객체지향 데이터베이스에서 XML 질의 처리를 위한 클래스 인덱스 기법

김은미 , 박 석
서강대학교 전자계산학과 데이타베이스 연구실

A class index scheme for processing XML queries in object-oriented database

Kim, Eun mi and Park, Seog
Database Research Lab., Dept. of Computer Science, Sogang University
{banny95, spark}@dblab.sogang.ac.kr

1. 서 론

XML(eXtensible Markup Language)은 웹 상에서 데이터 교환을 위해 제안된 표준 언어이다. 앞으로 많은 데이터가 XML로 표현될 것이므로, 관계형 데이터베이스(relational database: RDB)나 객체지향 데이터베이스(object-oriented database: OODB)를 이용하여 XML 데이터를 효과적으로 저장하고 검색하는 연구들이 최근 진행되고 있다. 특히 객체지향 데이터베이스는 데이터 모델이 XML의 특성을 자연스럽게 반영할 수 있고, XML에 대한 질의를 기존 객체질의(Object Query Language: OQL)를 확장하여 지원할 수 있기 때문에 이것을 이용하려는 연구들이 많이 이루어지고 있다.

그림 1-a는 XML 데이터의 스키마에 해당하는 DTD(Document Type Definition)의 한 예를 보인 것이고, 그림 1-b는 그림 1-a를 참고문헌[1]에서 제시된 방법을 사용하여 OODB 스키마로 변환한 것이다.

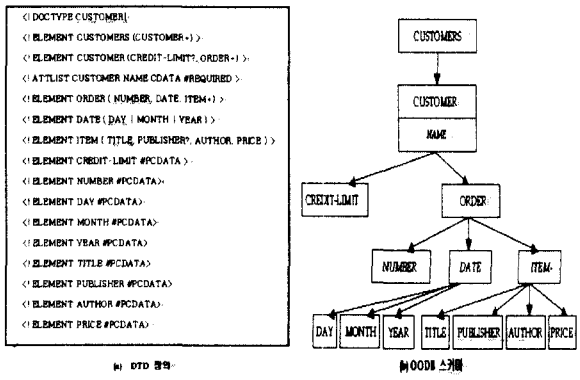
객체지향 데이터베이스는 경로식(path expression)을 포함하는 질의를 처리하기 위해서, 객체간의 참조관계를 따라 객체를 차례로 방문해야 하기 때문에 오버헤드를 유발한다. 객체지향 데이터베이스에 저장된 XML 데이터를 효과적으로 검색하기 위해서, 객체 탐색의 오버헤드를 감소시키도록 질의처리를 도와주는 인덱스 기법이 필요하다. 검색을 효과적으로 지원하려면 경로의 객체 모두를 저장하고 있어야 하는데(경로 인덱스(path index)[4]), 이는 저장장치에 오버헤드를 유발한다. 그러므로 경로의 객체를 모두 알 수 있으면서 인덱스의 저장공간은 줄이는 방법이 필요하다.

XML 데이터는 트리 구조를 나타내므로 이것을 이용하여 부모 객체를 알 수 있는 UID(Unique element Identifier)를 부여한다. 그리고 이것을 인덱스에 활용하면, 최하위 객체만을 저장하고도 부모객체 계산을 통해 경로의 객체 모두를 알 수 있다. 또한 이러한 부모객체 계산은 데이터베이스를 접근하지 않아도 검색을 가능하게 함으로 효과적이다. 제안한 UID 는 BUS(Bottom Up Scheme)[2]의 UID 와는 다르게 부모와 형제 객체에만 관계가 있도록 부여하기 때문에, 갱신 시 다른 객체의 UID에는 영향을 주지 않는다. 또한 UID 는 형제 객체간의 순서 정보도 포함하므로 순서 정보를 포함하는 질의처리를 할 수 있게 한다.

본 논문에서는 이러한 UID를 이용하여 텍스트 인덱스와 속성 인덱스를 구성하고, 텍스트와 속성에 해당하는 경로식의 정보를 스키마 경로 인덱스로 구성한 클래스 인덱스 기법을 제안한다. 비용모델로 성능평가를 해보면, 클래스 인덱스가 경로의 객체 모두를 저장하는 기존의 경로 인덱스에 비해 저장 공간을 많이 줄이면서 UID를 이용한 부모객체 계산을 통해 검색도 효율적으로 하는 것을 알 수 있다.

논문의 나머지는 다음과 같이 구성되어 있다.

