論文
5-1-10

XML을위한효율적인저장구조및인덱싱기법설계 (Design of Efficient Storage Structure and Indexing Mechanism for XML Documents)

신 판 섭(Pan-seop Shin)[1)]

요 약

최근에 인터넷의 급속한 발전과 더불어 대량의 정보를 효과적으로 표현 및 교환할 수 있는 새로운 데이터 표준으로 XML (eXtensible Markup Language)이 제안되었으며, XML 문서에 대한 저장과 인덱싱에 대한 연구가 활발하게 진행되고 있다. 본 논문에서는 실시간 XML 문서 처리에 효율적인 주기억장치 기반의 XML 전용 저장 시스템을 설계하고, 사용자 질의에 포함된 엘리먼트 타입 정보를 이용하여 XML 문서 트리에 대한 순회를 최소화시킬 수 있는 구조적 검색 기법을 설계한다. 또한, 엘리먼트의 삭제 및 삽입 등 동적인 변경에 빠르고 유연하게 대처할 수 있는 인덱스 구조와, 링크 정보를 가지고 있는 XML 문서의 질의 처리를 위해 XLink 표준을 준수하여 테이블 형식의 링크 정보 인덱스 구조를 설계한다.

ABSTRACT

XML has recently considered as a new standard for data presentation and exchange on the web, many researches are on going to develop applications and index mechanism to store and retrieve XML documents efficiently. In this paper, design a Main-Memory based XML storage system for efficient management of XML document. And propose structured retrieval of XML document tree which reduce the traverse of XML document tree using element type information included user queries. Proposed indexing mechanism has flexibilities for dynamic data update. Finally, for query processing of XML document include Link information, design a index structure of table type link information on observing XLink standards.

논문접수 : 2004. 1. 13.
심사완료 : 2004. 1. 26.

1) 정회원 : 대진대학교 컴퓨터공학과

* 이 논문은 2003년도 대진대학교 학술연구비 지원으로 연구되었음

1.서론

인터넷의 발전과 www의 보편화로 인해 교류되는 정보의 양이 크게 증가함에 따라, 대량의 정보를 보다 효율적으로 보관 및 관리 할 수 있는 새로운 데이터 표준으로 XML (eXtensible Markup Language)이 제안되었다.[1]

이러한 이유로 최근에는 XML 문서에 대한 저장과 인덱싱에 대한 연구가 특히 활발하게 진행되고 있다[2,3]. XML은 기존의 데이터 표현 및 교환의 수단으로서 많이 사용되었던 HTML(Hyper Text Markup Language)과 달리 첫째, XML은 사용자가 직접 태그를 정의할 수 있도록 하여 XML 문서 내에 데이터뿐만 아니라 시맨틱을 함께 표현할 수 있는 특징을 지닌다. 둘째, XML은 DTD(Document Type Definition)나 XML Schema를 이용해 문서의 논리적 구조를 문서와 독립적으로 표현할 수 있고 XSL(eXtensible Style Sheet)를 이용해 문서 자체의 데이터 독립성을 유지할 수 있다. 따라서 XML은 이러한 다양하고 새로운 특성으로 인해 많은 응용분야에서 XML 기반 플랫폼 개발 연구가 주목받고 있다. 특히, 웹 환경에서 XML문서의 효과적 저장, 접근 및 검색 기법에 대한 많은 연구가 진행되어왔으며[4][5], 이러한 연구는 기존의 관계형DBMS나 객체지향형(또는 객체 관계형)DBMS를 이용하는 것이 주류를 이루어 왔다. 기존의 상용 DBMS 기반의 XML 문서 저장 시스템은 이전에 개발된 많은 응용 시스템과의 호환성 및 연동 측면의 장점이 있으나, XML 문서가 내포하고 있는 의미와 구조 정보를 완벽하게 표현하기 어렵고, 갱신 연산이나 질의 처리 과정이 복잡해지며, XML 문서의 재구성 비용이 추가적으로 발생하는 등의 단점이 있다. 따라서, 최근에는 XML 문서의 의미는 물론 구조적 정보도 완벽히 지원할 수 있는 전용의 XML 저장 시스템에 대한 관심이 높아지고 있다. 그러나 기존 XML 전용 저장 시스템들은 XML 문서의 특성에 따른 분류를 고려하지 않고, 단순히 일반적인 XML 문서를 대상으로 하는 경우가 주를 이루고 있으므로, 보다 효율적인 XML 문서의 저장 및 검색을 위해서는 XML 문서 자체의 특성에 따라 XML 문서를 분류하고, 각 특성을 고려한 XML 문서 전용 저장 시스템에 대한 연구가 필요하다. 특히 XML 문서를 그 특성에 따라 데이터 중심 XML과 문서 중심 XML로 분류할 경우, 데이터 중심 XML에 비해 상대적으로 연구가 수행되지 않은 문서 중심 XML 저장을 위한 전용 저장 시스템에 대한 연구가 중요 시 되고 있다. 한편, 모든 산업 분야에 걸쳐 실시간 응용이나 무선 응용을 위한 중요성이 커지는 만큼 실시간 처리 요구를 만족할 수 있는 주기억장치 기반의 XML 문서 저장 시스템에 대한 연구 또한 그 필요성이 증가하고 있다. 기존에 연구 개발된 XML 저장 및 검색 시스템은 대부분 디스크 기반으로 개발되어 실시간 처리가 요구되는 응용 분야에는 적합하지 않다. 따라서 본 논문에서는 첫째, 문서 중심 XML의 특성을 충분히 지원하여, 그 특성에 맞게 효과적으로 저장 및 접근할 수 있도록 하기 위한 데이터 모델을 정의한다. 제안한 논리적 데이터 모델은 문서 중심 XML의 특성을 충분히 지원하여 모델링 후 XML 문서 자체가 가지는 원래의 의미와 구조적 특징이 그대로 반영되고, 다시 원문으로의 복원 시, 손실이 발생하지 않도록 설계한다.

또한, 최근에는 효율적인 XML정보의 검색 연구도 활발히 진행되고 있다. 이러한 연구들에서는 XML 문서를 구조화 문서(structured document)의 인스턴스로 해석하고 전통적인 정보검색 기법들을 적용하는 연구[6][7][8]가 진행되고 있다. 특히, XML 문서에 대한 인덱싱 및 검색 기법들은 구조적 특징을 고려한 내용 검색이 많은 부분을 차지하고 있다. 이러한 연구들에서는 주로 내용 검색의 정확도를 높이기 위해 구조적 정보를 이용하는 방법을 제시하고 있으나, 순수한 구조 정보만을 대상으로 하는 처리 방법에 대한 연구는 미비한 실정이다. 키워드 등의 내용 요소 없이 순수하게 구조적 엘리먼트에 대한 결합만으로 구성된 검색을 '구조적 검색(structural retrieval)'이라 하며, 이러한 구조적 검색 질의의 기본적인 처리 방법으로 상향식 또는 하향식 트리 순회를 기반으로 하는 반복적 혹은 재귀적 처리를 필요로 한다[9]. 그러나 이 경우 트리 전체를 순회하여야 하기 때문에 처리 영역이 넓게 되어 효율성이 떨어진다. 또한, [3]의 연구들에서는 XML 문서를 트리 형태로 표현하고, 각 노드로 매핑된 엘리먼트에 구조적 정보를 포함하는 특별한 식별자를 부여하여 트리의 순회 없이 구조적 검색 질의 처리가 가능하도록 설계하였다. 그러나 일부 구조적 검색 질의 유형에는 효율적이나 다른 유형

