Vol.2009-DBS-148 No.2
Vol.2009-FI-95 No.2
2009/7/28

ハイブリッド型 XML－関係データベースにおけるキーワード検索

張麗茹† 大森匡† 星守†

構造化/半構造データベースにおけるキーワード検索は，ユーザーがスキーマの詳細やクエリ言語を知る必要がなく情報を取り出す有効な手段である．既存の構造化データベース（伝統的な関係データベース）や XML データベースなどと異なって，ハイブリッド型 XML－関係データベースは XML データをそのまま格納でき，XML のクエリ言語（XQuery）も利用できる関係データベースである．本稿は，XML データを用いながら，関係データベースのスキーマ情報を利用する観点から考えて，ハイブリッド型 XML－関係データベースにおけるキーワード検索方式を提案する．この方式を実現するために，新しい XRjoin 演算を提案し，ハイブリッド型 XML－関係データベースでの結合演算をスムーズにする．本稿は，IBM DB2 V9.5 上でこの方式を実験し，既存手法では検索できない結果を得られることを示す．

A Study of Keyword Search on Hybrid XML-Relational Databases

Liru Zhang† Tadashi Ohmori† and Mamoru Hoshi†

Keyword search on structured databases has been popularized since it enables users to get information under databases without any knowledge of schema and query languages. Different from existing structured databases (traditional relational databases) and XML databases, hybrid XML-Relational databases system can not only store XML data directly as a type of a column, but also use XML query language (such as XQuery). From the standpoint using the schema of relational databases and the features of XML, we propose an algorithm of keyword search on hybrid XML-Relational databases. To realize it, we propose a new join operator, named XRjoin. We test this method by utilizing database IBM DB2 V9.5 to show it can get results that had not been got.

1. はじめに

近年，関係データベースのような構造化データベースや XML データベースにおいて，複雑な問い合わせ言語（SQL や XQuery など）の代わりにキーワード入力による検索方式に関する研究が注目されている．

従来の研究では，XML データ表現を考慮してデータベースのキーワード検索を行うには典型的な手法が二種類ある．

一つはピュアな XML データベースを用いて，XML データをそのままツリーの形で格納し，キーワード検索を行う手法 XRANK[1]である．

二つ目は関係データベースを利用し，XML データの情報を実体（Entity）と関連（Relationship）にマッピングして，管理する方式である．すなわち，XML データを分割して格納した関係データベース（RDB）で，RDB のキーワード検索方式 DBXplorer[2]の手法を用いれば，キーワード検索が実現できる．

著者らは文献[3]で，既存の手法 XRANK と DBXplorer を分析して，同じキーワード入力に対して両者の検索結果が一致していないことと，それぞれの検索結果が不十分であることを示してきた．そして， XML データをそのまま格納できるハイブリッド（Hybrid）型 XML－関係データベースを利用し，データベースのスキーマ情報による結合演算を行うことで，新しいキーワード検索方式を提案してきた．具体的には，従来のリレーションデータと新しく格納できるようになった XML データを結合するため，XRjoin という新しい演算を提案した．XRjoin を用いることによって，ハイブリッド型 XML－関係データベースでの結合演算が可能になる．本稿は，XRjoin の設計と実装方法及び XRjoin によるハイブリッド XML－関係データベース上でのキーワード検索の実行方法を説明し，IBM DB2 V9.5 上で実験を行った結果によって，既存手法では検索できない結果を得られることを示す．

以下では，2 節で DEWS2008 の時の話[3]を要約して，本研究の目的として解決する問題点を指摘する．3 節で，ハイブリッド型 XML－関係データベースにおけるキーワード検索について述べる．4 節で，XRjoin を説明し，幾つかの例を示す．最後に，5 節にまとめと今後の課題を述べる．

2. 従来方式について

構造化データベースにおけるキーワード検索と XML ドキュメントに対するキーワード検索の研究として，ここでは従来手法として XML 向けの XRANK と RDB 向けの DBXplorer の能力を[3]に沿って述べる．

† 電気通信大学大学院情報システム学研究科
Department of Information Systems Fundamentals, The University of Electro-Communications

Vol.2009-DBS-148 No.2
Vol.2009-FI-95 No.2
2009/7/28

2.1 XML データベースにおけるキーワード検索

XRANK[1]はツリー構造を利用し，キーワードを含むノードからなる subtree のツリーサイズ（ツリーの高さや横幅などによって計算されるもの）の昇順で検索結果を出す．検索結果の階層関係は XML データベースで格納した元の XML データと同じで，一つだけである．ほかの関係性があるとき，XLink や XPointer などを使って付加するが，その関係性が高くなっても XRANK の検索結果として考えない．そのため，データ間の関係性が多数あるとき，XRANK の検索結果は不十分である．

