경로 정보 축약 레이블링 기법을 이용한 효율적인 XML 분기 질의 처리[1)]

서세훈° 배진욱 이석호
서울대학교 전기컴퓨터공학부
{ulyss°, jinuk}@db.snu.ac.kr, shlee@snu.ac.kr

Efficient XML Twig Query Processing based on Path-summarized Labeling Schemes

Sehoon Seo°, Jinuk Bae, Sukho Lee
School of Electrical Engineering and Computer Science, Seoul National University

요 약

지금까지 제안된 XML 문서상의 분기 질의(twig query) 처리 기법들의 중요한 흐름 중 하나는 지역 인코딩 기법을 이용하는 것이다. 하지만 이 기법에 기반한 분기 질의 처리는 분기 질의상의 단말 노드와 분기 노드의 엘리먼트를 반드시 읽어야 하는 단점이 있다. 그러나 지역 인코딩 기법과는 달리 경로 정보를 축약하는 방식의 레이블링 기법(예: 듀이 인코딩)은 지역 인코딩에 의한 레이블에 비해 더 많은 정보를 담고 있어서, 이 기법과 구조 인덱스를 이용하여 XML 문서를 인덱싱하면 질의상의 단말 태그의 엘리먼트만을 읽어도 분기 질의를 처리할 수 있다. 이를 이용하여, 본 논문에서는 경로 정보 축약 레이블링 기법과 구조 인덱스를 이용한 분기 질의 처리 기법을 제안한다. 제안된 알고리즘은 디스크 입출력을 줄일 수 있으며 불필요한 중간 결과도 생성하지 않는다.

1. 서 론

최근 XML을 이용한 데이타 저장의 중요성이 점차 증대되고 데이타의 규모가 점차로 커짐에 따라 XML 문서의 인덱싱 및 질의처리가 XML 분야의 핵심적인 연구 분야로 자리잡게 되었다. XML 분기 질의(twig query) 처리 기법의 한가지 흐름은 지역 인코딩(region encoding) 기법[1]을 이용하는 것이다. 이 기법을 이용한 레이블은 작고 간단하지만 담고 있는 정보가 지나치게 적다는 단점이 있다. 이 때문에 지역 인코딩에 기반한 질의 처리에서는 질의상의 분기 태그와 단말 태그에 해당하는 엘리먼트를 모두 읽어야만 한다.

최근에는 지역 인코딩에 비해 더 많은 정보를 포함하고 있는 경로 정보 축약 방식의 레이블링 기법들이 꾸준히 제안되고 있는데, 본 논문에서는 경로 정보 축약 방식의 레이블링[2][3]과 구조 인덱스 기법[4]을 이용하여 질의상의 분기 태그의 엘리먼트를 읽지 않고, 단말 태그의 엘리먼트만 읽는 분기 질의를 처리하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 살펴본 후, 3장에서 문제를 정의하고 기반이 되는 인덱스 구조와 기본적인 원리들을 살펴본다. 구체적인 질의처리 알고리즘은 4장에서 제시하고, 5장에서 결론을 맺는다.

2. 관련연구

2.1. 레이블링 기법

지금까지 제안된 XML 데이타에 대한 분기 질의 처리 기법에는 XML 엘리먼트들에 레이블을 부여하고 이를 이용하여 엘리먼트들 사이의 구조관계를 식별하는 연구가 있다.

레이블링 기법은 크게 두가지 갈래로 나눌 수 있다. 먼저 레이블이 엘리먼트까지의 경로 정보를 축약해 가지고 있지 않은 기법의 대표적인 예는 지역 인코딩 기법[1]으로, [5]는 지역 인코딩 기법을 이용한 최적화된 분기 질의 처리 알고리즘을 제안했다.

다른 한가지 흐름은 레이블에 그 엘리먼트의 경로 정보를 축약해서 담는 방식이다. 대표적으로, 정적인 방식인 듀이 인코딩 기법[2], 동적인 방식인 프리픽스 기반 레이블링 기법[3] 등이 있다. 그 중 듀이 인코딩 기법은 루트 엘리먼트에 레이블 1을 부여하고, 그 이후로는 한 엘리먼트의 n번째 자식에 그 엘리먼트의 레이블 뒤에 .n을 붙이는 방식으로 레이블을 부여한다. 예를 들면 루트 엘리먼트의 두번째 자식은 1.2라는 레이블을 갖고, 다시 이 엘리먼트의 네번째 자식은 1.2.4라는 레이블을 갖는다.

