801	67,706

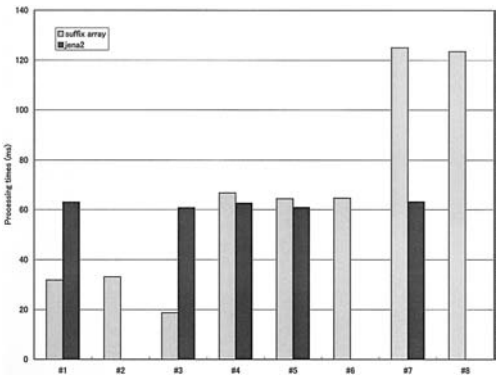

図 13 処理時間
Fig. 13 The processing times.

RedHat Linux 9.0 で行った．

4.2 実験結果

表 3 に Gene Ontology の RDF 文書から生成した経路式数とその接尾辞数を示す．この表から分かるように，インスタンス述語グラフでは，接尾辞数が経路数よりも少ない．これは，重複した接尾辞を削除するためで，多重辺を多く含むグラフであることが確認できる．また，クラス継承グラフやプロパティ継承グラフは非常に小さなグラフであるにもかかわらず，スキーマ述語グラフは，巡回を含むためにその値は大きくなっている．

図 13 に Jena2 の処理時間と提案手法を用いたときの処理時間の比率を示す．また，表 2 に示した問合せのうち，問合せ 2，6，8 は RDQL では表現できないため，空欄になっている．図 13 から，我々の手法の方が高速であるのは問合せ 1 と 2 で，他の問合せでは，Jena2 の方が高速であったと判断できる．しかしながら，問合せ 2，6，8 を RDQL で表現できれば，我々の手法の方が効果的であることが予測できる．その理由を次に示す．Jena2 は，どの検索もほぼ一定の処理時間であり，これは，RDQL で表現できた問合せは，経路の長さが 3 以下のものであるため，一定の処理時間になったと考えられる．いい換えれば，仮に RDQL で長い経路式の問合せを表現できれば，Jena2 の処理速度は，現在の処理速度よりも遅くなることは明確である．一方，我々の手法でも同じ部分グラフに対して問合せを発行する場合は，一定の検索速度であることが分かる．これは，同じ部分グラフから生成された同じ接尾辞配列を用いて問合せを行っているためである．このことは，提案手法での処理速度は，対象の部分グラフから抽出された経路式の数とその配列の長さにのみ依存し，問合せ経路式の長さに依存しないことを示している．したがって，問合せ 2 と 6，8 において，RDQL で表現できれば，我々の手法の方が効果的になることが予測できる．

5. 関連研究

非巡回有向グラフを含む構造化文書の索引は，Sacks-Davis ら[27] によると，位置に基づいた索引と経路に基づいた索引に分類される．本提案手法は，後者の手法であることから，経路に基づいた関連研究について触れる．

経路に基づいた索引には，Yamamoto ら[35] や Qun ら[25]，Cooper ら[6] が提案した手法がある．Yamamoto らは，本論文の手法のアイデアの基礎を提案している．XML の根から葉までの経路を文字列に置き換えて，接尾辞配列による索引を提案している．Qun らは，これまでに McHugh ら[20]，Milo ら[22]，および Kaushik ら[14] が提案した Structural Summary をより拡張した手法を提案した．Structural Summary とは，有向グラフの類似した要素を併合することで，よりコンパクトなモデルを生成し，検索における探索領域の減少を図っている．併合処理のときに，経路に基づいた親子関係を比較して併合処理を行う手法である．Cooper ら[6] は Patricia trie[15] を基にした Index Fabric という索引を提案してる．Patricia trie は大きなサイズの文字列を扱うことのできるコンパクトで効果的な索引である．Index Fabric は Patricia trie を B 木のように左右の枝をつりあわせ，半構造モデルに対応するように拡張した索引である．また，Sacks-Davis ら[27] は，位置と経路の両方に基づいた手法を提案している．彼らの経路に基づいた索引は，文字列が出現するまでの経路を要素名とその位置で表した素朴な手法である．