例えば，図 1 に示す例（Conference-Session-Paper-Citation に関する情報）では，会議 VLDB2004 の論文“ ObjectRank ”(ID 番号は P005)が会議 ICDE2002 の論文“ DBXplorer ”（ID 番号は P003）を引用した情報を表す．ここの引用関係は XLink（赤い点線）で表現している．検索キーワードが“ system ”と“ authority ”とすれば，図 1 のような異なった会議でヒットした subtree の連係の結果は XRANK では得られない．つまり，この二つのキーワード間の引用関係は XRANK で取り出すことができない．つまり，重要な関係性を持つ情報が求められず，検索結果は不完全である．

2.2 伝統的な関係データベースにおけるキーワード検索

DBXplorer[2]は Entity と Relationship を用いて，すべての関係性を平等に扱う．問い合わせが発行されたら，関係性の強い順でタプル集合から検索結果を取り出すことができるので，XRANK で落とした答えは DBXplorer で拾える．しかし，RDB で探せないものがある．XML データを分割して，RDB に格納するため，元の階層関係はばらばらになってしまう．キーワードが出現した Entity 以外には，XML 階層関係のようなキーワード間の関連を自動的に探索しない．つまり，下位の階層でヒットした情報について，上位の階層関係を求めない．その影響で検索結果の情報は不完全になる．

RDB でのキーワード検索のやり方はキーワードを入力したら，まず，RDB のスキーマでキーワードが出現した Entity を確定する．次に，これらの Entity をスキーマ上で探し，繋がる経路も調べる．スキーマ上で連結した部分は，Entity をノードとし，Relationship をエッジとする木になる．最後に，この連結木の中の Entity 間で結合演算する SQL 文を生成、発行し，検索結果のタプル集合を取り出す．この連結木は Jointree といい，結合（Join）する Entity と Relationship の情報を木の形で表現するものである．

例えば，図 2 のような会議とそのセッションの情報を RDB で格納するとき，キーワードが“ tuning ”と“ analysis ”を入力して，検索を行う．二つのキーワードが同じ Entity **Sessions** で出現するため，Jointree は **Sessions** ノードだけである．この Jointree による SQL ステートメントを生成、発行し，結果を取り出す．Entity **Conference** は Jointree に含まれないので，これに関しては何の結果も出されない．図 2 の点線で囲まれたところを見ると，二つのキーワードが同じ会議であることがわかるが，この情報は検索結果にならない．そのため，XML の階層関係で表される当たり前の関係性は RDB では自動的に取り出せない．

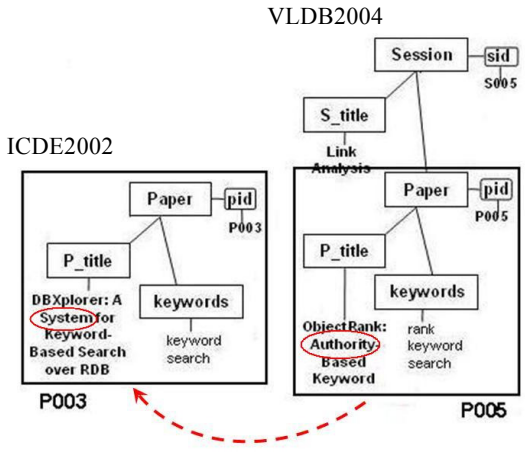


図 1　XRANK 手法によって論文間の引用関係を表す XLink の例

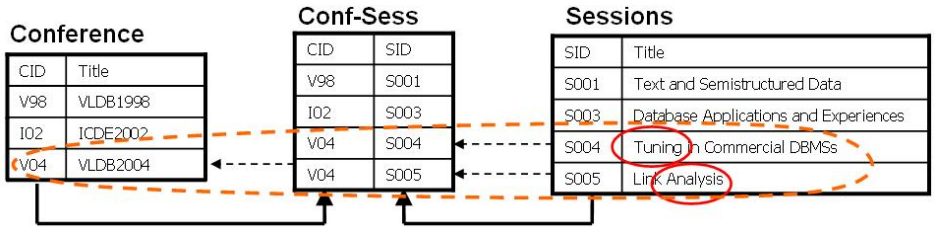


図 2　論文と会議の従属関係を伝統的な関係データベースで格納する例

3. Hybrid 型 XML－関係データベース

両者の問題を解決するため，著者らは XML データの格納を許した関係データベースに基づいて，Hybrid 型 XML-関係データベース上でのキーワード検索を提案する．

