DEIM Forum 2009 D1-5

注釈を利用したXMLデータのためのトレーサビリティ機構の提案

永元　芳幸†　　天笠　俊之†,††　　北川　博之†,††

† 筑波大学大学院システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学計算科学研究センター　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †nagamoto@kde.cs.tsukuba.ac.jp, ††{amagasa,kitagawa}@cs.tsukuba.ac.jp

あらまし　現在，XML に対する問合せ言語（XQuery 等）を利用することで，簡単に複数の XML 情報源を組み合わせて新たな XML データを作成することが可能となっている．しかし XML データが次々と加工，統合されて伝播していく状況において，複雑に伝搬した XML データの起源や伝搬先を追跡することは既存の問合せ言語だけでは困難である．そこで本稿では，XML データに対するトレーサビリティ機構を実現するため，XML データに含まれる全てのノードに対して注釈と呼ばれるメタデータを付与した新たな XML データモデルを提案する．また，複雑に伝播した XML データを検索するために，注釈を利用した XML データの検索方法を提案し，実験によってその計算速度を検証する．さらに，検索結果のランキング処理と表示処理について述べる．

キーワード　XML, XQuery，metadata, トレーサビリティ

A System for XML Data Traceability based on Annotations

Yoshiyuki NAGAMOTO†, Toshiyuki AMAGASA†,††, and Hiroyuki KITAGAWA†,††

† Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
†† Center for Computational Sciences, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
E-mail: †nagamoto@kde.cs.tsukuba.ac.jp, ††{amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract　Recently, it has become easy to generate a new XML data from othrer plural XML data using XML query languages, such as XQuery. However, in such cases, it is difficult to trace the origin or the data flow of that XML data. In this paper, we propose a new XML data model on which all nodes are annotated with metadata called annotaion to construct the Traceability System for XML data. We show the method of retrieval using annotaion and make analysis of perforamnce of the method. In additon, we also show the ranking method for results of retrieval and how to display the results.

Key words　XML, XQuery, metadata, traceability

1. はじめに

現在，Web 上の情報をはじめとして，様々な情報を記述するためのフォーマットとして XML (Extensible Markup Language) [1] が使用されている．たとえば e-サイエンスにおいては，研究データなどに XML という共通のフォーマットを利用することにより，従来各研究者や研究グループ内などの閉じた環境で使用されてきた研究データを公開し，情報の共有化をはかり研究の生産性をあげようという動きが活発化してきている．そのため，他の研究者が公開した XML データを自分の研究に利用し，研究結果をさらにまた XML で記述，公開するといったことが広く行われるようになってきている．また，XML に対する問合せ言語（XQuery 等）を利用することで，簡単に複数の XML 情報源を組み合わせて新たな XML データを作成することが可能となっている．

このように，様々な情報を表現する XML データが，次々と加工，統合されて伝播していくといった状況において，ある XML データにおいて何らかの問題が発見された場合，その XML データの生成に関連する XML データや，間違いを含む XML データを組み込んで作成された XML データなどを全て探し，修正を行う必要がある．これには，複雑な問合せによって伝搬した XML データに対して，その情報の起源や伝播先を追跡するデータのトレーサビリティ機構の実現が必要であると考えられる．

本稿では，XML データに対するトレーサビリティ機構を実現するため，XML データに含まれる全てのノードに対して注釈 (annotation) と呼ばれるメタデータを付与した新たな XML

データモデルを提案する．さらに，注釈を利用した XML データの検索方法について述べる．また，検索結果のランキング処理と，検索結果から XML データを再構築し表示する処理についても検討する．

本稿の構成は次の通りである．2 章では関連研究について述べ，3 章では前提となる XML 関連技術について説明する．4 章で注釈を付与した XML データモデルとその格納方法を説明する．5 章ではトレーサビリティ機構の概要を説明する．6 章で注釈を利用した XML データの検索方法について述べる．7 章では，検索結果に対する処理として，検索結果のランキング処理と，検索結果から XML を再構築し表示する処理について検討する．最後に，8 章においてまとめと今後の課題について述べる．

