DEWS2004 3-A-03

経路式に基づく RDF データの関係データベースへの格納と検索

的野　晃整†　　天笠　俊之†　　吉川　正俊††　　植村　俊亮†

† 奈良先端科学技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
†† 名古屋大学情報連携基盤センター　〒 464–8601 名古屋市千種区不老町
E-mail: †{akiyo-ma, amagasa, uemura}@is.aist-nara.ac.jp, ††yosikawa@itc.nagoya-u.ac.jp

あらまし　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 では，RDF で記述したメタデータを用いて，高度な処理を提供できる．RDF 文書は急激に増加しており，大量の RDF データを効率よく管理できる RDF データベースが今後重要になる．本論文では，RDF データを関係データベースへ経路式に基づいて格納する手法とその検索手法を提案する．本手法では，これまでの RDF データベースとは異なり，RDF データをそのスキーマに基づいて管理する．まず，RDF スキーマからクラスとプロパティを抽出し，その後，それらのインスタンスである RDF データから資源を抽出する．最後に，それぞれの識別子と経路を生成し，関係表に格納する．これによって，スキーマ情報を含む経路に基づいた問合せを高速に処理することができる．
キーワード　RDF, データベース, 経路式, Semantic Web, メタデータ管理

A Path-based Approach for Storage and Retrieval of RDF Data Using Relational Databases

Akiyoshi MATONO†, Toshiyuki AMAGASA†, Masatoshi YOSHIKAWA††, and Shunsuke UEMURA†

† Graduate School of Information Science, Nara Institute of Science and Technology (NAIST)
8916–5 Takayama-cho, Ikoma-shi, Nara 630–0192, Japan
†† Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-shi 464–8601, Japan
E-mail: †{akiyo-ma, amagasa, uemura}@is.aist-nara.ac.jp, ††yosikawa@itc.nagoya-u.ac.jp

Abstract　Semantic Web is very anticipated as a next generation web. Semantic Web can provide high-level processings using metadata which is described by RDF. Today, RDF documents are extremely increasing. For this reason, an RDF database that can manage a large number of RDF data will be important in the near future. In this paper, we propose a scheme of storage and retrieval based on path expressions for RDF data using relational database. We manage RDF data according to the RDF schema data, unlike most of the previous RDF databases. We first extract all classes and properties from RDF schema data, extract all resources as those instances, next create the identifiers and path expressions, and finally store them into relational tables. Our proposal makes it possible to process path-based queries including schema information efficiently.
Key words　RDF, database, path expressions, Semantic Web, metadata management

1　はじめに

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 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ため，人と計算機，計算機と計算機の間でより知的なコミュニケーションを実現することができる．現在の Web と Semantic Web とのもっとも大きな違いは，メタデータの質と量である．現在では，Web 上に散在する資源の情報を表現するメタデータ記述言語の表現能力が貧困であるため，メタデータを利用した高度な処理を期待できない．一方，Semantic Web では，資源間の関係を構造的に表現した高度なメタデータが豊富に存在するため，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，一般に RDF (Resource Description Framework) [19] に基づいて記述される．RDF と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，RDF グラフを表現する構文とその意味論が定義されている．RDF グラフは，資源間の二項関係を表現する文と呼ばれる基本単位によって構成されている．文は，主語と述語，目的語の三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．したがって，文の集合によって表現される RDF グラフは，有向グラフ構造となる．RDF グラフを表現する構文には，XML (Extensible Markup Language) [20] が採用されており，RDF/XML と呼ばれる．また，RDF グラフを表現する際に用いる語彙の定義を行うための仕様として，RDF Schema [21] が提案されている．RDF Schema を用いることで，資源のクラスや資源間のプロパティを定義できる．すなわち，RDF Schema で記述された情報は，RDF グラフのためのスキーマ情報となる．

現在，RDF はさまざまなメタデータを記述するために利用されはじめている．たとえば，RSS 1.0 [17] や FOAF [7], CC/PP [22] などといったメタデータ記述言語は， RDF に基づいて定義されている．今後，これらの RDF に基づいたメタデータ記述言語の普及に伴い，Web 上のメタデータの質が向上するとともに，RDF の利用価値が認識され，RDF で記述されたメタデータ（以降，RDF データと呼ぶ.）が増加することが予想される．このため，大量の RDF データを高速に処理することのできる RDF データベースが重要である．

RDF データベースを実現する一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，この手法には本質的な問題がある．それは，RDF/XML 文法で記述した場合，一つの RDF データであっても，平坦に文を列挙する方法や関連する文を入れ子にする方法など，複数の表現が存在することである．この問題によって，意図する RDF データに対する問合せを，XML に対する問合せに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，関係データベースや Berkeley DB を利用する方法が挙げられる．これまで，いくつかの RDF データベースが提案されており，それらのほとんどがこれらのデータベースを用いて実現されている [2,3,5,12,14,16,18]．これまでの RDF データベースは，RDF データから文を抽出し，それらを平坦に列挙して格納する手法を採用している．これによって，RDF グラフ全体を一括して管理できるが，これらに対して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言えない．また，この手法を採用すると，RDF のスキーマ情報を通常の RDF データと同様に扱うため，RDF スキーマデータ に対する処理を効率的に行うことはできない．