Hybrid 型 XML-関係データベースとは，従来のリレーションデータと XML データ両方を扱い，XML に対する問い合わせ言語 XQuery，リレーションデータに対する問い合わせ言語 SQL, XML とリレーション両方に対する問い合わせ言語 SQL/XML（DB2 の場合）の三つの問い合わせ言語をすべて利用できるシステムである．

3.1 データモデル

Hybrid 型 XML-関係データベースのデータモデルは関係データベースの ER 図に由来するものである．具体例として，図 3 は従来の関係データベースの ER 図で表した

Vol.2009-DBS-148 No.2
Vol.2009-FI-95 No.2
2009/7/28

“会議，論文，著者”の関係を示す．図 4 は“会議-論文”，“論文-著者”の関係を XML データに変えた Hybrid 型 XML-関係データベースのスキーマである．

図 4 の Hybrid 型 XML-関係データベースのスキーマで変わってない Entity は“Paper”である．論文に関する情報はリレーションデータで格納するのに適切だからである．図 3 と比べて，図 4 で変化したのは XML1 と XML2 の部分である．XML1 では“会議-論文”の階層情報と論文の ID だけ（論文のタイトルなど情報を含まない）を含む．同様に，XML2 は“論文-著者”の階層情報と論文の ID だけである．“Part-of”は「一部の」を意味する．図 4 のスキーマによるインスタンスの一部を図 5 に示す．

このような情報を XML で格納するのはスキーマ設計の一案にすぎない．特定のデータに対して，データベースを築くとき，設計者が分析し，状況によって設計すべきである[4].

3.2 キーワード検索

3.2.1 準備

Hybrid 型 XML-関係データベースでキーワード検索を行うため，前処理として，補助テーブルを作成する．補助テーブルを使ってスキーマでキーワードが出現した Entity を分析し，結合演算をどう行うか決める．

補助テーブルは三つ必要になる．

一つ目はすべてのリレーションデータに関する情報で，各タプルについて属性値、タプル ID、コラム名、Entity 名から構成される．（表 1 の Symtbl_relation で示す．）

二つ目はすべての XML データに関する情報で，各テキスト要素のバリュー、DeweyID、XML データを含むハイブリッド Entity 名から構成される．（表 2 の Symtbl_XML で示す．）

ここの DeweyID というのは， XML データの一つの Index 方法[5]によって，すべてのノードに対して配分される唯一の ID である．補助テーブルを構成する前に，データベースでのすべての XML データをスキャンして，DeweyID を属性として XML データに付加する．XML を格納した Entity はハイブリッド Entity と呼び，従来のリレーションデータを保存する Entity はリレーション Entity と呼ぶ．

三つ目はリレーションデータの ID と対応する XML データのノードの ID 情報のテーブルである．この表は，タプル ID と DeweyID から構成される．（表 3 の Symtbl_link で示す．）

