XML 캐쉬의 점진적 갱신을 위한 XML 변경 처리 모델†

한승철 황대현 강현철
중앙대학교 컴퓨터공학부
{schan, dhhwang}@dblab.cse.cau.ac.kr hckang@cau.ac.kr

Models of XML Update Processing for Refreshing XML Cache Incrementally

Seungchul Han Dae Hyun Hwang Hyunchul Kang
School of Computer Science and Engineering, Chung-Ang University

요 약

XML이 웹 상에서 데이터 교환의 표준으로 부각된 이래 XML 데이터의 효율적 관리 기법에 관한 연구가 활발히 수행되고 있다. XML 질의의 표준화 작업도 활발히 이루어져 현재 XQuery가 유력한 표준으로 부각되었다. 그러나 XQuery 등이 완전한 XML 질의어가 되기 위해서는 변경 연산을 제공해야 하는데 XML 변경어의 표준화 작업이나 XML 변경 처리 기법에 대한 연구는 아직 미미한 실정이다. 본 논문에서는 e-Commerce 등 XML 데이터베이스 기반 웹 응용의 효율적 지원을 위한 XML 캐쉬를 점진적으로 갱신하는 과정에서 발생하는 XML 변경 연산 처리의 세가지 기본 모델, TD (텍스트/DOM 기반), PD (PDOM 기반), 그리고 IT (인덱스된 텍스트 기반)를 제안한다. 캐쉬된 XML 문서를 어떤 포맷으로 저장하는가는 이후 XML 변경 연산의 처리 및 요청된 XML 문서 반환의 효율성에 영향을 미친다. 이들 모델들은 캐쉬된 XML 문서의 저장 포맷에 따라 구분한 것이다. 이들 모델 각각에 대하여 캐쉬된 XML 문서의 갱신 및 반환 기능을 제공하는 XML 캐쉬 관리 시스템을 구현하여 이들 기능의 성능 및 공간 부담을 평가한 결과를 기술한다.

1. 서론

XML이 웹 상에서 데이터 교환의 표준으로 부각된 이래 XML 데이터 관리 기법이 활발히 연구되고 있다. e-Commerce 등 XML 데이터베이스 기반의 웹 응용(XML Database-backed web application)은 웹 상에 산재해 있는 XML 소스로부터 자주 접근하는 부분을 캐쉬하여 성능 향상을 기할 수 있다 [1-5]. 예를 들어, 그림 1의 XML 캐쉬는 XML 데이터베이스 기반의 웹 응용을 지원하기 위한 웹 서버, 응용 서버, 데이터 서버로 구성된 다층 구조(multi-tiered architecture)에서 중간층 캐싱(middle-tier caching)을 나타낸 것인데, 웹 페이지에 포함된 XML 질의 중 자주 제기되는 것에 대해서 그 결과를 응용 서버에 캐쉬해 두기 위한 것이다.

질의 결과를 캐쉬한 후 동일 질의 또는 관련된 후속 질의의 처리에 재사용하기 위한 핵심 요소 기술 중의 하나는 질의의 하부 소스 데이터에 발생한 변경을 캐쉬에 점진적으로 반영하여 갱신(incrementally refresh)함으로써 캐쉬의 일관성을 유지하는 것이다. 이들 문제는 90년대 후반부터 반구조적(semistructured) 데이터 및 XML 데이터에 대해서 연구되고 있다 [6-8][1][2].

본 논문에서는 XML 캐쉬를 점진적으로 갱신하는 과정에서 발생하는 XML 변경 연산의 처리 문제를 다룬다. 98년에 W3C의 XML 사양이 발표된 이래 XML 질의의 표준화 작업이 활발히 이루어져 현재 W3C의 XQuery가 유력한 표준으로 부각되었다. 또한 XML 데이터의 저장 및 질의 처리 기법에 대해서도 많은 연구가 수행되어 오고 있다. 그러나 XQuery 등이 완전한 XML 질의어가 되기 위해서는 변경 연산을 제공해야 하는데 XML 변경어의 표준화 작업이나 XML 변경 처리 기법에 대한 연구는 아직 미미한 실정이다. [9]에서는 XQuery의 구문을 확장하여 다양한 XML 변경 연산을 제시하고, XML 데이터가 관계 데이터베이스에 분할 저장되어 있을 경우 제시된 변경 연산의 처리 문제를 다루었다. XML:DB의 XUpdate Working Group에서는 XML 변경어인 XUpdate 사양 [10]의 표준화 작업을 수행하고 있다.

