アクセス頻度を考慮した XML 文書分割方式の提案

A Scheme for Partitioning XML Documents Based on Access Frequency

中尾 伸章♡ 天笠 俊之♢
的野 晃整♠ 植村 俊亮♣

Nobuaki NAKAO Toshiyuki AMAGASA
Akiyoshi MATONO Shunsuke UEMURA

本論文では，アクセスの偏りを考慮して XML 文書を分割する手法および分割された XML データにおける問合せ処理手法を提案する．本手法では，既知の問合せセットとその発行頻度を元にコスト関数を定め，その値によって分割方法を決定する．これにより，問合せ処理を効率化するような分割を行うことができる．また，Strong DataGuide に基づく構造概要を利用して，分割後の XML 文書への問合せ変換を実現している．

In this paper, we propose schemes for partitioning XML data by taking data access locality into account, and querying partitioned XML data. An XML data is partitioned by a cost function, that is defined by a given set of frequent queries and their frequencies, so that the queries can be processed efficiently. At query processing, a query is translated for the partitioned XML data by using a structural summary of the XML data based on the Strong DataGuide.

1. はじめに

近年，XML (Extensible Markup Language) [4] の普及とともに，様々な種類の XML 文書が生成されている．数百 MB から数 GB のサイズを持つ大規模な XML 文書はその一例であり，観測データ，サーバのアクセスログ，Web ディレクトリ，オントロジなどの記述などの応用例がある．このことから，規模の大きな XML 文書を高速に処理することは今後重要になると考えられる．

XML 文書の処理では，DOM (Document Object Model) や SAX (Simple API for XML) などの API が主として使用される．DOM は XML 文書を主記憶上にオブジェクトとして展開した上で処理を行う．SAX は，XML 文書を先頭から最後までパースしながら要素の開始や終了などのイベントを発生させ，アプリケーションがイベントを受け取った時に処理を行う．また，XML データベースを用いた処理においてもデータサイズが性能に与える影響は大きい．特に，XML に固有な構造結合処理では，データサイズとノード数の増加により計算コストが大きく増加することが知られている．以上のことから，XML 文書全体のデータサイズは，処理コストに大きな影響をおよぼすことが分かる．

この問題の解決手段として，本論文ではアクセスの局所性に着目する．一般に，XML 文書の問合せ処理に必要な部分は必ずしも文書全体ではなく，局所的であることが多い．また，問合せが発行される頻度も異なることから，XML 文書内のアクセスはより偏っていると考えられる．そこで本論文では，XML 文書内のアクセスの偏りを考慮し，問合せ処理が効率化されるように XML 文書を分割する手法を提案する．本手法では，XML 文書への典型的な問合せセットとそれらの発行頻度が既知であることを仮定し，それらを元にアクセスの偏りと処理コストを考慮し，XML 文書を任意の数へと分割する．まず各問合せに必要な部分を断片の候補とし，それらを任意の数にまで併合することで，分割後の断片を規定する．併合の判断には問合せの処理コストと，問合せの発行頻度に基づいて算出するコスト関数を用いる．コスト関数が低くなる断片の候補の組合せを，欲張り法を用いて決定する．これにより，問合せ時に必要なデータが記述された断片のみを処理することができるため，処理を効率化することができる．また，Strong DataGuide [2] に基づく構造の概要を生成し，断片を規定する問合せ式の決定や，分割後の問合せ処理に利用する．問合せ時には，構造概要を元に問合せを解析，変換し，必要なノードが存在する断片のみに問合せを発行することで処理を効率化できる．

2. 関連研究

まず，XML 文書の分割に関する研究として Bremer 等の研究 [1] について述べる． [1] は，XML 文書を分散処理するための研究であり，Repository Guide と呼ばれる索引付けされた構造概要を利用している．まず XML 文書の水平・垂直分割を XPath 式によって定義し，分割された文書を Repository Guide によって管理する．Repository Guide は三種類の索引に関連付けられ，分散環境での問合せ処理を効率化している．しかし，この論文は分散環境下での処理に注目しており，分割方法や問合せの変換，アクセスの偏りを考慮していない点が本研究と異なる．