本論文では，RDF データを関係データベースに効率よく格納，検索する手法を提案する．提案手法は，RDF データがラベルを持つ有向辺で構成された有向グラフ構造であることに着目した手法である．まず，RDF データからすべての文を抽出し，述語の種類によって文を分類する．具体的には，クラスの継承関係を表現する述語，プロパティの継承関係を表現する述語，プロパティの定義域と値域を表現する述語，資源のタイプを表現する述語，および，これら 4 種類を除く残りの述語（一般述語）である．これらの分類に基づいて，同一種に分類された文で構成される 5 種類の部分グラフに対して，述語の意味を踏まえた適切な手法で関係表に格納する(注1)．具体的には，クラスやプロパティの継承関係で構成される部分グラフは，任意の 2 要素間の接続関係の有無を知るためのナンバリングスキームを適用する．また，一般述語で構成される部分グラフは，経路式に基づいて関係表に格納することで，経路に基づいた問合せにおいて結合演算の回数を減少させることができる．

本論文で提案する関係スキーマは次の六つの関係表から成る．1) 資源の名前空間の情報を管理するための関係表，2) クラス情報を格納する関係表，3) プロパティ情報を格納する関係表，4) 資源の情報を格納する関係表，5) プロパティのみで構成される経路式を格納する関係表，6) 資源間の関係を文の形式で管理する関係表である．これらの関係スキーマは，対象の RDF スキーマ情報に依存しない．

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 章で関連した研究について議論し，4 章では，提案する RDF データの関係データベースへの格納法について述べる．主に，分類する 5 種類の述語で構成される部分グラフや，クラスとプロパティ継承関係グラフに適用するナンバリングスキーム，一般述語で構成される部分グラフを表現する経路式に関して述べる．また，5 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．最後に 6 章でまとめる．

2 RDF の概要

RDF (Resource Description Framework) [19] 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 [19] では，資源間の構造的な情報である RDF グラフとその記述と移植のための文法，および，その意味論を定義している．RDF を用いることで資源に対して構造的なメタデータを付加することができる． 資源とは，URI (Uniform Resource Identifier) [4] を用いて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．また，RDF Schema [21] 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 Schema を用いることで，RDF グラフを表現する際に用いるクラスやプロパティを定義できる．

RDF グラフは，資源間の二項関係を表現することができる文 (*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，文を用いることで，“This paper is authored by Akiyoshi MATONO.” という情報を表現することができる．文は，主語 (*rdf:subject*, “This paper”) と述語 (*rdf:predicate*, “is authored by”)，目的語 (*rdf:object*, “Akiyoshi MATONO”) という三つの部分によって構成されている（図 1）．文の集合を用いることで，主語と目的語を頂点とし，述語が有向辺に対応した有向辺にラ

(注1)：前提条件として， 一般述語で構成される部分グラフの構造を，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．

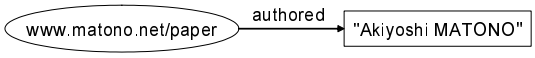

図 1　単純な文の例.

Fig. 1　A simple statement example.

ベルを持つ有向グラフ構造の情報を表現することができる.

RDF Schema [21] は，RDF データで用いる語彙を定義するための仕様で，RDF Schema を用いることで，クラス (*rdfs:Class*) やプロパティ (*rdf:Property*) を定義することができる．さらに，プロパティの定義域 (*rdfs:domain*) や値域 (*rdfs:range*) ，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．これらのクラスやプロパティは，他の RDF データ内で表現される資源のタイプ (*rdf:type*) として利用する．すなわち，RDF データは，RDF スキーマデータのインスタンス情報であり，RDF Schema を用いて定義した情報は，RDF データのためのスキーマ情報であると言える．RDF スキーマデータも通常の RDF データと同様に，文によって構成されている．そのため，RDF スキーマデータで用いる語彙(注2) は，RDF Schema の仕様で暗黙的に定義されている．すなわち，RDF Scheme を用いて定義するための暗黙的な文は，メタスキーマ情報であるといえる．また，RDF では，URI で記述できるものはすべて資源 (*rdfs:Resource*) であるため，クラスやプロパティはもちろん，RDF Schema の仕様で定義された暗黙的な語彙もすべて資源である.

RDF と RDF Schema を用いたやや複雑な RDF グラフの例を図 2 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，値域として *Artifact* クラスを持つ(注3)．また，*Painter* クラスは，*Artist* クラスのサブクラスで，*paints* プロパティは，*creates* プロパティのサブプロパティである．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．*www.picasso.net* や *www.picasso.net/guernica* はクラスのインスタンスで，言い換えると，この資源のタイプ (*rdf:type*) は，*Painter* クラスと *Painting* クラスである．また，“Picasso” などの文字列は，リテラル (*rdfs:Literal*) である.

2.2　RDF データの特徴

本節では，RDF データの特徴について考察する．データサイズに関する面と，RDF グラフの構造に関する面の二つの側面から，現在普及している代表的な RDF 文書 RSS [17], FOAF [7], Dublin Core [8]，Wordnet [13], OPD (Open Directory Project) [15]，Gene Ontology [9] を検討する.

