역 인덱스를 사용한 스트리밍 XML 필터링 기법*

이경한[0] 박 석
서강대학교 컴퓨터학과
{KyounghanLee[0], spark}@dblab.sogang.ac.kr

Streaming XML Filtering Technique using Inverted Index

Kyounghan Lee[0] and Seog Park
Department of Computer Science, Sogang University

요 약

스트리밍 XML 필터링 기법은 사용자가 등록한 질의를 만족하는 XML 문서를 찾아 사용자에게 XML 문서의 복사본을 돌려주는 것을 목적하고 있다. 본 연구는 Xfilter와는 차별된 방법으로 역 인덱스를 사용하여 Xfilter처럼 역 인덱스가 XML 필터링 동안 동적으로 변하는 특성을 제거한다. 또한 늦은 질의 삭제 전략을 이용함으로써 질의 삭제 시간을 50% 이상 줄인다. 따라서 본 기법은 Xfilter에 비해 적은 필터링 시간과 질의 추가/삭제 시간을 보여준다. 또한 역 인덱스를 사용한 기법들의 제한점을 Yfilter와 비교하여 보여준다.

1. 서 론

스트리밍 XML 문서는 실시간으로 빠르게 계속적으로 들어오는 XML 문서다. 온라인 뉴스, 온라인 옥션 환경에서 사용자가 원하는 정보를 얻기 위해 질의를 등록하면 빠르게 정보를 제공받는 것을 목적으로 하는 시스템을 Publish-Subscribe 시스템이라고 한다. 그 시스템의 한 부분으로 스트리밍 XML 필터링 기법은 사용자가 보낸 질의에 대해 매칭되는 XML 문서를 찾아 사용자에게 그 질의 결과가 아닌 XML 복사본을 돌려주는 것을 목적하고 있다.

대표적인 XML 필터링 기법으로 Xfilter[1]와 Yfilter[2]가 제안되었다. Xfilter는 ***경로노드분할***(path node decomposition)이라고 부르는 자료구조를 이용하여 XPath 질의를 유한 상태 기계로 변환한다. 본 연구도 Xfilter에서 사용한 역 인덱스를 차별된 방법으로 사용하여 다수의 질의를 동시에 처리한다. Xfilter의 ***질의 인덱스***(query index)는 XML 필터링 동안 ***동적***으로 변하는 특징을 갖는다. Yfilter는 Xfilter의 문제점을 해결하기 위해 다수 질의의 ***앞부분 공유***(prefix sharing)에 기반한 하향식 비결정적 오토마타를 사용하여 Xfilter에 비해 좋은 처리량을 보여주고 있다.

본 연구는 <질의번호,경로번호> 쌍을 탐색하기 위해 역 인덱스를 이용함으로써 Xfilter처럼 질의 인덱스가 동적으로 변하는 특성을 제거한다. 또한 역 인덱스를 사용한 XML 필터링 기법들의 제한점을 Yfilter와 비교하여 보여준다. 마지막으로 필터링 시간, 질의 추가/삭제 시간을 측정함으로써 본 기법의 특징을 살펴본다.

본 논문의 구성은 다음과 같다. 2장에서는 역 인덱스를 사용한 새로운 XML 필터링 기법의 질의 인덱스 구조와 질의 매칭 알고리즘을 서술한다. 3장에서는 본 기법에서 질의 추가/삭제 알고리즘을 기술하고 4장에서는 본 기법의 성능을 필터링 시간, 질의 추가/삭제 시간 측면에서 평가한다. 마지막으로 5장에서는 본 연구의 결론과 향후 연구 방향을 제시한다.

2. 역 인덱스 필터링 기법

본 장에서는 역 인덱스를 사용한 XML 필터링 기법의 질의 인덱스 구조를 살펴보고 그에 따른 질의 매칭 알고리즘에 대하여 기술한다.