2장에서는 관련 연구들에 대해 살펴보고, 3장에서는 제안하는 클래스 인덱스의 구성과 구현방법을 살펴본다. 그리고 4장에서는 클래스 인덱스의 검색과 갱신을 예를 통해 살펴보고, 5장에서는 비용모델을 이용하여 저장, 검색, 갱신비용 측면에서 다른 인덱스들과 비교 분석한다. 마지막으로 6장에서 결론을 맺는다.



[그림 1] XML DTD 와 변환된 OODB 스키마

2. 관련 연구

객체지향 데이터베이스의 데이터 모델은 XML의 모델과 거의 유사하므로, XML 데이터의 저장과 검색이 객체지향 데이터베이스에서 자연스럽게 이루어질 수 있다. 객체지향 데이터베이스를 이용하는 방식은 XML의 요소(element)를 주로 객체단위로 저장하기 때문에 서로 참조관계에 있는 객체들을 효과적으로 탐색하는 것이 검색 성능과 관련하여 중요한 관건이 된다. 이와 같이 경로에 존재하는 객체들을 효과적으로 탐색하기 위해, 경로 인덱스(Path Index)[4]와 중첩 인덱스(Nested Index)[4] 등의 기법이 제안되었다..

먼저, 경로 인덱스는 어떤 속성 값과 그것의 경로의 객체 모두를 저장하는 형태로 중첩된 프레디키트가 포함되는 질의를 처리할 수 있으나 저장비용이 너무 비싸다. 그리고, 객체간의 연결이 한 부분만 바뀌어도 연관되는 모든 객체를 인덱스에서 갱신해줘야 하므로 갱신비용이 비싸다. 반면에 중첩 인덱스는 어떤 속성 값과 그것의 경로의 시작하는 객체를 함께 저장하는 형태로, 저장 공간을 줄이지만 프레디키트가 포함되는 질의를 처리할 수가 없다. 그리고, 인덱스를 갱신할 때 끝 객체의 속성 값과 시작 객체를 얻기 위해 시스템 상에 하향식, 상향식 링크를 모두 유지해야만 한다.

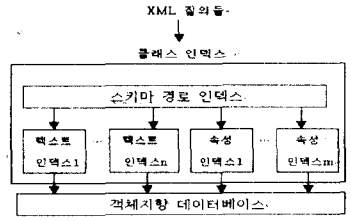
XML은 구조적 문서이므로 이러한 구조적 문서에 대해 제안되었던 인덱스 기법을 살펴보면, 정보검색 시스템에서 구조적인 부분을 처리하기 위해 제안되었던 인덱스 기법들이 있다. 그 중, 저장 공간을 줄이면서 데이터베이스에 저장되어 있는 문서들의 모든 객체에 접근하는 것을 지원하는 BUS(Bottom Up Scheme)[2]가 있다. 이 기법은 저장되는 문서들을 트리 구조화 했을 때, 최하위의 텍스트 레벨 노드(보통 하나의 노드는 하나의 객체로 저장된다)만을 인덱스로 구성하는 기법이다. 이 인덱스는 Lee[3]에서 제안된 UID(Unique element IDentifier)를 이용하여, 상향식 접근으로 모든 부모 노드를 접근할 수 있게 하는 것이다. UID 부여방법은 다음과 같다. 먼저, 문서를 k-ary 완전 트리(k-ary complete tree)로 표현한다. 여기서 k는 트리 구조화된

문서에서 노드가 가지는 최대의 자식 수이며, k-ary 완전 트리를 만드는 방법은 모든 노드가 k개의 자식 노드를 가지도록 가상 노드를 첨가하는 것이다. 그다음, 레벨 순서에 따라 트리를 탐색하면서 각 노드에 UID를 부여한다. 이제 문서 트리에서 어떤 노드의 UID를 알면, 다음 식을 통해 부모 노드의 UID를 계산할 수 있다.

Parent (i) = ⌊ (i-2) / k + 1 ⌋

3. 클래스 인덱스 기법

본 논문에서는 객체지향 데이터베이스에 저장된 XML 데이터에 대한 질의를 효율적으로 처리하기 위한 클래스 인덱스 기법을 제안한다. 그것은 크게 두 부분으로 나뉜다. 하나는 스키마 경로식을 저장하는 스키마 경로 인덱스이고, 다른 하나는 각 스키마 경로식에 대한 내용정보를 저장하는 텍스트 인덱스와 속성 인덱스이다. 스키마 경로 인덱스는 질의에 포함되는 경로식을 지원하기 위한 것이고, 텍스트 인덱스와 속성 인덱스는 프레디키트 평가를 지원하기 위한 것이다. 스키마 경로 인덱스가 텍스트 인덱스와 속성 인덱스 위에 만들어진다. 그림 2는 XML 질의가 들어왔을 때 그것을 처리하는 클래스 인덱스의 구조를 보여준다.