まず，現在の RDF 文書はデータサイズの面から，大きく二つに分類できる．一つは，Wordnet や OPD，Gene Ontology で，

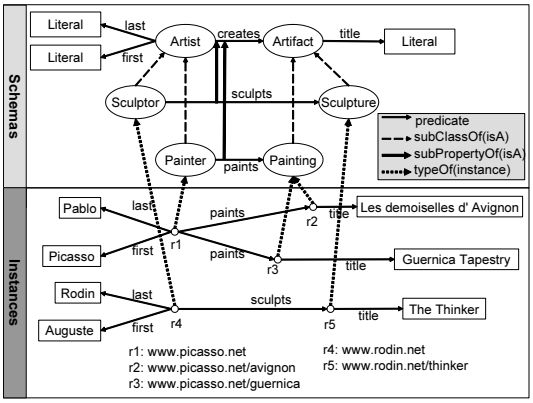

図 2　RDF と RDF Schema を用いた RDF グラフの例.

Fig. 2　An example using RDF and RDF Schema.

語彙やゲノムなどの資源を体系化するための RDF データであるため一般に大きなサイズを持つ．他方は，RSS や FOAF, Dublin Core で，Web 上に実際に存在する画像やページなどの資源に対してメタデータを付加することを目的とした RDF であるため，一般にそのサイズは小さい．また，後者の RDF 文書は単体で用いることはあまりなく，複数の RDF 文書を用いて初めて効率的なメタデータとしての役割を果たすことができる.

RDF グラフの構造に関して考察する前に，RDF グラフを形成する述語（有向辺）の種類に関して考察する必要がある．RDF データは，述語によって構造化されている．述語は，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，ある文を経由すると有向グラフの構造が発散するものを発散述語，ある文で終端する述語，あるいはその文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ことが多い.

次に，RDF グラフの構造に関して考察する．現在普及している RDF 文書のグラフの構造は，ほとんど巡回を含まないような単純な構造であることが一般的で，単体の RDF 文書で巡回を多く含んだものは，我々の知る限りない．実際，前述した六つのメタデータ規格は，いずれも単一文書では，木あるいは非巡回有向グラフの構造で，ほとんど巡回を含まない．この理由として，RDF 文書内で用いられる述語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ためである．これは，一つの 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要因であると考えられる.

3　関 連 研 究

これまでにいくつかの，RDF データベースが提案されてきた [2, 3, 5, 12, 14, 16, 18]．これらのほとんどは，内部で関係デー

(注2)：*rdfs:Resource*, *rdfs:Class*, *rdf:Property*, *rdfs:subClassOf*, *rdfs:subPropertyOf*, および，*rdf:type* など

(注3)：図 2 の上部の RDF スキーマデータはわかりやすくするために，いくつかの表現を省略してある．具体的には，*creates* などのプロパティは，RDF スキーマ定義では，述語ではなく，主語として記述し，その定義域や値域にクラスを定義している．そのため，図として表現する場合，有向辺として表現するのではなく，頂点として表現するほうが本来は正しい.

表 1　RDF データベースの格納手法と問合せ言語.
Table 1　Storage methods and query languages of RDF databases.

	格納手法	問合せ言語
RDFSuite [2]	flat, schema	RQL
Redland [3]	hash	Triple-matching
Sesame [5]	schema	SeRQL
Jena2 [12]	flat	RDQL
Inkling [14]	flat	SquishQL
RDFStore [16]	hash	SquishQL
rdfDB [18]	hash	SquishQL

タベース，あるいは Berkeley DB を利用して実現されている．関係データベースを利用した RDF データの格納方法には大きく分けて 2 種類あり，一方は文を単一関係表に平坦に列挙する手法で，他方は RDF スキーマデータ に依存して定義されたクラスとプロパティごとに関係表を生成し，関連する資源を格納する手法である．また，Berkeley DB を利用した RDF データの格納では，3 種類のハッシュ表を用いる．具体的には，主語，述語，目的語をそれぞれのハッシュ表のキーとして用い，値には文を用いる．実装によっては，キーと値に採用しているデータは多少異なるが，基本的には文による問合せを意識した格納方法である．

表 1 に これまで提案されてきた代表的な RDF データベースの格納手法と問合せ言語を示す．本論文では，関係データベースを用いて単一表に平坦に格納する手法のことを flat アプローチ，RDF スキーマデータ に依存して定義されたクラスとプロパティごとに関係表を生成する手法を schema アプローチ，Berkeley DB を利用した 3 種類のハッシュ表を用いる手法を hash アプローチと呼ぶ.

これまでの手法は文単位で格納しているため，単一の文の検索を行うには効率的な結果を得ることが期待できる．しかしながら，RDF データの検索の多くは，部分グラフを与え，それに一致する部分グラフを抽出するものであるため，比較的大きな部分グラフの発見を行う場合，これらの手法では，結合演算を頻繁に行う必要があり，大きなコストを要する．また，flat アプローチと hash アプローチといった文を単一表で平坦に管理することは，RDF のスキーマ情報を通常の RDF データと同様に扱うことを意味しており，RDF スキーマデータ に対する効率的な処理を行うことはできない．RDFSuite [2] は，RDF Schema に対応している数少ない RDF データベースの一つである．しかしながら，RDFSuite は，RDF スキーマデータに依存して関係スキーマが変化する．このため，RDF スキーマデータが変化するたびにデータベースを再構築しなければないらない.

