제23회 한국정보처리학회 춘계학술발표대회 논문집 제12권 제1호 (2005. 5)

시맨틱 XML 질의 캐쉬의 교체 기법†

홍정우, 강현철
중앙대학교 컴퓨터공학부
e-mail : jwhong@dblab.cse.cau.ac.kr, hckang@cau.ac.kr

A Scheme of Semantic XML Query Cache Replacement

Jung-Woo Hong, Hyunchul Kang
School of Computer Science and Engineering, Chung-Ang University

요 약

웹 상에서 XML 로 기술된 데이터가 증가하고, 이를 이용하여 의미 있는 데이터를 검색하는 것의 중요성이 커지고 있다. 웹 상에서 더 좋은 검색 성능을 보이기 위해 XML 질의 결과를 캐쉬하는 방법에 관한 연구들과 캐쉬의 저장 공간과 다양한 질의를 캐쉬에 저장하는 것에 한계가 있기 때문에 캐쉬 교체 기법에 관한 연구들이 있었다. 기존의 XML 캐쉬 교체 정책에는 질의 결과를 교체 단위로 하는 방법과 질의 결과 내의 각 경로들을 교체 단위로 하는 방법이 있는데, 첫번째 방법은 효율이 상대적으로 낮고 두번째 방법은 높은 효율에 비해 교체 연산을 수행하는 부담(overhead)이 크다는 단점이 있었다. 본 논문에서는 위 두 방법의 단점을 해결하기 위해 2 단계로 교체 희생자를 선택하는 방법을 제시한다. 질의 결과들 중에서 교체 희생자를 찾고, 그 희생자 내의 모든 경로들 중에서 다시 교체 희생자를 찾는다. 이는 각 질의 내의 경로가 교체 희생자가 되어 캐쉬 효율을 향상시키고, 질의 결과에 대해 먼저 교체 대상을 찾으므로 교체 희생자를 찾기 위한 연산을 수행하는 부담을 줄인다. 또한 캐쉬 적중률, 최근 접근 시간, 인출 지연 시간, 객체 크기를 고려하여 교체 희생자를 선택하는 교체 함수를 제시한다. 가상의 시맨틱 데이터에 대한 캐쉬 교체 시스템을 구현하여 본 논문에서 제시한 교체 기법과 교체 함수를 평가한 결과를 기술한다.

1. 서론

웹 상에서 XML 로 기술된 데이터가 증가하고, 이를 이용하여 의미 있는 데이터를 검색하는 것의 중요성이 커지고 있다. XML 데이터를 관리하기 위해 많은 연구가 수행되었다. 웹 상에서 더 좋은 질의 처리 성능을 보이기 위해 XML 질의 결과를 캐쉬에 저장하는 방법에 관한 연구들이 수행되어왔다. XCacher[1]에서는 메인 메모리에 캐쉬되는 내용을 저장하였고, ACE-XQ[2]에서는 XML 문서 형식으로 디스크에 저장하였다.

그러나 캐쉬는 가용 저장 공간보다 많은 수의 XML 질의 결과를 저장할 수 없고, 요청되는 XML 질의의 종류도 여러 가지 이다. 또한 캐쉬가 많은 양의 XML 질의 결과를 가지고 있을 때 캐쉬의 질의 처리 속도가 감소한다. 이와 같은 문제를 해결하기 위해 캐쉬 교체가 필요하다. 즉, 캐쉬의 XML 질의 결과를 적절하게 교체하여 준다면 질의 처리 성능을 향상시킬 수 있다. XCacher 에서는 캐쉬 내용의 정의에서 하부 트리를 교체 단위로 한다. 그리고 캐쉬 교체 희생자 선택을 위한 교체 함수로는 LRU 방법을 사용하였다. ACE-XQ 시스템의 캐쉬 교체 기법[3][4]에서는 XQuery 결과가 가질 수 있는 모든 경로를 교체단위로 한다. 그리고 각 경로의 적중률, 인출 지연 시간, 객체 크기, XML 문서 크기를 교체 함수에서 참조하였다.

캐쉬를 교체하는 방법에서 XCacher 에서는 캐쉬 내용 정의의 하부 트리를 교체 단위로 하고, 캐쉬 교체 함수로 LRU 를 사용함으로써 캐쉬의 효율을 높이지 못하였다. ACE-XQ 에서는 XQuery 결과가 가질 수 있는 모든 경로를 교체 단위로 하므로 XQuery 결과 전체를 교체 단위로 하는 것 보다 좋은 효율을 가지지만, 교체 단위의 수가 증가되면서 교체 희생자를 찾기 위한 연산의 횟수도 많아지게 되어, 캐쉬 교체 연산을 수행하기 위한 부담이 커지는 단점이 있다. 이를 보완하기 위한 방법으로 본 논문에서는 시맨틱 XML 질의 캐쉬의 교체 기법을 다룬다.