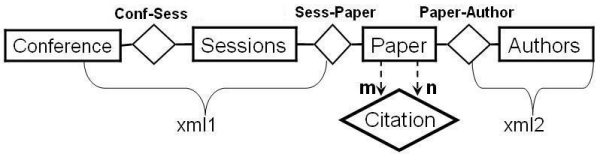


図 3　伝統的な関係データベースの ER モデルの例

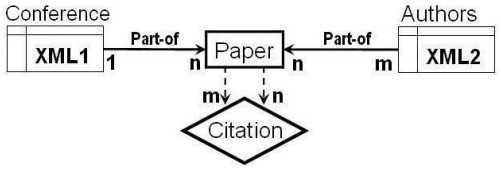


図 4　ハイブリッド XML－関係データベースのデータモデルの例

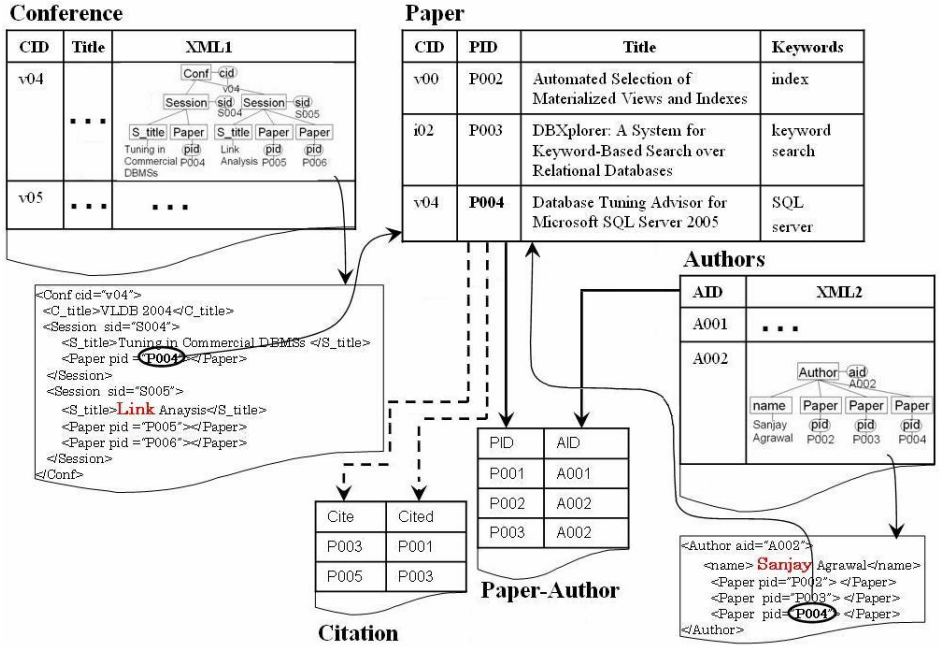


図 5　ハイブリッド XML－関係データベースのインスタンスの例

表 1　Symtbl_relation

Value	Tuple id	Column name	Entity name

表 2　Symtbl_XML

Value	DeweyID	Hybrid Entity name

表 3　Symtbl_link

Tuple id	DeweyID

Vol.2009-DBS-148 No.2
Vol.2009-FI-95 No.2
2009/7/28

3.2.2 キーワードが与えられたときの動作の概要

STEP 1．補助テーブル表 1 と表 2 を参照して，キーワードが出現した Entity を確定する．

STEP 2．確定した Entity をスキーマ上で連結木 Jointree を構成する．

STEP 3．連結木 Jointree でハイブリッド Entity $\mathbf{X_i}$ があった場合は，その $\mathbf{X_i}$ と繋がるリレーション Entity $\mathbf{E_i}$ との間で XRjoin を行い，中間結果をつくる．

STEP 4．連結木 Jointree で XRjoin をした中間結果と残りの Entity の間で自然結合を行って，キーワードに関するタプル集合を取り出す．

（ここで，XRjoin とはハイブリッド Entity の XML とリレーション Entity のデータの中で，キーワードと関連する情報（XML の階層情報と Relationship の情報を両方含む）を取り出す操作をする演算である．XRjoin((Hybrid Entity, keyword1), (Relation Entity, keyword2))と記す．詳しい説明を 4 節で述べる．ハイブリッド XML－関係データベースで Jointree による結合演算は XRjoin を用いれば実現できるものである．）

STEP 5．XML データと従来のリレーションデータ両方を含む答えタプル集合を変形し，ユーザーインターフェースへ出力する．

3.3 具体例

クエリ {“ link ”, “ sanjay ”}（キーワード K1 は“ link ”で，キーワード K2 は“ sanjay ”である）の具体例では，次の処理になる．

STEP 1．Symtbl_relation（表 1）と Symtbl_XML（表 2）の Value で K1 もしくは K2 があった Entity を探し出す．“ link ”の場合は **Conference** と **Paper** 両方にある．“ sanjay ”の場合は **Authors** で出現した．

STEP 2．STEP 1 の結果を反映したスキーマを図 6 で示す．図 6 のスキーマ情報を展開していくと，図 7 の二つの Jointree が得られる．Jointree1 と Jointree2 は二種類の結合パターンを決める．

STEP 3．図 7 の Jointree1 によって，ハイブリッド Entity **Authors** があると，XRjoin((Authors, “sanjay”), (Paper, “link”))を行う．

図 7 の Jointree2 によって，ハイブリッド Entity が **Conference** と **Authors** 二つがあると，XRjoin((Conference, “link”), (Paper))と XRjoin((Authors, “sanjay”), (Paper))を行う．

STEP 4．Jointree1 の場合は，XRjoin した後残りの Entity がないので，そのままで結合演算が終る．Jointree2 の場合は STEP 3 の二つの中間結果テーブルを自然結合することになる．

STEP 5．インターフェースの処理で，データベース検索結果となるタプル集合をユーザーの要求に応じて変形し，表示する．

4. XRjoin

この節では XRjoin の設計理念と振る舞いについて説明する．

4.1 アルゴリズム

XRjoin という結合演算はハイブリッド Entity **X** と XML データを含まないリレーション Entity **E** の間で，**X** の XML に格納したタプル ID を利用して，キーワードに関する必要な情報抽出を行う．