[그림 2] 클래스 인덱스

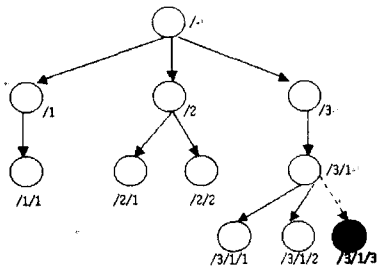
3.1 스키마 경로 인덱스

XML 문서를 객체지향 데이터베이스에 저장했을 때, 저장된 문서에 나타날 수 있는 경로식들은 변환된 OODB 스키마에 있는 경로식들이다. 예를 들어, 그림 1-b 를 보면 텍스트 인덱스에 해당되는 경로식은 9 개가 있으며, 속성 인덱스에 해당되는 경로식은 1 개가 있다. 경로식은 루트 클래스 다음 클래스로부터 텍스트 레벨 또는 속성 레벨 클래스까지를 표현한다. 구성된 스키마 경로 인덱스는 질의가 들어왔을 때, 질의에 맞는 경로식을 찾아서 해당 경로의 텍스트 또는 속성 인덱스로 연결시킨다. 스키마 경로 인덱스의 경로식 정보는 복합적인 질의를 처리하기 위해, 텍스트나 속성 인덱스에서 경로 객체를 계산할 때 이용될 수 있다.

3.2 UID(Unique element Identifier)

텍스트 인덱스(속성 인덱스)를 텍스트 값(속성 값)과 텍스트 레벨(속성 레벨)의 객체에 부여된 UID로만 구성함으로써 저장공간을 줄이는데, UID를 부여하는 방법은 [6]에서 제시된 방법을 사용한다. 그것은 다음과 같다. 먼저, 저장하려는 XML 문서를 트리 구조화한다. 그 다음, 루트 노드로부터 디렉토리 형식으로 각각의 노드에 UID를 부여한다. 이때, UID는 그림 4와 같이 부모 노드의 UID에 형제 노드 간의 순서 정보를 덧붙여서 표현된다. 이제 어떤 노드의 UID가 있다면, 다음의 식을 통해서 부모 노드의 UID를 계산할 수 있다.

If UID of any node is < /a/b/c/d >, then UID of parent node of that node is < /a/b/c >



[그림 4] UID들을 부여한 문서 트리

이렇게 부여된 UID는 부모 노드와 형제 노드에만 관련되므로 다른 노드의 UID에는 아무런 영향을 주지 않아 갱신 비용에 대한 오버헤드가 없다. 어떤 노드가 삽입될 때 그것의 UID는 단지 부모 노드와 형제 노드만을 참조해서 부여해주면 된다. 이 때 다른 노드의 UID에는 아무런 영향을 주지 않는다. 예를 들어, 그림 4에서 색칠된 노드가 삽입된다고 했을 때, 그것의 부모 노드와 형제 노드를 참조해서 '/3/1/3'이라는 UID를 부여해주면 된다.

3.3 텍스트 인덱스

스키마 경로 인덱스가 질의에 포함된 경로식을 평가하고, UID로 경로객체를 계산하기 위한 경로식 정보를 제공하지만, 이것은 질의 경로식에 포함된 프래디키트 평가는 지원하지 못한다. 그러므로 질의 경로식에 포함된 프레디키트 평가를 지원하기 위해, 경로식에 일치하는 텍스트 값들을 텍스트 레벨 객체에 부여된 UID와 함께 해당 경로에 매핑하는 것이 텍스트 인덱스이다. 어떤 스키마 경로식에 대한 텍스트 인덱스는 다음과 같다.

< 텍스트 값 1, UID1, ... UIDn >, ... , < 텍스트 값 m, UID1, ... UIDk >

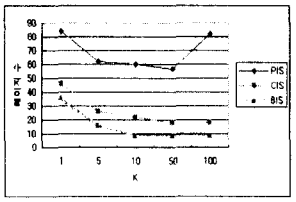
텍스트 값을 해당 경로식의 객체들과 직접 매핑하는(중첩 인덱스, 경로 인덱스처럼) 대신에, 텍스트 인덱스는 스키마 경로 인덱스에 저장된 경로식에 텍스트 값과 UID를 매핑한다. 스키마 경로 인덱스는 경로객체 탐색을 위한 정보로서 그리고 해당 경로식에 대한 텍스트 값들로의 간접적인 연결레벨로서 이용된다. 즉, 질의에 포함된 경로식과 프레디키트의 평가를 위한 탐색을 분리함으로써 저장 공간의 오버헤드와 갱신 비용을 낮추었다. 텍스트 인덱스는 B+-trees와 같은 트리 구조로 구성될 수 있다. 이때, 스키마 경로 인덱스는 해당 경로식에 관련되는 텍스트 인덱스의 시작 위치를 포인터로 연결한다.