* 본 연구는 한국과학재단 목적기초연구
(R01-2003-000-10385-0) 지원으로 수행되었음

Q_1 = /r/b//d
Q_2 = /r/*/d
Q_3 = /r//c/r/c
Q_4 = //b/c[@a1="v1"]
Q_5 = //b/c[text()=3]
Q_6 = /r/b/[position()>2]
Q_7 = /r[b]//d[e/f[text()=4]/c

Figure 1: XPath queries

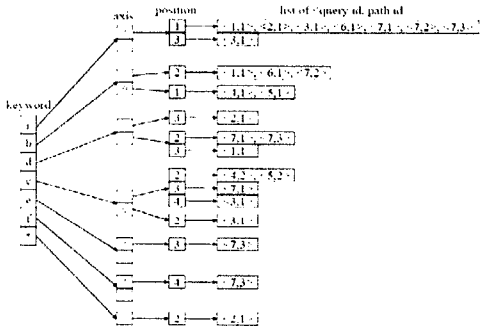
2.1 질의 인덱스 구조

질의 인덱스는 ***질의 역 인덱스***(query inverted index)와 ***질의 테이블***(query table)로 구성되어 있다. 그림 1의 질의에 대해 질의 역 인덱스와 질의 테이블을 생성한 것은 그림 2 (a),(b)와 같다.

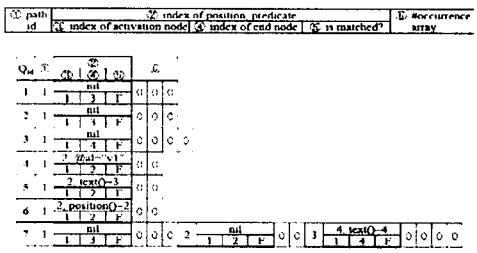
질의 역 인덱스의 ***키워드***(keyword) 부분은 모든 질의에서 나타나는 엘리먼트(element)들과 한 개의 "*" 로 채워진다. 질의처리 동안 질의 역 인덱스의 빠른 탐색을 위해 각 키워드는 ***부모-자식***과 ***조상-자손 축***(axis)으로 구분되어 있고 각 축은 다시 질의에서 그 엘리먼트의 ***위치***(position)로 나누어져 있다. 각 위치에는 <질의번호,경로번호> 쌍들로 이루어진 리스트가 존재한다. 키워드부터 위치까지는 ***동적 해쉬 테이블***을 이용하여 구현되고 <질의번호,경로번호> 쌍들은 ***리스트***로 구현된다.

질의 테이블은 각 질의번호에 대해 분해된 경로질의 배열로 구성된다. 분해된 경로질의는 그것을 ***유일하게 나타내는 번호***(①)와 ***경로에서 얻을 수 있는 술어(predicate)의 정보***(②) 그리고 ***질의 처리 동안 이용되는 보조 필드***들(③, ④, ⑤, ⑥)로 이루어져 있다. 따라서 XML 문서의 엘리먼트가 시작되면 질의 역 인덱스에 의해 <질의번호,경로번호> 쌍을 알 수 있고 그 엘리먼트가 질의에서 어디에 존재하는지 알 수 있다. 또한 <질의번호,경로번호> 쌍은 질의 테이블에서 유일하기 때문에 그 질의의 술어 정보와 보조 필드들을 구할 수 있다.

각 필드가 의미하는 바를 자세히 살펴보면 다음과 같다. ②는 술어 정보를 저장하는 필드로 질의에서 나타난 술어의 위치와 내용을 저장한다. ③은 질의처리를 완료하기 위해 반드시 통과해야 할 활성노드(activation node)의 위치 첨자를 나타내며 초기에는 1로 설정된다. ④는 단말노드(end node)의 위치 첨자를 나타낸다. ⑤는 경로질의가 현재 문서에 매칭이 완료되었는지 나타내는 필드이다. 마지막으로 ⑥은 XPath의



(a) query inverted index



(b) query table

Figure 2: Query inverted index and its query table

position() 함수를 처리하기 위해 각 위치의 엘리먼트가 같은 부모 하에서 발생한 회수를 저장한 배열이다

예제 1 그림 1의 질의 Q_1을 고려하면 엘리먼트 r의 축은 "/" 이고 위치는 1이 된다. 또한 엘리먼트 b의 축은 "/" 이고 위치는 2가 된다. 마지막으로 엘리먼트 d의 축은 "//" 이고 위치는 3이 된다. 이러한 정보는 그대로 질의 역 인덱스에 <1,1>로 저장된다. 또한 질의번호는 1이며 중첩 경로 표현식(nested path expression)이 아니므로 질의를 분해할 필요없다. 따라서 질의 테이블에서 질의번호1이고 경로번호1인 지점에 Q_1의 모든 정보가 저장된다. Q_1은 술어가 없으므로 ②는 nil 값을 갖고 ③은 1을 ④는 3을 저장하게 된다. ⑤는 문서를 처리하기 전이므로 아직 매칭이 완료되지 않았기 때문에 false 값을 갖고 마지막으로 ⑥은 모든 원소가 0 값을 가지고 있는 크기 3인 배열로 설정된다.