의 처리를 위해서 추가적인 연산 및 반복적 접근인 요구되는 한편 부가적인 필터링 과정이 요구된다. 또한, XML 문서에 대한 엘리먼트 삽입 및 삭제와 같은 구조적인 변경이 자주 발생하는 환경에서는 구조적 정보를 포함하는 엘리먼트 식별자에 대한 많은 재할당이 요구되므로 동적인 환경에서는 매우 부적합하다[10]. 따라서 본 논문에서는 둘째, XML 문서에 대해 엘리먼트의 삽입 및 삭제 등 동적인 부분 변경을 빠르게 처리하고, 엘리먼트 타입 정보를 이용하여 트리의 순회를 최소화할 수 있는 구조적 질의 처리 기법을 설계한다.

한편 기존 인덱싱 연구들은 독립적인 단일 XML 문서를 대상으로 하는 경우가 대부분이므로 XML 문서간의 관계를 정의한 XLink 개념을 지원하지 못하는 제약이 발생한다[2][3]. 그러므로 여러 문서간의 관계를 이용하여 사용자의 질의를 보다 효과적으로 검색할 수 있도록 XLink를 이용한 인덱싱 기법에 대한 연구가 필요하다. 본 논문에서는 셋째, XLink의 애트리뷰트 중 링크의 의미정보를 포함하고 있는 title과 inbound link를 지원하는 인덱스를 설계한다.

2. 관련연구

효율적인 XML 문서의 저장과 검색을 위해 다양한 XML 저장 구조와 인덱싱 기법의 연구가 활발히 진행되고 있다. 본 장에서는 XML 문서를 위한 저장 시스템과 인덱싱 기법에 대한 기존 연구를 다룬다.

2.1 XML 저장 시스템

XML 문서는 사용 목적 및 문서 자체의 특성에 따라 크게 데이터 중심 XML과 문서 중심 XML로 분류되어진다. 본 절에서는 이러한 XML 문서 분류를 기준으로 기존 관계형 DBMS 및 객체지향형(또는 객체관계형) DBMS 기반의 XML 문서 저장 시스템과 전용 저장 시스템에 대한 연구 내용을 소개한다.

(1) XML 문서 분류

<표 1> XML 문서의 분류

문서 중심 XML	데이터 중심 XML
비정형화된 구조	매우 정형화된 구조
많은 혼합 요소 존재(매우 혼재된 형태)	내용과 구조의 혼재 적음 혹은 거의 없음
구조 및 형제 엘리먼트 간의 순서 중요	구조 및 형제 엘리먼트 간의 순서 정보 불필요
인간 이해 중심의 문서적 특성	기계 처리 중심의 특성

XML 문서는 특성이나 사용 목적에 따라 크게 <데이터 중심 XML>과 <문서 중심 XML>로 분류할 수 있다[5]. 데이터 중심 XML과 문서 중심 XML의 특성에 대한 비교 내용은 <표 1>과 같다.

가. 문서중심 XML

문서 중심 XML은 XML 문서 자체가 중요한 의미를 가지는 인간 이해 중심의 문서적 특성을 지닌다. 데이터 중심 XML에 비해 정형화된 구조 측면이 약하며, 많은 혼합 요소를 포함하는 등 문서의 내용 정보과 구조적 정보가 매우 혼재하고 있는 경우가 많다. 이러한 XML문서는 문서 내에 표현되어 있는 데이터뿐만 아니라 문서 자체의 구조적 정보도 중요한 역할을 하게 된다. 예로는 편지, 책, 광고 등을 들 수 있다.

나. 데이터중심 XML

데이터 중심 XML은 주로 데이터 교환의 형식으로 이용되는데, 문서 자체보다는 문서 내에 표현되어 있는 데이터가 중요한 특징을 지닌다. 정형화된 구조 측면이 강하고, 혼합 요소가 많지 않아 내용과 구조의 혼재가 적고, 형제 엘리먼트간 순서와 같은 정보는 중요하지 않게 다루어지는 경우가 많다. 데이터 중심 XML 예로는 전자상거래 문서, 비행 스케줄, 식당 메뉴 등을 들 수 있다. <표 2>는 문서 중심 XML과 데이터 중심 XML 문서 예를 비교하여 보인 것이다..

<표 2> 문서 중심 XML과 데이터 중심 XML 문서 예

데이터중심XML	문서중심XML
<Flights> <Flight> <Airline>AEC Airways</Airline> <Origin>Dallas</Origin> <Destination>Fort Worth</Destination> <Flight> <Departure>02:15</Departure> <Arrival>03:16</Arrival> </Flight> <Flight> <Departure>11:15</Departure> <Arrival>11:16</Arrival> </Flight> <Flight> <Departure>13:15</Departure> <Arrival>13:16</Arrival> </Flight> </Flights>	<Product> <Name>Turkey Wrench</Name> <Developer>Full Fabrication Labs, Inc.</Developer> <Summary>Like a monkey wrench, but not as big.</Summary> <Description> <Para>The turkey wrench, which comes in both right-and left-handed versions (stylish options!), is made of the finest stainless steel. The Read-grip rubberized handle quickly adapts to your hands, even in the greasiest situations. Adjustment is possible through a variety of custom dials. </Para> <Para>You can:</Para> <List> <Item><Link URL="Order.html">Order your own turkey wrench</Link> </Item> <Item><Link URL="Wrenches.html">Read more about wrenches</Link> </Item> </List> <Para>The turkey wrench costsjust $19.99and, if you order now, comes with anhand-crafted shrimp hammer as a bonus gift.</Para> </Description> </Product>

(2) 관계형 DBMS 기반 문서 저장 시스템

RDBMS를 이용한 XML 문서 저장 시스템은 XML에 대한 저장 기법에 대한 연구가 시작되면서부터 이용된 형태로, 기존에 개발된 많은 응용들과의 연동이 쉽다는 장점이 있다[11]. 하지만 문서 중심 XML이 가지고 있는 의미와 구조 정보를 완벽하게 표현하기가 어렵고, 검색 결과 XML 문서를 재구성할 때 변환에 대한 비용이 추가적으로 발생하는 문제점을 가지고 있다.

(3) 객체지향 DBMS 기반 문서 저장 시스템

객체지향 데이터 모델의 개념은 XML 문서의 논리적 구조 정보와 많은 공통점을 지니고 있기 때문에 관계형 DBMS에 비해 XML 문서의 의미를 표현하기 쉽고, XML 정보를 객체로 변환하는데 소요되는 비용을 줄일 수 있는 장점이 있다[13]. 그러나 XML 문서에 대한 질의를 객체지향 질의어(OOQ)로 변환해야 하고 XML 문서의 갱신이 복잡해지는 문제점을 지니게 된다.