본 논문에서 캐쉬 교체를 위해 두가지 이슈를 다룬다.

† 본 논문은 정보통신부의 정보통신기초기술연구지원사업(정보통신 연구진흥원)으로 수행 되었음 (과제번호: 04-기초-082).

첫번째 이슈는 교체 대상 단위가 무엇이고, 어떤 방법으로 교체 희생자를 선택할 것인가 이다. 본 논문에서는 질의의 결과로 나타내어지는 시맨틱 XML 영역 내의 모든 경로를 교체 단위로 하되, 희생자의 선택은 시맨틱 XML 영역 중에서 1 차적으로 선택하고, 시맨틱 XML 영역 내부의 경로들 중에서 2 차적으로 선택하는 2 단계 교체 희생자 선택 방법을 제시한다. (XML 시맨틱 영역에 관한 정의는 3.1 절에서 자세하게 다룬다.) 두번째 이슈는 교체 희생자를 선택하기 위해 어떤 교체 함수를 사용하는가 이다. 본 논문에서는 적중률, 인출 지연 시간, 객체 크기를 사용하고 추가될 XML 질의 결과와 캐쉬된 XML 질의 결과의 크기를 비교하여 희생자를 선택하는 교체 함수를 제시한다.

본 논문은 XML 질의 결과를 캐쉬하는 시스템에서 질의 처리 성능을 향상 시킬 수 있는 교체 기법을 다루었고, 제시된 기법은 실용적인 측면에서 실제 웹 환경에서도 향상된 질의 처리 성능을 보이는데 활용 될 수 있다.

본 논문의 구성은 다음과 같다. 2 절에서는 관련 연구들의 내용을 알아보고, 본 논문과의 차이점을 기술 한다. 3 절에서는 본 논문에서 제시하는 시맨틱 XML 질의 캐쉬와 교체의 내용을 기술하고, 4 절에서는 3 절에서 제시한 내용을 구현하고 기존 기법과 성능을 비교 평가한 결과를 기술한다. 마지막으로 5 절에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. 관련 연구

XML 데이터에 대한 질의 결과는 시맨틱 영역이다. [5]에서는 시맨틱 영역에 대한 캐쉬 교체 방법에 관한 연구를 하였고, [6]에서는 웹 캐쉬에서 교체 함수에 관한 비교 연구를 하였다. 그리고 XML 질의 캐쉬 교체에 관해서는 [1][3][4]에서 연구 되었다.

[1]에서는 XQuery 결과에서 캐쉬에 추가될 내용을 추가하고 캐쉬의 내용을 나타내는 정의를 재구성한다. 재구성된 정의 내에서 하부 트리로 나타내어 지는 부분을 캐쉬 교체 단위로 하고, 희생자를 선택하기 위한 교체함수로는 LRU 를 사용하였다. 이 방법에서 교체 단위의 개수는 작고 크기는 크다. 따라서 희생될 교체 단위의 크기가 추가될 XQuery 결과 크기보다 많이 클 경우가 자주 발생하고, 이로 인하여 캐쉬의 활용도가 떨어져서 성능 저하의 원인이 된다.

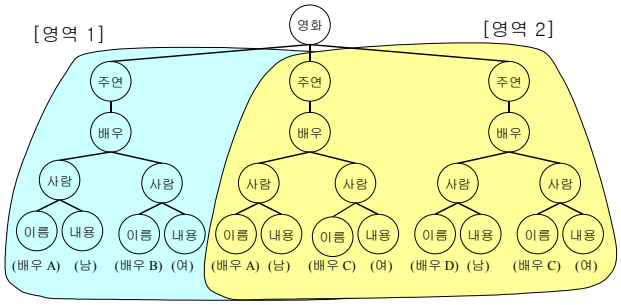
[3][4]에서는 캐쉬 교체의 단위로 XQuery 결과가 가질 수 있는 모든 경로를 사용하고, 교체 희생자를 선택하기 위한 교체 함수로는 각 경로의 적중률, 인출 지연 시간, 객체 크기, XML 문서 크기를 사용하였다. 이는 XQuery 결과 전체를 교체 단위로 하는 것 보다 교체 단위의 개수가 많고, 크기가 작아서 높은 적중률을 나타낸다. 그러나 교체 단위의 크기가 작아서 새로 추가될 XQuery 결과로 인해 희생되어야 할 교체 단위의 수가 많아지게 된다. 이로 인해 교체 희생자를 찾기 위한 연산을 수행하는 부담이 크다. 이를 해결하기 위해 본 논문에서 2 단계로 희생자를 선택한다. 이에 대한 자세한 설명은 3.2 절에서 한다.