2.2. 구조 인덱스와 경로 정보를 축약하지 않는 레이블링 기법

XML 질의 처리에서 또 다른 이슈는 구조 인덱스(structure index)를 구축하여 질의 처리에 이용하는 것이다[4]. 그러나 구조 인덱스는 경로 질의 처리에는 효과적이나 실제 데이타의 분기정보를 담고 있지 않아 분기 질의를 처리하는 데에는 한계가 있다.

최근에 구조 인덱스와 지역 인코딩 기법을 접목하여 모든 엘리먼트를 읽을 필요없이 질의의 분기 태그의 엘리먼트와 단말 태그의 엘리먼트만 읽어도 정확한 답을 구할 수 있는 기법이 제안되었다[6]. 예를 들어 /A/B[//C/D]//E라는 질의가 주어졌다면, 태그 A, B, C, D, E에 해당하는 모든 엘리먼트를 읽을 필요 없이, 분기 태그 B와 단말 태그 D, E에 속하는 엘리먼트들만 읽어도 질의의 답을 구할 수 있다.

한편 [7]는 구조 인덱스를 이용하여, 구조 인덱스 상의 노드별로 엘리먼트의 데이블을 구성하는 방식을 통해 불필요한 디스크 접근을 최소화하는 기법을 제안하였다.

3. 경로 정보 축약 레이블링을 이용한 질의 처리 기법

이 장에서는 구조 인덱스와 경로 정보 축약 레이블링을 접목하여 분기 노드의 디스크 입출력을 제거한 분기 질의 처리 기법을 제안한다.

3.1. 문제 정의

본 논문에서 해결하고자 하는 문제는 일반적인 XML 분기 질의 처리 문제이다. XML 분기 질의 처리 문제는 다음과 같이 기술할 수 있다.

> 분기 질의 Q와 XML 문서 D가 주어졌을 때, D에 포함된 모든 엘리먼트들 중에서, Q에 나타난 모든 태그에 이름과 구조가 모두 일치하는 엘리먼트의 집합을 매치 집합이라고 한다. 이 때 이 질의의 답은, 구분되는(distinct) 매치 집합에 대해 질의 Q의 단말 태그에 해당하는 엘리먼트 ID를 포함하는 튜플 $t = (e_1, \ldots e_n)$의 집합이다. (여기서 n은 Q에 포함된 단말 태그의 수이다.)

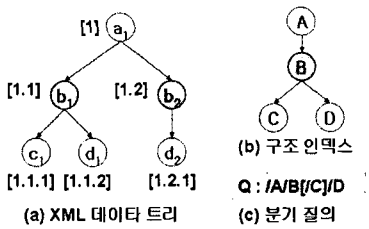
기존 기법들에서 답 튜플은 질의에 나타난 각각의 태그에 대응하는 모든 엘리먼트를 포함하지만 본 논문에서 제안하는 기법은 단말 태그에 대응하는 엘리먼트만을 읽고 질의를 처리하는 것이 목적이므로, 비단말 태그에 대응되는 엘리먼트를 답 튜플 형식에서 배제하였다.

3.2. 인덱스 구조

본 논문에서 제안하는 기법은 [5]에서 채택한 구조 인덱스와 [7]에서 제안한 PPS(prefix-path streaming) 기법을 차용하고 있다.

구조 인덱스는 XML 문서에서 같은 경로를 가지는 엘리먼트드들을 한 노드로 병합하는 방식을 통해 구축한 자료구조이다. 예를 들어 그

1) 본 연구는 2005년도 두뇌한국 21 사업과 정보통신부의 대학 IT 연구센터(ITRC)의 지원을 받아 수행되었습니다.



[그림 1] 예제 데이타

림 1의 (b)는 (a)에 대한 구조 인덱스를 구성한 것이다. 이 예에서 엘리먼트 d_1, d_2는 A/B/D라는 경로를 갖기 때문에 구조 인덱스 노드(이하 SI 노드) 상에서 하나의 노드 D1으로 나타내어진다.

PPS는 XML 엘리먼트들을 구조 인덱스 노드별로 구분하여 테이블을 구성하는 방식이다.

각 테이블(이하 SI 노드 테이블)의 엔트리에는 듀이 레이블만이 저장된다. 각 SI 노드상의 엘리먼트들이 각각의 테이블에 저장되므로, 엘리먼트의 레벨 정보는 따로 저장하지 않아도 무관하다. 또한 테이블 내에서 엘리먼트들은 전위순회 순서대로 저장한다.