e-Commerce 등 XML 데이터베이스 응용의 완전한 지원을 위해서는 문서 단위의 삽입 및 삭제 외에 문서 내 텍스트(PCDATA) 또는 속성값의 변경이나 엘리먼트 단위의 구조 변경 기능은 필수적이며, 이들을 구현하기 위한 여러 기법들(implementation alternatives)에 대한 연구와 그들 간의 성능 트레이드오프를 검토하는 작업은 중요하다. 특히, XML 문서에 대한 대단위의 변경보다는 단말(leaf) 엘리먼트 단위의 변경과 같은 소규모의 변경을 효율적으로 수행하는 기법이 빈도나 실용성 면에서 더 중요하다.

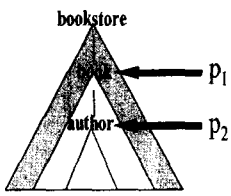

그림 1. XML 데이터베이스 기반 웹 응용을 위한 다층 구조

본 논문에서는 질의 처리 결과 등으로 캐쉬된 XML 문서를 점진적으로 갱신하는 과정에서 발생하는 엘리먼트의 삽입 및 삭제 그리고 엘리먼트의 텍스트(PCDATA) 값 수정 연산의 처리를 다룬다. 이들 XML 변경은 가장 핵심적인 변경 연산들로서 모두 [9]에서 제안된 XML 변경에 포함되며 XUpdate 사양에도 명시되어 있는 것들이다.

캐쉬된 XML 문서의 특정 일부분을 변경하는 데 가장 효율적인 저장 기법은 문서를 파싱한 후 문서를 구성하는 여러 요소들을 분할 저장하는 것이다. 예를 들어, [11] 등에서 많이 연구된 것과 같이 관계 데이터베이스에 XML 문서를 분할 저장하는 기법을 생각할 수 있다. 그러나 본 논문에서는 캐쉬된 XML 문서의 비분할 저장 경우만을 고려한다. 비분할 저장은 저장 포맷이 compact하므로 공간 사용 효율이 높다. 따라서 가용 공간 내에서 보다 많은 수의 질의 결과를 캐쉬할 수 있는 이점을 가지는데 이는 웹 응용의 효율적 지원을 위한 캐쉬라는 점에서 확장성(scalability)을 갖추는 데 유리하다. 그리고 캐쉬 결과를 반환할 때의 효율성은 분할 저장의 경우보다 높을 뿐 아니라 최적이거나 최적에 가깝다. 또한, 캐쉬의 갱신 시에 변경되는 부분이 전체 문서의 소량 부분에 한정된다고 할 때 DBMS의 사용 등 이를 위한 분할 저장의 부담이 정당화되지 못할 수 있다는 점에서 비분할 저장의 고려는 의미가 있다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문의 XML 캐쉬 및 캐쉬의 점진적 갱신 요청 모델을 기술한다. 3절에서는 XML 변경 처리 모델을 기술한다. 4절에서는 이들의 구현과 성능 평가 결과를 기술한다. 마지막으로 5절에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. XML 캐쉬 및 캐쉬 갱신 요청 모델

† 본 논문은 한국과학재단 특정기초연구사업(R01-2003-000-10395-0) 지원으로 수행 되었음.