一方，大量の RDF データを効率的に検索するための手法として索引が提案されている [24] [6]. 我々は，これまで，経路式に基づく RDF データのための接尾辞配列を提案してきた [24] . 具体的には，RDF グラフから意味の異なる述語で構成された 4 種類の部分グラフを抽出し，部分グラフから生成された経路式に対して，接尾辞配列を生成する手法である．これによって経路式に基づいた問合せを効率的に検索することができる．また，Christophides 等 [6] 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ている．具体的には，Agrawal 等 [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し，それに対してラベルをつける手法である．

4　提 案 手 法

本章では，RDF データを関係データベースを用いて，経路式に基づいた格納と検索を行う手法を述べる．提案する格納法は，RDF データが有向辺にラベルを持つ有向グラフ構造であることに着目した手法である．まず，RDF グラフから，種類の異なる述語を持つ文で構成される部分グラフを抽出する（4.1 節）. 具体的には，クラス継承グラフ，プロパティ継承グラフ，タイプグラフ，プロパティ定義域値域グラフ，および一般述語グラフの 5 種類である．次に，抽出した部分グラフを，提案する関係スキーマに基づいて格納する．この際，それぞれの部分グラフに対して，適切な手法を用いて格納する．たとえば，クラス継承グラフとプロパティ継承グラフは，4.3 節で述べるナンバリングスキームを用いて格納し，一般述語グラフは，4.4 節で述べる経路式に基づいて格納する．これによって，スキーマ情報に基づく問合せや経路に基づく問合せを効率的に処理することができる.

我々が対象とする RDF データは，2.2 節で述べた分類において，データサイズが大きい RDF 文書を対象とする．つまり，Wordnet や OPD，Gene Ontology といった語彙や情報を体系化し，単一文書で多数の資源に対するメタデータを表現する RDF 文書である．また，一般述語グラフ（4.1 節で述べる）に巡回を含まないことも条件とする．前述したようにこのような RDF 文書は，多く Web 上に流通しており，本手法の適用範囲にそれほど制限を受けないと言える.

4.1　部分グラフの抽出

RDF データは，RDF Schema の仕様で定義された暗黙的な語彙の RDF メタスキーマデータと RDF Schema を用いて定義する RDF スキーマデータ，それらのインスタンスである RDF インスタンスデータで構成されている．これまでの RDF データベースでは，これらを同等に扱っており，すべてを一つの RDF グラフとして扱っていた．しかしながら，この手法では一つのグラフですべてを表現しているため，その構造が複雑になり，その結果格納する単位を文単位としなければならず，非効率的である．これまでの手法に対し，我々は文の述語の種類によって RDF グラフをいくつかの部分グラフに分割する手法を提案する．それぞれの部分グラフは，元の RDF グラフと比べ構造を単純化でき，また，それぞれに適した格納手法を利用することができる．具体的には RDF データ全体から得られる部分グラフは以下の 5 種類になる.

- **クラス継承グラフ**

この部分グラフは，クラス間の継承を表現する述語 *rdfs:subClassOf* を含む文で構成される．継承の関係を表現しているグラフであるため，非巡回有向グラフ構造であり，クラスが頂点となり，有向辺はラベルを持たない.

- **プロパティ継承グラフ**

この部分グラフは，プロパティ間の継承を表現する述語 *rdfs:subPropertyOf* を含む文で構成される．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，プロパティが頂点となり，有向辺はラベルを持たない．

- **タイプグラフ**

この部分グラフは，資源とその資源のタイプを表現した述語 *rdf:type* を含む文で構成される．このグラフは巡回を含まない平坦な有向グラフ構造で，頂点はクラスあるいインスタンスであり，有向辺はラベルを持たない．

- **プロパティ定義域値域グラフ**

この部分グラフは，プロパティの定義域を表現する述語 *rdfs:domain* と値域を表現する述語 *rdfs:range* を含む文で構成される．定義域や値域を定義する場合，主語はプロパティ，述語は *rdfs:domain* あるいは *rdfs:range*，目的語はクラスである．このため，頂点がプロパティあるいはクラス，有向辺は *rdfs:domain* あるいは *rdfs:range* である．このグラフの構造は巡回を含まない平坦な有向グラフである．

- **一般述語グラフ**

この部分グラフは，RDF データのグラフ全体から上記の 4 種類の特殊な述語を含む文を除く，残りのすべてのプロパティで構成される．このグラフは，主語と目的語を頂点とし，述語を有向辺としたグラフで，複数の種類のプロパティで構成される部分グラフとなる．本来は，巡回を含む有向グラフ構造であるが，本論文では，この一般グラフに巡回を含まない RDF データを対象とする．

4.2 提案スキーマの基本方針

我々が提案する関係スキーマは，4.1 節で述べた部分グラフをそれぞれ別々に関係表に格納することを前提に設計した（4.5 節）．しかし，プロパティ継承グラフとプロパティ定義域値域グラフは，ともに RDF データを構成する基本要素である *rdf:Property* の情報を表現しているため，同一関係表に格納する．さらに，タイプグラフと一般述語グラフも *rdfs:Resource* の情報を表現しているため，同一関係表に格納する．したがって，提案するスキーマは，クラス継承グラフを表現する class 表，プロパティ継承グラフとプロパティ定義域値域グラフを表現する property 表，タイプ継承グラフと一般述語グラフを表現する resource 表の三つの関係表を基本とする．また，それらを補うための関係表として，namespace 表，triple 表，および ppath 表があり，提案スキーマは六つの関係表で構成される．