図 8 のハイブリッド Entity **X** にキーワード K1 がヒットし，リレーション Entity **E** にキーワード K2 がヒットした場合を例に，表 4 に示した XRjoin のアルゴリズムを説明する．

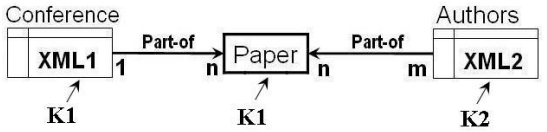


図 6　ヒットした K1: “ link ”，K2: “ sanjay ” のスキーマ情報

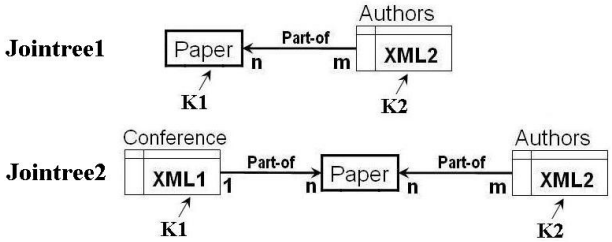


図 7　ヒットした K1:“ link ”, K2:“ sanjay ”のスキーマ情報から展開した連結木 Jointree

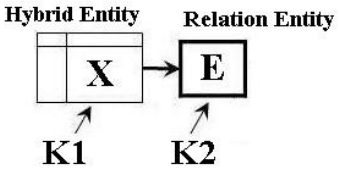


図 8　XRjoin((**X**, K1), (**E**, K2))

表 4　XRjoin((X, K1), (E, K2))のアルゴリズム

Input: **X**, K1, **E**, K2
Output: **T**
1: for each e ∈ σ_{K2}(**E**)
2: for each x ∈ **X**
3: insert a tuple [LCA-T(e_id, K1) on x, e] into **T**
4: endfor
5: endfor
6: return **T**

表 4 の XRjoin((X, K1), (E, K2))は，ハイブリッド Entity **X**、キーワード K1、リレーション Entity **E**、キーワード K2 が入力データとし，新たに生成したハイブリッド Entity **T** を出力とするアルゴリズムを示す．K1、K2 ともに単一キーワードとする．

1 行目の操作はキーワード K2 を条件にして，セレクトしたすべてのタプル e に対し，2～4 行の繰り返し処理を利用する．2～4 行の処理はすべての XML データ x に対して，[LCA-T(e_id, K1) on x, e]の結果が新しいタプルとして **T** に挿入する．

[LCA-T(e_id, K1) on x, e]の操作は，XML x についてその subtree を求める処理 LCA-T(e_id, K1) on x と，e_id に対応するタプル e の全体を取り出す処理の二つからなる．LCA-T(e_id, K1) on x の処理は，与えられた x に対して，リレーションタプル ID e_id のノードとキーワード K1 を含むノードの最小共通先祖（Lowest Common Ancestors，以下 LCA と略す）を計算し，LCA を subtree の根ノードとする．subtree 全体（LCA-T）を結果の XML 部とする．その後，得られた XML 部と対応するリレーションタプル e の全体を合わせて，一つのタプルとして **T** に挿入する．

表 4 のアルゴリズムは XRjoin の基本タイプの定義である．一般形としては，K1 は一個以上のキーワードの集合を許し，K2 は 0 もしくは 1 個のキーワードである場合に拡張する．K2 がφのときは 1 行目の操作をリレーション Entity のすべてのタプルに対して行う，すなわち，

1: for each e ∈ **E**

とすれば，対応できる．XML 部の subtree を取る条件 K1 が空ではない集合で，キーワード要素が二個以上の集合であるときは，取り出し（3 行目）を

3: insert a tuple [LCA-T(e_id, k_1, k_2…k_n) on x, e] into **T**

へ変更して対応する．（K1＝φは許さない）

現在はデータベースのスキーマ上で与えたキーワード集合にヒットするすべての Entity E_1, E_2…E_n からなる最小コストシュタイナ木（a minimum cost steiner tree）を Jointree として作って，それに沿って XRjoin と Join で動かして結果を求めている．