次に，問合せに必要な部分の局所性に注目した研究について述べる．中島等は，処理に必要な部分のみを主記憶上に展開して処理できる SPlitDOM [9] を提案している． Marian 等は，XQuery [5] 問合せを静的に解析し，XML 文書内の必要な部分のみを射影して処理する手法を提案している [3]．横山等は，発生イベントとそれに付随するデータの保存先をアクセス頻度に応じて切り分けることで SAX における処理時間を短縮する手法を提案している [8]．また，我々は関係データベースに格納された大規模 XML データの処理において，問合せ処理に必要な部分のみを動的にマッピングして使用することで高速化する手法を提案している [7]．これらの研究では，問合せの発行頻度によって生じるアクセスの偏りについて議論していない点で本研究とは異なる．

3. 提案手法

3.1 XML 文書の分割及び構造概要の定義

まず，問合せ式に基づく XML 文書の垂直分割 (*vertical partitioning*) を定義する [1]．本論文では簡単のため “/” 及び “//” 軸とノードテストからなり，述語を含まない XPath のサブセットを問合せ式として用いる．以下では XPath 式はこの XPath のサブセットを指すものとする．

【定義 (1)】 XML 文書の垂直分割
垂直分割は f は $f = sf - \{ef_1, ..., ef_m\}$ で定義される．ここで，sf は選択される断片，ef は除かれる断片を示す．ただし ef は sf からの相対パスである．また，$\{ef_1, ..., ef_n\}$ は無くてもよい．

♡ 正会員 奈良先端科学技術大学院大学情報科学研究科．現在，（株）NTT ドコモ関西．
♢ 正会員 奈良先端科学技術大学院大学情報科学研究科．現在，筑波大学大学院システム情報工学研究科．
amagasa@cs.tsukuba.ac.jp
♠ 正会員 奈良先端科学技術大学院大学情報科学研究科．現在，独立行政法人産業技術総合研究所グリッド研究センター．
a.matono@aist.go.jp
♣ 正会員 奈良先端科学技術大学院大学情報科学研究科．
uemura@is.naist.jp

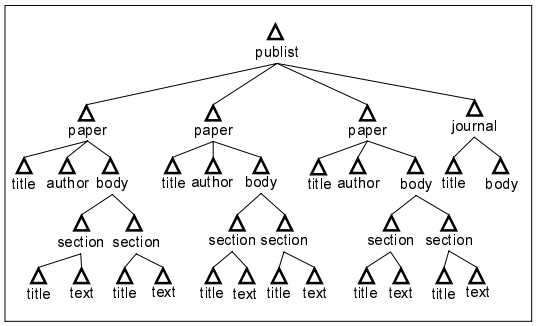

図 1: XML 文書の例.
Fig. 1: An example of XML document.

表 1: 頻繁に発行される問合せとその頻度.
Table 1: Frequent queries and their probabilities.

q	問合せ式	発行頻度
q_1	//title	0.5
q_2	//section	0.3
q_3	//author	0.1

[1] では XML 文書の水平分割 (*horizontal partitioning*) も定義しており，それは垂直分割で規定した断片 f において，sf, ef を表現する問合せ式が述語を含む時，f を水平分割としている．本論文では水平分割は考えない．

次に，本手法で用いる XML 文書の構造概要を示す．XML を含む半構造のための構造概要はこれまで多く提案されているが，ここでは最も単純な Strong DataGuide [2] を用いる．

【定義 (2)】 Strong DataGuide
情報源 s の DataGuide d とは，1) 全てのラベル経路が d において一つのデータ経路インスタンスを持ち，2) d の全てのラベル経路は s のラベル経路になっているものをいう．ここで，ノード o のラベル経路とは，o から辿ることのできるエッジ $(e_1, e_2, ..., e_n)$ のラベル系列 $(l_1, l_2, ..., l_n)$ である．また，ノード o のデータ経路とは，o から辿ることのできるノードとラベルの系列 $l_1, o_1, l_2, o_2, ..., l_n, o_n$ である．Strong DataGuide sd とは，このような d のうち，情報源における全ての経路式を任意の経路式のターゲット集合という視点から，s と d が区別できないものをいう．