クラス継承グラフとプロパティ継承グラフは，任意の 2 要素間の接続関係を容易に知ることが重要であるため，4.3 節で述べるインターバルナンバリングスキームを用いる．また，一般述語グラフは，経路に基づいた問合せを効率的に処理するため，4.4 節で述べる経路式によって表現する．また，タイプグラフとプロパティ定義域値域グラフの構造は平坦であるため，特殊な手法を用いることなく容易に表現できる．

4.3 ナンバリングスキームによる継承関係の表現

本節では，インターバルナンバリングスキーム [11] を非巡回有向グラフに適用する手法について述べる．インターバルナンバリングスキームとは，XML データなどの木構造のデータに対し，各頂点に数字の識別子を割り当てる手法の一つで，具体的には，深さ優先探索の前置順と後置順，および木の根からの距離の三つの数字を割り振る．本手法では，この手法を適用する前処理として，非巡回有向グラフを木構造に展開し，その後，インターバルナンバリングスキームを適用する．

ナンバリングスキームのアルゴリズムを図 3 に，例を図 4 に示す．まず，入次数が 0 であるすべての頂点（図 4 では A と D）の親となる仮想的なルート頂点を設定する．その後，入次数が 1 以上であるすべての頂点 (E) を多重辺を含まないようにコピーを生成する (E', E")．このとき生成されたグラフは，木構造となる（図 4 右）．最後に，生成された木に対して，XML データに対するインターバル手法と同様に (pre-order, post-order, depth) の形で数字を割り振る．このとき，木を生成する以前は一つであった頂点 (E) に対しては，複数の異なる数字が割り振られる．この手法をクラス継承グラフとプロパティ継承グラフに適用し，各頂点の数字を調べることで二つのクラス（プロパティ）間に継承関係があるかどうかを容易に知ることができる．

インターバルナンバリングスキームを用いて数字を割り振られた頂点間の接続関係の定理について述べる．

［**定理** 1］ 任意の頂点 v と u において，$(pre(v_i), post(v_i), depth(v_i)),\ i = \{1, 2, \ldots, m\}$ と $(pre(u_j), post(u_j), depth(u_j)),\ j = \{1, 2, \ldots, n\}$ は，それぞれ v と u に割り振られた数字である．頂点 v が，頂点 u の先祖であるならば，次の条件を満たす (i, j) が少なくともひとつ存在する．

$$pre(v_i) < pre(u_j) \wedge post(u_j) < post(v_i)$$

このときの v と u の距離は，$depth(u_j) - depth(v_i)$ である． □

図 4 の 2 頂点 A と E に対して，定理 1 を適用した例を挙げる．A の数字は (2,7,1)，E の数字は (4,1,3)，(6,3,3)，(8,5,2) である．このとき，両者の数字のすべての組み合わせを比較する．まず，A の 数字と E の 1 番目の数字で比較すると，$2 < 4 \wedge 7 > 1$ が成り立つため，A から E に向かう長さ 2 の経路が存在する．また，E の 2 番目の数字で比較すると $2 < 6 \wedge 7 > 3$ が成り立つため，A から E に向かう長さ 2 の経路が他にも存在する．さらに，E の最後の数字で比較すると $2 < 8 \wedge 3 > 5$ は成立しないため，A から E に向かう経路は二つあることが確認できる．

4.4 経 路 式

本節では，一般述語グラフの情報として関係表に格納する経路式を定義する．我々は，RDF グラフに対する問合せも経路式で行うことを前提としているが，本論文では問合せ経路式の定義に関しては割愛する．まず，それぞれの経路を定義する．

［**定義** 1］（有向辺経路） 非巡回有向グラフにおいて，$a(v_m, v_n)$ は頂点 v_m から頂点 v_n へ向かう a というラベルを持つ有向辺を指す．始点 v_0 から終点 v_k へ向かう有向辺経路とは，有向辺の有限列

$$\langle a_0(v_0, v_1), a_1(v_1, v_2), \ldots, a_{k-2}(v_{k-2}, v_{k-1}), a_{k-1}(v_{k-1}, v_k)\rangle$$

とする．有向辺経路の経路式は有向辺のラベルを列挙した

$$\langle a_0, a_1, \ldots, a_{k-2}, a_{k-1}\rangle$$

のように定義する． □

```
Input: A directed acyclic graph G.
Output: G′ including node numbers.

V = the set of nodes in G
pre = 1 : Integer;
post = 1 : Integer;
depth = 0 : Integer;
foreach ( v in V){
   if the in-degree of v == 0 {
      assignNumbers (v);
   }
}
return G′;

function assignNumbers (v) {
   pre = pre + 1;
   depth = depth + 1;
   Append the number pre to node v in G.
   Append the number depth to node v in G.
   U = a set of nodes to which adjoined from v.
   foreach(u in U){
      assignNumbers(u);
   }
   Append the number post to node v in G.
   post = post + 1;
   depth = depth – 1;
}
```

図 3　非巡回有向グラフに対するナンバリングスキームのアルゴリズム

Fig. 3　Algorithm for assigning node numbers to DAGs.