(4) 전용 저장 시스템

최근에는 기존의 상용 관계형 DBMS나 객체지향 DBMS를 기반으로 XML 저장 시스템을 구축할 때 발생하는 여러 문제를 해결하기 위해 XML 문서를 위한 전용 저장 시스템에 대한 연구가 활발히 이루어지고 있다. XML 문서의 전용 저장 시스템은 XML 문서가 내포하고 있는 내용과 의미뿐만 아니라 그 구조를 완전하게 표현하기 위해, XML 문서의 데이터 모델을 논리적으로 정의하고, 이 모델에 따라 XML 문서를 저장하고 검색함으로써 XML 문서의 재구성 비용이나 추가적 질의 처리 비용을 줄이고 있다. 현재까지 연구된 디스크 기반 XML 전용 저장 시스템으로는 Natix, Lore, Tamino 등이 대표적이다[8][14]. 그러나 이러한 기존의 XML 전용 저장 시스템들은 디스크 기반에서 이루어졌기 때문에 실시간 응용이나 무선 응용 처리에 대한 한계점을 가지고 있다.

2.2 XML 문서의 인덱싱

(1) 구조적 검색

구조적 검색은 그 처리 방향성에 따라, 조상/후손, 부모/자식 검색과 같은 계층적(수직적) 관계 질의와, 형제 검색과 같은 수평적 관계 질의로 분류되며, 순서 또는 레벨 등의 부가적 조건을 포함하여 상세 분류될 수 있다[10]. 구조적 검색 질의의 대상은 문서 또는 엘리먼트가 될 수 있으며, XML 문서에서의 최소 검색 단위인 엘리먼트가 중요하게 다루어진다. .

구조적 검색 질의는 XPath와 같은 경로식을 이용하여 표현된다[9]. Xpath는 단일 XML 문서내의 엘리먼트 등에 대한 접근을 제공한다. 구조적 문서를 지원하는 일반적인 정보 검색 시스템은 문서 컬렉션 전체를 대상으로 특정 조건을 만족하는 문서 또는 엘리먼트를 검색해야 하므로 XPath를 기반으로 하는 XML 질의어의 확장이 필요하다.

구조적 검색 질의의 형태는 '<검색 기준 엘리먼트> 방향 조건<검색 대상 엘리먼트>'와 같이 표현될 수 있다. 검색 조건 엘리먼트는 질의의 시작이 되는 엘리먼트이며, 검색 대상 엘리먼트는 질의의 최종 목표가 되는 엘리먼트이다. 방향 조건은 부모/자식, 조상/후손, 형제 등의 순으로 서술된다.

이러한 구조적 검색 질의는 XML 문서에 대해 각 엘리먼트를 트리 형태의 노드로 매핑한 후, 각 노드에 식별자를 할당하고 인덱스를 구축하여, 주어진 경로식 조건에 따라 XML 문서 트리를 상향식 또는 하향식으로 순회하여 처리할 수 있으며, 이때, 반복적 혹은 재귀적 처리를 필요로 한다. 그러나 이 경우 트리 전체를 순회하여야 하기 때문에 처리 영역이 넓게 되며, 대상 엘리먼트가 아닌 엘리먼트들을 제거하기 위한 추가적인 필터링 과정이 요구되는 등 효율성이 떨어진다. Lore 시스템에서는 루트로부터 시작하는 단순 경로들에 대한 구조적 요약

정보를 유지하는 DataGuide[8]를 바탕으로 주어진 경로식으로 도달할 수 있는 모든 엘리먼트들을 유지함으로써, 트리를 직접 순회하지 않고 해당 엘리먼트만을 접근할 수 있도록 하고 있으나, 정규 경로식 등을 포함하거나 루트가 아닌 엘리먼트로부터 시작되는 질의 처리에 대해서는 언급하지 않고 있다.

연구[2]에서는 트리로 표현된 XML 문서의 각 노드에 구조적 정보를 포함하는 특별한 식별자를 부여하여 트리의 순회 없이 구조적 검색 질의 처리가 가능하도록 설계하였다. 그러나 일부 구조적 검색 질의 유형에는 효율적이나 다른 유형의 처리 과정에서는 추가적인 연산 및 반복적 접근이 요구된다. 연구[2]에서는 한 문서내의 엘리먼트에 대한 식별자 값을 부여하고 처리하는 방안에 대해서만 언급하고, 문서 컬렉션에 대한 처리는 다루고 있지 않다. 연구[3]에서는 임의의 두 엘리먼트의 이름이 주어진 경우에만 엘리먼트간의 자손/후손 관계를 구조적 정보를 지닌 식별자를 통해 빠르게 계산할 수 있으나, 부모/조상의 구별이 없는 경우나 엘리먼트 이름이 하나만 주어지면, 형제 검색, 부모 및 자식 검색이 반복적으로 발생하게 된다. 또한, 엘리먼트 이름을 기반으로 검색하기 때문에 사용자가 원하지 않는 엘리먼트 들을 구조적 검색 대상에서 제거하는 필터링 과정이 항상 요구된다. 또한, [2][3]의 방법은 XML 문서에 대한 엘리먼트 삽입 및 삭제와 같은 구조적 변경이 자주 발생하는 환경에서는 구조적 정보를 포함하는 엘리먼트 식별자에 대해 많은 재 계산이 요구되므로, 동적인 환경에서는 매우 부적합하다.

(2) XLINK를 지원하는 인덱싱

연구[16]에서는 XLink의 링크 속성 중 actuate와 show를 이용한 가중치 부여 방법을 사용한다. 이를 이용해 링크 된 문서의 중요성을 판단해서 우선 순위를 부여한다.

인덱싱 시에는 역화일을 이용한 색인어 기반의 포스팅(posting)을 생성하고, 문서의 링크 정보는 <그림 1>과 같은 링크 정보 화일에 따로 저장해 놓는다.

element	id	frequency	href	Remote_link	Insert_link

<그림 1> 링크 정보 화일

문서에서 인덱싱이 끝나면, 임시 화일에 들어있는 링크 정보를 이용하여 링크 된 문서에서의 엘리먼트 정보를 포스팅에 삽입한다. 이러한 방법으로 링크 된 모든 문서들을 인덱싱 하기 위해서 링크 식별자 테이블과 원격 문서의 링크에 대한 가중치 부여 테이블의 가중치 값이 '0'일 때까지 반복적으로 이루어진다. 그러나 Remote_Link 값을 결정해 주는 판단 기준이 모호하며, 가중치를 부여가 단순 링크(simple link)와 확장링크(extended link)에서의 인라인 링크(inline link)만을 가정하였기 때문에 최신의 XLink 표준을 만족시키지 못하게 되며, 문서 혹은 엘리먼트 단위의 검색이 용이하지 않다.

연구[17]에서는 기존 XML 문서 데이터 모델에, 링크 정보를 포함한 데이터 모델을 추가하여 XLink로 표현된 XML 문서 데이터 모델을 새롭게 제안하였다. 그러나 제안된 데이터 모델은 단방향 링크만을 지원하여 XLink가 가지는 확장된 링크를 표현하지 못하는 단점이 있다.