3. 2 分割処理

本論文では，XML 文書に対する問合せセットとそれらの発行頻度が既知であることを仮定する．発行頻度は問合せの発行履歴の統計から算出した確率を元に，上位 n 件の問合せを典型的な問合せセットとし，その確率を用いる．図 1 の XML 文書に対する典型的な問合せセットとそれらの発行頻度の例を表 1 に示す．

問合せセットが既知である場合，あらかじめそれぞれの問合せ式を元に定義 (3) による分割を行い，断片を規定しておけば，問合せ処理時にはそれを返すだけで済むので効率的である．しかし，問合せセットは一般に多数あるので，そのまま分割を行なうと，断片の数が余りにも多くなってしまう．また，後で述べる重複部分の問題も生じるので，一般にはうまく行かない．そこで本論文では任意の n 個の断片へと分割することを考える．ここで n は，想定する環境によって自由に決めることができる．例えば，分散環境ではプロセッサの数に基づいて決めれば良いし，分散ディスクの場合はスピンドルの数などである．

以上より，問題は XML 文書を m 個の問合せ式に基づいて n 個の断片に分ける問題となる．ここで一般に $m > n$ である．提案手法では，まず与えられた m 個の問合せ式で規定される断片を候補の断片としておき，その m 個の断片のいくつかを併合することで n 個にまで減らすことを考える．併合の決定は，問合せ処理のコストに問合せの発行頻度を考慮したコスト関数を用いる．これにより，問合せ処理を効率化する分割を行うことができる．

3. 2. 1 重複したノードの取り扱い

一般に問合せ式に対応するノード集合は独立ではなく互いに重複部分を持つことがある．例として，図 1 において問合せ q_1 = //title と q_2 = //section を考える．この場合，title 要素のいくつかが section 要素に含まれている．このように重複部分を持つ複数の問合せ式あら断片を規定する場合，重複部分を複製し各断片に重複部分を持たせる方針と重複部分を複製せずに扱う方針が考えられる．重複部分の複製を行う方針は，問合せの処理速度の面から有利であるが，余分な記憶容量を必要とすることや更新時の処理が複雑になる．以上の理由から本論文では，重複部分の複製は考えないこととする．重複部分の複製を行わない方針はさらに二つのケースに分けることができ，重複部分を新たな断片として扱うケースと重複部分の記述先を一意に決定するケースが考えられる．本論文では後者を選択し，重複部分は発行頻度が最も高い問合せ式に基づく断片が持つこととする．

以上により，問合せ式の数だけ候補となる断片を規定することができる．表 1 の問合せセットを用いて 図 1 の XML 文書を分割する場合，候補となる断片は表 2 のようになる．ただし，f_{remain} は，全ての問合せ式から指定されないノード集合からなる断片を示すものとする．

3. 2. 2 断片の併合

断片の候補を併合することによって n 個の断片を決定する．断片の併合を考慮した XML 文書の垂直分割を，定義 (1) の拡張によって次のように定義する．

【定義 (1)'】 併合を考慮した XML 文書の垂直分割
断片の併合によって規定される断片 f は $f = \{sf_1, ..., sf_n\} - \{ef_1, ..., ef_m\}$ の式で定義される．sf は選択される断片を指定する絶対 XPath 式，ef は除かれる断片を指定する絶対 XPath 式である．ただし，ef は sf の問合せ式のいずれかを接頭辞として持つものとする．また，$\{ef_1, ..., ef_n\}$ は無くてもよい．

本手法では，問合せセットで問合せ処理を行った場合の処理コストと，各問合せの発行頻度からコスト関数を定め，その値から併合の組合せを決定する．コスト関数が最小となる組合せで併合を行うことで，問合せ処理を最も効率化するように併合することができる．問合せセット全体を処理するコスト関数 C_{total} は次式で与えられる．

$$C_{total} = \sum_{k=1}^{m} freq(q_k) \cdot C(q_k)$$

ここで $freq(q_k)$ は q_k の発行頻度，$C(q_k)$ は q_k の評価にかかる処理時間コストである．

$C(q_k)$ は問合せに必要な全ての断片に問合せを発行し，評価するコスト C_{eval} と，必要に応じて結果を統合するためのコスト $C_{integration}$ の和で次のように与えられる．