따라서 같은 태그 이름을 갖는 엘리먼트들을 하나의 테이블로 구성하는 방식에 비해 테이블이 세분화되어 불필요한 디스크 접근을 줄일 수 있는 이점이 있다.

3.3. 기본 원리

구조 인덱스를 이용하면 같은 경로를 갖는 엘리먼트들을 쉽게 구할 수 있다. 구조 인덱스는 그 정의상, 같은 경로를 갖는 엘리먼트들을 통합해서 가지고 있는 자료구조이기 때문이다. 따라서 주어진 질의에 대해, 질의를 구성하는 각각의 단일 경로를 만족하는 엘리먼트들을 구할 수 있다. 예를 들어 그림 1에서 주어진 질의 Q는 단말 /A/B/C와 /A/B/D라는 두 개의 단일 경로 질의로 나누어진다. 그리고 구조 인덱스를 통해 그림 2(b)의 C와 D가 각각의 단일 경로 질의를 만족시킨다는 것을 알 수 있다. 하지만 C와 D에 속한 모든 엘리먼트의 조합이 Q의 답이 되지는 않는다. 질의 Q의 답이 되기 위해서는, 태그 B에 해당하는 엘리먼트에서 분기해야 하기 때문이다.

구조 인덱스가 데이타의 실제 분기 정보를 담고 있지 않다는 문제는 듀이 레이블을 통해 보완할 수 있다. 듀이 레이블을 통해서 임의의 두 엘리먼트가 공유하는 조상 엘리먼트의 수를 알 수 있기 때문이다. 두 엘리먼트가 공유하는 조상 엘리먼트의 수는 두 엘리먼트 레이블이 공유하는 프리픽스의 단위 수와 같다. 예를 들어 그림 1의 (a)에서, c_1과 c_1은 각각 1.1.1과 1.1.2의 레이블을 가지고 있다. 이 두 레이블의 공통 프리픽스는 1.1이며 따라서 두 엘리먼트는 두 개의 조상 노드를 공유함을 알 수 있다. 같은 이유로 c_1과 d_2는 한 개의 조상 노드만을 공유함을 알 수 있다. 이와 같이 구조 인덱스와 듀이 레이블을 함께 사용함으로써 (c_1, d_1)은 질의 Q의 답이고, (c_1, d_2)는 답이 아니라는 것을 판별하는 것이 가능해진다.

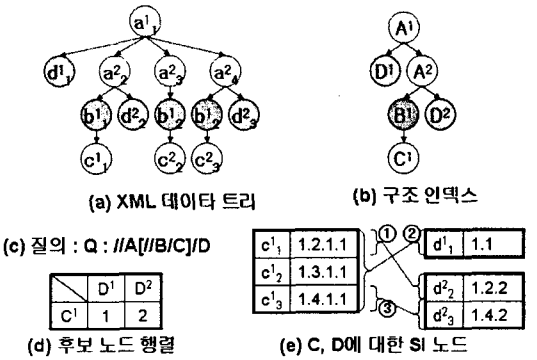
본 논문에서는 이러한 구조 인덱스와 듀이 인코딩을 이용하여 분기 엘리먼트의 테이블을 읽지 않고서도 분기 질의를 처리하는 기법을 제시하고자 한다. 그러나 제시하는 기법은 단지 듀이 인코딩에만 적용되는 기법이 아니라, 레이블에 경로 정보를 축약해서 담고 있고, 레이블을 이용해 임의의 두 엘리먼트간의 전위순회 순서를 판단할 수 있는 모든 레이블링 기법에 일반적으로 적용가능하다.

4. 질의 처리 기법

질의 처리는 크게 두 단계로 나누어진다. 먼저 구조 인덱스를 이용해서 후보 노드 행렬을 구하고, 두번째 단계에서 후보 노드 행렬을 참고하여 각 SI 테이블을 스캔하며 질의의 답을 구한다.

4.1. 후보 노드 행렬 구하기

후보 노드 행렬이란 구조 인덱스에서 어떤 SI 노드가 답이 될 가능성이 있는 후보 엘리먼트를 포함하는 노드이며, 이 엘리먼트들이 답이 되려면 몇 개의 조상 엘리먼트를 공유해야 하는지 등의 정보를 담고 있는 자료구조이다.