3. 주기억장치기반의 XML 저장시스템

3.1 논리적 데이터 모델

본 논문에서는 문서 중심 XML 문서를 위한 데이터 모델을 정의한다. 제안한 데이터 모델은 XML 문서 자체가 가지는 원래의 의미와 그 구조적 특성을 반영할 수 있도록 설계하였다. 데이터 모델은 크게 노드와 링크로 구성되고, 노드에 레이블된 방향성과 순서를 지닌 그래프 형태로 표현한다. 노드 간의 순서 정보는 그래프 상에서 왼쪽 노드에서부터 오른쪽 노드 순으로 표현된다.

(1) 노드

노드는 XML 문서의 주요 구성 요소인 엘리먼트, 애트리뷰트, 데이터, 엔터티 등을 표현하며, 엘리먼트 노드, 애트리뷰트 노드, 데이터 노드, 엔터티 노드, 참조 노드 타입으로 구분된다. 노드는 한 문서 내에서 유일하게 구별되는 NID 값과 노드 자체의 의미를 설명하는 레이블 정보를 갖는다. 레이블은 보통 엘리먼트의 태그 이름, 애트리뷰트의 태그 이름, 엔터티 이름을 이용한다.

(2) 링크

링크는 노드간의 다양한 관계를 표현하며, 중첩 타입(Nested type) 링크, 참조 타입(Reference type)

링크, 링키지 타입(Linkage type) 링크로 분류된다. 중첩 타입 링크는 엘리먼트와 그 자식 엘리먼트 간의 관계나 엘리먼트와 엘리먼트 내의 텍스트 정보와의 관계, 엘리먼트와 애트리뷰트 간의 관계를 나타낸다. 참조 타입 링크는 내부 참조 관계와 외부 참조 관계로 나뉘는데, 내부 참조 관계는 엘리먼트와 다른 엘리먼트간의 참조 관계를 나타내며, IDREF 타입의 애트리뷰트를 가지는 엘리먼트가 이 애트리뷰트 값과 일치하는 ID 타입의 애트리뷰트를 가지는 엘리먼트를 참조하는 의미를 지닌다. 외부 참조 관계는 확장 링크 사용 시, href 애트리뷰트와 같은 locator 정보를 이용해 엘리먼트와 원격 자원 간의 참조 관계를 나타낸다. 링키지 타입의 링크는 XML 문서 내의 일반 Xlink를 이용한 명확한 링킹 정보를 표현한다.

(3) 데이터 모델 변환 예

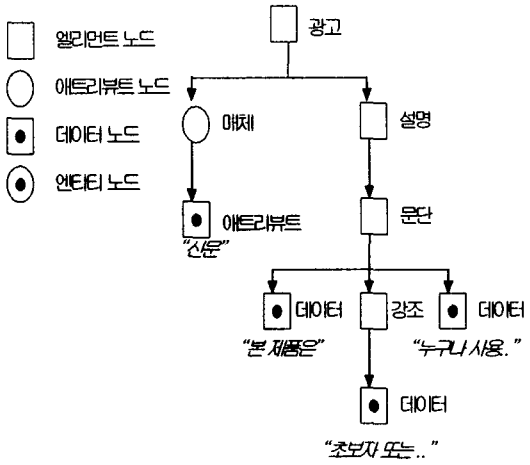
<그림 2>는 간단한 문서 중심 XML의 예를 타나내고 있다. 이 XML 문서는 XML 문서의 일반적 구성 요소 중 엘리먼트, 애트리뷰트, 엔터티를 포함하고 있다. <그림 2>의 XML 문서를 제안한 논리적 모델로 변환하게 되면 노드 레이블된 방향성과 순서가 존재하는 그래프 형태로 표현 가능하다. <그림 3>

광고 문구 예

```
<광고 매체="신문">
  <설명>
    <문단> 본 제품은<강조>초보자 또는 기술자</강조>
           누구나 사용할 수 있는 공구입니다.</문단>
  </설명>
</광고>
```

<그림 2> 간단한 XML 문서 예

<그림 2>의 "문단" 엘리먼트의 경우에는 많은 PCDATA 텍스트와 또 다른 엘리먼트 노드를 자식 노드로서 가지고 있는데, "문단" 엘리먼트의 자식 노드들 간의 순서 정보는 문서 중심 XML에서 중요한 정보라 할 수 있다. 이러한 형제 노드 간의 순서는 <그림 3>의 모델링 변환 후 결과에도 그대로 반영되고 있음을 확인할 수 있다. <그림 3>에서의 모델링 결과는 <그림 2>의 XML 문서가 가지는 의미와 그 구조적 정보를 그대로 표현하고 있다.



<그림 3> 논리적 데이터 모델로의 변환 결과

3.2 XML 전용 저장 시스템 설계

본 논문에서 제안한 XML 전용 저장 시스템은 문서 중심 XML의 의미와 구조 정보를 그대로 유지하면서 효과적인 저장과 검색을 지원하며, 실시간 처리를 요구하는 이동 통신이나, 무선 응용 분야에 활용 가능하도록 주기억장치 기반으로 설계한다.

(1) 시스템 구성

제안된 XML 문서 저장 시스템은 크게 XML 문서를 입력받아 문서 중심 XML을 위한 논리적 데이터 모델로 변환하는 변환 모듈(Conversion Module)과 변환된 데이터 모델을 실제 메모리에 저장하는 하부 저장 모듈(Repository Module), 사용자의 질의를 실제로 처리하기 위한 질의 처리 모듈(Query Processing Module)로 구성되어 있다.

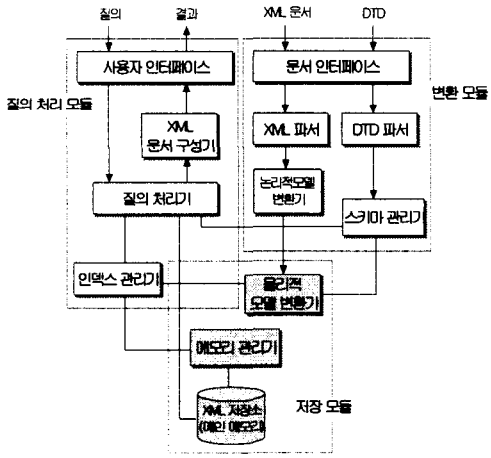
<그림 4>는 본 논문에서 제안한 XML 문서 전용 저장 시스템의 전체 구성도이다.

변환 모듈은 XML 문서를 입력받아 앞서 제안한 논리적 모델로 변환하고, 입력받은 DTD 정보를 XML 문서 저장 및 질의 처리를 위해 저장 및 관리한다. 변환 모듈은 XML 문서와 DTD 정보를 입력받는 문서 인터페이스, XML과 DTD의 유효성을 검사하는 XML 파서와 DTD 파서, XML 문서를 데이터 모델로 변환하는 논리적 모델 변환기, 스키마 관리기로 구성되어 있다.

질의 처리 모듈은 사용자로부터 XML 문서에 대한 여러 질의를 입력받아 실제적인 질의 처리를 담당한다. 질의 처리 모듈은 사용자로부터 질의를 입

력받는 사용자 인터페이스, 질의 처리를 담당하는 질의 처리기, 질의 최적화를 위한 인덱스 관리기, 결과 XML 문서를 구성해주는 XML 문서 구성기로 구성된다.

