D-037

経路式に基づくRDFデータのための索引手法とその構築法

An Indexing Scheme for RDF Data Based on Path Expressions and its Construction.

的野 晃整 † 天笠 俊之 † 吉川 正俊 ‡ 植村 俊亮 †
Akiyoshi Matono Toshiyuki Amagasa Masatoshi Yoshikawa Shunsuke Uemura

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Resource Description Framework (RDF) [3] 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，RDFが普及することで，メタデータの質が向上するとともに，RDFデータが増加することが予想される．このため，大量のRDFデータに対する問合せの高速な処理が重要になる．

大量のRDFデータの問合せを高速に処理することを目的とした研究がこれまで行われてきた．主に，RDFデータベース [2] や索引 [1] を用いる手法がある．RDFデータベースの多くは，関係データベースかBerkeley DBを用いて実現しており，RDFデータを文に細かく分解し，それ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，つまり経路式に基づく問合せでは，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．

本稿では，根や葉ではなく中間頂点から経路を辿って到達できる中間頂点の検索の高速化を目的とする．また，提案する索引の構築手法についても議論する．

2. RDFの概要

RDF [3] 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データが表現するグラフ（RDFグラフ）は，頂点の集合によって表される．葉を除く頂点，つまり中間頂点（資源）は，一意に識別できるURIをもち，葉頂点（リテラル）は値として文字列を持つ．各頂点は，ラベルつき有向辺（述語）によって接続されており，そのラベルはURIによって表されるが，一意に識別されない．図1にRDFグラフの例を示す．本稿では，簡単のために葉頂点と中間頂点を区別しない．また，対象とするRDFグラフは閉路を含まない非巡回グラフとする．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．

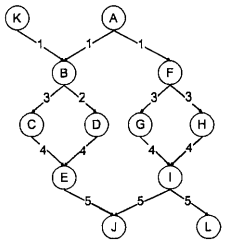

図1: RDFグラフの例

3. 経路式に基づく索引手法

本節では，RDFデータを効率的に検索することができる索引手法を提案する．提案索引を用いることで，対象のデータベースを特定することなく高速な検索を提供することができる．つまり，文に基づいてRDFデータを格納するアプローチのRDFデータベースであっても提案手法を適用することができる．

3.1 経路式

RDFグラフに対する問合せは，与えられた部分グラフと一致する部分グラフを発見する問合せや，与えられた順路をたどって到達できる要素集合を求める問合せが一般的である．これらの問合せで用いる検索のキーは経路式である．そのため，経路式に基づく問合せを効率的に処理できることがRDFデータ検索の効率化に直接的につながる．

RDFグラフは頂点だけでなく，有向辺にもラベルを持つ有向グラフ構造であるため，その問合せは，有向辺を列挙した問合せが頻出すると考えられる．そのため，われわれが提案する経路式に基づく索引手法は，有向辺のみを列挙した**有向辺経路式**を格納する手法である．以下で有向辺経路を定義する．

定義1 (有向辺経路) 非巡回有向グラフ g における頂点集合を $V(g)$，有向辺集合を $E(g)$ とする．$(v_m, v_n) \in E(g)$ は，頂点 $v_m \in V(g)$ から頂点 $v_n \in V(g)$ へ向かう有向辺であるとする．始点 v_0 から終点 v_k へ向かう有向辺経路は有向辺の有限列で表される．

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

このとき，(v_0, v_1) を始辺，(v_{k-1}, v_k) を終辺と呼ぶ．

有向辺経路の経路式は，有向辺のラベルの列で表す．

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

このとき l は，有向辺をラベルへ変換する関数である．

†奈良先端科学技術大学院大学 情報科学研究科
‡名古屋大学 情報連携基盤センター

表 1: 提案する索引の構造

経路式	始辺	終辺	(head-tail)
1	1	1	(A-B),(A-F),(K-B)
2	2	2	(B-D)
3	3	3	(B-C),(F-G),(F-H)
4	4	4	(C-E),(D-E),(G-I),(H-I)
5	5	5	(E-J),(I-J),(I-L)
1/2	1	2	(A-D),(K-D)
1/2/4	1	4	(A-E),(K-E)
1/2/4/5	1	5	(A-J),(K-J)
1/3	1	3	(A-C),(A-G),(A-H),(K-C)
1/3/4	1	4	(A-E),(A-I),(K-E)
1/3/4/5	1	5	(A-J),(A-L),(K-J)
2/4	2	4	(B-E)
2/4/5	2	5	(B-J)
3/4	3	4	(B-E),(F-I)
3/4/5	3	5	(B-J),(F-J),(F-L)
4/5	4	5	(C-J),(D-J),(G-J),(H-J),(G-L),(H-L)