[그림 2] 예제 데이타

그림 2(a)의 데이타에 그림 2(c) 질의 Q1이 주어졌다면, 이 경우의 후보 노드 행렬은 그림 2(d)가 된다.

이 행렬의 각 행과 열은 각각 질의의 단말 태그인 C와 D의 구조 인덱스 노드에 대응된다. 이 행렬이 의미하는 바는, C^1과 D^1에 속하는 엘리먼트들은 서로 최소 1개의 조상 노드를 공유해야 질의 Q의 답이 될 수 있으며, 마찬가지로 C^1, D^2의 경우 최소한 2개의 조상 노드를 공유해야 한다는 것이다.

만약 질의에 조상 자손 관계가 포함되어 있고, 동시에 공통 조상 노드들에서 질의의 분기 태그에 대응되는 노드가 여러 번 나타날 때 (nesting)는 그 노드들 중 가장 상위 노드의 깊이가 행렬에 기록된다.

이러한 후보 노드 행렬은 구조 인덱스를 하나의 XML 문서로 보고 [2]의 알고리즘을 수행하면 구할 수 있으며, 구조 인덱스는 통상적으로 XML 문서보다 훨씬 작을 것이므로 수행비용은 그리 크지 않다.

여기서는 편의상 후보 노드 행렬을 테이블 형태로 표현하였으나, 실제 구현 시에는 메모리 공간 절약을 위해 해시 테이블에 저장하는 것이 좋다.

4.2. 테이블 스캔

일반적으로 질의의 각 단말 태그에 대응되는 후보 노드가 여러 개 존재하게 되므로 후보 노드 행렬은 mxn이 된다. 따라서 질의 처리는 m:n 조인이라 할 수 있다(m과 n은 각각 단말 태그에 해당하는 테이블 수). 이를테면 그림 2의 데이타 (a)에 대한 질의 Q의 처리는 1:2 조인이다. 단말 태그 C와 D에 해당하는 후보 노드가 각각 1개, 2개이기 때문이다.(C^1, D^1과 D^2) 그러나 C^1과 D^1 테이블을 메모리로 읽어들이며 조인을 수행하고, 다시 C^1과 D^1을 읽으며 조인을 수행하면 C^1 테이블을 두 번 읽게 되어 디스크 입출력 수가 커지게 된다. 일반적인 m:n 조인에서 각 테이블이 한번에 메모리에 올라올 정도로 작지 않다면 디스크 입출력 회수는 O(m*n)이 된다.

이 문제를 해결하기 위해 이러한 중복된 테이블 읽기를 피하고, 각각의 테이블을 최대 1번만 읽으며 조인을 수행하는 알고리즘을 제안한다.

이 기법은 질의의 답을 엘리먼트의 튜플 형태가 아닌, 구간의 쌍 형태로 저장한다. 그림 2(e)는 구간 쌍 형태의 답을 SI 노드 테이블상에 그림으로 표현한 예이다. 예를 들어 구간의 쌍 (2)는 $[c^1_1, c^1_3]$-$[d^1_1, d^1_1]$으로 나타낼 수 있는데, 이는 엘리먼트 튜플 (c^1_1, d^1_1), (c^1_2, d^1_1), (c^1_3, d^1_1)에 해당한다.

이러한 답 구간 쌍에는 다음 성질이 성립한다.

> 1) 한 SI 노드 테이블의 한 구간은 다른 하나의 SI 노드 테이블에서 두개 이상의 구간에 대응하여 답이 되지 않는다.
> 2) 임의의 두 SI 노드 사이에서 답 구간 쌍은 교차하지 않는다.
> 3) 알고리즘의 각 단계에서 엘리먼트들을 트리상의 전위순회 순서대로 읽어들인다면, 임의의 두 SI 노드 사이에는 매 단계마다 둘 이상의 구간 쌍을 유지하지 않는다.