저장 모듈은 XML 문서를 주기억 장치에 저장하고 이를 관리한다. 저장 모듈은 크게 물리적 모델 변환기, 메모리 관리기, 물리적 저장소로 구성되어 있다. 물리적 모델 변환기는 변환 모듈에 의해 구성된 XML 문서의 논리적 모델을 물리적 저장 노드 구조로 변환한다. 메모리 관리기는 주기억 장치 내에서 동적으로 메모리를 할당하고 반환하는 역할을 담당한다. 물리적 저장소는 주기억 장치 내에 XML 문서를 저장하는 공간을 의미한다. 주기억 장치에 XML 문서가 저장되는 형태는 물리적 노드를 그 기본 단위로 한다.



<그림 4> 시스템 전체 구성도

물리적 노드는 내부 노드와 리프 노드로 나뉘며, 내부 노드는 앞서 설명한 논리적 데이터 모델에서 엘리먼트 노드와 애트리뷰트 노드를 의미하고, 리프 노드는 자식 노드를 가지지 않고 데이터 내용을 담고 있는 노드로서 논리적 데이터 모델의 데이터 노드, 엔터티 노드, 참조 노드가 이에 속한다.

Node ID	Node Type	Node Label	Offset	Point to Parent	Point to Child	Point to Sibling

<그림 5> 내부 노드 구조

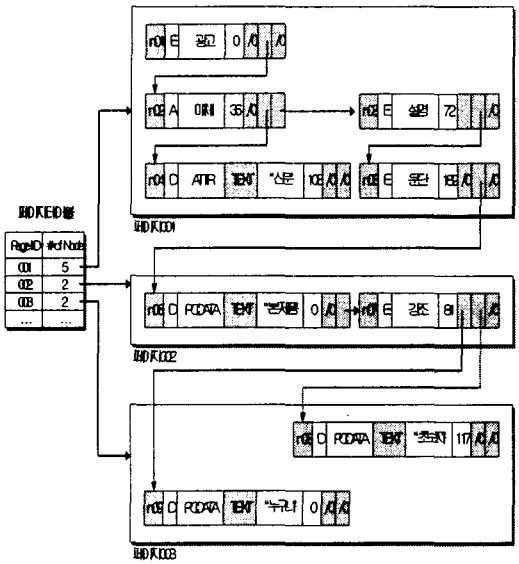
<그림 5>는 노드의 내부 구조를 나타낸다. 노드 구조에는 노드를 문서 내에서 유일하게 구별해주는 노드 아이디(node id), 저장되는 노드의 형태를 나타내는 노드 타입(node type), 노드를 설명하는 노드 레이블(node label), 부모 노드를 가리키는 포인터, 첫 번째 자식 노드를 가리키는 포인터, 첫 번째 형제 노드를 가리키는 포인터로 구성된다.

Node ID	Node Type	Node Label	Content Type	Offset	Point to Parent	Point to Next Sibling

<그림 6> 리프 노드 구조

<그림 6>은 리프 노드의 구조이다. 리프 노드 구조는 노드 아이디(node id), 리프 노드가 논리적 데이터 모델로서 엔터티 노드인지 데이터 노드인지, 참조 노드인지를 나타내는 노드 타입 (node type), 노드의 데이터 정보 타입 (content type), 노드의 실제 데이터 정보(content), 부모 노드를 가리키는 포인터, 첫 번째 형제 노드를 가리키는 포인터로 구성된다.

페이지는 주기억 장치 내에 기본 저장 구조로서, 하나의 페이지 내에는 여러 개의 물리적 노드가 저장될 수 있다. 일반적인 디스크 기반 저장 시스템의 경우에는 부모-자식 관계 또는 형제 관계와 같이 관련 있는 데이터들을 클러스터링하여 동일 페이지에 저장하여 전체 성능을 향상시키고 있다. 그러나 주기억 장치 기반 저장 시스템은 디스크 I/O의 부담이 없으므로 데이터 클러스터링을 고려하지 않았다. 따라서 데이터 클러스터링에 대해 부가적인 오버헤드 및 단편화 문제를 줄일 수 있다. 또한 문서 중심 XML에서는 형제 관계의 순서 정보가 중요하게 이용되는데, 본 논문에서 제안한 물리적 노드 구조는 모든 형제 노드에 대한 메모리 포인터를 유지하지 않고, 첫 번째 형제 노드에 대한 메모리 포인터만을 유지하고 있기 때문에 불필요한 기억 공간의 낭비를 줄일 수 있고, 형제 노드의 순차적인 검색이 자연스럽게 이루어진다.



<그림 7> 페이지 내에 물리적 노드를 저장하는 실제 예

<그림 7>에서는 <그림 2>의 간단한 XML 문서를 실제로 주기억 장치 내 페이지에 저장하는 예를 보여준다. 점선은 형제 노드에 대한 관계를 표현하는 메모리 포인터를 나타낸 것이고, 실선은 첫 번째 자식 노드에 대한 관계를 표현하는 메모리 포인터이다. 노드 ID "n07"을 가지는 노드는 첫 번째 자식 노드로서 노드 ID "n09"의 노드를 가리키고, 첫 번째 형제 노드로서 노드 ID "n08"를 갖는 노드를 가리킨다. 이때 이 세 노드는 특별한 관련성을 가지고 있으나 실제 저장 예에서는 다른 페이지에 저장되고 있음을 <그림 7>에서 확인할 수 있다.

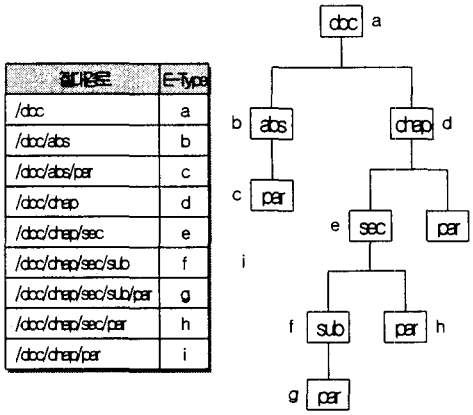
페이지 테이블은 여러 페이지를 관리하기 위한 방법으로서 페이지 ID, 페이지 내에 저장되어 있는 노드 수, 노드에 대한 포인터로 구성되어 있다. 문서 테이블은 여러 페이지 내에 저장되어 있는 여러 XML 문서에 대한 관리를 위한 것으로, 문서 테이블의 각 엔트리는 전체 시스템에서 유일하게 구별되는 문서 ID, 해당 페이지에 저장되어 있는 첫 번째 노드에 대한 포인터로 구성된다.

4. 확장된 인덱스 기법

4.1구조적검색지원

(1) 경로 정보 테이블의 구성

주어진 DTD 등을 분석하여 <그림 8>의 오른쪽처럼 명세된 논리적 구조 정보의 단순 절대 경로들을 바탕으로 <그림 8>의 왼쪽과 같이 경로 정보 테이블(Path Information Table)을 구성한다.