2. 関 連 研 究

リレーショナルデータベースに対して注釈を付与する方法として，pSQL [2] がある．pSQL では，SQL を拡張し，付与された注釈をどのように問合せで伝播させるかを記述する PROPAGATE 句を提案している．また，関連する別の技術としては，データの起源を求めるために Lineage なる概念を用いている研究がある．Trio System [3] では，Lineage の概念を取り入れたデータモデルである ULDB を提案し，Lineage を利用してデータの起源を問合せることが出来る TriQL という SQL ベースの問合せ言語を提案している．さらに，XQuery を拡張し，XML に付与されたメタデータを問合せに利用することができるようにした MetaXQuery [4] が提案されている．MetaXQuery では，メタデータとして，各要素にセキュリティ情報やタイムスタンプなどを記述し，その情報を問合せに条件として取り入れる方法が述べられている．

3. 関 連 事 項

XML データには，仮想的なノードであるルートノード，要素を示す要素ノード，その従属ノードである属性ノード，および要素ノードの子供ノードとして，テキストデータを示すテキストノードが存在する．さらに，これらのノードからなる木構造として表現することができる．XML データに対して本研究では，XQuery などの XML 問合せ言語によって既存の XML データから抽出された部分データが伝搬していく状況を想定している．XQuery [5] は 2007 年に W3C 勧告として公開された，XML データに対する関数型データベース問合せ言語である．XQuery では，FLWOR 表現式と呼ばれる，for 句，let 句，where 句，order by 句，return 句の五つの表現式を用いることで，リレーショナルデータベースに対する SQL のように柔軟で強力な問合せを行うことができる．また，構築子を利用することで，問合せ結果を組み込んだ新たな XML データを容易に作成することが可能である．

4. 注釈を付与した XML データモデル

本研究では，XML データのためのトレーサビリティ機構を実現するため，注釈を付与した XML データモデルとその格納方法を提案する．以下，まず注釈を付与した XML データモデルについて説明する．次に，注釈を付与した XML データを経路アプローチに基づき関係表へマップする手法について述べる．

4.1 注釈付き XML データモデル

本研究では，新しく作成する XML データのルートノードを除く全てのノードに対して，そのデータ内で一意となる ID をもつ注釈を付与する．また，XML データはデータベース内で一意となる XML データ ID を付けてデータベースに格納するため，XML データ ID と注釈の ID を指定すれば注釈を一意に識別することができる．図 1 に新しく作成された XML インスタンスの例を示す．たとえば，要素<cities>のいちばん左の子である要素<city>には，1-2 という注釈が付与されている．これは，XML データ ID が 1 である XML データで作成された 2 という ID をもつ注釈であることを示している．

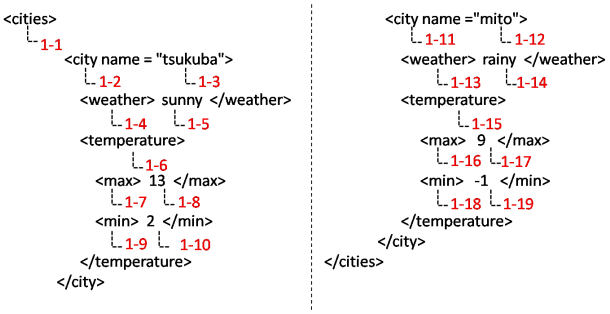

図 1　注釈を付与した XML インスタンスの例

また本研究では，XQuery などの XML に対する問合せによって，様々な XML データが統合，加工され新たな XML データが作成されていく状況を想定している．そこで問題となるのが，いくつかの XML データを組み合わせて新たな XML データを作成した際に，どのように注釈を付与するかである．本研究では，新たな XML データを作成する際，他の XML データから組み入れたデータには，元々の注釈をそのまま付与しておき，新しく作成した部分については，そのドキュメント独自の新たな注釈を付与するという方針をとる．たとえば，図 1 の XML データに対して XQuery による問合せを実行した場合の，実行結果への注釈の伝播の様子を図 2 に示す．なお，図 1 の XML データのファイル名を”sample.xml”とする．

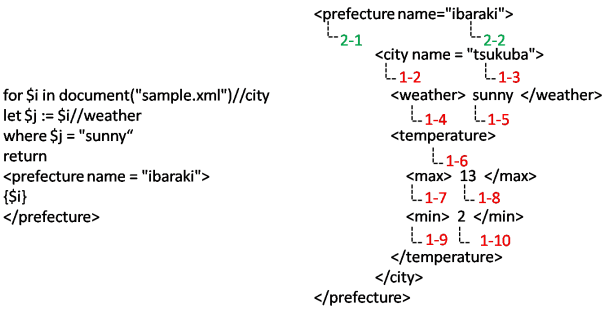

図 2　XQuery による注釈の伝搬例

この例では，新たに作成した要素である「prefecture」とその属性である「name=”ibaraki”」について新たな注釈を付与している．このようにすることで，データの作成者の所在をはっきりさせることができるだけでなく，同じデータを利用している全ての XML データを検索することが可能になる．