첫번째 성질은 임의의 두 SI 노드 테이블에서, 답 구간은 항상 1:1 대응이 된다는 의미이다. 그림 2(e)에서 SI 노드 C^1과 D^2의 테이블 사이에는 두개의 답 구간 쌍이 존재한다.(구간 쌍(1)과 (3)) 구간 쌍 (1)의 D2쪽 구간은 엘리먼트 d^2_2에서 끝나는데, 이는 c^1_1은 SI 노드 D^2상의 다른 어떤 구간과도 답이 될 수 없음을 의미한다. (물론 D^2 이외의 다른 SI 노드의 엘리먼트와는 답이 될 수 있다)

이유는 다음과 같다. 하나의 구간 쌍은 하나의 서브트리에 대응된다. 그리고 각각의 구간 쌍에 대응되는 서브트리의 루트는 모두 같은 SI 노드에 속하기 때문에(이 SI 노드의 깊이가 바로 후보 노드 행렬의 해당 셀에 기록된 깊이이다) 각각의 서브트리는 서로 겹치지 않는다. 예를 들어, d^2_2를 포함한 구간 쌍 (1)은 엘리먼트 a^2_2를 루트로 하는 서브트리에 속해 있다. 그리고 D^2에서 d^2_2의 다음 엘리먼트인 d^2_3부터는 a^2_3를 루트로 하는 서브트리에 속하게 된다. 그런데 C^1과 D^2에 속한 엘리먼트는 적어도 서로 2개의 조상을 공유해야 답이 되는데, a^2_2를 루트로 하는 서브트리의 엘리먼트는 a^2_3를 루트로 하는 서브트리의 엘리먼트는 하나의 조상밖에 공유하지 않으므로 절대 답이 될 수 없다. 따라서 c^1_1은 $[d^2_2, d^2_2]$ 이외의 구간과는 답이 될 수 없다.

두번째 성질에서 교차란, 답 구간 쌍 (I_C-I_D)에 대해 I_C에 선행하는 구간과 I_D에 후행하는 구간이 또 다른 답 구간 쌍을 이루는 상황을 말한다. 그림 2의 예에서 $[c^1_1, c^1_1]$이 $[d^2_2, d^2_2]$와 답이 되는데, 만약 이 상황에서 C^1에서 $[c^1_1, c^1_1]$ 이후의 구간이 D^2에서 $[d^2_2, d^2_2]$ 이전의 구간과 답이 된다면 교차가 발생하는 것이다.

임의의 두 SI 노드에 대해 교차가 존재하지 않는다는 것은 하나의 답 구간 쌍이 하나의 서브트리에 대응된다는 것을 생각하면 알 수 있다. 이 성질은 알고리즘 수행시 하나의 구간이 시작되면, 그 구간 이전의 엘리먼트들은 더 이상 메모리에 올려둘 필요가 없음을 의미한다.

세번째 성질은 두 SI 노드 사이에서, 하나의 구간이 열려있는 상태에서 다른 구간이 열리는 일은 발생하지 않는다는 것이다. 이 역시 답 구간 쌍 하나가 서브트리 하나에 대응된다는 것을 통해 알 수 있다. 서로 다른 두 개의 답 구간 쌍에 대응되는 서브트리는 겹치지 않기 때문에 전위순회 순서대로 읽는다면 반드시 한 서브트리의 엘리먼트가 모두 읽힌 후에 다른 서브트리가 시작된다. 따라서 구간 쌍 역시 하나가 닫힌 후에야 다른 구간 쌍이 열린다는 것이 보장된다.

이 세가지 성질에 의해, 알고리즘이 전위순회 순서대로 단말 엘리먼트들 읽어들이며 수행된다면 알고리즘 수행중에 발생하는 모든 답 구간 쌍을 mxn 행렬 하나로 유지하는 것이 가능하다. 행렬의 각 셀에는 그 셀의 행과 열에 대응하는 SI 노드 사이의 답 구간 쌍이 저장된다.

알고리즘의 전반적인 구조는 다음과 같다. 먼저 질의에 관여하는 모든 SI 노드의 엘리먼트를 전위순회 순서대로 읽어들인다. 그리고 이렇게 읽힌 각 엘리먼트(그림 3의 i)마다 그에 대응되는 상대방 후보 노드 각각에서 가장 최근에 읽힌 엘리먼트(그림 3의 e)를 읽는다. i와 e의 특성에 따라 동작은 네가지로 나뉜다.

1) i와 e가 답이 되며, 결과 행렬의 해당 셀이 비어있는 경우 : i와 e를 시작으로 하는 구간 쌍을 연다.
2) i와 e가 답이 되며, 셀에 구간 쌍이 이미 존재하는 경우 : 구간 끝을 수정하여 i와 e를 각 구간에 추가한다.
3) i와 e가 답이 안되고 셀이 비어있는 경우 : 아무것도 하지 않는다.
4) i와 e가 답이 안되고 셀에 구간 쌍이 유지되고 있는 경우 : 현재 유지되는 구간 쌍이 끝났다는 의미이므로, 결과로 출력한다.