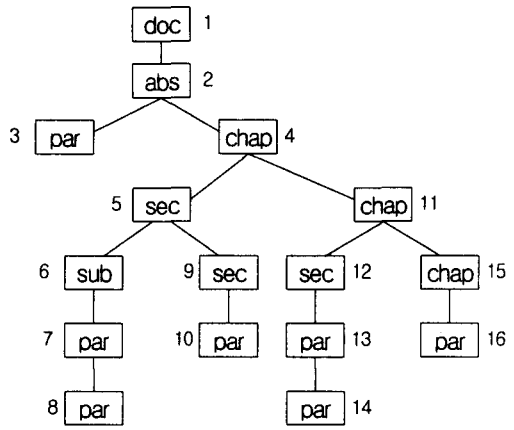
절대경로	E-Type
/doc	a
/doc/abs	b
/doc/abs/par	c
/doc/chap	d
/doc/chap/sec	e
/doc/chap/sec/sub	f
/doc/chap/sec/sub/par	g
/doc/chap/sec/par	h
/doc/chap/par	i

<그림 8> 논리적 구조와 경로 정보 테이블

<그림 9>에서 엘리먼트 타입 즉, ETY(Element Type)는 각 경로마다 고유하게 할당된 값으로 해당 경로의 마지막 엘리먼트에 대한 유일한 식별자 값을 의미한다.

(2) XML 문서 모델링 및 인덱스 구조

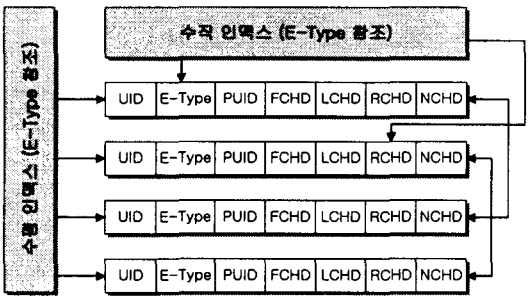
먼저 소스 XML 문서를 <그림 9>와 같이 왼쪽 자식 오른쪽 형제의 구조를 갖는 이진 트리 형태로 표현하고, 각 엘리먼트에 임의의 고유한 식별자(UID)를 할당한다. 그리고 각 엘리먼트에 대해 <UID, E-TYPE, PUID, FCHD, LCHD, RCHD, NCHD> 엔트리를 매핑한다. 여기서 PUID, FCHD, LCHD, RCHD는 각각 부모 엘리먼트의 UID, 첫 번째 자식의 UID, 바로 왼쪽 형제의 UID, 그리고 바로 오른쪽 형제의 UID를 의미하고, NCHD는 자식 엘리먼트의 수를 의미한다. 예를 들어 <그림 9>에서 UID가 4인 'chap' 엘리먼트에 대한 엔트리는 <4, d, 1, 5, 2, 11, 2>가 된다.



<그림 9> XML 문서 트리의 예

모든 엘리먼트에 대해 생성된 엔트리들은 UID를 기준으로 역 과정을 통하여 <그림 10>과 같은 역 인덱스로 구성되는데, 이때 동일한 E-TYPE를 갖는 엔트리들을 그룹핑하기 위해 E-TYPE을 키로 하는 새로운 차원의 인덱스 파일을 추가한다. UID를 키로하는 인덱스 파일을 '수직 인덱스', E-TYPE을 키로 하는 인덱스를 '수평 인덱스'라 한다. 즉, 수직 인덱스를 통해 특정 E-TYPE에 대한 모든 엘리먼트만을 직접 접근할 수 있다. 인덱스의 구현에는 일반적으로 사용되는 B+ 트리 등을 활용할 수 있다.

이렇게 인덱스를 구성할 경우, 특정 엘리먼트에 대한 삭제 또는 새로운 엘리먼트의 삽입에 대해 유연하게 대처할 수 있으며[10], 기존 방법의 문제점이었던 삭제 또는 삽입된 노드에 대한 오른쪽 형제를 기준으로 하는 서브트리 전체에 대한 UID의 변경이 발생하지 않는다. 따라서 삭제 또는 삽입이 발생하는 엘리먼트에 직접 연관된 엘리먼트들에 대한 UID만 적절하게 수정하면 되므로 갱신에 대한 오버헤드를 대폭 줄일 수 있다.



<그림 10> 제안 인덱스 구조

(3) 구조적 검색 질의 처리 방안

본 논문에서는 사용자로부터 주어진 질의에 포함된 엘리먼트 이름과 타입 정보를 이용하여 트리 순회를 하지 않거나 최소화하여 구조적 검색 질의를 처리할 수 있도록 하였다.

이를 위해 사용자 질의를 적절한 스트링 형태로 표현하고, <그림 8>에서 설계된 경로 정보 테이블의 단순절대경로 필드의 값들과 스트링 매칭을 통해 최종 결과에 포함될 엘리먼트의 타입 값을 추출하여 목표 대상만을 직접 접근하게 함으로써 질의 처리 영역을 축소하였다. 스트링 매칭을 위한 표현에 사용된 표기 중 와일드 카드 문자인 '*'는 0개 이상의 문자를 의미한다. 각 질의의 최종 반환 값은 UID의 집합으로 한다.

가.자식검색질의

<그림 9>와 같이 트리로 표현된 XML 문서에서 'chap' 엘리먼트의 자식 중 'sec' 엘리먼트를 검색하라는 질의(//chap/sec)는 다음과 같이 처리된다. 여기서 검색 기준 엘리먼트는 'chap'가 되며 검색 대상 엘리먼트는 'sec'이 된다. 또한, 괄호내의 표현은 질의문에 대한 이해를 돕기 위한 XPath에 따른 경로식 표현이다. 먼저 사용자 질의를 '*/chap/sec' 스트링으로 변환하고 <그림8>의 경로 정보 테이블의 각 단순절대경로 필드에 대한 각 엔트리의 스트링 값들과 스트링 매칭을 수행하여 '/chap/sec'으로 끝나는 엔트리들만을 추출하여 후보 E-TYPE 집합을 추출한다(이 예에서는{e}). 추출된 각 E-TYPE값을 키로, 수직 인덱스 의 해당 E-TYPE 값을 갖는 포스팅들만을 접근하여 UID들을 최종 결과 집합에 포함시켜 반환한다(이 예에서는 {5, 9, 12}).

결국, 'chap'의 자식이면서 'sec'이 아닌 타입을 갖는 엘리먼트들 즉, 주어진 질의와는 다른 경로상에 있는 엘리먼트들에 대한 접근이나 필터링 과정이 요구되지 않는다.

나.후손검색질의

'chap' 엘리먼트의 후손 중 모든 'par' 엘리먼트를 검색하라는 질의(//chap/descendant::par) 는 다음과 같이 처리된다. 사용자 질의를 '*/chap*/par' 스트링으로 변환하고 경로 정보 테이블에서 스트링 매칭을 통해 후보 E-TYPE 집합 {g, h, i}를 구해 낸다. 각 ETY를 키로 수직 인덱스를 통해 포스팅들을 접근하여 UID들의 집합 즉, {7, 8}, {10, 13,14}, 그리고 {16}를 추출하고 최종적으로 이들에 대해 합집합 연

산을 수행하면 최종 결과 집합인 {7, 8, 10, 13, 14, 16}을 구할 수 있다. 여기서도 후손 검색에 대한 이진 트리 순회 등의 별도 필터링 과정 없이 목표 엘리먼트 타입에 대한 엘리먼트들 만을 직접 접근하여 결과를 구한다.