3. 시맨틱 XML 질의 캐쉬와 교체

본 절에서는 XCacher 와 ACE-XQ 에서 캐쉬를 교체하는데 나타나는 문제점을 해결하기 위해 시맨틱 XML 영역을 정의하고, 교체 희생자를 선택하는 과정과 방법을 제시한다. 3.1 절에서 시맨틱 XML 영역을 정의하고, 3.2 절에서 시맨틱 XML 질의 캐쉬의 교체 방법을 제시한다.

3.1. 시맨틱 XML 영역

시맨틱 XML 영역이란 XPath 등의 XML 질의어로 표현되는 질의 결과의 하위 경로까지 모두 포함한 영역을 말한다. 본 논문에서 캐쉬에 저장되는 단위는 시맨틱 XML 영역이다. (그림 1)은 시맨틱 XML 영역의 예이다. 여기서 사용된 문서 형식(DTD)은 [7]에서 사용한 영화 문서이다. 영역 1 은 '/영화/주연[/배우/사람/이름="배우 A"]'로 정의된 시맨틱 XML 영역이고, 영역 2 는 '/영화/주연[/배우/사람/이름="배우 C"]'로 정의된 시맨틱 XML 영역이다.



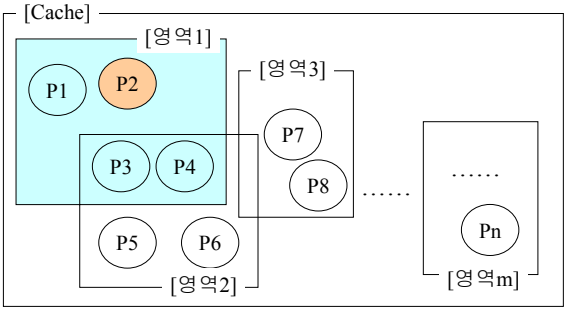
(그림 1) 시맨틱 XML 영역의 예

3.2. 시맨틱 XML 질의 캐쉬의 교체

시맨틱 XML 질의 캐쉬의 교체를 위해 두가지 이슈가 있다. 첫번째 이슈는 교체 대상 단위가 무엇이고, 어떠한 방법으로 교체 희생자를 선택할 것인가 이다. [3][4]에서는 XQuery 결과를 교체 대상으로 하는 Total Replacement 와 XQuery 결과 내의 각 경로를 교체 대상으로 하는 Partial Replacement 를 비교 평가 하였다. 비교 평가 결과로 Partial Replacement 에서 더 좋은 성능을 보였으나 교체 희생자를 선택하기 위한 연산의 부담이 컸다.

본 논문에서는 캐쉬된 시맨틱 XML 영역 내의 각 경로를 교체 대상으로 한다. 그리고 교체 대상의 선택을 위해 2 단계 선택을 한다. 캐쉬에 저장된 시맨틱 XML 영역들 중에서 교체 희생자를 먼저 선택하고, 선택된 영역 내의 각 경로들 중에서 최종 교체 희생자를 선택한다. 이는 시맨틱 XML 영역을 교체 대상으로 하는 것 보다 교체 단위의 크기가 작기 때문에 우수한 효율을 얻을 수 있고, 모든 시맨틱 XML 영역 내의 경로에 대해서 교체 희생자 선택을 위한 연산을 하지 않음으로 교체 처리 시간을 줄인다. 본 논문에서는 이를 Partial Replacement in Semantic Region 이라 한다. (그림 2)에서 예를 보여준다. 캐쉬 내에서 먼저 모든 시맨틱 XML 영역을 대상으로 교체 희생자 영역(예: 영역 1)을 선택한다. 그 영역 내의 모든 경로들 중에서 최종 교체 희생자(예: P2)를 선택한다. 예를 들어, 캐쉬에 m 개의 영역이 있고, 각 영역은 평균 n 개의 경로를 포함하고 있다고 하자. [3][4]의 Partial Replacement 에서는 교체 희생자를 찾기 위해 m×n 개의 교체 단위에 대한 연산을 해야 한다. 반면 본 논문의 Partial Replacement in Semantic Region 에서는 m 개의 영역에 대한 연산을 먼저 하고 그 안에서 n 개의 경로에

대한 연산을 하게 되어 총 연산은 m+n 번 하게 된다.



