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

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

的野 晃整♡ 天笠 俊之♢
吉川 正俊♠ 植村 俊亮♢

Akiyoshi MATONO Toshiyuki AMAGASA
Masatoshi YOSHIKAWA Shunsuke UEMURA

本稿では，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，RDF データをいくつかの部分グラフに分割し，それぞれの部分グラフに対して適切な手法を用いて関係表に格納する．これによって，スキーマ情報に基づいた問合せや経路に基づいた問合せを効率的に処理することができる．

In this paper, we propose a path-based scheme for storage and retrieval of RDF data in relational databases. We first divide an RDF graph into sevral subgraphs, and then store them into relational tables by applying appropriate techniques for representing the subgraphs. It is possible to process queries including schema information and/or path expressions efficiently.

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．Semantic Web におけるメタデータは，一般に RDF (Resource Description Framework) [11] に基づいて記述される．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おり，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 で記述されたメタデータ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 [1–3, 6–9]．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，単一の文の検索は効率的に行うことが予想できるが，連続した複数の文を検索する処理には，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．

♡ 学生会員 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
akiyo-ma@is.naist.jp
♢ 正会員 奈良先端科学技術大学院大学情報科学研究科
{amagasa,uemura}@is.naist.jp
♠ 理事 名古屋大学情報連携基盤センター
yosikawa@itc.nagoya-u.ac.jp

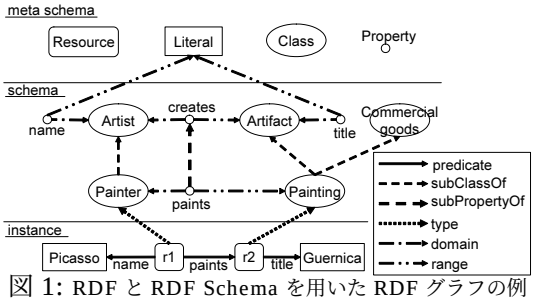

図 1: RDF と RDF Schema を用いた RDF グラフの例
Fig.1 An example of RDF graph using RDF and RDF Schema.

本稿では，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．次に同一種に分類された文で構成される部分グラフを構成する．我々はこれらの部分グラフは基本的に異なる関係表に格納することを前提とした．このとき各部分グラフを関係表の上で表現するために，述語の性質を踏まえた適切な手法を採用する．具体的には，経路式による問合せが頻繁に行われる部分グラフに対しては経路式に基づく手法を採用し，任意の 2 要素間の接続関係を容易に知る必要がある部分グラフではインターバルナンバリングスキームを拡張した手法を採用する．これによって，経路式に基づく問合せや継承関係に基づく問合せを効率的に処理できる．

2. RDF の概要

RDF [11] は，メタデータ記述と処理のための基礎となる枠組みを提供している．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．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グラフ構造の情報を表現することができる．RDF Schema [10] は，RDF データのスキーマ情報を定義するための仕様で，クラスやプロパティなどを定義できる．さらに，プロパティの範囲と定義域，クラスやプロパティの継承を定義することができる．

RDF と RDF Schema を用いた RDF グラフの例を図 1 に示す．図の上部は，RDF Schema の仕様自体であるメタスキーマデータを示した図である．図の中部は，RDF Schema を用いて定義された RDF スキーマデータを示しており，図の下部はインスタンスとしての資源とそれらの関係を示した RDF データである．*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，*Painter* クラスは，*Artist* クラスを継承している．*r1* や *r2* はそれぞれ *Painter* と *Painting* クラスのインスタンスである．また，“Picasso” などはリテラルである．

3. 関連研究および本研究との相違

これまでにいくつかの，RDF データベースが提案されている [1–3, 6–9]．これらは内部で関係データベース，あるいは Berkeley DB を利用している．関係データベースを利用した RDF データの格納方法には，文を単一関係表に平坦に列挙する手法と，定義

表 1: RDF データベースの格納手法
Tab.1 Storage methods of RDF databases.