4.2 注釈付き XML データモデルのデータベースへの格納

本研究では，注釈を付与した XML データを関係データベースに格納するアプローチを取るものとする．ただし，提案手法は，関係 XML データベースだけではなく，一般の XML データベースに対しても応用可能である．XML データを関係データベースに格納する手法はこれまで多数提案されているが，本研究では中でも経路アプローチを利用する．これは DTD に依存することなく XML データを格納，検索することが可能となる手法である．代表的な手法としては XRel [6] がある．本稿では XRel を参考にして，注釈付き XML データを関係表に格納する． 表 1 に，図 1 で示した XML データを関係表にマップする例を示す．

tid	did	type	name	value	nodenum	path	anno
1	1	element	cities	null	1	/cities	1-1
2	1	element	city	null	1.1	/cities/city	1-2
3	1	attr	name	tsukuba	1.1	/cities/city/@name	1-3
4	1	element	weather	null	1.1.1	/cities/city/weather	1-4
5	1	text	#PCDATA	sunny	1.1.1.1	/cities/city/weather/text()	1-5
...	...	...	...	...	...	...	...

表 1　注釈付き XML データの関係表への格納

ここでは，XML データの各ノードの親子，兄弟関係を表すために Dewey Order を用いる．これは，親ノードのラベルの後に何らかのデリミタ（多くは ”. ”）を挟んで親子，兄弟間の順序を示すコードを付与する方法である．

5. トレーサビリティ機構の概要

本研究では，注釈が付与された複数の XML データを格納しているデータベースが多数ある状況を想定している．また，データベース内やデータベース間で XQuery 問合せによる XML データの統合や加工が行われ，その結果がデータベース内に格納されるものとする．そこで，起源や伝搬先を追跡したい XML データに対して，同一の注釈をもつ XML データを各データベースの中から検索することを可能とすることで，XML データのトレーサビリティ機構の実現を目指す．図 3 に全体図を示す．

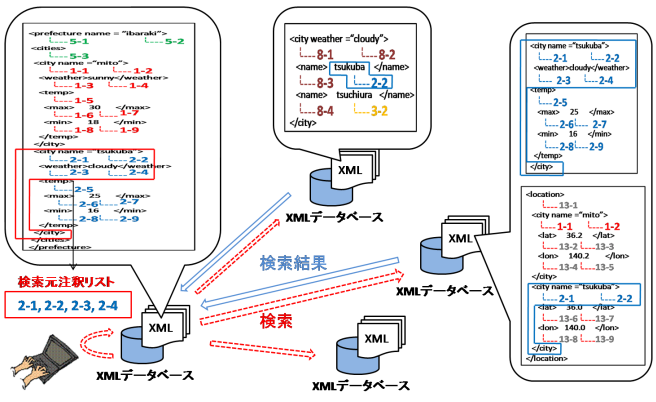

図 3　トレーサビリティ機構の概要

図 3 に示すように，本研究では，検索を行いたいデータから注釈集合を抽出し，各データベースに対して，それぞれに含まれる複数の XML データの中から，共通の注釈が付与されているデータを検索し，検索結果として返す．そのために，大きく分けて以下の三つの処理を行う．

- 注釈を利用して XML データの検索を行う処理
- 検索結果をランキングする処理
- 検索結果から XML データを再構築し，表示する処理

本稿では，以降，上記の三つの処理について詳細を述べる．

6. 注釈を利用した XML データの検索

注釈を利用した XML データの検索処理は,「検索対象データベースの絞り込み処理」と「各データベース内の検索処理」の二つに分けられる．本研究では，検索要求として，起源や伝搬先を追跡したいノードに付与されている注釈集合に対して,「一つでも共通の注釈を含むデータを検索したい」といった要求の他に,「全ての注釈を含むデータを検索したい」といった要求も想定している．また，検索対象となるデータベースが多数ある場合，検索要求を満たさないデータベースも多く存在していると考えられる．そこでまず，検索対象データベースの絞り込み処理を行う．その後，各データベース内の検索を行っていく．以下では，まず検索モデルについて述べる．次に，検索に利用するインデックスについて述べ，その後，データベースの絞り込み処理について述べる．最後に，データベース内の検索処理について説明する．