(그림 2) Partial Replacement in Semantic Region 의 예

두번째 이슈는 교체 희생자를 선택하기 위해 어떤 교체 함수를 사용하는가 이다. [6]에서는 웹 캐쉬에서 사용되는 여러 교체 함수를 비교하였다. [3][4]에서는 적중률, 인출 지연 시간, 객체 크기, XML 문서 크기를 교체 함수에서 참조하였다. 본 논문에서는 적중률, 인출 지연 시간, 최근 접근 시간, 객체 크기를 고려해서 희생 우선순위를 정하고, 교체 대상을 찾을 때 캐쉬에 추가되는 객체의 크기와 희생자의 크기를 비교한다. 본 논문에서 이 방법을 Least Recently Frequently Fetch – Large Size – Minimum(LRFF–LS–MIN)이라 한다. 이는 웹 캐쉬 교체 기법인 LRU-MIN 방법[8]에서 크기를 고려하는 부분을 적용한 것이다. LRU-MIN 은 캐쉬된 객체를 가장 오래 전에 사용한 순서대로 정렬하고, 정렬된 객체 크기와 추가될 객체 크기를 비교하여 추가될 객체보다 크기가 큰 캐쉬된 객체를 희생자로 선택한다. 만족되는 결과가 없을 때 캐쉬된 객체를 추가될 객체의 1/2 크기와 비교하여 교체 희생자를 선택하는 것을 반복한다. LRFF–LS–MIN 은 (식 1)의 교체 함수에 따른 우선순위를 정하고, 상위 50%의 캐쉬된 객체와 추가될 객체를 비교하여 추가될 객체보다 큰 캐쉬된 객체를 희생자로 선택하여 우선순위가 낮은 객체를 보호한다. 만족하는 결과가 없을 때 캐쉬된 객체는 추가될 객체의 4/5 크기와 비교하여 교체 희생자를 선택하는데, 이는 추가될 객체가 1/2 이 되면서 필요 이상으로 작은 희생자가 선택되는 것을 방지하여, 희생자로 선택되는 객체의 크기를 추가되는 객체와 비슷하게 한다. 이러한 과정을 반복 수행하여 필요한 캐쉬 공간을 확보한다.

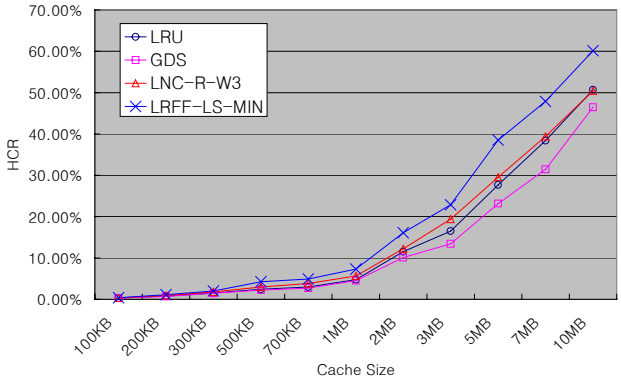
$$Rep-fun = \frac{Hit\ Count \times Larst\ Access\ Time \times Fetch\ Time}{Object\ Size^2} \quad \text{(식 1)}$$

4. 구현 및 성능평가

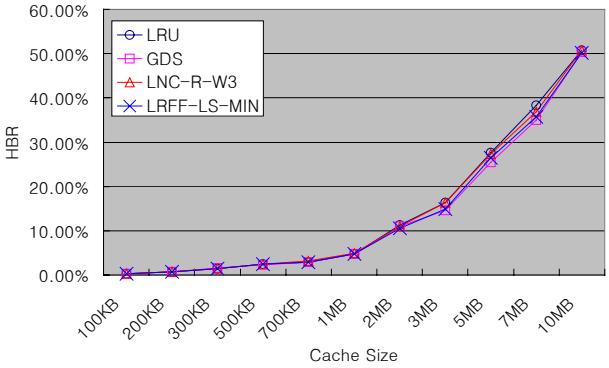
이전 절에서 캐쉬 교체 단위로서 Partial Replacement in Semantic Region 을 제시하였다. 그리고 캐쉬에서 적용 가능한 교체 함수로 LRFF-LS-MIN 을 제시하였다.

