XML에서 브랜칭 노드를 이용한 효율적인 Twig Pattern 처리*

류병걸, 박상현, 하종우, 이상근
고려대학교 정보통신대학 컴퓨터통신공학부
e-mail:{smart123, condols, okcomputer, yalphy}@korea.ac.kr

Efficient Processing of Twig Pattern Matching using Branching Node

Byung-Gul Ryu, Sang-Hyun Park, Jong-Woo Ha, SangKeun Lee
Division of Computer and Communication Engineering, Korea University

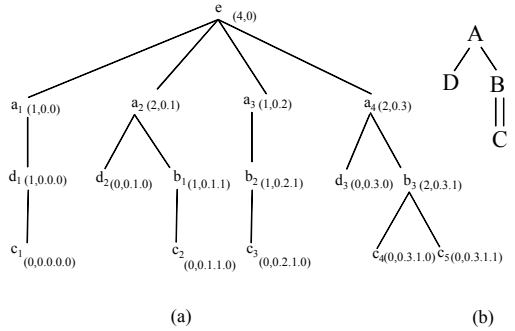
요 약

현재 웹상에서 데이터 표현을 위한 XML의 사용이 늘어나면서, XML 문서의 효율적인 질의 처리에 대한 관심이 증가하고 있다. 기존에 질의 처리 연구는 단일 경로에 대한 질의 처리가 연구되었고, 최근에는 두 개 이상의 경로를 가지는 Twig Pattern 질의 처리 연구가 이루어졌다. 따라서 본 논문에서는 기존에 제안된 기법들 보다 효율적으로 Twig Pattern 질의 처리를 할 수 있는 O-TJFast (*Optimal TJFast*) 기법을 제안한다. 또한, 본 논문에서는 XML 문서의 파싱(parsing)시 제공되는 정보를 가공하여 스트림과 포인터 구조를 얻어내어 기존에 제안된 기법들 보다 입출력 시간(I/O Time)과 처리 시간(Execution Time)을 효율적으로 감소시킬 수 있는 기법을 제안한다. 그리고 성능평가를 통해 제안한 기법이 처리시간에 많은 이득을 볼 수 있음을 보인다.

1. 서론

현재 웹상에서 데이터 표현을 위한 XML의 사용이 늘어나면서, XML 문서의 효율적인 질의 처리에 대한 관심이 증가하고 있다. 기존의 질의 처리 관련 연구는 단일 경로를 가지는 질의 처리 연구를 많이 하였으나 최근에는 처리 시간을 많이 요구하는 두 개 이상의 경로를 가지는 Twig Patten 질의 처리가 최근 들어 중요한 문제로 인식되고 있다. 이러한 Twig Pattern 질의를 효율적으로 처리하기 위해서 많은 레이블링 (labeling) 기법과, Twig Pattern 처리 기법들이 제안되었다 [1, 2, 4, 5].

XML 문서는 일반적으로 일정한 Pattern들의 집합 형태로 나타난다. 이런 Pattern 형식으로 이루어진 XML 문서에서 특정한 Pattern을 찾아내기 위해 일반적으로 사용하는 Twig Pattern 질의는 XPath와 XQuery 이다. 그림 1은 XML 문서와 Twig Pattern 질의를 보여준다.



(그림 1) XML 문서(a)와 Twig Pattern 질의(b)

기존에 제안된 Twig Pattern 질의 처리 기법인 TJFast는 새로운 레이블링 기법을(*Extended Dewey*) 제안했다 [2]. 이 레이블링 기법을 사용하여 TJFast는 단말 엘리먼트만을 읽음으로써 각 엘리먼트가 가지고 있는 정보들을 얻어 올 수 있다. 따라서 기존에 연구되었던 다른 기법들보다 효율적인 질의 처리를 가능하게 하였다. 그러나 TJFast는 질의의 단말노드에 상응되는 모든 엘리먼트들을 읽고 처리해야 하기 때문에 많은 입출력 시간(I/O Time) 및 총 처리 시간(Execution Time)을 갖게 된다. 따라서 본 논문에서는 불필요한 입출력 시간을 줄임으로써 총 처리 시간을 줄일 수 있는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 관련 연구와 본 연구의 동기를 살펴본다. 3장에서는 제안하는 Twig Pattern 처리 기법을 기술한다. 4장에서는 제안하는 기법의 성능평가를 분석하며, 5장에서는 향후 연구에 대해 살펴보고 결론을 내린다.