経路に基づいた索引はオブジェクト指向データベースでも利用されている[4),16),34]．これらの手法は，継承や述語の関係を経路として扱い，問合せを高速化している．この点は本手法と類似している．最も異なる点は，経路を表現するとき，これらの手法では複数回の結合を必要とするが，本手法では文字列として経路を扱っているため，結合をほぼ必要としない点である．

Christophides ら[5] は，RDF に対する効果的な検

索を目的としている点で，本論文と最も関連している．彼らの手法は，これまで提案されてきた木構造のデータに対するラベリングスキームを RDF スキーマデータに適用させる方法である．具体的には，彼らの手法は，Agrawal ら[1] が提案した各ノードの先祖数に基づいて非巡回有向グラフから全域木を生成させる，それに対してラベルをつける手法である．Christophides ら[5] は，大まかに bit vector，prefix および interval scheme の 3 つに分類している．bit vector scheme[28] は，全ノード数と同数のビットを各ノードに割り当て，各ノードに対応するビットと祖先ノードに対応するビットを 1 にする手法である．prefix scheme は，Dewey Scheme[24] に代表される，親ノードのラベルを接頭辞として継承し，その後ろに，そのノードの番号を付ける手法である．interval scheme[1),8),17)] は各ノードが (start, end) の形の interval ラベルを持ち，その値が親の interval ラベルに包含されるようなラベリングスキームである．Christophides ら[5] は，検索の高速化の対象を RDF のスキーマのみにしているが，彼らの手法は，DAG に限定したインスタンスであれば，適用することが可能である．我々の手法と彼らの手法の比較は，今後の課題としたい．

6. おわりに

本論文では，RDF インスタンスデータと RDF スキーマデータに対する効果的な索引手法を提案した．我々は，前提として，対象の RDF データを非巡回有向グラフに限定する．その RDF グラフから，まず 4 種類の部分グラフを抽出した．さらに，有向グラフに対する接尾辞配列を定義し，抽出した部分グラフをそれに適用することで，4 種類の経路式集合とその接尾辞配列を生成した．また，本論文では，実験によって，提案手法が効果的であることを確認した．

今後の課題として，インスタンス述語グラフに対する制限を解除し，巡回を含むデータに対する手法について考察する必要がある．また，RDF グラフから経路を生成し，その接尾辞配列を生成する処理の高速化や更新，問合せ最適化に関しても考察する必要がある．

謝辞 本研究の一部は，文部科学省科学研究費補助金（課題番号 15017243），日本学術振興会科学研究費補助金（課題番号 15200010，15700097）の支援によるものである．ここに記して謝意を表す．

参考文献

1) Agrawal, R., Borgida, A. and Jagadish, H.V.: Efficient Management of Transitive Relationships in Large Data and Knowledge Bases, *Proc. 1989 ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, May 31–June 2, 1989, Clifford, J., Lindsay, B.G. and Maier, D. (Eds.), pp.253–262, ACM Press (1989).
2) Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D. and Tolle, K.: The RDFSuite: Managing Voluminous RDF Description Bases, *Proc. 2nd International Workshop on the Semantic Web* (*SemWeb'2001*), Hongkong, China (2000).
3) Berners-Lee, T., Fielding, R.T. and Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax (1998). RFC2396. http://www.isi.edu/in-notes/rfc2396.txt
4) Bertino, E.: Index Configuration in Object-Oriented Databases, *VLDB Journal*, Vol.3, No.3, pp.355–399 (1994).
5) Christophides, V., Plexousakis, D., Scholl, M. and Tourtounis, S.: On labeling schemes for the semantic web, *Proc. 12th International Conference on World Wide Web*, pp.544–555, ACM Press (2003).
6) Cooper, B., Sample, N., Franklin, M.J., Hjaltason, G.R. and Shadmon, M.: A Fast Index for Semistructured Data, *The VLDB Conference*, pp.341–350 (2001).
7) Dan Brickley and Libby Miller: FOAF Vocabulary Specification RDFWeb Namespace Document 16 (Aug. 2003). http://xmlns.com/foaf/0.1/
8) Dietz, P. and Sleator, D.: Two algorithms for maintaining order in a list, *Proc. 19th Annual ACM Conference on Theory of Computing*, pp.365–372, ACM Press (1987).
9) Dublin Core Metadata Element Set, Version 1.1 (2003). http://dublincore.org/documents/dces/
10) GENE ONTOLOGY CONSORTIUM. http://www.geneontology.org/
11) Hewlett-Packard Company: RDQL—RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.
12) Jiang, H., Lu, H., Wang, W. and Yu, J.X.: Path Materialization Revisited: An Efficient Storage Model for XML Data, *13th Australasian Database Conference* (*ADC2002*), Zhou, X. (Ed.), Melbourne, Australia, ACS (2002).
13) Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D. and Scholl, M.: RQL: A Declarative Query Language for RDF, *Proc. 11th International Conference on World Wide*