다.부모검색질의

'par' 엘리먼트의 부모 중 'sec' 엘리먼트를 검색하라는 질의(//par/parent::sec)는 다음과 같이 처리된다. 'par' 엘리먼트를 자식으로 갖고 있는 'sec' 엘리먼트를 검색하라는 질의(//sec[child:par])도 이 범주에 속한다. 사용자 질의를 '*/sec/par' 형태로 변환하고, 경로 정보 테이블과의 스트링 매칭을 통해 E-TYPE 후보 집합 {h}를 추출한다. 그리고 E-TYPE 값을 키로 하여 수직 인덱스를 참조하여, 연결된 포스팅들을 접근한 후, 각 포스팅에서 PUID들의 집합인 {9, 12}를 추출하여 반환한다. 이 경우에도 타겟 엘리먼트의 타입에 대한 필터링이 없이 바로 원하는 엘리먼트 타입에 대한 엘리먼트들만을 구해낼 수 있다.

라.조상검색질의

'par 엘리먼트의 모든 조상 중 'sec' 엘리먼트들 검색하라 는 질의(//par/ancestor::sec)는 다음과 같이 처리된다. 먼저 사용자 질의를 '*/sec*/par' 형태로 변환하고, E-TYPE 후보 집합 {g, h}을 추출한다. 그리고 각 E-TYPE 값을 키로 하여 수직 인덱스를 통하여 포스팅들을 접근한 후, PUID들을 추출하고 결과 집합에 포함시킨다. 추출된 PUID를 키로 수평 인덱스를 이용하여 엔트리 포스팅을 접근하여 PUID 추출한다. 이러한 과정을 각 E-TYPE에 대한 sec 레벨까지 반복 수행 후, 그때의 UID만 최종 결과 엘리먼트 집합에 포함시키면 {5, 9, 12}을 구할 수 있다.

마.형제검색질의

경로식은 표현 자체가 계층적인 수직 관계성만을 정의하므로, 수평적 관계에 대한 형제 질의는 경로식을 이용하여 처리할 수 없다. 우선, 경로식에 대한 스트링 매칭을 통해 검색 기준 엘리먼트 타입에 대한 후보 집합을 추출하고, 포스팅내의 FCHD 추출 후 LCHD 또는 RCHD에 대한 반복적인 접근으로 최종 결과 UID 집합을 구해낸다.

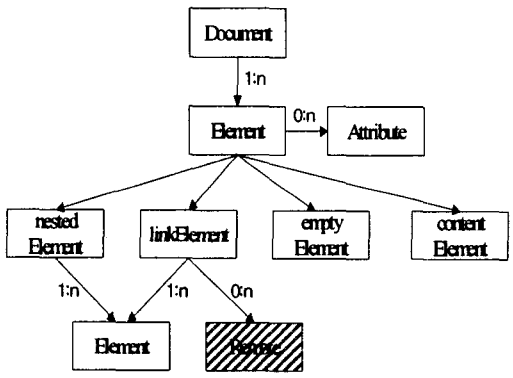
4.2 XLink 지원 인덱싱

기존 XLink의 애트리뷰트를 이용한 인덱스는 Actuate, Show등의 애트리뷰트만을 이용하였기 때문에 실제 의미정보를 가지고 있는 애트리뷰트에 대한 인덱싱은 이루어지지 않았다. 또한 링크 되어 있는 여러 문서 중에서 어떠한 문서를 색인 할 지에 대한 명확한 판단 기준이 분명하게 정의되지 않았다[16]. 한편, [2]에서는 한 문서에 대한 인덱싱은 가능하나 웹 환경에서와 같은 여러 문서의 인덱싱은 이루어지지 않았다[18]. 따라서 본 절에서는 타입과 애트리뷰트에 대한 인덱싱과 outbound Link와 inbound link 모두를 지원하는 인덱싱 기법을 제시한다.

(1) 데이터 모델

가.구조

XLink로 표현된 링크 정보를 고려한 XML 문서의 데이터 모델을 위한 구조적 측면은 <그림 11>과 같다.



<그림 11> 링크 정보를 포함한 XML 문서의 데이터 모델

하나의 문서(Document)는 문서의 논리적 구조 정보를 표현하는 엘리먼트(Element)와 1:n의 관계를 갖는다. 엘리먼트는 애트리뷰트(Attribute)와 0:n의 관계에 있고, 중첩 엘리먼트(nestedElement), 빈 엘리먼트(emptyElement), 내용 엘리먼트(contentElement)와 XLink로 표현된 링크 정보를 가질 수 있다. <그림 11>에서 엘리먼트와 링크 관계에 있는 대상을 Remote로 표현하였으며, 이는 원격 자원(remote resource)를 의미한다. 여기서 자원이란 문서, 이미지, 오디오, 화일 등의 도달 가능한 정보의 단위를 의미[11]하며, 본 논문에서는 XML 문서 또는 엘리먼트로 그 의미를 한정하겠다. 링크

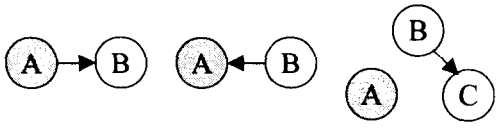
정보를 포함하고 있는 엘리먼트는 원격 자원과 0:n의 관계를 가진다. 내용은 엘리먼트가 포함하고 있는 텍스트 값이고, 애트리뷰트는 엘리먼트의 속성이다. 그리고, 중첩 엘리먼트는 엘리먼트 안에 내포되어 있는 엘리먼트 이다.

나.링크참조무결성

링크는 도달 불가능한 자원(resource)과 관계성을 가질 수 없다. 도달 불가능한 자원은 링크 정보에 명시된 자원의 위치에 해당 자원이 없는 것을 말한다. 본 논문에서는 XLink[11]에 근거하여 3가지 링크 참조 무결성(link referential integrity)을 정의한다.

A.outbound링크참조무결성

지역 자원에서 원격 자원으로의 링크 참조 관계에서 지역 자원의 링크 정보는 NOT NULL이고 유효한 원격 자원의 위치 정보를 가져야한다. <그림 12>의 a와 같이 A→B로 링크 된 경우, B에 대한 위치 정보를 가지고 있는 A의 링크 정보 값은 널 값을 가질 수 없고, 반드시 B에 대한 실제 위치 주소 값을 가져야 한다. 여기서 음영으로 표시된 원은 지역 자원을 의미하며, 흰색으로 표시된 원은 원격 자원을 의미한다.



a. outbound link b. inbound link c. third-party link

<그림 12> 링크 참조 관계 예

B.inbound링크참조무결성

원격 자원에서 지역 자원으로의 링크 참조 관계에서 지역 자원의 링크 정보는 NOT NULL이고 유효한 원격 자원의 위치 정보를 가져야한다. <그림 12>의 b와 같이 B→A로 링크 되어 있을 경우, A에 정의된 위치 정보 값은 B에 대한 유효한 위치 정보 값을 가져야 한다. 즉, B는 A의 위치 정보 값에서 명시한 위치에 반드시 존재해야 한다.