3.2 索引の構造

提案する索引手法は，すべての有向辺経路式を列挙し，各経路の始辺と終辺の組と，各経路の始点と終点の組 (*head-tail*) の集合を保存する手法である．図 1 の例を提案する索引手法で表現した例を表 1 に示す．提案手法では，有向辺経路式と始辺，終辺をキーとした B 木によって索引を構築しており，さらに，各行の head-tail に対しても，始点と終点をそれぞれ，B 木を用いて格納する．つまり，B 木の値に B 木を持つ構造である．

3.3 索引を用いた検索手法

本節では，本索引を用いた検索手法を述べる．本手法を用いることで，葉や根ではなく中間頂点から中間頂点までの経路式に基づく検索を効率的に実現することができる．

例を挙げて述べる．図 1 において，頂点 F から経路 3/4 を辿って到達可能な頂点集合を求める方法について考える．まず，経路式 3/4 を B 木を用いて索引情報から検索し，その head-tail 中に F を始点とする組を B 木によって発見し，その終点の集合 $\{I\}$ が解となる．

すべての問合せを索引のみを用いて検索するわけではなく，データベースに格納されているデータと併用して検索する場合もある．たとえば，頂点 B から頂点 E までの有向辺経路式をすべて列挙する問合せについて考える．データベースの構造は，単一の関係表に主語，述語，目的語の組を格納していると仮定する．まず，関係表から主語に B を持つ述語 $\{2,3\}$ を検索し，目的語に E を持つ述語 $\{4\}$ を検索する．それぞれが始辺と終辺であるような経路式 $\{2/4, 3/4\}$ を検索し，それらの head-tail に (B,E) を含むかどうかを確認する．

4. 索引の構築

本節では，RDF データの更新，主に追加について議論する．RDF データは，文に分解することができることから，追加処理では一つの文が追加されることを前提とする．

Algorithm 1 に RDF の文が追加されたときのアルゴリズムを示す．1 行目では，文の集合 T に，文 *stmt* を追加し，2 行目の関数は，*stmt* の主語と一致する資源を目的語として持つ文を B 木によって検索し，その述語の集合 *Preds* を返す．4 行目の関数は，末尾に p を持つ経路式を B 木を用いて検索し，その経路式と head-tail の集合

Algorithm 1: `AddStatement(`$T, I,$ *stmt*`)`

Data: T は文の集合, I は経路と head-tail の集合の組の集合, *stmt* は追加する文
Result: *stmt* 追加後の文の集合 T', *stmt* 追加後の索引 I'

```
1  T ← T ∪ {stmt}
2  Preds ← getPredicatesMatchedObj(stmt.subj)
3  for p ∈ Preds do
4      PathHTs ← getPathHTsMatchedPathend(p)
5  PathHTs ← PathHTs∪
   {getPathHTsMatchedPath(stmt.pred)}
6  for pht ∈ PathHTs do
7      oldPath ← pht.path
8      OldHTs ← pht.HTs
9      Heads ← getHeadsMatchedTail(OldHTs, stme.subj)
10     if Heads ≠ ∅ then
11         newPath ← appendPath(oldPath, stmt.pred)
12         NewHTs ← getHTsMatchedPath(newPath)
13         if NewHTs = ∅ then
14             I ← I ∪ {(newPath, NewHTs)}
15         for head ∈ Heads do
16             NewHTs ← NewHTs ∪ {(head, stmt.obj)}
```

の組 (*path*, *HTs*) を返す．5 行目の関数は，追加される文の述語である経路式を B 木を用いて検索し，その経路式と head-tail の集合の組 (*path*, *HTs*) を返す．9 行目の関数は，*OldHTs* 中から，追加した文の主語 *subj* と一致する終点を持つ head-tail を B 木を用いて検索し，その始点集合を返す関数である．11 行目では，取得した各経路 *oldPath* の末尾に，追加する文の述語を付加し，新たな経路式 *newPath* を生成している．12-14 行目において，`getHTsMatchedPath` 関数は I から経路式 *newPath* と一致する経路式の head-tail を B 木によって検索し，その集合 *NewHTs* を返す関数で，もし *newPath* を一致する経路が存在しなければ，新たに空の *NewHTs* を作成する．16 行目では，*Heads* の各要素 *head* と追加する文の目的語の組を *NewHTs* に追加している．なお，このアルゴリズムでの計算量は $O(M \log N)$ となる．

5. まとめ

本稿では，追加の更新を考慮した RDF のための経路式に基づく索引を提案した．今後の課題としては，実験による評価を行う必要がある．

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 15017243，16016243），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] 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.

[2] A. Magkanaraki, G. Karvounarakis, T. T. Anh, V. Christophides, and D. Plexousakis. ONTOROGY STORAGE AND QUERYING. Technical report, Computer Science Informatino Systemns Laboratory, 2002.

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