Web, pp.592–603, ACM Press (2002).
14) Kaushik, R., Shenoy, P., Bohannon, P. and Gudes, E.: Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data, *ICDE* (2002).
15) Knuth, D.E.: *The Art of Computer Programming, Volume 3 Sorting and Searching*, Second Edition, Addison-Wesley (1998).
16) Lee, W. and Lee, D.: *Path Dictionary: A New Approach to Query Processing in Object-Oriented Databases* (1995).
17) Li, Q. and Moon, B.: Indexing and Querying XML Data for Regular Path Expressions, *The VLDB Journal*, pp.361–370 (2001).
18) Matono, A., Amagasa, T., Yoshikawa, M. and Uemura, S.: An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays, *1st International Workshop on Semantic Web and Databases* (*SWDB*), Berlin, Germany (2003).
19) McBride, B.: Jena: Implementing the RDF Model and Syntax Specification, *Proc. 2nd International Workshop on the Semantic Web* (*SemWeb'2001*) (2001).
20) McHugh, J., Widom, J., Abiteboul, S., Luo, Q. and Rajamaran, A.: Indexing Semistructured Data, Technical Report, Stanford University, Computer Science Department (1998).
21) Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K.: Introduction to WordNet: An On-Line Lexical Database (1993). http://www.cogsci.princeton.edu/~wn/
22) Milo, T. and Suciu, D.: Index Structures for Path Expressions, *Database Theory* (*ICDT '99*), *Proc. 7th International Conference, Jerusalem, Israel, January 10-12, 1999*, Beeri, C. and Buneman, P. (Eds.), Lecture Notes in Computer Science, Vol.1540, pp.277–295, Springer (1999).
23) Netscape: Open Directory Project. http://dmoz.org/
24) Online Computer Library Center: Dewey Decimal Classification. http://www.oclc.org/dewey/
25) Qun, C., Lim, A. and Ong, K.W.: D(k)-Index: An Adaptive Structural Summary for Graph-Structured Data, *Proc. 2003 ACM SIGMOD International Conference on Management of Data* (*SIGMOD2003*), San Diego, California, USA, pp.134–144 (2003).
26) RSS-DEV Working Group: RDF Site Summary (RSS) 1.0 (2000). http://web.resource.org/rss/1.0/
27) Sacks-Davis, R., Dao, T., Thom, J.A. and Zobel, J.: Indexing Documents for Queries on Structure, Content and Attributes, *Proc. International Symposium on Digital Media Information Base*, Nara, Japan, pp.236–245 (1997).
28) Wirth, N.: Type Extensions, *ACM Trans. Prog. Lang. Syst.*, Vol.10, No.2, pp.204–214 (1988).
29) World Wide Web Consortium: Resource Description Framework(RDF) Model and Syntax Specification W3C Recommendation 22 February 1999 (1999). http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
30) World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000 (2000). http://www.w3.org/TR/REC-xml
31) World Wide Web Consortium: Resource Description Framework(RDF) Schema Specification 1.0 W3C Candidate Recommendation 27 March 2000 (2000). http://www.w3.org/TR/2000/CR-rdf-schema-20000327/
32) World Wide Web Consortium: Semantic Web (2001). http://www.w3c.org/2001/sw/
33) World Wide Web Consortium: Survey of RDF/Triple Data Stores (2001). http://www.w3.org/2001/05/rdf-ds/DataStore
34) Xie, Z. and Han, J.: Join Index Hierarchies for Supporting Efficient Navigations in Object-Oriented Databases, *VLDB'94, Proc. 20th International Conference on Very Large Data Bases*, September 12–15, 1994, Santiago de Chile, Chile, Bocca, J.B., Jarke, M. and Zaniolo, C. (Eds.), pp.522–533, Morgan Kaufmann (1994).
35) Yamamoto, Y., Yoshikawa, M. and Umeura, S.: On Indices for XML Documents with Namespaces, *Conference Proc. Markup Technologies '99, GCA*, Philadelphia, U.S.A. (1999).
36) Yoshikawa, M., Amagasa, T., Shimura, T. and Uemura, S.: XRel: a path-based approach to storage and retrieval of XML documents using relational databases, *ACM Trans. Internet Technology* (*TOIT*), Vol.1, No.1, pp.110–141 (2001).
37) 的野晃整, 天笠俊之, 吉川正俊, 植村俊亮：RDF のための経路式に基づいた索引方式, 第 14 回データ工学ワークショップ（DEWS2003）論文集 (2003). http://www.ieice.org/iss/de/DEWS/proc/2003/papers/8-A/8-A-02.pdf