4.2 振る舞い

IBM DB2 V9.5 を利用して，図 5 の DB インスタンスを用いて，具体例をあげて XRjoin の表 4 に沿った実現方法と動作例を説明する．

図 9 はハイブリッド Entity **Conference** でキーワード“ tuning ”が出現し，リレーション Entity **Paper** にキーワード“ base ”があった場合の Jointree の一つである．それに関するインスタンス情報は図 10 になる．図 10 では，キーワード K1“ tuning ”を赤い丸で，キーワード K2“ base ”を紺色の丸で囲んで，XML1 の属性 DeweyID も表示してある．このとき，図 9 に対応する XRjoin((Conference, “tuning”), (Paper, “base”))は次のように動作する．

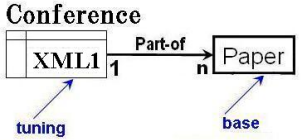


図 9　XRjoin((Conference, “tuning”), (Paper, “base”))

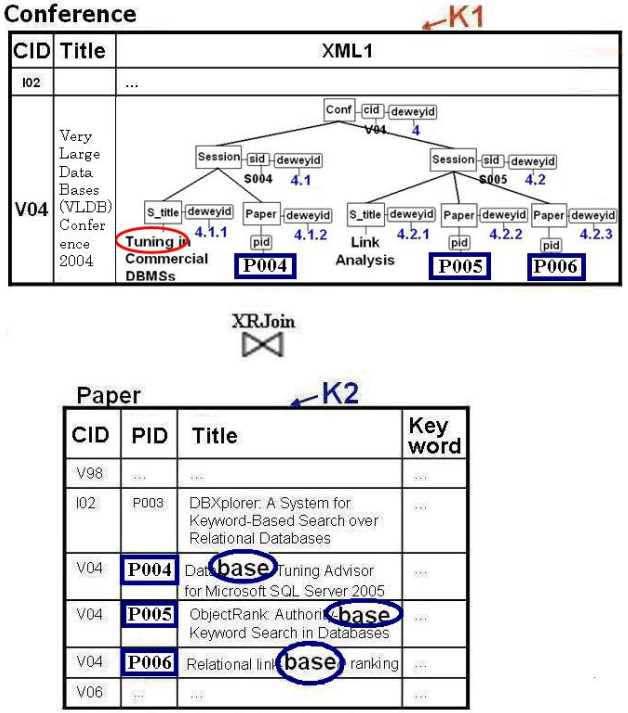


図 10　K1：tuning、K2：base のインスタンス情報

まず，補助テープル Symtbl_XML からキーワード“ tuning ”に関する情報を取り出す（図 11 に示す）.“ tuning ”を含むテキストの親ノードの DeweyID は 4.1.1 である．同じように，図 12 は補助テーブル Symtbl_relation と Symtbl_link を使って取得した“ base ”に関する情報である．図 11 と図 12 の DeweyID を使用して,“ tuning ”と“ base ”の関連ノードの最小共通先祖 LCA の DeweyID を取得できる．図 13 は取得した LCA の情報である．

次に，LCA を取得すると同時にスコアを与える．DeweyID が長ければ長いほどノードは XML のツリーでの位置が深いので，LCA の DeweyID の長さと K1“ tuning ”、K2“ base ”に関連するノードの DeweyID の長さの差は subtree の深さを表す．ここは，“ . ”を区切って各 DeweyID の長さを計算し，長さの差に対応して，スコアを与える（スコアが小さいほど良いとする）．スコアの大きいほうが LCA の subtree が深い（階層が多い）．現段階は subtree の横幅は計算していない．この例では，DeweyID 4.1.1 と 4.1.2 の LCAID は 4.1 で，スコアは一番小さいので，LCA 4.1 の subtree の縦幅は一番短いことがわかる．図 13 のテーブル LCA のスキーマは[K1 の DeweyID, K2 の DeweyID, LCA の DeweyID，Subtree の score]になる．

最後に，ハイブリッド Entity **Conference**，リレーション Entity **Paper** とテーブル LCA を用いて，下記のような SQL/XML を自動生成し，実行すると，図 14 の結果が得られる．

DB2 V9.5 の SQL/XML ステートメントの例：

```
SELECT cf.id , cf.ctitle , lca.lcaid , X.* , paper.*, lca.score
FROM conference cf , lca , symtbl_link sl , paper ,
XMLTABLE ('$c/Conference//descendant-or-self::node()[@DEWEYID=$l]'
passing cf.xml1 as "c", lca.lcaid as "l"
COLUMNS "subtree" XML path 'document{.}'    ) as X
WHERE lca.k2nodeid = sl.deweyid and sl.pid = paper.pid
ORDER BY score
```