6.1 検索モデル

本研究では，ある XML データに含まれるデータの起源や伝搬先を調べるため，その XML データと共通の注釈をもつ XML データの検索を行う．そこでまず検索モデルについて述べる．

まず，入力として，データの起源や伝搬先を調べたい XML データのノード情報を与え，このノードに付与されている注釈の集合を取得する．次に，その注釈集合を元に，検索対象となるデータベースの絞り込みを行う．その後，各データベース内を検索し，共通の注釈をもつノードのノード情報を出力として返す．また，本節で述べた検索モデルを実現するためには，注釈をキーとし，値としてその注釈が付与されているノード情報を持つインデックスが必要であると考えられる．よって，次節でそのインデックスについて述べる．

6.2 インデックスの作成

本研究では各データベースごとに，注釈を付与した XML データをマッピングした関係表から，事前に注釈に対する転置インデックスを作成しておき，そのインデックスを利用して XML データの検索を行う．以下，このインデックスを注釈インデックスと呼ぶ．表 2 にインデックスの例を示す．

表 2 において，たとえば注釈 (2-1) は,「XML データ ID=2 の XML データに含まれ，関係表中でタプル ID=1 のタプルとして格納されているノード」と「XML データ ID=5 の XML データに含まれ，タプル ID=14 のタプルとして格納されているノード」に付与されていることを表している．また，ノード情報は XML データ ID，タプル ID によってソートされている．

注釈	ノード情報（XML データ ID，タプル ID）
2-1	(2,1), (5,14)
2-2	(2,2), (5,15)
4-27	(13,33)
4-28	(13,34)
...	...

表 2　注釈インデックスの例

6.3　検索対象の絞り込み処理

本研究では，各注釈インデックスに格納されている注釈の集合を事前に抽出しておき，検索の際に，検索元の注釈集合との積集合を計算し，その計算結果によって検索対象とするかどうかを判断する．たとえば，全ての注釈を含むデータを検索したい場合，検索元の注釈集合との積集合が，検索元の注釈集合と等しくなるデータベースのみを検索対象とする．また，積集合の計算には，積集合を高速に検索するアルゴリズムである Baeza-Yates のアルゴリズム [7] を利用する．Baeza-Yates のアルゴリズムでは，ソートされた二つのリストに共通に含まれる値を，二つのリストの分割を繰り返しながらバイナリサーチによって探索する．そこでまず，起源や伝搬先を追跡したいノードに付与されている注釈集合を検索元リストとする．次に，各データベースごとに，それぞれがもつ注釈インデックスから注釈集合を抽出し検索対象リストとする．その後，検索元リストと複数の検索対象リストの積集合を順次計算していく．しかしこの方法では，検索対象リストを変えるたび，検索元リストに対し何度も同じ走査を行うことになってしまうため，処理効率が低下してしまうことが予想される．そこで本研究では，一つの検索元リストと複数の検索対象リストの積集合を同時並行に計算するために，Baeza-Yates のアルゴリズムを拡張したアルゴリズムを提案する．以下にその拡張したアルゴリズムの概要を示す．

検索元となるリストを Q，検索対象となるリストの集合を D[] とする．まず最初に，D[] の各要素に対して Q の中間にある値をバイナリサーチにより探索する．D[] の各要素において，もし同じ値が見つかった場合，その値をそれぞれの要素の結果シーケンスに追加する．また，値が見つかったか見つからなかったかによらず，Q を Q の中間にある値よりも左にある値のリストと右にある値のリストの 2 つに分割する．D[] の各要素についても同様に，Q の中間値よりも左にある値のリストと右にある値のリストの 2 つに分割する．さらに，左側と右側のリストそれぞれに対し，同様のアルゴリズムで検索元のリストが空になるか，検索対象となるリスト全てが空になるまで，再帰的に探索，分割を繰り返していく．最後に検索結果として，それぞれの結果シーケンスを返す．ただし，元のアルゴリズムでは，検索元のリスト Q の大さが検索対象のリスト D よりも大きい場合に，Q と D を交換することでバイナリサーチの効率を上げていたが，拡張したアルゴリズムでは，一つのリストの値について同時に複数のリストに対しバイナリサーチを行うため，交換処理は行っていない．