$$C(q_k) = C_{eval}(q_k) + C_{integration}(q_k)$$

$C_{eval}(q_k)$ は 必要なノードが含まれる断片全てに対して q_k を評価するためのコストである．たとえば，断片 f_a 内に q_k の評価に必要な全てのノードがあれば，評価する断片は f_a のみでよい．しかし，そうでない場合，f_a だけでなく重複部分を持つ他の断片も評価する必要がある．f_a 以外で q_k にどの断片が必要になるかは，問合せセットから規定される断片の情報（表 2）から判断することができる．また，$C_{integration}(q_k)$ は，複数の断片から得た結果を結合やソートによって統合するためのコストである．C_{eval} 及び $C_{integration}$ がノード数等，何に依存し，さらにどのような計算量で変化するかは処理方法やアルゴリズムによって異なる．よって，処理方法やアルゴリズ

表 2: 問合せ式から規定した断片とその情報.
Table 2: Fragments derived from a query set and their descriptions.

断片	断片を規定する式	ノード総数	処理対象とする問合せ
f_1	//title	10	q_1, q_2
f_2	//section – {./title}	12	q_2
f_3	//author	3	q_3
f_{remain}	/publist – {./paper/title, ./journal/title, ./paper/author, ./paper/body/section}	9	-

表 3: 問合せセットに基づく断片の候補.
Table 3: Candidate fragments derived from the query set.

断片の候補	断片を規定する式	イベント数	処理対象とする問合せ
f_1	//date	3033	q_1, q_4, q_6
f_2	//interval/start	360	q_2, q_3
f_3	//interval – {./start}	962	q_3
f_4	//closed_auction – {./date}	7583	q_4
f_5	//person/name	766	q_5
f_6	/site/region/asia – {./item/mailbox/mail/date}	1597	q_6
f_7	/site/catgraph	30	q_7
f_8	/site から f_1〜f_7 を除いた部分（紙面の都合上省略）	53410	-

ムに応じた情報を断片の候補を決定する際に収集しておき（表 2 のノード総数等），推測値を計算する．

断片の合計が n となるような断片の候補の組合せ毎に C_{total} を計算する．C_{total} が最小となる組合せが問合せ処理を最も効率化する組合せとなる．しかし，m 個の断片を n 個に併合するとき，その全てのコスト関数を算出する計算量は $O(n^m)$ となってしまう．よって，本手法では，組合せ最適化問題の近似解法である欲張り法 (Greedy Method) を用いることで，近似解となる組合せを決定する．具体的には，二つの断片の候補のみを併合した場合の C_{total} を部分問題とし，C_{total} が最も低くなる断片の候補同士を併合する．これを $m-n$ 回行うことで最終的な併合の組合せを決定する．

併合後，断片を規定する式に基づいて XML 文書を n 個の断片へと分割する．ただし，断片は問合せ式から規定されるため，そのままでは最上位の要素が複数存在する場合がある．XML では最上位の要素を一つに定めているため，各断片の根要素となる要素を生成しておく必要がある．

3. 2. 3 問合せ処理のための情報付加

分割時に，分割によって破断されるエッジの両端のノードと構造概要にそれぞれ情報を付加しておく．破断されるエッジの両端のノードには，二つの情報を付加する．一つはエッジの識別子であり，破断したエッジ同士を再び結合する際に用いる．本手法では，元の XML 文書における文書順をエッジの識別子として用いる．これにより，結合だけでなく，文書順でのソートが必要な場合でも利用することができる．二つ目は，構造概要上でのノード識別子である．この値は，根ノードからの経路式が異なるノードを識別する値であり，分割により構造が変化し，同名であるが根ノードからの経路が異なるノードを識別するために用いる．構造概要には，ノード毎に記述先となる断片の情報のみを付加し，問合せの解析時に用いる．

3. 3 問合せ処理

分割後の各格納先への問合せ処理は，次のように問合せの解析と変換，結果の統合から実現される．

3. 3. 1 問合せの解析と変換

分割後の問合せ処理では，構造概要を用いて問合せの解析と変換を行う．分割後，構造概要上の各ノードには 3.2.3 節で述べたように，各ノードの記述先である断片を示す値が付加されている．この値と，構造概要の持つ構造（経路）情報を用いて問合せの解析，変換を行うことができる．具体的には，問合せ式の経路と一致するノードを元に構造概要をたどり，記述先の断片と，そこに発行すべき問合せを生成する．

3. 3. 2 結果の統合

問合せの変換後，発行先が複数の断片となった場合，それぞれの断片から得た結果をソートや結合などの処理によって統合することで，最終的な問合せ結果を得ることができる．

表 4: 実験で用いた問合せセットとそれらの発行頻度.
Table 4: Query set and their frequencies used in the experiment.