본 논문에서는 XML 질의 캐쉬(query caching)를 고려한다. 즉, 그림 1의 웹 응용을 위한 다층 구조에서 데이터 서버에 저장되어 있는 복수개의 XML 문서들에 대하여 응용 서버로부터 XQuery와 같은 XML 질의어에 의한 질의가 제기되었을 때 그 결과 XML 문서를 응용 서버에 캐쉬한다.
캐쉬되는 XML 문서는 그림 2의 형식을 가진다고 가정한다. 엘리먼트 <cache>를 루트로 하고 그 아래에 <qdocument> 서브 엘리먼트가 n개 존재한다. 여기서 n은 XML 질의의 소스 XML 문서 m개 중 질의의 조건을 만족하는 것의 개수이다 ($n \le m$). 즉, XML 질의 결과로 검색되는 소스 문서 각각에 대하여 <qdocument> 엘리먼트가 생기며 그것의 서브 엘리먼트들의 형식은 예를 들어 XQuery의 RETRUN 절이 규정하는 형식에 의한다. <qdocument> 엘리먼트의 did 속성은 해당 소스 XML 문서의 식별자이다. 캐쉬되는 질의 결과에 did 속성을 유지하는 이유는, 캐쉬 이후 데이터 서버에서 해당 문서에 변경이 발생했을 경우 이를 캐쉬에 반영하여 점진적으로 갱신하고자 할 때 갱신 대상 <qdocument> 엘리먼트를 찾기 위해서이다.
캐쉬 갱신을 위한 변경 요청 때 명시되어야 하는 정보에는 (1) 변경이 발생한 소스 XML 문서의 식별자(did), (2) 그것의 해당 <qdocument> 엘리먼트 내의 변경 대상 목적(target) 엘리먼트를 지정하기 위한 경로 및 조건, 그리고 (3) 변경에 필요한 데이터 (예를 들어, 엘리먼트 삽입의 경우 삽입될 엘리먼트가 주어져야 하고, PCDATA 수정의 경우 새 값이 주어져야 한다.) 등이 있다.

3. XML 변경 처리 모델

그림 1의 XML 데이터베이스 기반 웹 응용을 위한 응용 서버에서 중간층 캐쉬를 제공하기 위한 XML 캐쉬 관리 시스템의 API는 두 기능으로 대별된다. 첫째는, 캐쉬된 XML 문서의 검색을 요청 받아 그것을 XML 화일 (텍스트 화일)로 반환하는 것이다. 이때 캐쉬의 갱신을 위해 유지하고 있던 did 속성은 반환 전에 모두 제거된다. 질의 결과로 요구되는 XML 문서의 형식이 <qdocument> 태그를 포함하지 않을 경우 did 속성 제거와 더불어 <qdocument>와 </qdocument> 태그 쌍도 제거할 수 있다. 예를 들어, eXcelon의 경우 복수개 XML 문서에 대한 질의 결과의 디폴트 형식은 검색된 소스 문서 간의 구분이 없는 것이다. 이상의 작업은 XML 문서 검색 모듈이 담당한다. 둘째는, 캐쉬의 갱신을 위한 XML 변경을 요청 받아 해당 XML 문서를 변경하는 것으로 이는 XML 문서 변경 모듈이 담당한다. 캐쉬된 XML 문서의 저장 포맷에 따라 다음과 같이 세 가지의 XML 변경 처리 모델을 생각할 수 있다.

3.1. 모델-TD

그림 2의 형식을 가진 캐쉬된 XML 문서는 평범한(plain) 텍스트 화일로 저장된다. 따라서 캐쉬될 XML 문서가 텍스트 화일로 주어질 경우 이를 캐쉬 저장소에 저장하는 데 있어 아무런 처리 과정이 필요 없다는 장점이 있다. 그러나 캐쉬 갱신을 위한 XML 변경 연산을 처리하기 위해서는 해당 문서를 DOM 파서를 통해 메모리 내에 DOM 트리로 생성하여 적재한 후 작업한다. 따라서 가용 메모리 제약에 따라 변경되는 문서 크기에 한계가 있는 단점이 있다.
문서 변경 모듈은, (1) 변경할 XML 문서를 DOM 파서를 통해 DOM 트리로 변환한 후, (2) 주어진 did 속성 값으로 요청된 XML 변경 연산을 적용할 <qdocument> 엘리먼트 내의 노드를 찾는다. (3) 적절한 DOM API 함수를 호출하여 요청된 변경을 수행하고, 마지막으로 (4) 변경된 DOM 트리를 다시 XML 텍스트 화일로 복원하여 저장한다. 문서 검색 모듈은 요청된 문서를 반환하기 위해, (1) 해당 XML 텍스트 화일을 읽어 DOM 트리로 구성한 후, (2) 문서 내의 did 속성을 모두 제거한다. (3) 이후 그 결과 DOM 트리를 XML 문서 (텍스트 화일)로 재구성하여 반환한다.