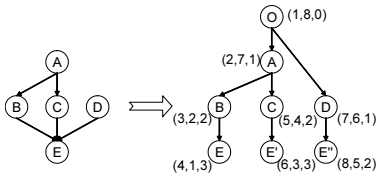

図 4　非巡回有向グラフに対するナンバリングスキームの例.

Fig. 4　An example of numbering scheme algorithm for DAG.

```
propertyPath ::= (property '.')* property
property     ::= [0-9]+       // 一般述語グラフのプロパティの識別子
```

図 5　経路式の文法（EBNF 表現）.

有向辺経路式の文法を図 5 に示す．図において，propertyPath は有向辺経路式を表し，また，property はプロパティを示す識別子である．経路式の区切り文字としては‘.’を使う．

4.5　提案スキーマ

図 6 に提案する関係スキーマを示す．namespace 表は RDF 文書の名前空間情報を格納し，property 表や class 表は RDF スキーマデータで定義されたプロパティやクラスを列挙している．property 表の *domain* と *range* はプロパティ定義域値域グラフを表現するために用い，property 表と class 表の *pre* や *post*, *depth* に，4 節で述べたナンバリングスキームを用いて数字を割り振ることで，プロパティ継承グラフとクラス継承グラフを表現する．このとき，入次数が複数存在する頂点は，複数の異なる数字を持つため，複数行にわたって情報を格納する．triple 表には一般述語グラフを形成する文を格納する．resource 表には一般述語グラフの各資源を格納し，*type* にその資源のクラスの識別子を格納することで，タイプグラフを表現する．ppath 表には一般述語グラフの有向辺経路式を格納する．resource 表の *ppathID* は ppath 表の *pathID* を参照することで，その資源への

```
CREATE TABLE namespace ( nsID int,
                         URI  varchar(512))
CREATE TABLE class ( classID   int,
                     nsID      int,
                     className varchar(512),
                     pre       int,
                     post      int,
                     depth     int )
CREATE TABLE property ( propertyID   int,
                        nsID         int,
                        propertyName varchar(512),
                        domain       int,
                        range        int,
                        pre          int,
                        post         int,
                        depth        int )
CREATE TABLE triple ( subject   int,
                      predicate int,
                      object    int )
CREATE TABLE ppath ( ppathID  int,
                     ppathexp varchar(256) )
CREATE TABLE resource ( resourceID   int,
                        nsID         int,
                        resourceName text,
                        ppathID      int,
                        type         int )
```

図 6　提案する関係スキーマ.

Fig. 6　Our proposing relational schema.

ppath

pathID	*pathexp*
1	”
2	’2’
3	’2.4’
4	’3’

namespace

nsID	*prefix*	*URI*
1	’rdf’	’http://www.w3.org/1999/02/22-rdf-syntax-ns#’
2	’rdfs’	’http://www.w3.org/2000/01/rdf-schema#’
3	’cul’	’http://Examples/schema.rdf#’

property

pID	*nsID*	*pName*	*domain*	*range*	*pre*	*post*	*depth*
0	0	”	0	0	1	5	0
1	3	’creates’	2	3	3	3	1
2	3	’paints’	4	5	4	2	2
3	3	’name’	2	1	2	1	1
4	3	’title’	3	1	5	4	1

class

cID	*nsID*	*cName*	*pre*	*post*	*depth*
0	0	”	1	6	0
1	2	’Literal’	6	5	1
2	3	’Artist’	2	2	1
3	3	’Artifact’	4	4	1
4	3	’Painter’	3	1	2
5	3	’Painting’	5	3	2

resource

rID	*nsID*	*rName*	*ppathID*	*type*
1	0	’www.picasso.net’	1	4
2	0	’www.picasso.net/guernica’	2	5
3	0	’Guernica’	3	1
4	0	’Picasso’	4	1

triple

subject	*predicate*	*object*
1	3	4
1	2	2
2	4	3

図 7　RDF データ格納の例.

Fig. 7　An example of storing RDF data

到達経路を表現する．このとき，ある資源への到達経路が複数存在する場合は，そのすべての出現を異なる行として resource 表に格納する．図 7 に図 2 で示した RDF の例の一部を格納した例を示す(注4)．

4.6　問　合　せ

RDF はスキーマ情報を用いてメタデータを体系的に表現している．そのため，RDF に対する問合せには，スキーマ情報に基づいた問合せが必要であると考えられる．また，RDF データは，有向グラフ構造であるため，今後は，経路式を含む問合せがより重要になると思われる．実際， RDF データに対する経路式に基づいた問合せ言語がいくつか提案されている [23]．今回提案するスキーマでは，これらのどちらの問合せでも処理することができる．紙面の都合で例を挙げるだけに留めるが，例えば 図 9 に示す RDQL [10] の問合せは図 8 のように表現する

(注4)：ここに示した例は，簡単のためにメタスキーマデータに関しては省略してある．

```
SELECT DISTINCT result.resourceName
FROM   ppath AS pp, resource AS result
WHERE pp.ppathexp LIKE
                ( SELECT '%' || p1.propertyID || '.'
                             || p2.propertyID
                  FROM ( SELECT DISTINCT propertyID
                           FROM property AS p
                           WHERE p.propertyName = 'part_of' )
                         AS p1,
                         ( SELECT DISTINCT propertyID
                           FROM property AS p
                           WHERE p.propertyName = 'n_associations' )
                         AS p2 )
AND    pp.ppathID = result.ppathID
```

図 8　経路式に基づいた問合せ（SQL）.