6.3.1　検索処理のスレッド化

本研究では，拡張したアルゴリズムにより，複数のリストに対し同時に探索を行う．そこで，スレッドを用いて検索対象となるリストを分割し，各スレッド毎に分割されたリストの集合に対して並列に探索を行う．

6.3.2　Bloom Filter による検索対象のフィルタリング処理

6.3 節では，注釈集合の積集合を計算することで，検索対象となるデータベースの絞り込みを行った．しかし，データベースの中には，検索元の注釈集合と共通の注釈を一つも含んでいないデータベースも存在すると考えられる．このようなデータベースは検索対象とはなり得ないため，積集合を計算すること自体が非効率である．そこで本研究では，Bloom フィルタ [8] を利用して，検索対象となるデータベースをフィルタリングし，検索元の注釈を一つも含んでいないデータベースを検索対象から除外する処理を行う．Bloom Filter を利用すると，ある要素がある集合に含まれているかどうかを，空間的に小さいデータ構造で高速に判定することができる．よって，各 XML データベースごとに Bloom Filter を用意し，Bloom Filter 同士の AND をとることで，共通の注釈が含まれているかどうかを判定する．

しかし，単純に各データベースに含まれている注釈を全て Bloom Filter に格納すると，注釈の数が非常に大きい場合に Bloom Filter による判定ができなくなってしまうという問題が生じる．そこで本研究では，各注釈インデックスに格納されている注釈から XML データ ID を抽出し，その XML データ ID を Bloom フィルタに格納する．この方法では，同じ XML データ ID をもつ異なる注釈を含んでいるデータベースを検索対象から除くことはできない．しかし，本研究が想定している XQuery などの問合せ言語によって注釈が伝搬していく状況において，データベースが数多くある場合，一つの Bloom フィルタに格納される XML データ ID の数は各データベースに含まれる XML データの総数に比べてそれほど多くはならないであろうことが予想されるため，十分検索対象を減らし検索速度を向上させることができると考えられる．

6.3.3　評 価 実 験

本節では，実験により，検索対象データベースの絞り込み処理の計算速度を検証する．本研究では，計算速度を検証するため，計算に利用する注釈インデックスをプログラムにより作成した．注釈インデックスを作成する際には，パラメータとしてインデックスのサイズ，インデックス数の他に，ノード引用率（その XML データの何割が他の XML データから組み込まれたノードなのか）と XML データ引用率（組み込まれたノードの注釈の起源となる XML データ数が，総 XML データ数の何割にあたるのか）を与える．実験環境として，CPU:Intel(R) 1.60GHz x4，OS:Red Hat Enterprise Linux AS release 4，メモリ:5GB の計算機を 1 台用いた．また，注釈インデックスを格納するデータベースとして PostgreSQL 8.3.5 を使用した．

実験は，ノード引用率，XML データ引用率をともに 10%とし，総インデックス数が 25, 50, 100 のそれぞれの場合について，インデックスサイズを 1000〜64000 まで変化させて行った．総インデックス数が 50 の場合の実験結果を図 4 に示す．