	格納手法	database
RDFSuite [1]	flat / schema	relational
Redland [2]	hash	Berkeley
Sesame [3]	schema	relational
Jena2 [6]	flat	relational
Inkling [7]	flat	relational
RDFStore [8]	hash	Berkeley
rdfDB [9]	hash	Berkeley

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の 2 種類がある．一方，Berkeley DB を利用した手法では，3 種類のハッシュ表を生成し，主語，述語，目的語をそれぞれキーとして用い，値には基本的に文を用いる．

表 1 に従来の代表的な RDF データベースとその格納手法を示す．本稿では，関係データベースを用いて文を平坦に単一表へ格納する手法のことを flat アプローチ，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．

これまでの手法の問題点を次にまとめる．1) flat アプローチと hash アプローチでは，スキーマ情報とインスタンス情報を同一の RDF グラフとして扱っているため，一部のスキーマ情報に基づく問合せを行うことができない．2) schema アプローチではスキーマ情報に基づいて関係スキーマを決定しているため，スキーマ情報の変更に伴って関係スキーマの変更が必要になる．3) 従来の手法はこれまでの手法は文単位で格納しているため，単一の文を検索する処理は効率的であるが，連続する複数の文を検索する処理は，結合演算の回数が増加してしまうために非効率的である．本稿では，このような連続する複数の文を検索する問合せを経路式に基づく問合せと呼ぶ．

4. 提案手法

本章では，関係データベースを用いた，RDF データの経路式に基づく格納と検索に関して述べる．3 章であげた従来の RDF データベースの問題を解決するために，我々はまず RDF グラフを五つの部分グラフに分割し，それぞれに適切な手法を採用して関係表に格納する．部分グラフを関係表に格納する際，任意の 2 要素間の接続関係の有無を知る検索が頻繁に行われる部分グラフでは，ナンバリングスキームを採用してそれぞれの関係表に格納し，経路式に基づいた問合せが頻繁に行われる部分グラフでは，経路式に基づいて関係表に格納することで，結合演算の回数を減少させることができる．

4.1 部分グラフの抽出

RDF データは，RDF メタスキーマデータと RDF スキーマデータ，および，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．これまでの手法では，一つの RDF グラフで扱っていたが，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．これによって，スキーマ情報に基づいた問合せを実現でき，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される．また，それぞれで異なる手法を採用して関係表に格納することが可能になり，各部分グラフに適した手法を採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．

- **CI (Class Inheritance) 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．クラスが頂点で，有向辺は *rdfs:subClassOf* のみの非巡回有向グラフ構造である．
- **PI (Property Inheritance) 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．プロパティが頂点で，有向辺は *rdfs:subPropertyOf* のみの非巡回有向グラフ構造である．
- **T (Type) 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．この文は，主語がインスタンスとなる資源，目的語がクラス，述語は *rdf:type* であるため，深さ 1 の非巡回有向グラフ構造となる．
- **DR (Domain Range) 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．このグラフを構成する文は，主語がプロパティで目的語がクラス，述語が *rdfs:domain* あるいは *rdfs:range* であるため，深さ 1 の非巡回有向グラフとなる．
- **G (Generic) 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，残りのすべての述語で構成される．ユーザが定義した述語や他の述語（*rdfs:label* や *rdfs:isDefinedBy* など）を含む．本来は，巡回を含む有向グラフ構造であるが，本稿では，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[1]．

我々が提案する関係スキーマは，これらの部分グラフをそれぞれ別々の関係表に格納することを基本方針とし，それぞれの部分グラフに適した手法を採用して関係表に格納する．CI グラフと PI 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，G 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，4.2 節で述べる経路式によって表現する．最後に関係スキーマの詳細は 4.4 節で述べる．

4.2 経路式

本節では，G グラフを関係表に格納する際に用いる経路式に関して述べる．経路式による問合せは，有向辺が連続するような経路式で行われることが一般的であるため，有向辺のみによって構成される経路を以下のように定義する．

定義 1 (有向辺経路) 非巡回有向グラフにおいて，頂点 v_m から頂点 v_n へ向かうラベルが a である有向辺を $a(v_m, v_n)$ と表す．始点 v_0 から終点 v_k までの有向辺経路とは，有向辺の有限列