Fig. 8　A query based on path expression (SQL.)

```
SELECT ?b
WHERE (?a, <part_of>, ?c),
      (?c, <n_associations>, ?b)
```

図 9　経路式に基づいた問合せ（RDQL）.

Fig. 9　A query based on path expression (RDQL.)

ことができる.

5 性 能 評 価

本章では，実験による本手法の性能評価を行う．評価実験は，提案する手法と flat アプローチに基づく RDF データベースである Jena2 [12] の処理時間を比較する.

実験環境は，関係データベースとして，PostgreSQL 7.4.1 を用い，Athlon 1.4 GHz の CPU と 1024 MB のメモリ，OS に Gentoo Linux 1.4 (注5) を搭載した計算機を用いた.

5.1 実験で用いた問合せ

実験では RDF に対する問合せとして，経路式に基づく問合せと RDF スキーマ基づく問合せの 2 種類の問合せで比較を行った．実験で用いた RDF スキーマに基づく問合せを以下に列挙する．経路式に基づく問合せは，経路の長さを 2 から 8 まで変えたものを用いた.

（1） あるクラスを継承したすべての子孫クラスの検索
（2） あるクラスを直接継承した子孫クラスの検索
（3） あるクラスを継承したすべての祖先クラスの検索
（4） あるクラスを直接継承した祖先クラスの検索
（5） あるプロパティを継承したすべての子孫プロパティの検索
（6） あるプロパティを直接継承した子孫プロパティの検索
（7） あるプロパティが継承したすべての祖先プロパティの検索
（8） あるプロパティが継直接承した祖先プロパティの検索
（9） あるクラスを定義域とするプロパティの検索
（10） あるクラスを値域とするプロパティの検索
（11） あるプロパティの定義域となるクラスの検索
（12） あるプロパティの値域となるクラスの検索
（13） あるクラスをタイプとする資源の検索
（14） ある資源のタイプであるクラスの検索

5.2 実験データ

本手法に適した RDF 文書は，経路式が長くてかつその RDF グラフに多重辺を多く含まない RDF 文書である．そのため，2.2 節で挙げた RDF 文書のうち，もっとも我々の手法に適していると考えられるのは OPD [15] 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 [13] 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 [9] を実験データとして採用した．なお，そのデータサイズは 1 MB とした．Gene Ontology は，多重辺を多く含む非巡回有向グラフ構造であるため，経路式の数が他の二つに比べ増加の割合が高く，我々の手法では，効率的な性能を期待できない．しかし，逆に言えば，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.

(注5)：Linux version 2.6.1-rc1 (gcc version 3.2.3)

表 2　RDF 文書の資源数，文数および各関係表の行数.

Table 2　The numbers of resources and statements of dataset, and sizes of the tables.

資源数	10,630
文数	16,388
namespace 表の行数	6
class 表の行数	14
property 表の行数	26
resource 表の行数	173,977
triple 表の行数	15,527
ppath 表の行数	113

表 3　RDF スキーマ情報に基づく問合せの処理時間 (msec).

Table 3　The processing times of the queries based on RDF schema data.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
jena2	—	2.37	—	1.97	—	2.07	—	1.99	2.56	2.43	2.1	2.0	10.52	2.02
ours	2.84	2.76	2.33	2.51	2.42	2.55	2.47	2.55	2.79	2.64	2.42	2.38	42.14	2.42

5.3 実 験 結 果

表 2 に Gene Ontology の RDF 文書に含まれる資源数と文数，およびその RDF 文書を提案スキーマに格納したときの各関係表の行数を示す．この表からわかるように，resource 表行数が，資源数や文数に比べ，非常に大きいことがわかる．これは，実験に用いた Gene Ontology の RDF 文書が，多くの多重辺を持っているため，経路数が膨大であることを意味している.

表 3 に，4.6 節で挙げた RDF スキーマ情報に基づく 14 種類の問合せを発行したときの処理速度を示す．flat アプローチでは，問合せ 1，3，5，7 を処理することはできないため，計測することはできなかった．この表を見る限り，我々の手法は flat アプローチに比べ，やや劣っているがほぼ同じ値を示していることが確認できる．その理由としては，我々の関係スキーマでは，識別子を用いて一意に管理しており，資源の URL を namespace 表や class 表といった異なる関係表間で格納しているため，結合演算を行う必要があるためである．例外的に，問合せ 13 は他と比べ処理時間に開きが見られる．これは，前述したように resource 表で重複行があり，行数が膨大であることが原因であると考えられる.

図 10 に経路式に基づく問合せの処理時間を示す．縦軸は処理時間（ミリ秒），横軸は問合せに用いた経路式の長さである．その処理時間の詳細を表 4 に示す．これらが示すように，提案手法は経路式に基づく問合せに対し，非常に効率的であることが分かる．この理由は以下の通りである．Jena2 が採用してい

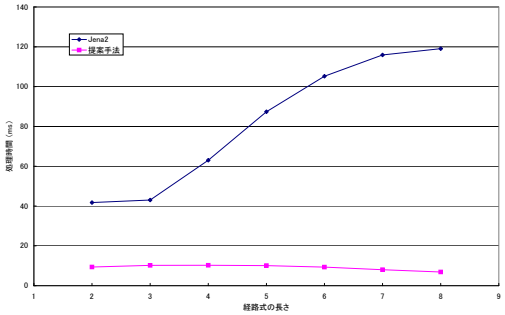

図 10　経路式問合せの処理時間.