본 절에서는 앞서 제시한 기법을 구현하고 성능을 비교 평가하였다. 실험 환경으로 CPU 는 Celeron 2.4GHz, 메모리는 1GB, OS 는 Windows 2000 Server, 개발 도구로는 Java 1.4 를 사용하였다. 실험을 위한 데이터로 4900 개의 각기 다른 크기와 인출 지연 시간을 가진 객체를 생성하고, 질의로는 여러 개의 객체로 이루어진 시맨틱 영역을 생성하였다. 그리고 캐쉬에 추가되는 영역은 질의 결과 중에서 캐쉬에서 적중되지 않은 시맨틱 영역이다. 이를 기반으로 캐쉬에 추가 또는 교체하는 시스템을 구현하였다. 본 구현에서는 질의 결과 중에서 캐쉬에 적중되지 않은 시맨틱 영역은 모두 캐쉬에 추가하도록 하였다. 실험은 두가지로 진행하였다.

- 첫째, Partial Replacement in Semantic Region 교체 기법에서 LRU, GDS, LNC-R-W3[6]와 LRFF–LS–MIN 교체 함수를 사용할 때 Hit Count Ratio(HCR)와 Hit Byte Ratio(HBR)를 비교한다.
- 둘째, LRFF–LS–MIN 교체 함수를 이용하여 Total Replacement, Partial Replacement, Partial Replacement in Semantic Region 의 Hit Count Ratio, Hit Byte Ratio, 캐쉬 교체 연산을 위한 부담(overhead)을 비교한다.

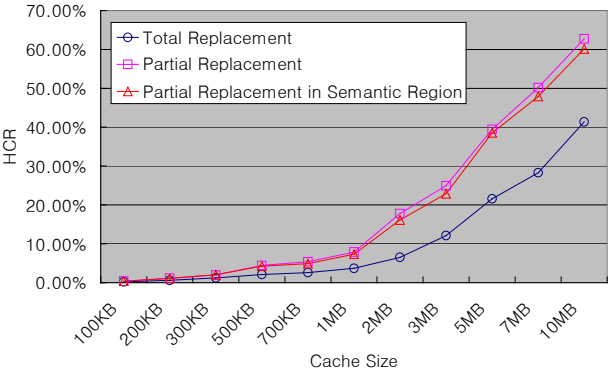


(그림 3) 교체 함수 비교(Hit Count Ratio)

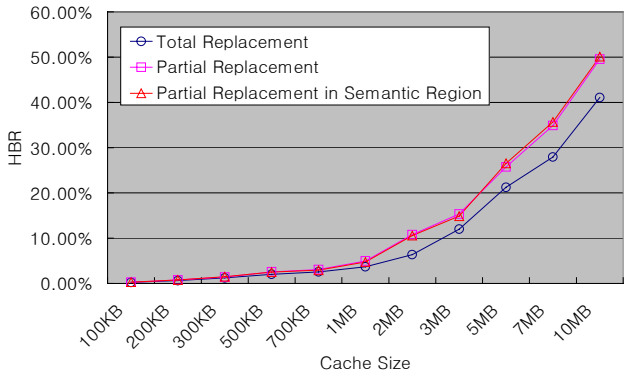


(그림 4) 교체 함수 비교(Hit Byte Ratio)

(그림 3), (그림 4)는 여러 교체 함수의 Hit Count Ratio 와 Hit Byte Ratio 를 나타낸 것이다. Hit Byte Ratio 에서는 크게 차이 나지 않으나 Hit Count Ratio 에서 LRFF-LS-MIN 의 성능이 나은 것을 볼 수 있다.

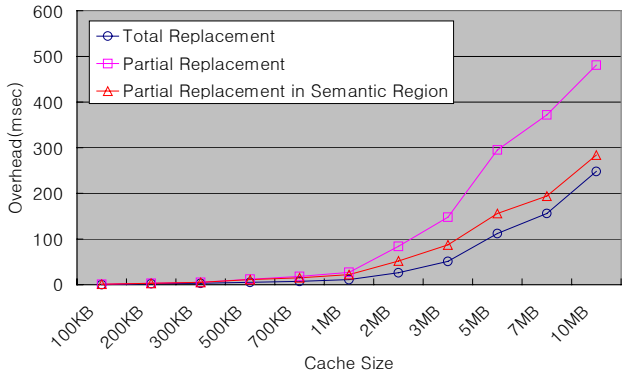


(그림 5) 교체 단위 비교(Hit Count Ratio)



(그림 6) 교체 단위 비교(Hit Byte Ratio)

(그림 5), (그림 6)은 LRFF–LS–MIN 교체 함수를 사용할 때 Total Replacement, Partial Replacement, Partial Replacement in Semantic Region 세가지 교체 기법의 Hit Count Ratio 와 Hit Byte Ratio 를 비교한 것이다. Total Replacement 교체 기법에서 성능이 좋지않고, 나머지 두 교체 기법에서는 거의 비슷한 성능을 나타냄을 볼 수 있다.