現在の XRjoin は以上の 3 パートから構成される．

図 14 の subtree 列の情報は **Conference** の XML から得たものである．1 行目の結果は LCA のスコアが一番小さいもので,“ tuning ”がセッションのタイトルで出現して,“ base ”が同セッションの P004 番の論文のタイトルにあったことを表す．つまり，VLDB2004 の“ base ”に関する論文は“ tuning ”に関するセッションで発表されたということがわかる．

VALUE	PARENTDEWEYID	HYBRIDNAME
tuning in comm...	4.1.1	Conference

図 11　Symtbl_XML 中のキーワード“ tuning ”に関する情報（DB2 の結果表示）

PID	DEWEYID	VALUE	RELATIONNAME	COLUMNNAME
P003	1.1.2	dbxplorer: a system for keyword-based search over relational DBs ...	Paper	title
P003	12.3	dbxplorer: a system for keyword-based search over relational DBs ...	Paper	title
P004	12.4	database tuning advisor for microsoft sql server 2005 ...	Paper	title
P005	13.2	objectrank: authority-based keyword search in DBs ...	Paper	title
P006	14.2	relational link-based ranking ...	Paper	title
P004	4.1.2	database tuning advisor for microsoft sql server 2005 ...	Paper	title
P005	4.2.2	objectrank: authority-based keyword search in DBs ...	Paper	title
P006	4.2.3	relational link-based ranking ...	Paper	title

図 12　Symtbl_relation と Symtbl_link 中の K2“ base ”に関する情報（DB2 の結果表示）

K1NODEID	K2NODEID	LCAID	SCORE
4.1.1	4.1.2	4.1	100
4.1.1	4.2.2	4	200
4.1.1	4.2.3	4	200

図 13　テーブル LCA の例（DB2 の結果表示）

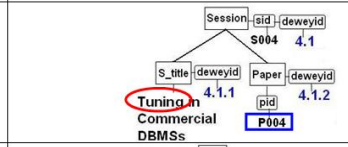
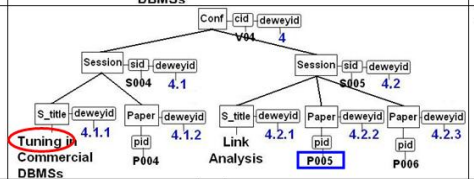
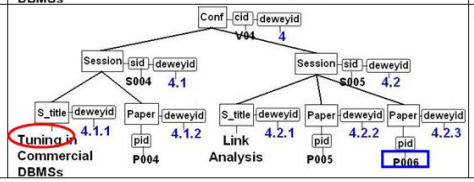
CID	CTITLE	LCA ID	SUBTREE	PID	TITLE	SCORE
V04	Very Large Data Bases (VLDB) Conference 2004	4.1		P004	database tuning advisor for microsoft sql server 2005	100
V04	Very Large Data Bases (VLDB) Conference 2004	4		P005	ObjectRank authority base keyword search in DBs	200
V04	Very Large Data Bases (VLDB) Conference 2004	4		P006	relational link-based ranking	200

図 14　XRjoin((Conference, “tuning”), (Paper, “base”))の結果

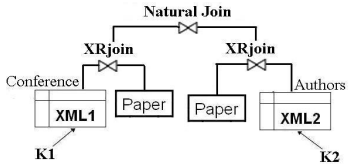


図 15　図 7 の Jointree2 による結合手順の例（K1:“ link ”，K2:“ sanjay ”）

Vol.2009-DBS-148 No.2
Vol.2009-FI-95 No.2
2009/7/28

最後にXRjoinを使ったJointreeの実行例として,もう一度3節の例について述べる.

キーワード {“ link ”, “ sanjay ”} の Jointree2（図 7）について，XRjoin と Join を使って実行する手順を図 15 に示す．図 15 の結合処理は下から実行される．つまり，XRjoin((Conference, “link”), (Paper))と XRjoin((Authors, “sanjay”), (Paper))を実行してから，二つの XRjoin の結果テーブルを自然結合する．この検索結果を図 16 で示す．

図 16 の結果は「Sanjay 氏が書いた Paper004 は “ link ” に関する Session を含む Conference VLDB2004 に出ている」ということを意味している．この例は従来手法ではできない検索であり,我々の提案したシステムでは可能であることを示している．

4.3 議論

現段階では，XRjoin をできるだけシンプルに動かすことを考えて実装している．我々の提案した方法は，DBXplorer の検索能力を全て含むが，XRANK の検索能力の全てを含んでいる訳ではない．