3.2. 모델-PD

캐쉬된 XML 문서는 파싱된 후 이진(binary) PDOM [12] 화일로 변환되어 저장된다. PDOM 화일은 DOM 트리를 이진 화일 형태로 저장하므로 변경될 XML 문서를 다시 파싱하여 DOM 트리로 구성하는 부담은 없다. 또한 문서 전체를 메모리에 DOM 트리로 적재하는 것이 아니라 할당된 메모리 크기만큼만 적재한 후 그 부분에 해당되는 DOM 트리 부분만의 처리가 가능하므로 [13] 모델-TD가 갖는 가용 메모리 크기에 따른 처리 가능한 문서 크기의 제약에서 벗어날 수 있다. 따라서 문서 전체 중 일부분만 변경하는 캐쉬의 점진적 갱신과 같은 연산에 아주 효율적이다. 그러나 문서 반환을 위해 did 속성을 모두 제거하는 작업과 같이 문서 전체를 대상으로 변경을 수행하는 경우에는 할당된 메모리가 적다면 현재 메모리 내에서 변경된 부분을 PDOM 화일로 저장한 후 그 이후 부분을 메모리에 적재해야 하므로 화일 입출력 부담이 가중될 수 있다. 따라서 가용 메모리보다 큰 문서의 경우에는 모델-TD에서와 같이 처리가 원천적으로 불가능한 것은 아니지만 비효율적일 수 있다.

```
<cache>
  <qdocument DID="1">
   ...
  </qdocument>
  <qdocument DID="5">
   ...
  </qdocument>
....
</cache>
```

그림 2. XML 질의 결과 캐쉬 형식

PDOM의 API도 기본적으로 DOM API이므로 캐쉬 내 XML 문서를 변경하는 것이나 did 속성을 제거한 후 반환하는 과정은 PDOM 화일을 메모리에 DOM 트리로 적재하는 부분 이외에는 모델-TD의 과정과 동일하다.

3.3. 모델-IT

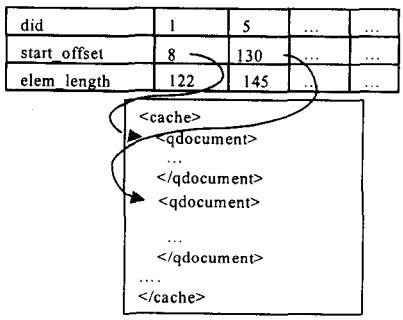

그림 3. 인덱스된 텍스트 화일의 구조

캐쉬된 XML 문서는 파싱된 후 그림 3과 같은 구조의 텍스트 화일로 저장된다. 즉, 그림 2의 XML 형식에서 did 속성을 모두 제거한 XML 문서 화일 (텍스트 화일) f와 f에 대한 인덱스 화일 i로 변환되어 저장된다. 화일 i에는 f 내 <qdocument> 엘리먼트 각각에 대하여 (1) 해당 소스 XML 문서의 식별자(did), (2) f의 시작부터 <qdocument> 엘리먼트 시작 위치까지의 바이트 오프셋 값, 그리고 (3) <qdocument> 엘리먼트의 바이트 길이로 구성된 인덱스 엔트리가 저장된다.
문서 변경 요청이 들어오면 문서 변경 모듈은, (1) 인덱스 화일 i를 읽어 해당 did 엔트리를 검색한 후 시작 오프셋과 길이 값으로 전체 문서 중 요청된 변경이 수행될 <qdocumet> 엘리먼트만 메모리로 적재한 후, (2) 이를 DOM 트리로 변환한다. (3) 모델-TD에서처럼 생성된 DOM 트리를 대상으로 변경 연산을 수행하고, (4) 그 결과를 XML 문서 화일 f에 반영한다. (5) 아울러 그에 따라 영향을 받는 인덱스 화일 i의 엔트리들을 모두 갱신한다. 문서 검색 모듈의 작업은 세 모델 중 가장 간단하다. 모델-TD와 모델-PD에서 캐쉬의 점진적 갱신을 위해 유지하고 있는 did 속성 값이 인덱스 화일 i에서 관리되므로 화일 f에 대해서는 아무런 부가적인 연산 없이 그대로 반환이 가능하기 때문이다.