예제 2 그림 1의 질의 Q_7을 고려하자. Q_7은 중첩 경로 표현식이므로 3개의 경로로 분해한다. 분해된 경로1은 /r//d/c, 경로2는 /r/b, 경로3은 /r//d/e/f[text()=4]이다. 경로1과 2는 예제 1과 유사하므로 설명을 생략한다. 경로3에서 엘리먼트 r의 축은 "/" 이고 위치는 1이 된다. 엘리먼트 d의 축은 "//" 이고 위치는 2가 된다. 엘리먼트 e의 축은 "/" 이고 위치는 3이 된다. 엘리먼트 f의 축은 "/ "이고 위치는 4가 된다. 위 정보는 그대로 질의 역 인덱스에 <7,3>으로 저장된다. 술어의 경우 위치 4에 존재하므로 질의 테이블에 있는 질의번호7이고 경로번호3인 지점의 ② 필드에는 4,text()=4가 저장된다. 나머지 ③, ④, ⑤, ⑥ 필드는 예제 1에서 설명한 바와 비슷하게 알맞은 값으로 채워진다.

2.2 질의 매칭 알고리즘

질의 매칭 알고리즘은 *SAX 파서*(parser) 인터페이스(interface)를 구현한다.

필터링할 XML 문서가 시작(StartDocumentHandler)되면 이전 XML 문서에 매칭된 질의번호를 저장하고 있는 리스트 자료구조 R을 초기화시킨다.

그 후 현재 필터링하고 있는 XML 문서의 각 엘리먼트가 시작(StartElementHandler)되면 SAX 파서는 그 엘리먼트 이름과 문서에서 깊이(depth)를 반환하기 때문에 이를 이용하여 질의 인덱스로부터 현재 고려해야 할 <질의번호,경로번호> 쌍과 경로질의에서 현재 엘리먼트 이름이 나타나는 위치를 알아낼 수 있다. 질의 역 인덱스를 탐색하는 방법은 그림 3의 *역 인덱스*

```
Input: D is an input XML document.
Output: R is the set of XPath queries that matches D.
Let ε be a element of D;
Let ε_axis be a axis of ε;
Let ε_level be a level of ε;
Let ε_pred be a predicate of ε;
// qid & pid are query id & path id in query table, respectively.
Let <qid,pid> be a posting list at ε, ε_axis, and ε_level;
Let ω be a end node index of <qid,pid>;
Let pred be a predicate of <qid,pid> at activation node index α;

Procedure StartDocumentHandler
  Initialize R to be empty set;

Procedure StartElementHandler
  // P is the set of posting lists of element ε.
  Initialize P to be empty set;
  Initialize <qid,pid> to be null;
  // α is a activation node index of <qid,pid>.
  Initialize α to be null;
  // μ is a IsMatched field of <qid,pid>.
  Initialize μ to be false;
  if(ε_axis=child-parent axis) then
    if(ε_level=1 or ε_level=2) then
      <qid,pid>=InvertedIndex.get(ε, child-parent axis, ε_level);
      Set α and μ from <qid,pid>;
      if(α=ε_level and μ=false) then
        P=P∪<qid,pid>;
      <qid,pid>=InvertedIndex.get("*", child-parent axis, ε_level);
      Set α and μ from <qid,pid>;
      if(α=ε_level and μ=false) then
        P=P∪<qid,pid>;
    else then
      for each level l∈ε_level−{1} do
        <qid,pid>=InvertedIndex.get(ε, child-parent axis, l);
        Set α and μ from <qid,pid>;
        if(α=ε_level and μ=false) then
          P=P∪<qid,pid>;
        <qid,pid>=InvertedIndex.get("*", child-parent axis, l);
        Set α and μ from <qid,pid>;
        if(α=ε_level and μ=false) then
          P=P∪<qid,pid>;
  if(ε_axis=ancestor-descendant axis) then
    for each level l∈ε_level do
    <qid,pid>=InvertedIndex.get(ε, child-parent axis, l);
    Set α and μ from <qid,pid>;
    if(α=ε_level and μ=false) then
      P=P∪<qid,pid>;
  for each <qid,pid>∈P do
    if(pred=ε_pred and ω=α) then
      Set IsMatched field μ to true;
    if(∀ IsMatched field μ of qid=true) then
      R=R∪{qid};

Procedure EndElementHandler
  decrease ε_pred.#occurrences;

Procedure EndDocumentHandler
  return R;
```