例えば，図 17 で示す例のように，キーワード K1，K2 がリレーション Entity **Paper** で出現する場合を考える．今の方式では，最小コストシュタイナ木として，リレーション Entity **Paper**（青い部分）だけを選び，これに対して SQL のステートメントを生成、発行し，検索結果を取る．本来 XRANK なら，**Paper** のタプル ID を格納した **Conference** の XML1 で, K1 の paper と K2 の paper の共通先祖を求めて subtree を取ることによって，キーワードを含む論文が同じ会議であるという情報がわかるが，本稿で述べた方式ではこのような階層情報による結果を取らない．故に，青い部分のリレーション Entity **Paper** から, ハイブリッド Entity **Conference** を含むように図 17 全体で示す Jointree を拡張すべきである．これは今後の課題になる．

5. おわりに

本稿は XML データベースと伝統的な関係データベースにおける既存のキーワード検索方式が不十分な答えしか取り出さないことを指摘し，それらの問題を解決するため, ハイブリッド XML－関係データベース上でのキーワード検索のアルゴリズムを提案した．データベースでの結合演算をスムーズにするため，XRjoin を設計し，実行した．IBM DB2 V9.5 上での実験によって，XRjoin を用いることによって，複雑なクエリ言語のステートメントを自動生成、発行し，従来手法では得られない検索結果を得られることを示した．

今後，XRANK の検索能力全てを含むように，Jointree を拡張し，かつ XRjoin のデザイン自体は変えない方針で，実装していく．ハイブリッド XML－関係データベースに大規模な実データを格納して，スケーラビリティ評価を行うことも課題である．

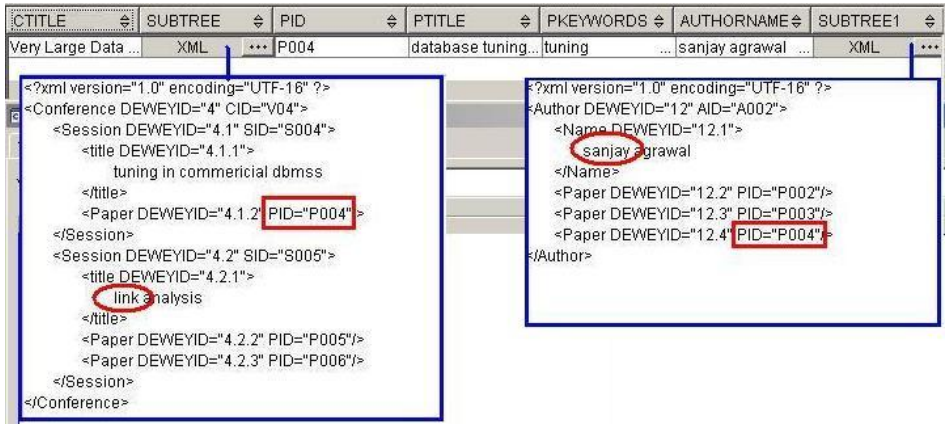


図 16 “ link ” と “ sanjay ” による検索結果（DB2 の結果表示）

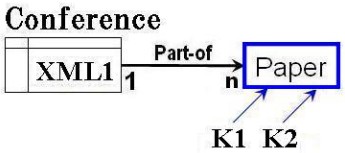


図 17 キーワード K1，K2 がリレーション Entity **Paper** で出現した例

参考文献

1) Lin Guo, Feng Shao, Chavdar Botev and Jayavel Shanmugasundaram, XRANK: Ranked Keyword Search over XML Documents, SIGMOD, pp.16–27(2003).

2) Sanjay Agrawal, Surajit Chaudhuri and Gautam Das, DBXplorer: A System for Keyword-Based Search over Relational Databases, ICDE, pp.05–17 (2002).

3) 張麗茹, 大森匡, 星守, XML データ表現を考慮した関係データベースのキーワード検索, DEWS, C6-2, 2008.

4) Bolin Ding, Jeffrey Xu Yu, Shan Wang, Lu Qin, Xiao Zhang and Xuemin Lin, Finding Top-k Min-Cost Connected Trees in Databases, ICDE, pp.836-845 (2007).

5) Liru Zhang, Tadashi Ohmori and Mamoru Hoshi, Keyword Search over Hybrid XML- Relational Databases, SICE, 1A04-5 (2008).

6) 日本アイビーエム,プロジェクトの流れで理解する XMLDB デザイン徹底解説－最新 DB2 9.5 pureXML 対応, 日経 BP 社 (2008).

7) Patrick E. O'Neil, Elizabeth J. O'Neil, Shankar Pal, Istvan Cseri, Gideon Schaller and Nigel Westbury, ORDPATHs: Insert-Friendly XML Node Labels, SIGMOD, pp.903-908 (2004).