모든 엘리먼트에 대해 수행한 후에는, 결과 행렬에 남아 있는 구간 쌍을 답으로 모두 출력하고 알고리즘은 끝난다.

구체적인 알고리즘은 그림 3에 제시하였다. 편의상 두 개의 가지로 분기하는 경우에 알고리즘을 제시하였으나, 이를 조금만 수정하면 일반적인 분기 질의에도 적용 가능하다.

이 알고리즘은 테이블을 SI 노드 단위로 분할하여 결과와 무관한 SI 노드 테이블은 읽지 않고, 또한 전위순회를 통해 한 엘리먼트를 두번 이상 읽지 않기 때문에 디스크 입출력을 최소화할 수 있다. 또한 메모리에는 후보 노드 행렬, 결과 행렬과 답과 관련된 SI 노드 테이블당 1개의 메모리 페이지만이 유지되므로 메모리 공간의 이용도 효율적이다. 구간형태의 답에서 튜플 형태의 답을 구하기 위해서는 구간에 해당하는 엘리먼트들을 알고 있어야 하지만, 일반적으로 이는 매우 작고, 만약 커질 경우 디스크에 저장할 수 있다.

```
Algorithm TwigJoinInterval
// Q : 주어진 질의
// C and D : 질의상의 단말 태그
// SI : 구조 인덱스

 1: CNM := { Q와 SI를 이용해서 구한 결과 행렬 }
 2: for each i := C, D에서 전위순회 순으로
                          읽어들인 엘리먼트
 3: do until C, D의 모든 엘리먼트를 읽음
 4:   INi := i를 포함하고 있는 SI 노드
 5:   N[] := CNM에서 INi와 대응되는 노드 리스트
 6:   for k = 1 to size of N[]  do
 7:     e = N[k]에서 가장 최근에 읽힌 엘리먼트
 8:     INe = e를 포함하는 SI 노드(=N[k])
 9:     if  i와 e의 공유 프리픽스 수>=
          CNM에서 구한 INi와 INe의 최소 깊이 then
10:       if RM(INi, INe)가 비어있음   then
11:         RM(INi, INe).start := (i, e)
12:       end if
13:       RM(INi, INe).end := (i, e)
14:     else
15:       if RM(INi, INe)에 유지되는 구간 있음  then
16:         RM(INi,INe)을 답으로 출력하고 셀을 비움
17:       end if
18:     end if
19:   end for
20: end for
21: 결과 행렬에 남은 구간을 모두 답으로 출력
```

[그림 3] TwigJoinInterval 알고리즘

5. 결론

본 논문에서는 경로 정보 축약 방식의 레이블링 기법에 기반한 분기 질의 처리 기법을 제안하였다. 듀이 인코딩과 같은 경로 정보 축약 방식의 레이블링 기법은 레이블에 지역 인코딩에 비해 많은 정보를 담고 있어서 질의 처리의 최적화에 이용될 수 있다. 뿐만 아니라, 와일드카드를 포함한 질의 처리, 자식 노드 순서와 관련된 질의 처리 등 보다 넓은 범위의 질의를 효율적으로 처리하는 데에도 이용될 수 있으나, 지면관계상 그에 대해서는 언급하지 못하였다.

6. 참고 문헌

[1] Q. Li and B. Moon, "Indexing, Querying XML Data for Path Expressions," VLDB Conference, 2001
[2] I. Tatarinov, et al., "Storing and Querying Ordered XML Using a Relational Database," ACM SIGMOD Conference, 2002
[3] E. Cohen, H. Kaplan, T. Milo, "Labeling dynamic XML trees," ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2002
[4] T. Milo and D.Suciu, "Index Structure for Path Expressions," ICDT, 1999
[5] N. Bruno, N. Koudas, and D.Srivastava, "Holistic Twig Joins: Optimal XML Pattern Matching,"ACM SIGMOD Conference, 2002
[6] R. Kaushik, R. Krishnamurthy, J. F. Naughton, R. Ramakrishnan, "On the integration of structure indexes and inverted lists," ACM SIGMOD Conference, 2004
[7] T. Chen, J. Lu and T.W. Ling, "On Boosting Holism in XML Twig Pattern Matching Using Structural Indexing Techniques," ACM SIGMOD Conference," 2005