속성 인덱스도 같은 방식으로 구성된다

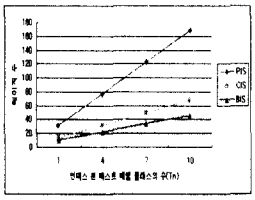
4. 저장비용 및 성능평가

이번 장에서는 성능을 분석하기 위해 다음의 두 가지 인덱스와 비교한다. 첫 번째는 기존의 객체지향 데이터베이스에서 제안되었던 인덱스 기법 중, 제안한 클래스 인덱스와 궁극적으로 저장하고자 하는 대상이 같은 경로 인덱스(Path Index)이고, 두 번째는 구조적 문서에 대한 인덱스 기법 중 제안한 클래스 인덱스와 기본 개념이 동일한 BUS(Bottom Up Scheme)의 UID 를 적용한 클래스 인덱스이다. 앞의 두 가지 인덱스와 제안하는 클래스 인덱스의 비용 모델(cost model)을 공식화하고, 세가지 인덱스의 저장, 검색 그리고 갱신 비용을 비교한다 비용 모델과 변수는 [4,7,8]에서 제시되었던 것을 XML 문서의 특징에 맞게 변형시켜 만든 것이다.

4.1 저장 비용



[그림 9] 저장 비용

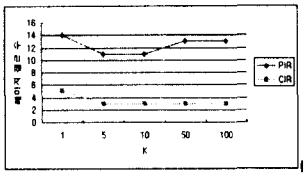


[그림 10] 텍스트 인덱스의 수에 따른 저장비용

그림 9는 N을 2000이라고 가정했을 때, K를 변화시켜가며 세 개의 인덱스의 저장비용을 측정한 것이고, 그림 10은 텍스트 인덱스들의 수가 많아질수록 변화하는 세 개의 인덱스의 저장비용을 측정한 것이다. 그 결과를 보면, 클래스 인덱스가 BUS의 UID를 적용한 클래스 인덱스보다 저장 비용이 조금 더 들지만 그리 큰 차이는 없음을 알 수 있고, 경로 인덱스보다는 저장 비용이 많이 줄었음을 알 수 있다.

4.2 검색 비용

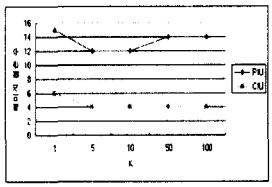
여기서는 분석을 간단하게 하기 위해, 다음의 두 가지 질의에 대한 검색 비용을 비교해본다. 첫번째는 프래디키트가 포함된 경로식의 시작하는 객체를 찾는 질의 형태이고, 두 번째는 프레디키트 값이 주어지지 않고, 주어진 경로에 맞는 텍스트 값을 구하는 질의이다. BUS의 UID 를 적용한 인덱스는 UID 만 다르게 적용한 것이므로, 검색 비용은 거의 비슷하다. 그러므로, 여기서는 경로 인덱스와 클래스 인덱스의 검색 비용만을 비교할 것이다

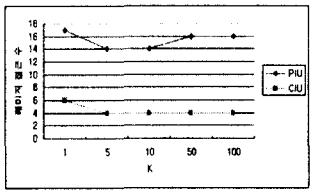


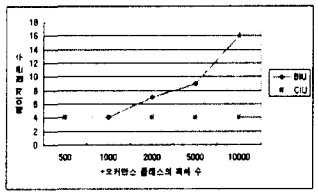
[그림 11] 검색 비용

4.3 갱신 비용

첫 번째 경우는 텍스트 값이나 속성 값이 변경되는 경우이고(여기서는 텍스트 값이 변했다고 가정한다), 두 번째 경우는 클래스의 객체가 삽입되는 경우이다.