Figure 3: Inverted index filtering algorithm

필터링 알고리즘 의사코드에 자세히 적어두었다. ***질의 역 인덱스 탐색으로 현재 필터링 중인 XML 문서에서 각 질의들이 구조적 매칭(structural matching)을 만족하는지 알 수 있고 질의 테이블을 이용하여 술어 조건이 만족하는지 확인한다.*** 만약 구조적 매칭과 술어 조건이 동시에 만족하면 활성노드 첨자를 하나 증가시켜 해당 경로의 다음 활성노드가 처리될 위치임을 저장한다. 이런 방법으로 활성노드 첨자를 증가시키게 되면 결국 그 첨자는 단말노드 첨자와 일치하게 되며 그 경로번호가 XML 문서에 매칭되는 시점이다. 따라서 isMatched 필드는 true값으로 바뀐다. 실제로 질의번호의 매칭을 판단하는 것은 이 후에 처리하는데 이것은 단순히 해당 질의번호가 가지고 있는 경로번호들의 isMatched 필드가 모두 true일 때 (*ANDing 연산*)이다. 만약 질의번호가 현재 XML 문서를 매칭한다면 리스트 *R*에 그 질의번호를 삽입한다.

XML 문서에서 StartElementHandler에 대응되는 엘리먼트의 마지막 태그를 만나게 되면 질의 매칭 알고리즘은 EndElementHandler 프로시저를 호출한다. 이때는 XPath의 position() 함수를 처리하기 위해 해당 엘리먼트에 대한 질의 테이블에서 발생(occurrence) 회수를 줄여준다.

마지막으로 현재 XML 문서의 SAX 파싱이 완료되면 문서가 종료(EndDocumentHandler)된다. 이때는 현재까지 리스트 *R*에 저장된 질의번호를 반환하고 다음 XML 문서를 처리하기 위해 준비한다.

3. 질의 추가와 삭제

본 장에서는 3장에서 설명한 역 인덱스 기반 XML 필터링 시스템에서 어떻게 질의를 추가하고 삭제하는지 알아본다. 또한 질의 삭제에 어떤 이점이 있는지 알아본다.

3.1 질의 추가

한 개의 XML 문서에 대해 SAX 파싱이 모두 완료되거나 시작되기 전에 질의 추가가 가능하다. 질의를 추가하기 위해서는 먼저 질의를 분해(2.1의 예제 3)하고 *주 경로*와 *중첩 경로*를 구한다. 그 후 각 경로의 엘리먼트 정보를 질의 역 인덱스에 삽입하고 각 경로의 술어 정보는 질의 테이블에 삽입한다.

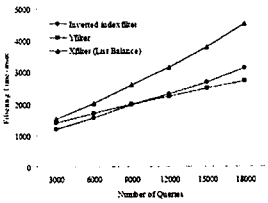
3.2 질의 삭제

질의의 삭제는 질의의 추가와 달리 SAX 파싱 도중에 수행될 수 있다. 그 이유는 질의 테이블에서 삭제할 질의번호에 해당되는 필드를 모두 nil로 설정함으로써 질의의 삭제 효과를 나타낼 수 있기 때문이다. 실제로 질의 역 인덱스에서 삭제된 질의에 해당 부분도 삭제해야 되나 그것은 상당히 비싼 탐색 비용이 필요하다. 따라서 실제 삭제될 <질의번호,경로번호> 쌍을 질의 역 인덱스에서 만나게 될 때 질의 역 인덱스에서 그 쌍을 삭제한다. 이러한 *늦은(lazy) 삭제 전략*은 질의 삭제 시간을 상당히 줄이지만 빈번한 삭제가 이루어질 경우 필터링 시간이 증가되는 단점이 있다.