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

と表す．有向辺経路式は，有向辺のラベルの列として次のように表現する．

$$a_0, a_1, \ldots, a_{k-2}, a_{k-1} \qquad \square$$

本稿では，非巡回有向グラフに対して，入次数が 0 である頂点を始点とした有向辺経路を絶対有向辺経路と呼ぶ．提案手法では，すべての頂点までの絶対有向辺経路式を抽出し，それらを関係表に格納する．つまり，ある頂点までの絶対有向辺経路が複数存在する場合は，複数行にわたって格納する．

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 [5] を用いて非巡回有向グラフ構造である CI グラフと PI グラフを表現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マとは，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ノード番号を各頂点に割り振る手法である．本手法を用い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．

ナンバリングスキームの例を図 2 に示す．まず，入次数が 0 であるすべての頂点（図 2 では A と D）の先祖となる仮想的なルート頂点（図 2 では O）を生成する．その後，多重辺を含まないよう木に展開する（図 2 右）．最後に，生成された木に対して，前置順，後置順，根からの深さの三つのノード番号を各頂点に割り振る．このとき，木に展開する以前は一つであった頂点 (E) に対しては，複数の異なるノード番号が割り振られる．

インターバルナンバリングスキームを用いて割り振られたノード番号と頂点間の継承関係には，次に示す包含定理が成り立つ．

[1]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．

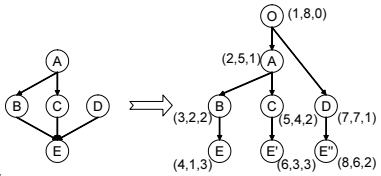

図 2: インターバルナンバリングスキームの例
Fig.2 An example of the interval numbering scheme for DAG.

定理 1 (非巡回有向グラフにおける包含定理) 非巡回有向グラフの頂点 v と u において，$(pre(v_i), post(v_i), depth(v_i))$, $i = \{1, 2, \ldots, m\}$ および $(pre(u_j), post(u_j), depth(u_j))$, $j = \{1, 2, \ldots, n\}$ は，それぞれ v と u に割り振られたノード番号であるとする．v が u の先祖であるならば，次の条件を満たす (i, j) が少なくとも一つ存在する．

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

また，v が u の直接の親であれば $|depth(u_j) - depth(v_i)| = 1$ が成り立つ． □

図 2 の 2 頂点 A，E 間および A，D 間の継承関係を，定理 1 を用いて調べる．A の ノード番号 (2, 5, 1) と E の 1 番目のノード番号 (4, 1, 3) で比較すると，$2 < 4$ かつ $5 > 1$ が成り立つため，A は E の 2 世代前の先祖である．次に，A の ノード番号 (2, 5, 1) と D のノード番号 (7, 7, 1) で比較すると，$2 < 7$ かつ $5 > 7$ が成立しないため，A は D の先祖ではない．

4.4 提案スキーマ

図 3 に 図 1 で示した RDF の例の一部を格納した図を示す[2]．property 表や class 表はプロパティやクラスを格納してスキーマ情報を表現する．property 表の *domain* と *range* は DR グラフを表現するために用い，property 表と class 表 の *pre* や *post*, *depth* に，4.3 節で述べたナンバリングスキームを用いてノード番号を割り振ることで，PI グラフと CI 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，複数の異なるノード番号を持つため，複数行にわたって情報を格納する．resource 表と path 表，triple 表を用いて G グラフを格納してインスタンス情報を表現する．resource 表には，G グラフの頂点の情報を格納し，path 表には，G グラフの絶対有向辺経路式を格納する．resource 表の *pathID* は path 表の *pathID* を参照することで，その資源への絶対有向辺経路を表現する．このとき，ある資源への絶対有向辺経路式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．さらに，triple 表に G グラフ内のすべての文を格納する．また，type 表に T グラフの情報を格納することで，インスタンス情報とスキーマ情報の関連を表現する．