[그림 12] 내용변경과 객체삭제 [그림 13] 객체삽입 시의 갱신비용
[그림 14] BUS의 UID와 제안한 UID를 적용한 클래스 인덱스의 갱신비용

그림 12과 그림 13를 보면, 제안한 클래스 인덱스가 기존의 경로 인덱스보다 갱신 비용을 많이 줄이고, K의 변화에 상관없이 일정한 비용을 보임을 알 수 있다. 그림 14는 문서의 최대의 자식노드 수가 바뀔 때의 BUS의 UID와 제안한 UID와의 비용 차를 측정한 것인데, 객체 수가 2000일 때부터 바뀐다고 가정하였다. BUS의 UID는 갱신비용이 급격하게 증가하므로 XML 문서와 같이 가변적인 문서는 제안한 UID를 적용하는 것이 갱신 측면에서 훨씬 효율적임을 알 수 있다.

5. 결 론

최근 객체지향 데이터베이스가 XML 데이터의 저장 장치로서 각광받고 있는데, 객체지향 데이터베이스에 저장된 XML 데이터를 효율적으로 사용하려면, 검색 성능을 향상시키는 효과적인 인덱스 기법이 필요하다. 본 논문에서는 객체지향 데이터베이스에 XML 데이터를 저장할 때, 저장 공간을 줄이면서 XML 질의처리를 효과적으로 지원하는 인덱스 기법인 클래스 인덱스 기법을 제안한다.

이 기법은 효과적인 검색을 위해 경로의 객체 모두를 저장하는 경로 인덱스와는 다르게 부모 객체를 알 수 있는 UID를 부여하여 경로의 끝 객체만을 저장함으로써 인덱스의 저장 공간을 줄인다. 그리고 XML 질의에 포함되는 경로식의 평가를 스키마 경로 인덱스에서 수행한 뒤, 일치하는 경로식에 연결된 텍스트 또는 속성 인덱스만을 탐색하게 함으로써 질의 경로식에 맞지 않는 불필요한 부분의 탐색을 방지한다. 마지막으로 기존 연구인 BUS의 UID보다 갱신 비용면에서 더 효율적인 방법으로 UID를 부여함으로써 갱신 비용을 감소시켰으며, UID가 형제 객체간의 순서정보를 포함하고 있으므로 순서정보를 포함하는 질의도 수행할 수가 있다.

하지만, 스키마에서 + 오커런스를 가지는 클래스 수가 많아지고, 그러한 클래스의 객체 수가 많아지면, UID의 크기가 증가하게 되어 저장비용이 늘어날 수 있다. 하지만 부여된 UID의 특징상 크기의 증가속도가 낮으므로 큰 오버헤드는 없다.

앞으로 DTD내에 사이클(cycle)을 포함하는 경로식와 정규식으로 주어진 경로식에 대한 인덱스 구성에 관한 연구가 이루어져야 할 것이다.

참고 문헌

[1] Vassilis Christophides, Serge Abiteboul, Sophie Cluet, and Michel Scholl. "From structured documents to novel query facilities", In Proceedings of the ACM SIGMOD International Conference on Management of Data. 23(2):313-324, June 1994.

[2] Dongwook Shin and Honglan Jin, " BUS: An Effective Indexing and Retrieval Scheme in Structured Documents", ACM DL 1998, pp. 235-243.

[3] Lee, Y.K. Yoo, S.J. Yoon, K.Berra, P.B, "Index Structures for Structured Documents", Proc. Digital Library'96 (1996), Pages 91-99.

[4] Elisa Bertino and Won Kim, "Indexing techniques for queries on nested objects", IEEE Trans on Knowledge and Database Eng. 1(2), 1(2):196--214, Jun. 1989.

[5] W3C Working Draft 15 February 2001, " XQuery : A Query Language for XML", http://www.w3c.org/TR/2001/WD-xquery-20010215

[6] Hyung-Il Kang, Jae Soo Yoo and ByoungYup Lee, " Design and Implementation of a XML Repository System Using DBMS and IRS", XML Asia Pacific 2000, 7th Annual Conference for XML, SGML and markup technologies, Sydney, 31 Oct- 3 Nov, 2000.

[7] Lee, W.C., and Lee, D.L., "Path Dictionary: A New Approach to Query Processing in Object-Oriented Databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 3, May/June 1998, 371-388

[8] Hirotaka Kanemoto, Hiroyuki Kato, Hiroko Kinutani and Masatoshi Yoshikawa: "An Efficiently Updatable Index Scheme for Structured Documents", DEXA Workshop 1998: 991-996.

[9] D.L. Lee & W.C. Lee, " Using Path Information for Query Processing in Object-Oriented Database Systems, " Proceedings of Conference on Information and Knowledge Managament, Washington, DC, Nov. 1994, 64-71.