(그림 7) 교체 단위 비교 (Overhead)

(그림 7)은 캐쉬 교체 연산을 수행하는 부담을 나타낸 것이다. Total Replacement 교체 기법은 교체 단위가 시맨틱 XML 영역으로 교체단위 개수가 가장 적기 때문에 캐쉬 교체 연산을 위한 부담이 가장 적었고, Partial Replacement 교체 기법은 시맨틱 XML 영역의 모든 경로들에 대해 교체 함수가 적용됨으로 캐쉬 교체 연산을 위한 부담이 가장 많았다. 반면에 Partial Replacement in Semantic Region 교체 기법은 교체 단위가 Partial Replacement 와 같이 시맨틱 XML 영역의 모든 경로이지만 모든 교체 단위에 교체 함수를 적용하지 않고, 2 단계로 교체 희생자를 선택함으로써 캐쉬 교체 연산을 위한 부담이 Partial Replacement 교체 기법보다 적었다.

위 실험 결과를 토대로 시맨틱 XML 질의 캐쉬 교체에서 LRFF–LS–MIN 교체 함수가 향상된 성능을 나타냄을 확인하였고, 교체 희생자 선택에서 2 단계 방법을 사용하는 Partial Replacement in Semantic Region 교체 기법을 사용함으로써 Partial Replacement 에 준하는 성능을 보임과 동시에 Partial Replacement 보다 캐쉬 교체 연산을 위한 부담이 많이 줄어듦을 확인할 수 있었다.

5. 결론 및 향후 연구

웹에 산재한 많은 양의 XML 문서에 대한 질의 처리에 있어서 성능의 향상을 보증하는데 시맨틱 XML 질의 캐쉬가 유용하다. 그러나 캐쉬의 저장 공간과 다양한 질의 결과를 다수 캐쉬하는데 한계가 있기 때문에 효율적인 캐쉬 교체가 필요하다. 본 논문에서는 시맨틱 XML 질의 캐쉬에서 캐쉬 교체 단위와 교체 방법을 정의한 교체 기법과 캐쉬 희생자를 선택하기 위한 교체 함수를 제시하였다. 그리고 위 교체 기법과 교체 함수를 시맨틱 영역을 표현한 가상의 데이터를 만들어 적용하여 기존의 교체 기법과 교체 함수들과 비교 평가하였다. 비교 결과 본 논문에서 제시한 방법이 더 우수한 것으로 나타났다.

향후 연구 과제는 다음과 같다. 본 논문에서는 가상의 데이터로 실험하였는데, 향후 제시한 방법을 실제 XML 문서에 대해 적용하여 그 성능을 확인하는 연구가 필요하다.

참고문헌

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

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

[3] L. Chen, S. Wang and E. A. Rundensteiner, "A Fine-Grained Replacement Strategy for XML Query Cache," Proc. 4th Intl. Workshop on Web Information and Data Management (WIDM'02), 2002, pp. 76-83

[4] L. Chen, S. Wang and E. A. Rundensteiner, "Replacement Strategies for XQuery Caching Systems," Proc. Elsevier Science Publishers B. V., 2004.

[5] S. Dar, M. J. Franklin and B. Jonsson, "Semantic Database Caching and Replacement," Proc. 22nd VLDB, 1996, pp. 330-341

[6] A. Balamash and M. Krunz, "An Overview of Web Caching Replacement Algorithms," IEEE Communications Surveys & Tutorials, 2004, pp. 44-56

[7] 박정기, 강현철, "웹에서 캐쉬를 이용한 XML 질의 처리 : 구현 및 성능 평가," 한국 정보과학회 2003 가을 학술발표논문집(II), 2003, pp. 133-135

[8] M. Abrams et al., "Caching Proxies: Limitations and Potentials," Proc. 4th Int'l. World Wide Web Conf., 1995.