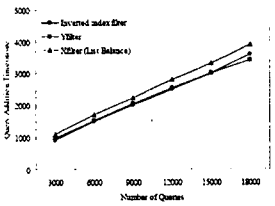
4. 성능 평가

본 연구의 실험은 3.2GHz Intel CPU와 1GB의 메모리를 장착한 Windows 2003 Standard Server에서 실행되었다. 자바 타이거를 이용하여 알고리즘을 구현했고 NITF XML 스키마를 가지고 Yfilter 질의 생성기와 ToXgene을 이용하여 XPath 질의와 100개의 XML 문서를 생성했다. 생성된 XML 문서와 질의의 최대 깊이는 6이며 한 개의 질의의 각 위치 단계에서 "*" 과 "//"이 발생할 확률은 0.2이고 각 질의 당 서술과 중첩 경로의 개수는 2로 설정하였다.

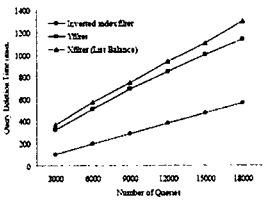
그림 4 (a)의 결과에서 알 수 있듯이 ***필터링 시간은 본 연구에서 제안한 기법이 Xfilter보다 더 적게 걸림을 나타내고 있다.*** 그 이유는 문서 한 개의 필터링이 끝날 때 Xfilter의 경우 다시 경로노드와 질의 인덱스를 초기화 시켜줘야 하나 본 기법의 질의 역 인덱스는 필터링 동안 *정적*인 특성을 갖고 있으며 단지 문서의 필터링이 끝나면 질의 테이블의 isMatched 필드만 모두 false로 초기화하기 때문이다. 그러나 ***Yfilter에 비해 질의 개수가 증가할수록 필터링 시간이 느려진다.*** 그 이유로써 역 인덱스를 사용한 기법들은 엘리먼트가 시작할 때 항상 엘리먼트 이름과 "*" 키워드를 질의 역 인덱스에서 탐색해야 하고 XML 문서에서 엘리먼트의 깊이가 커질수록 질의 역 인덱스를 탐색하는 범위가 넓어지기 때문이다.



(a) filtering time



(b) query addition time



(c) query deletion time

Figure 4: Filtering, query addition / deletion time, Varying Q

이러한 단점은 질의 추가시간(그림 4 (b))에서도 나타난다. 부언하자면 Yfilter의 경우 질의개수가 커질수록 공유되는 상태가 많아지나 역 인덱스를 사용하면 공유할 수 있는 부분이 사라지기 때문에 질의 개수가 증가함에 따라 질의추가 시간이 짧아진다.

마지막으로 ***질의 삭제시간(그림 4 (c))은 다른 기법보다 50% 이상 적게 걸린다.*** 그 이유는 3.2절에서도 설명했듯이 본 기법은 단순히 질의 테이블의 해당 질의번호의 내용을 nil로 설정해버리고 질의 역 인덱스 부분은 필터링 동안 늦게 삭제되기 때문이다.

5. 결론 및 향후 연구

본 논문에서 우리는 Xfilter와는 다른 역 인덱스를 사용한 XML 필터링 기법을 제안했다. 본 기법은 Xfilter보다 필터링 시간, 질의 추가/삭제 시간에서 우수한 성능을 나타낸다. 그러나 역 인덱스의 구조적 한계로 인하여 질의 개수가 증가함에 따라 필터링 시간과 질의 추가 시간이 Yfilter에 미치지 못한다. 그러나 질의 삭제의 경우 질의 역 인덱스 자료구조의 내용을 늦게 삭제하는 전략을 채택함으로써 기존의 기법들보다 질의 삭제 시 두 배 이상 높은 성능 향상을 이끌어 냈다.

추후 연구로써 스트리밍 XML 질의처리에서 논의되고 있는 조인 기법을 변형하여 본 기법에서 채택한 질의 언어인 XPath를 XQuery로 확장하는 연구와 Yfilter의 특징인 오토마타의 상태 공유를 이용할 수 있는 변형된 알고리즘을 가진 필터링 기법을 연구할 예정이다.

참고 문헌

[1] M. Altinel and M. J. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information," In proc. of VLDB Conf., pp. 53-64, 2000.

[2] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer, "Path Sharing and Predicate Evaluation for High-Performance XML Filtering," ACM TODS, Vol.28, No.4, pp. 467-516, 2003.