4.5 問合せ

RDF データベースは，従来の文に基づいた問合せのみではなく，RDF スキーマ情報に基づいた問合せや経路式に基づいた問合せを効率的に処理できる必要があると考えている．しかしながら，RDF データに対する問合せ言語の標準が未だ存在しないため，RDF 問合せ言語から SQL への変換の詳細を考察することができない．以下に，経路式に基づいた問合せとスキーマ情報に基づく問合せを示す．

例 1: ある人に描かれた (paints) あるもののタイトル (title) を検索

```
SELECT r.resourceName
FROM   path AS p, resource AS r
WHERE  p.pathID = r.pathID
AND    p.pathexp = 'paints>title'
```

例 2: 資源 `http://www.w3.org/2000/01/rdf-schema#Resource` の直接の親クラスを検索

[2]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．

表 2: RDF データの資源数と文数
Tab.2 The number of resources and statements of RDF data.

	1 MB	5 MB	10 MB
class	14	14	14
property	26	26	26
type	769	5,213	10,887
resource	251,136	4,869,497	9,304,132
path	113	165	8,526
triple	15,553	74,654	135,945

```
SELECT c1.className
FROM   class AS c, class AS c1
WHERE  c.pre < c1.pre
AND    c.post > c1.post
AND    c.depth = c1.depth - 1
AND    c.className =
         'http://www.w3.org/2000/01/rdf-schema#Resource'
```

5. 性能評価

5.1 実験環境

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 [6] の処理時間を比較する．実験には，Gene Ontology [4] を実験データとして採用した．実験では，1 MB, 5 MB および 10 MB の異なるデータサイズの RDF 文書を生成して，それぞれのサイズで実験する．表 2 にそれぞれの 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．

実験に用いる問合せは，スキーマ情報に基づく問合せと経路式に基づく問合せの 2 種類の問合せを用いた．前者には，次の三つに分類できる．1) 先祖子孫関係に関する問合せ，2) 範囲定義域に関する問合せ，3) 資源のタイプに関する問合せである．後者は，経路式の長さを 1 から 9 へと変更して，それぞれの処理時間を測定した．

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 を搭載した計算機を用いた．

5.2 実験結果

図 4 は，スキーマ情報に基づく問合せの実験結果を示している．図の横軸は問合せの種類を示し，縦軸は処理時間（ミリ秒）を対数目盛で表している．“class1” と “property1” に関しては，Jena2 では処理できない問合せであるため，空白になっている．図から “class2” や “type1” のとき特に我々の手法の方が勝っていることが確認できる．

表 3 は，経路式に基づく問合せの実験結果である．この結果から我々の手法の方が Jena2 よりも効率的であることが確認できる．また，経路式の長さが 1 から 6 までは処理時間が増加し，その後は減少していることがわかる．この理由は経路式の長さが増加するにつれて，解集合が減少するためである．この結果から，我々の手法が Jena2 よりもおよそ 2 倍から 13 倍早いことが確認できる．また，データサイズの増加するにつれて比率の差が開いていることも確認できる．

これらの実験により，我々の手法がスキーマ情報に基づく問合せや経路式に基づく問合せにおいて効率的であることを確認した,.

6. おわりに

本稿では，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．我々の手法では， 5 種類の部分グラフに分割して，それらの部分グラフを関係表に格納するために，経路式に基づく格納手法やインターバルナンバリングスキームといった，それぞれの部分グラフを表現するのに適切な手法を採用した．本手法によって，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．さらに，flat アプローチや hash アプローチでは実現できなかった RDF スキーマに基づく問合せを処

class

className	*pre*	*post*	*depth*
'Literal'	2	1	1
'Artist'	3	3	1
'Artifact'	4	2	1
'Painter'	5	5	2
'Painting'	6	4	2
'Painting'	8	6	2
'CommercialGoods'	7	7	1

property

propertyName	*domain*	*range*	*pre*	*post*	*depth*
'creates'	'Artist'	'Artifact'	3	3	1
'paints'	'Painter'	'Painting'	4	2	2
'name'	'Artist'	'Literal'	2	1	1
'title'	'Artifact'	'Literal'	5	4	1

type

resourceName	*className*
'r1'	'Painter'
'r2'	'Painting'

resource

resourceName	*pathID*	*datatype*
'r1'	1	0
'r2'	2	0
'Guernica'	3	1
'Picasso'	4	1

path

pathID	*pathexp*
1	”
2	'paints'
3	'paints.title'
4	'name'

triple

subject	*predicate*	*object*
'r1'	'name'	'Guernica'
'r1'	'paints'	'r2'
'r2'	'title'	'Picasso'

図 3: RDF データを格納した例
Fig.3 A strage example of RDF data.