4. 구현 및 성능평가

XML 캐쉬 관리 시스템을 세가지 모델에 대하여 Java로 Windows 2000 Server에서 구현하였다. DOM 파서로는 Apache Xerces Java Parser 1.4.3. Release [14]와 JDOM [15] 을 각각 모델-TD와 iT에 대해서 사용하였고, 모델-PD에 대해서는 GMD-IPSI XQL Engine Version 1.0.2 의 PDOM [13] 을 이용하였다. 성능 실험에 사용된 XML 데이터 즉, 캐쉬된 XML 문서로는 TPC-W의 order.xml 문서 2,000개에 대하여 각 문서로부터 order/order_lines 엘리먼트를 모두 검색하는 질의를 처리한 결과로서 그림 2의 형식을 갖는 XML 문서를 사용하였다. 이들 캐쉬된 XML 문서의 크기로는 100KB부터 1MB까지 100KB씩 증가시킨 것과 2MB, 그리고 4MB를 고려하였다. 실험은 Pentium IV 1.8GHz CPU와 256MB 메모리를

장착한 시스템에서 Java 프로그램의 가상 메모리 최대값을 500MB로, 그리고 모델-PD의 경우 PDOM 화일 처리에 소요되는 캐쉬 메모리를 문서 변경 시와 반환 시에 각각 100페이지와 3,000페이지로 설정하고 [13] 수행하였다. 평가 결과를 요약하면 다음과 같다 (그림 4-6 참조).