2. 관련연구

XML이 보급되면서 효율적인 Twig Pattern 질의 처리를 위해 많은 연구가 진행되었다. 초기의 Twig Pattern 질의 처리는 질의에 존재하는 각각의 경로를 분석하고, 여기서 나온 결과를 다시 통합하여 최종 결과를 보여준다 [3]. 하지만 이런 기법은 불필요한 중간 처리 시간이 필요로 하기 때문에 많은 처리 시간을 소요하게 된다. 따라서 Twig Pattern 질의 처리를 위한 새로운 기법이 제안되었다. Holistic Twig Pattern 질의 처리는 기존의 기법보다 입출력 시간(I/O Time)과 처리 시간(Exection Time)을 많이 줄일 수 있었다[1, 2, 5]. 대표적인 방법으로 첫째는, 레이블링을 사용한 기법이 있다. Lu et al. [2]은 레이블링 기법(*Extended Dewey*)을 이용한 TJFaat를 제안했다. 이 기법은 질의 안에 존재하는 단말 노드와 상응되는 엘리먼트만을 읽음으로써 기존의 다른 기법보다 효율적으로 질의를 처리 할 수 있다. 두 번째로는 자료구조를 이용한 방법이 있다. Twig^{2}Stack[4]은 Stack 구조를 이용함으로 처리시간을 줄일 수 있는 기법을 제안하였고, TwigList는 List 구조를 이용함으로써 처리시간을 줄일 수 있는 기법을 제안하였다[5]. 위와 같은 연구들은 XML 데이터 처리를 위한 효과적인 해결책으로 널리 인식되고 있다. 그러나, TJFast는 질의내에 있는 단말노드와 상응되는 모든 엘리먼트들을 읽어 들여야 되고, Twig^{2}Stack과 TwigList

* 이 논문은 서울시 산학협력단(10561)의 지원을 받았음.

의 경우에는 질의와 상응되는 모든 엘리먼트들을 읽어 들여야만 한다. 이것은 입출력 시간(I/O Time)의 증가와 Twig Pattern 질의 처리 시간(Execution Time)에 많은 영향을 미치게 된다. 따라서, 이를 해결하기 위해 불필요한 엘리먼트들을 확인하지 않는 기법이 요구된다.

3. 제안하는 Twig Pattern 처리 기법

TJFast는 불필요한 단말 엘리먼트들을 읽고 처리해야하고, Twig^{2}Stack과 TwigList는 질의에 상응되는 모든 엘리먼트들을 읽어 들여야 하기 때문에 많은 입출력 시간을 갖게 된다. 또한, 입출력 시간은 Twig Pattern 질의를 처리하기 위한 총 처리 시간에 영향을 미치게 된다. 따라서, 이러한 문제점을 극복하기 위해서 본 논문에서는 Twig Pattern 질의 내에 있는 브랜칭 노드를 이용하여 기존의 TJFast를 개선한 방법을 제안한다.

Twig Pattern 질의 처리를 위해 본 논문에서 제안한 스트림 구조는 XML 문서의 전처리(parsing) 과정을 통해 생성된다. 그림 1(a)에 있는 XML 문서에 관한 스트림 구조는 그림 2에서 보여준다.

Set of Streams	
A	(1, 0.0), (2, 0.1), (1, 0.2), (2, 0.3)
B	(1, 0.1.1), (1, 0.2.1), (2, 0.3.1)
C	(0, 0.0.0.0), (0, 0.1.1.0), (0, 0.2.1.0), (0, 0.3.1.0) , (0, 0.3.1.1)
D	(1, 0.0.0), (0, 0.1.0), (0, 0.3.0)
E	(4, 0)

Pointer
a_2 : (B,1,1), (C,2,2), (D,2,2) b_3 : (C,4,5) a_4 : (B,3,3), (C,4,5), (D,3,3)

(그림 2) 그림 1에 관한 스트림 및 포인터 구조

TJFast를 이용하여 Twig Pattern 질의 처리를 고려할 때, TJFast는 그림 1(b)에서 보여준 Twig Pattern 질의 안의 단말 노드인 C와 D에 상응하는 스트림들(stream$_c$, stream$_d$)을 그림 2로부터 가져와서 각 stream 안의 모든 엘리먼트들을 처리함으로써 최종 결과를 얻게 된다. 하지만 이럴 경우 최종 결과와 상관없는 엘리먼트인 "d_1", "c_1", "c_3"를 확인함으로 불필요한 입출력 시간을 갖게 된다. 따라서, 본 논문에서는 이러한 불필요한 엘리먼트들을 처리하는 것을 없애기 위해 첫째, Twig Pattern 질의 안에 있는 노드들 중 자식들을 두 개 이상 가지는 노드인 브랜칭 노드와 그에 상응하는 스트림을 이용하는 방법을 제안한다. 둘째, 자식수가 두 개 이상인 엘리먼트가 가지게 되는 포인터 구조를 이용한다. 셋째, TJFast에서 제안한 레이블링 방법에 각 엘리먼트가 가지는 가지 수를 추가한 레이블링 방법을 제안한다. 각 엘리먼트 가지 수는 질의 안에 있는 브랜칭 노드가 가지는 *부모-자식* 관계의 가지 수와 비교한다.