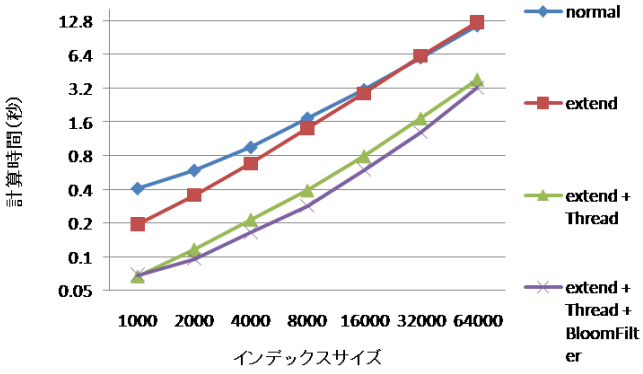

図 4　計算速度の比較

normal は，Baeza-Yates のアルゴリズムをそのまま利用した計算方法の結果を示している．extend は，拡張したアルゴリズムによる結果を示す．また，スレッドを 4 つ用いた場合と Bloom Filter によるフィルタリングを行った場合の結果も合わせて載せている．この実験結果から，インデックスサイズが小さい場合，拡張したアルゴリズムによる計算方法は元のアルゴリズムをそのまま利用する方法に比べて約 52%検索時間を短縮できていることが分かった．さらに，スレッドと Bloom Filter も併用した場合，元のアルゴリズムを単純に利用するだけの計算方法に比べて約 72〜83% 計算時間を短縮できた結果となった．また，インデックスサイズが大きい場合に，元のアルゴリズムと拡張したアルゴリズムとで差異がほとんど見られなくなっている．これは，Baeza-Yates のアルゴリズムでは，リストを分割していく中で，検索元のリストが検索対象のリストよりも大きくなった場合に，二つのリストを交換することによってバイナリサーチの効率を上げているが，拡張したアルゴリズムではこの処理を行っていない点が理由として考えられる．そのため，インデックスサイズが大きい場合に，検索元となるリストに対する走査が一度で済むことによって短縮される処理時間よりも，上記の処理を行わないことによって悪化する処理時間の方がわずかに上回ってしまったと考えられる．

6.4　データベース内の検索処理

本節では，各データベース内の検索処理について述べる．6.3 節で述べた，検索対象データベースの絞り込み処理を行った後，各データベース内の検索処理を行う．まず，各データベースの注釈インデックスから，検索元の注釈を検索し，そのノード情報を取得する．次に，取得したノード情報を，その中に含まれる XML データ ID でグルーピングし，検索結果を XML データ ID ごとに作成する．このようにすることで，7 章で述べる XML データの再構築を可能とする．

7.　検索結果の処理

7.1　類似度による検索結果のランキング

本研究では，注釈に基づいた XML データ間の類似度を計算し，その値を用いて検索結果のランキングを行う．類似度は，包含率と再現率の積として計算する．包含率は，検索対象の XML データが，検索元の XML データの何割を包含しているかを示す．よって「検索対象に含まれる検索元の注釈の数 / 検索元の注釈の数」で計算する．再現率は，検索対象の XML データのうち，何割が検索元の XML データで占められているかを示す．よって「検索対象に含まれる検索元の注釈の数 / 検索対象の注釈の数」で計算する．

7.2　検索結果の表示

本研究では，各検索結果に含まれるタプル ID を元に関係表に問合せを行い，そのノード情報を得る．その後，Dewey order に基づき XML データを再構築する．Dewey order には，各ノードの兄弟親子関係が記されているため，各ノードを組み合せて可能な限り大きな部分木を構築していく．さらに，部分的に再構築された XML データをスニペットとして，XML データ ID とともにランキング順に表示する．

8.　ま　と　め

本稿では，XML データに対するトレーサビリティ機構を実現するため，XML データに含まれる全てのノードに対して注釈と呼ばれるメタデータを付与した新たな XML データモデルを提案した．また，注釈を利用した XML データの検索方法について述べ，実験によって検索速度を検証した．さらに，検索結果のランキング処理と，検索結果から XML データを再構築し表示する処理について述べた．

今後の課題としては，トレーサビリティ機構全体の実装があげられる．

謝　　辞

本研究の一部は，科学研究費補助金特定領域研究 (#19024006)，若手研究 (#19700083) による．

文　　献

[1] World Wide Web consortium: Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml. W3C Recommendation 26 November 2008.

[2] Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew, TanGaurav Vijayvargiya: ”An Annotation Management System for relational Databases” ,VLDB Conference (2004).

[3] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha Nabar, Tomoe Sugihara, Jennifer Widom: ”Trio: A System for Data, Uncertainty, and Lineage” ,VLDB Conference (2006).

[4] Hao Jin, Curtis Dyreson: ”Supporting Proscriptive Metadata in an XML DBMS” ,DEXA (2008).

[5] World Wide Web consortium: XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery. W3C Recommendation 23 January 2007.

[6] M. Yoshikawa, T. Amagasa, T. Shimura and S. Uemura: ”XRel: A path-based approach to storage and retrieval of XML documents using relational databases” , ACM Transactions on Internet Technology (TOIT), 1, 1, pp. 110.141 (2001).

[7] R. Baeza-Yates: ”A Fast Set Intersection Algorithm for Sorted Sequences” ,Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM 2004), Springer LNCS 3109, pp 400-408, Istanbul, Turkey, July 2004.

[8] Bloom, Burton H: ”Space/time trade-offs in hash coding with allowable errors” ,Communications of the ACM 13 (7): 422–426 (1970).