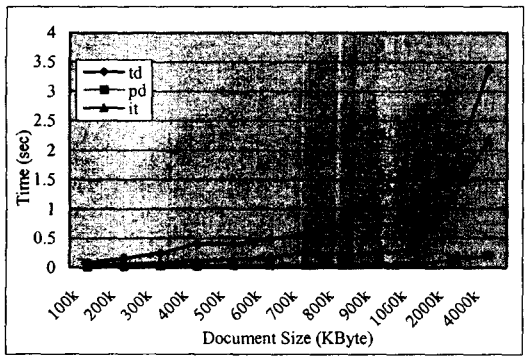

그림 4. 단말 엘리먼트의 삽입 시간

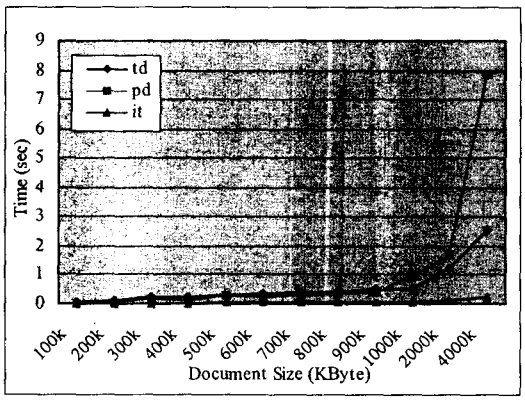

그림 5. 캐쉬된 XML 문서 반환 시간

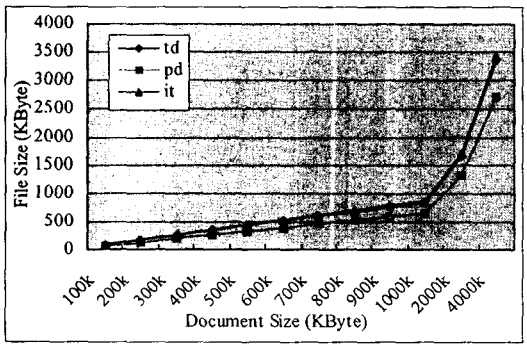

그림 6. 저장 공간

DOM을 사용하는 모델-TD의 경우, 캐쉬된 XML 문서 일부분의 변경을 위하여 문서 전체를 읽어 DOM 트리로 생성하는 비효율성이 크게 작용하는 것으로 나타났다. 따라서 캐쉬된 문서의 크기가 작을 때 이외에는 효용성이 떨어졌다. PDOM을 사용하는 모델-PD의 경우, 변경될 부분만 파싱 부담 없이 메모리로 적재하여 DOM 트리 조작이 가능하므로 변경 연산 처리에서 가장 좋은 성능을 보였으며 이진 PDOM 화일을 사용하므로 공간 효율이 가장 우수했다. 그러나 문서 반환 처리에서는, 캐쉬된 문서의 점진적 갱신을 위해 유지되는 did 속성의 제거 부담 때문에 did 속성 정보를 별도의 인덱스 화일로 분리하여 유지하고 있는 모델-IT에 비해 낮은 성능을 보였고 반환될 문서의 크기가 커질수록 그 성능 차이가 커졌다. 모델-IT의 경우, 변경 처리에서는 세 모델 중 중간 정도의 성능을 보였지만 반환 처리에서는 부가적인 연산이 필요하지 않기 때문에 최적의 성능을 보였다. 대신 공간 부담이 가장 높았다.

5. 결론 및 향후연구

본 논문에서는 XML 데이터베이스 기반의 웹 응용 등의 효율적 지원을 위한 XML 캐쉬를 점진적으로 갱신하는 과정에서 발생하는 XML 변경 연산의 처리의 기본 모델 세가지를 제안하고 구현한 후 그 성능을 비교, 평가하였다.

향후 연구 과제는 다음과 같다. 첫째, 모델-PD와 모델-IT를 조합한 향상된 모델을 사용하여 문서의 변경과 반환 처리 모두에서 좋은 성능을 보이고 아울러 공간 효율도 높은 기법을 제시하는 것이다. 둘째, 캐쉬된 XML 문서를 분할 저장하는 경우와의 성능 비교도 필요하다.

참고문헌

[1] L. Chen and E. Rundensteiner, "Aggregate Path Index for Incremental Web View Maintenance," Proc. 2nd Int'l Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, 2000.

[2] L. Quan et al., "Argos: Efficient Refresh in an XQL-Based Web Caching System," Proc. Workshop on the Web and Databases, 2000, pp. 23-28.

[3] V. Hristidis and M. Petropoulos, "Semantic Caching of XML Databases," Proc. Workshop on the Web and Databases, 2002.

[4] L. Chen and E. Rundensteiner, "ACE-XQ: A CachE-aware XQuery Answering System," Proc. Workshop on the Web and Databases, 2002.

[5] P. Marron and G. Lausen, "Efficient Cache Answerability for XPath Queries," Proc. the 2nd Int'l Workshop on Data Interation over the Web(DIWeb), 2002, pp. 35-45.

[6] S. Abiteboul et al., "Incremental Maintenance for Materialized Views over Semistructured Data," Proc. Int'l Conf. on VLDB, 1998, pp. 38-49.

[7] D. Suciu, "Query Decomposition and View Maintenance for Query Languages for Unstructured Data," Proc. Int'l Conf. on VLDB, 1996, pp. 227-238.

[8] Y. Zhuge and H. Garcia-Molina, "Graph Structured Views and Their Incremental Maintenance," Proc. Int'l Conf. on Data Engineering, 1998, pp. 116-125.

[9] I. Tatarinov et al., "Updating XML," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2001, pp. 413-424.

[10] http://www.xmldb.org/xupdate/

[11] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. Int'l Conf. on VLDB, 1999.

[12] G. Huck, I. Macherius, and P. Fankhauser, "PDOM: Lightweight Persistency Support for the Document Object Model," Proc. OOPSLA Workshop "Java and Databases: Persistence Options", November 1999.

[13] G. Huck, and I. Macherius, "GMD-IPSI XQL Engine" http://xml.darmstadt.gmd.de/xql/.

[14] http://xml.apache.org/xerces-j/

[15] http://jdom.org

[16] http://db.uwaterloo.ca/~ddbms/projects/xbench/

[17] http://www.cs.toronto.edu/tox/toxgene/