(平成 15 年 9 月 20 日受付)
(平成 16 年 1 月 6 日採録)

（担当編集委員 定兼 邦彦）

的野　晃整（学生会員）

2000 年岡山県立大学情報通信工学科卒業．2002 年同大学大学院情報系工学研究科博士前期課程修了．同年奈良先端科学技術大学院大学情報科学研究科博士後期課程入学，在学中．RDF データ検索に関する研究に従事．ACM，日本データベース学会各学生会員．

天笠　俊之（正会員）

1994 年群馬大学工学部情報工学科卒業．1999 年同大学大学院工学研究科修了．博士（工学）．同年から奈良先端科学技術大学院大学情報科学研究科助手．XML データベース，装着型コンピュータにおけるデータベース応用等の研究に従事．電子情報通信学会，ACM，IEEE Computer Society，日本データベース学会各会員．

吉川　正俊（正会員）

1980 年京都大学工学部情報工学科卒業．1985 年同大学大学院工学研究科博士後期課程修了．工学博士．同年京都産業大学計算機科学研究所講師．同大学工学部助教授を経て 1993 年より奈良先端科学技術大学院大学情報科学研究科助教授，2002 年より名古屋大学情報連携基盤センター教授，現在に至る．1989 年〜1990 年南カリフォルニア大学客員研究員，1996 年〜1997 年ウォータルー大学客員准教授．XML データベース，多次元空間索引等の研究に従事．電子情報通信学会，ACM，IEEE Computer Society 各会員．日本データベース学会理事．

植村　俊亮（フェロー）

1964 年京都大学工学部電気工学科卒業．1966 年同大学大学院工学研究科修士課程修了．同年電気試験所（産業技術総合研究所）．1970 年マサチューセッツ工科大学電子システム研究所客員研究員，1981 年ソフトウェア部プログラム研究室長，1988 年東京農工大学教授を経て，1993 年から奈良先端科学技術大学院大学情報科学研究科教授．データ工学，データベースシステムの研究に従事．工学博士．IEEE Fellow，電子情報通信学会フェロー．現在，情報処理学会理事，日本情報考古学会理事，データベース振興センター評議員等．