먼저 본 논문에서 제안한 기법은 Twig Pattern 질의 안에 있는 브랜칭 노드를 확인하고, 브랜칭 노드가 가지는 부모-자식 관계의 수를 확인한다. 그림 1(b)에서 노드 A는 자식의 수가 두 개이기 때문에 브랜칭 노드가 되고, *부모-자식* 관계인 가지 수는 2가 된다. 그 다음 단말 노드 C와 D를 확인하고 그림 2로부터 브랜칭 노드 스트림 stream$_a$와 단말 노드 스트림 stream$_c$와 stream$_d$를 가져 온다. 다음으로 브랜칭 노드 스트림 stream$_a$안에 있는 엘리먼트들의 가지수와 브랜칭 노드가 가지는 *부모-자식* 관계의 가지 수를 비교하여 필요한 브랜칭 엘리먼트를 확인한다. 브랜칭 노드 A가 가진 *부모-자식* 관계의 가지수는 2이므로 stream$_a$의 엘리먼트 중 자식수가 2를 가지는 엘리먼트 a_2와 a_4가 필요한 브랜칭 엘리먼트인 것을 쉽게 확인할 수 있다. 다음으로 단말 노드 C와 D에 상응하는 stream$_c$와 stream$_d$안의 엘리먼트들 중 브랜칭 엘리먼트 a_2와 a_4의 포인터가 가리키는 단말 엘리먼트들("d_2", "d_3", "c_2", "c_4", "c_5")을 확인한다. 마지막으로 질의와 단말 엘리먼트를 비교 처리 함으로 최종 결과("a_2,d_2,b_1,c_2", "a_4,d_3,b_3,c_4", "a_4,d_3,b_3,c_5")를 얻어낸다.

4. 성능 평가

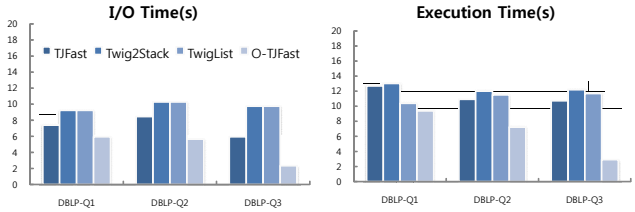
본 논문에서는 기존에 제안된 Twig Pattern 기법인 TJFast, Twig^{2}Stack, TwigList와 본 논문에서 제안한 기법인 O-TJFast (*Optimal-TJFast*)의 성능을 비교하였다. 또한, 성능 측정을 위해 I/O time과 Execution time을 측정하였다. 그림 3에서는 각 측정에 관한 결과를 보여준다.

<표 1> 실험에서 사용한 XML 데이터

	Data size (MB)	Nodes (million)	Max. / Avg. depth
DBLP	130	3.3	6/2.9

<표 2> 실험에서 사용한 질의

Dataset	Twig Query
Q1	//dblp/inproceedings[cite][title]/author
Q2	//dblp/author[title]//year
Q3	//dblp/article[author][title]//booktitle



(그림 3) 측정결과

I/O Time 평가에서 본 논문에서 제안한 방법은 TJFast보다 평균 약 38% 가량 좋아진다. 왜냐하면, O-TJFast는 브랜칭 엘리먼트와 연결된 단말 엘리먼트들만을 읽기 때문이다. Execution Time에 관해서 본 논문에서 제안한 기법은 기존의 제안된 방법보다 I/O Time에서 많은 성취를 이뤘기 때문에 Excution Time에서 또한 약 44%가량 시간이 단축된 것을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 새로운 Twig Pattern 질의 처리 기법을 제안했다. 제안된 기법은 기존의 TJFast에서 불필요하게 처리하던 단말 엘리먼트를 브랜칭 엘리먼트와 그들이 갖는 포인터 구조를 이용해 필터링 함으로써 보다 효율적으로 Twig Pattern 질의를 처리할 수 있다. 실험 평가를 통하여 본 논문에서 기법이 TJFast 보다 효율적으로 Twig Pattern 질의를 처리함을 보였다.

향후 연구로는 보다 많은 불필요한 엘리먼트들을 필터링하는 방법과 Twig Pattern 질의의 처리 시간을 현재보다 향상 시킬 수 있는 기법을 연구할 것이다.

참고문헌

[1] N. Bruno, N. Koudas, and D. Srivastave. Holistic twig joins: optimal xml pattern matching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 310-321, 2002.

[2] J. Lu, T. W. Ling, C.-Y. Chan, and T. Chen. From region encoding to extended dewey: on efficient processing of xml twig pattern matching. In *Proceeding of the 31st International Conference on Very Large Data Bases*, pages 193-204, 2005.

[3] J. McHugh and J. Widom. Query optimization for xml. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 315-326, 1999.

[4] S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig2Stack: bottom-up processing of generalized-tree-pattern queries over xml documents. *In Proceeding of the 32nd International Conferenceon Very Large Data Bases*, pages283-294,2006.

[5] L. Qin, J. X. Yu, and B. Ding. TwigList : Make twig pattern matching fast. *In Proceeding of the 12th International Conferenceon Database Systems for Advanced Applications*, pages850-862,2007.