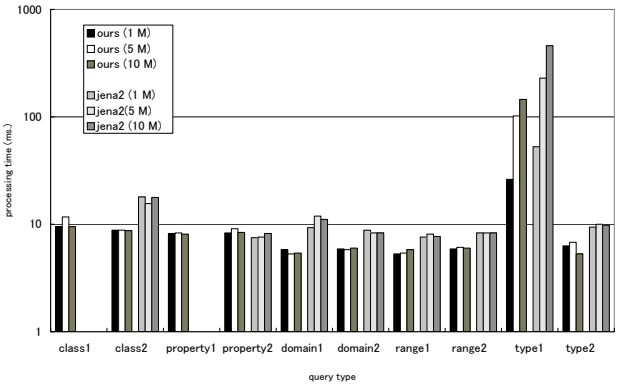

図 4: スキーマ情報に基づく問合せの処理時間（ミリ秒）
Fig.4 The processing times of the schema-based queries (ms.)

表 3: 経路式に基づく問合せの処理時間（ミリ秒）
Tab.3 The processing times of path-based queries (ms.)

	Proposed approach			Jena2		
length	1 MB	5 MB	10 MB	1 MB	5 MB	10 MB
1	53.3	98.3	117.0	98.5	364.3	623.2
2	26.6	96.0	123.1	137.1	777.0	1576.1
3	221.1	494.3	799.0	911.9	3686.8	7117.2
4	193.9	880.0	1684.8	985.1	8228.5	17261.7
5	151.6	3302.3	4289.4	794.3	23424.5	36452.1
6	103.2	5380.3	7643.1	590.5	35645.4	57646.5
7	61.8	1870.7	4062.8	330.2	13374.7	32703.8
8	19.5	623.7	1734.8	153.9	6004.5	16117.9
9	10.0	677.8	1222.7	134.6	5650.8	11742.1

理することができた．今後の課題として，問合せ言語の検討などが挙げられる．

[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] 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)*, May 1 2000.

[2] D. J. Beckett. The design and implementation of the redland RDF application framework. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, pp. 449–456, 2001.

[3] 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.

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

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

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

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

[8] A. Reggiori. RDFStore: Perl API for RDF Storage. `http://rdfstore.sourceforge.net/`.

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

[10] World Wide Web Consortium. RDF Vocabulary Description Language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema/`. W3C Recommendation 10 February 2004.

[11] World Wide Web Consortium. Resource Description Framework (RDF). `http://www.w3.org/RDF/`. W3C Recommendation 10 February 2004.

的野 晃整 Akiyoshi MATONO
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在学中. RDF の研究に従事. 情報処理学会学生会員. 日本データベース学会学生会員.

天笠 俊之 Toshiyuki AMAGASA
奈良先端科学技術大学院大学情報科学研究科助手. データベースシステムの研究に従事. 情報処理学会正会員. 電子情報通信学会正会員, 日本データベース学会正会員.

吉川 正俊 Masatoshi YOSHIKAWA
名古屋大学情報連携基盤センター教授. データベースシステムの研究に従事. 情報処理学会正会員. 電子情報通信学会正会員, 日本データベース学会理事.

植村 俊亮 Shunsuke UEMURA
奈良先端科学技術大学院大学情報科学研究科教授. データベースシステムの研究に従事. 情報処理学会フェロー. 電子情報通信学会フェロー, IEEE Fellow, 日本データベース学会正会員. 著書に「データベースシステムの基礎」（オーム社）など.