Fig. 10　The processing times for path-based queries.

表 4　経路式に基づく問合せの処理時間 (msec).

Table 4　The processing times of the path-based query(msec.)

長さ	2	3	4	5	6	7	8
jena2	41.74	43.02	63.03	87.38	105.23	115.92	119.06
ours	9.39	10.19	10.23	10.09	9.34	8.03	6.89

る flat アプローチでは，RDF データは文単位で関係表に格納されている．このため，経路式の文を一つ辿るたびに一回の結合演算を行うことになる．これに対して，提案手法では経路をあらかじめ列挙した上で経路式として関係表に格納しているので，問合せとして与えられた経路式の検索は単純な文字列マッチングで実現することができる．この差は経路式の長さが長くなったときに顕著になる．

また，Jena2 では経路式が増すにつれて処理が遅くなっているのに対し，我々の手法ではむしろ処理が早くなる傾向が見られる．これは，経路式が長くなることによって，問合せの選択度 (selectivity) が低くなり，解集合のサイズが小さくなることに起因していると思われる．

6　おわりに

本論文では，RDF データを関係データベースに格納する効率的な手法を提案した．これまで提案されてきた RDF データベースは，スキーマデータとインスタンスデータを区別なく扱っており，また RDF グラフを文に細分化して，平坦に格納している．このため，結合演算を多用する必要があり，膨大なデータに対して検索コストが高くなると考えられる．我々の手法では，特殊な述語を含む文で構成される 5 種類の部分グラフを抽出して，スキーマ情報とインスタンス情報を区別して扱っているため，flat アプローチや hash アプローチでは実現できない RDF スキーマに基づく問合せを処理することができた．また，経路式に基づく問合せに対して，提案手法は効率的であることを確認できた．今後の課題として，より大規模な RDF 文書を用いた性能評価を行う必要がある．また，問合せ言語の検討および巡回を含む RDF データへの対応などが挙げられる．

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．

文　献

[1] R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In J. Clifford, B. G. Lindsay, and D. Maier eds., *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989*, pp. 253–262. ACM Press.

[2] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The RDFSuite: Managing Voluminous RDF Description Bases. In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001 Hongkong, China.*, May 1 2000.

[3] D. J. Beckett. The design and implementation of the redland RDF application framework. In *World Wide Web*, pp. 449–456, 2001.

[4] T. Berners-Lee, R. T. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. http://www.isi.edu/in-notes/rfc2396.txt, August 1998. RFC2396.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. Hendler eds., *Proceedings of the First Internation Semantic Web Conference*, No. 2342 in Lecture Notes in Computer Science, pp. 54–68. Springer Verlag, July 2002.

[6] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the semantic web. In *Proceedings of the twelfth international conference on World Wide Web*, pp. 544–555. ACM Press, 2003.

[7] Dan Brickley and Libby Miller. FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/. RDFWeb Namespace Document 16 August 2003.

[8] Dublin Core Metadata Element Set, Version 1.1. http://dublincore.org/documents/dces/, June 2 2003.

[9] GENE ONTOLOGY CONSORTIUM. http://www.geneontology.org/.

[10] Hewlett-Packard Company. RDQL – RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.

[11] Q. Li and B. Moon. Indexing and querying XML data for regular path expressions. In *The VLDB Journal*, pp. 361–370, 2001.

[12] B. McBride. Jena: Implementing the RDF Model and Syntax Specification. In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001 Hongkong, China*, May 2001.

[13] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to WordNet: An On-Line Lexical Database. http://www.cogsci.princeton.edu/ wn/, 1993.

[14] L. Miller. Inkling: RDF query using SquishQL. http://swordfish.rdfweb.org/rdfquery.

[15] Netscape. Open Directory Project. http://dmoz.org/.

[16] Rdfstore – perl api for rdf storage. http://rdfstore.sourceforge.net/.

[17] RSS-DEV Working Group. RDF Site Summary (RSS) 1.0. http://web.resource.org/rss/1.0/, December 2000.

[18] R.V.Guha. rdfDB : An RDF Database. http://guha.com/rdfdb/.

[19] World Wide Web Consortium. Resource Description Framework(RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, Febuary 1999. W3C Recommendation 22 February 1999.

[20] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/REC-xml, October 2000. W3C Recommendation 6 October 2000.

[21] World Wide Web Consortium. Resource Description Framework(RDF) Schema Specification 1.0. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, March 2000. W3C Candidate Recommendation 27 March 2000.

[22] World Wide Web Consortium. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, January 2004. W3C Recommendation 15 January 2004.

[23] 松山, 北川, 斎藤. パスを利用した RDF 問い合わせ言語の提案. 情報処理学会第 45 回プログラミングシンポジウム, 1 2004.

[24] 的野, 天笠, 吉川, 植村. 接尾辞配列に基づいた RDF データのための索引手法. 情報処理学会論文誌: データベース, 45(SIG(TOD21)), 2004.