問合せ	問合せ式	発行頻度
q_1	//date	0.3
q_2	//interval/start	0.2
q_3	//interval	0.15
q_4	//closed_auction	0.125
q_5	//person/name	0.1
q_6	/site/region/asia	0.05
q_7	/site/catgraph	0.025

表 5: 導出された断片.
Table 5: The resulting fragments.

断片	併合した断片の候補	イベント数	処理対象とする問合せ
f_A	f_1, f_4	10616	q_1, q_4, q_6
f_B	f_2, f_3, f_5, f_6, f_7	3715	q_2, q_3, q_5, q_6, q_7
f_C	f_8	52410	-

4. 実験

分割における断片の候補の併合と，分割前後での問合せ処理時間の比較を行った．実験環境は，Pentium 4 1.8 GHz の CPU と 1024 MB のメモリを搭載した計算機を用い，問合せ処理は SAX を用いた問合せプログラムを Java 言語により実装して用いた．なお，SAX パーサは Xerces2 Java[1] を用いた．

4. 1 用いた XML 文書と問合せセット

実験に使用したデータは XMark プロジェクト [6] で公開配布されている XML 文書生成プログラム xmlgen によって 1 MB，10 MB のデータを生成して用いた．

問合せセット及びそれらの発行頻度として，表 4 に示す問合せ式を用いた．これらの問合せ式に基づく表 3 のような 8 個の断片の候補を，3 つの断片へと併合し，分割を行った．なお，今回は SAX によって問合せ処理（結果の統合を含む）を行うことから，コスト関数の処理コストは，イベント数に基づいて計算し，C_{eval} は処理する断片をパースする上で発生するイベント数，$C_{integration}$ は問合せ結果となるデータのパースで発生するイベント数に基づいて計算した．

4. 2 実験結果

4. 2. 1 断片の併合

併合の結果決定した三つの断片は表 5 のようになった．処理コストの面から併合が起こりやすい組合せは二つ考えられる．一つは併合によって統合の必要が無くなる組合せであり，統合にかかる処理コストが減少するため併合が起こりやすくなる．表 5 では，断片の候補 f_1 と f_4，および f_2 と f_3，はこの理由によって併合が行われたと

[1] http://xml.apache.org/xerces2-j/

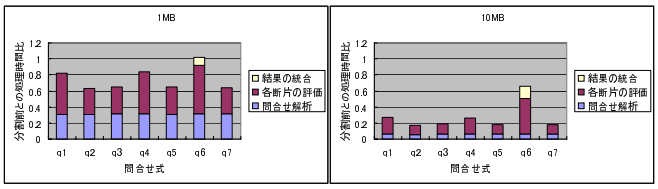

図 2: 処理時間
Fig. 2: Processing time.

考えられる．二つ目は，処理コストの比較的ない組合せであり，表 5 では，比較的処理コストの少ない f_2, f_3, f_5, f_6, f_7 がこの理由によって併合されたと考えられる．しかし，併合によって統合の必要が無くなるにもかかわらず，断片 f_1 と f_6 の併合は行われなかった．これは，発行頻度による影響が強いと考えられる．問合せ q_1 と q_6 では発行頻度の差が大きいため，併合によって q_1 にかかる処理時間のコスト増加の比重が大きくなるため併合が行われなかったと思われる．

4. 2. 2 問合せ処理時間

次に，問合せセットの 7 つの問合せにかかる問合せ処理時間を分割の前後で比較した．実験結果はデータサイズごとに 図 2 のようになった．ただし，処理時間を分割前後の比で表現しているため，グラフの高さが 1 以下のとき，分割によって処理が高速化したことを示している．なお，問合せ処理時間を問合せの解析，断片の評価，結果の統合のそれぞれにかかった処理時間によって色分けしてある．

10 MB の XML 文書の場合に処理がより高速化しているのは，パースやファイルの読み込みにかかるオーバヘッド分の処理時間の影響が少なくなることに起因すると考えられる．問合せ処理がどの程度高速化されるかは分割後の断片のデータサイズに依存し，これは問合せに必要な部分の局所性によって変化する．今回の実験では，元の XML 文書に対して f_A が約 17 %，f_B が約 6 %のサイズとなり，元の XML 文書のサイズが異なってもほぼ一定であった．このことから，元の XML 文書のサイズを大きくした場合，およそこれらの割合の程度まで高速化されると考えられる．

問合せの解析にかかる処理時間は元の XML 文書のサイズに関係無くほぼ一定であった．これは，使用したデータが同じスキーマに基づいて生成されているため，データサイズが増加しても構造概要のサイズがほぼ一定なためだと考えられる．

結果の統合にかかる処理時間は 3.2.2 節で述べたように統合の処理方法によって異なる．今回の実験 (q_6) では，問合せ結果の規模が小さいことや，統合に用いた SAX での処理の影響などの要因からそれほど統合の時間はかかっていないが，統合の処理方法によってはより処理時間がかかるケースも考えられる．それらを判断し，併合時の処理時間コストを算出することで，処理コストが高い統合を併合によって回避できる．

5. おわりに

本論文では，アクセスの偏りを考慮して XML 文書の分割し，問合せ処理を効率化する手法を提案した．問合せセットの問合せ式毎に，評価に必要な部分を断片の候補とし，それらを処理コストと発行頻度を考慮して併合した．これにより，問合せ処理を効率化するよう，XML 文書を任意の数へと分割することができる．さらに，分割後の問合せを，構造概要によって解析，変換することによって可能にした．今後の課題として，述語を含む問合せやデータ更新への対応が挙げられる，また，組合せ最適化のアルゴリズムとして欲張り法以外の手法の検討，重複部分の複製を許した場合の分割戦略等について検討する予定である．

[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 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．

[文献]

[1] J.-M. Bremer and M. Gertz. On Distributing XML Repositories. In *In 6th International Workshop on the Web and Databases WebDB2003*, pp. 73–78, 2003.

[2] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pp. 436–445. Morgan Kaufmann, 1997.

[3] A. Marian and J. Siméon. Projecting XML Documents. In *VLDB 2003: Proceedings of 29th International Conference on Very Large Data Bases, September 9–12, 2003, Berlin, Germany*, pp. 213–224, Los Altos, CA 94022, USA, 2003. Morgan Kaufmann Publishers.

[4] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Third Edition). `http://www.w3.org/TR/2004/REC-xml-20040204/`, February 2004. W3C Recommendation 04 February 2004.

[5] World Wide Web Consortium. XQuery 1.0: An XML Query Language. `http://www.w3.org/TR/xquery/`, October 2004. W3C Working Draft 29 October 2004.

[6] An XML Benchmark Project (XMark). `http://monetdb.cwi.nl/xml/index.html`, 2003.

[7] 天笠, 植村. リージョンディレクトリを用いた関係データベースによる大規模 XML データ処理. 日本データベース学会 Letters, pp. 33–36, September 2004. Vol.3, No.2.

[8] 横山, 太田, 片山, 石川. SAX-GTR: 高速 XML ストリーム読み込み手法. 研究報告 - データベースシステム, 第 2004-71 巻, July 2004.

[9] 中島, 小田切, 井谷, 吉田. XML 高速処理技術 SPlitDOM の機能拡張と Web アプリケーションへの適用評価. 電子情報通信学会第 15 回データ工学ワークショップ (DEWS2004), March 2004. I-5-5.

中尾 伸章 Nobuaki NAKAO
2005 年 3 月奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．XML データベースの研究に従事．現在，（株） NTT ドコモ関西．日本データベース学会正会員．

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． 2005 年 4 月より，筑波大学大学院システム情報工学研究科講師．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．

的野 晃整 Akiyoshi MATONO
2005 年 3 月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修了．現在，独立行政法人産業技術総合研究所グリッド研究センターポスドク研究員． Semantic Web データベースの研究に従事．

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．