C.third-Party링크참조무결성

원격 자원에서 원격 자원으로의 링크 참조 관계를 정의하는 지역 자원의 링크 정보는 NOT NULL이거나 유효한 원격 자원의 위치 정보를 가져야한다. <그림 12>의 c와 같이 B→C로 링크 되어 있고, A가 이에 대한 링크 정보를 가지고 있을 경우, A에 정의된 링크 정보 값은 반드시 유효한 B와 C의 위치 정보 값을 가져야 한다. 즉, B, C는 A의 링크 정보 값에서 명시한 위치에 반드시 존재해야 한다.

다.링크정보화일

본 절에서는 XML 문서에서 링크 정보를 고려하여 여러 문서간의 관계성을 이용한 검색 처리에 적합한 인덱스 구조를 제안한다. 인덱스의 기본적인 구조는 <그림 13>과 같다.

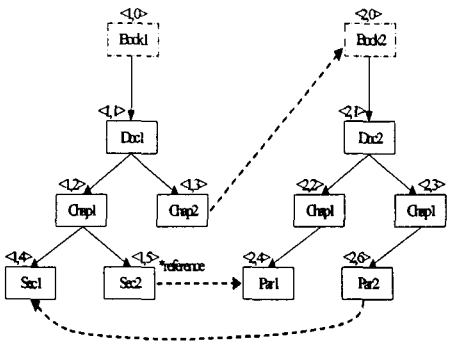
문서의 엘리먼트는 <DID, UID>쌍을 엘리먼트의 고유 식별자로 이용한다. DID는 문서 식별자이며, UID는 한 문서내의 엘리먼트에 할당되는 정수의 식별자로 모두 숫자 값을 갖는다.

fromName	toName	fromID	toID	title

<그림 13> 링크 정보 화일의 구조

<그림 13>에서 fromID는 링크 참조 관계가 시작되는 자원에 대한 식별자를 의미하며, toID는 링크 참조 관계의 목표가 되는 자원에 대한 식별자를 의미한다. fromID와 toID는 <DID, UID> 쌍을 값으로 갖는다. 여기서 자원이 문서 단위인 경우 UID 값은 '0' 값을 갖는다.

fromName은 fromID가 가리키고 있는 자원의 이름이고, toName은 toID가 가리키고 있는 자원의 이름이다. 단, 문서 단위의 링크 참조 관계의 경우 fromName와 toName은 문서의 화일 이름을 값으로 가진다. title은 fromName의 값인 엘리먼트가 가지는 애트리뷰트 중 링크를 사람이 식별할 수 있는 의미 정보를 가진 title 애트리뷰트의 내용을 의미한다. <그림 14>는 링크 참조 관계가 존재하는 두 문서간의 상태를 트리 형태로 나타낸 것이다. 여기서 방향을 갖는 점선으로 표시된 간선은 링크 관계를 나타낸다. *는 엘리먼트의 애트리뷰트 중 title 값을 의미한다.



<그림 14> 링크 관계가 포함된 두 문서

본 논문에서는 모든 문서가 가상의 루트 노드를 가지고 있다고 가정한다. 가상의 루트 노드에 대한 식별자는 <DID, UID>를 갖는데, DID는 문서 번호이고, UID는 '0' 값을 갖고, 노드의 이름은 화일 이름과 동일한 이름을 갖는다. 이는 문서 단위의 검색에 이용한다.

fromName	toName	fromID	toID	title
Chap2	Book2	<1,3>	<2,0>	
Sec2	Par1	<1,5>	<2,4>	reference
Par2	Sec1	<2,6>	<1,4>	
...	...	...	...	...

<표 3> 제안한 링크 정보 화일의 예

본 논문에서 제안한 인덱싱 기법은 XML 문서를 파싱하여 각 엘리먼트에 식별자를 할당한 후, 링크 정보가 정의된 엘리먼트 혹은 문서를 대상으로 <그림 13>에 각 필드에 대한 정보를 추출하여 인덱스 테이블로 생성한다. <표 3>은 <그림 13>에 대해 생성된 인덱스 테이블이다.

element	id	frequency	href	Remote_link	Insert_link
chap2	1	α	Book2.xml	Y	
Sec2	2	α	Book2.xml	Y	
Par2	2	α	Book1.xml/Doc1 /Chap1/Sec1	N	6
...	...	...	...	...	...

<표 4> 기존 링크 정보 화일의 예

<표 4>는 [16]에서 제안된 인덱싱 과정을 <그림 13>에 적용하여 생성한 링크 정보 화일의 예이다.

(2) 질의 예제

본 절에서는 본 제안한 인덱스와 [16]에서 제안한 인덱스를 이용한 몇 가지 대표적인 링크 질의 예와 이에 관한 처리 과정을 비교 설명한다.

A.특정엘리먼트가링크하고있는원격자원검색

예) Chap2가 링크하고 있는 문서를 검색하라.

① 기존 링크 정보 화일

'Chap2'가 링크하고 있는 문서를 찾기 위해서는 href 정보를 이용하여야 한다. 여기서는 'Book2.xml'이 검색된 결과로 반환된다. 하지만, href 정보는 링크하고 있는 리소스의 위치 및 경로 정보로서 링크의 대상이 되는 문서 혹은 엘리먼트 단위에 대한 보다 세부적인 검색이 용이하지 않다.

② 제안된 링크 정보 화일

링크 정보 화일에서 fromName의 값이 'Chap2'이고, toID의 UID 값이 '0'인 엔트리들을 추출한다. 추출된 엔트리들에 대해 toName 필드의 값만 추출하면 최종 결과 문서들의 집합이 된다. 여기서는 Chap2가 링크하고 있는 문서는 'Book2'인 것을 알 수 있다.

B.특정엘리먼트를링크하고있는원격자원검색

예) Sec1을 링크하고 있는 문서의 엘리먼트를 검색하라.

① 기존 링크 정보 화일

링크 정보 화일에서 'Sec1'을 링크하고 있는 문서의 특정 엘리먼트를 검색하기 위해서는 href 정보를 보고 판단해야 한다. 하지만, 앞의 예제에서 언급한 것처럼 href 정보만으로는 엘리먼트 단위의 검색이 어렵다. 따라서, 'Sec1'을 링크하고 있는 모든 문서의 엘리먼트를 효과적으로 검색할 수 없다.

② 제안된 링크 정보 화일

링크 정보 화일에서 toName의 값이 'Sec1'인 엔트리들을 추출한다. 추출된 엔트리들의 fromName과 fromID의 값을 최종 결과 집합으로 반환한다. 여기서는 Sec1을 링크하고 있는 문서는 식별자가 <2, 6>인 Par2 엘리먼트인 것을 알 수 있다.

C.title정보를이용한검색

예) 'reference' 타이틀을 가진 링크 설정된 두 개

의 엘리먼트를 검색하라.

① 기존 링크 정보 화일

링크 관계를 설명하는 title 애트리뷰트 정보를 저장하고 있지 않기 때문에 검색이 불가능하다.

② 제안된 링크 정보 화일

링크 정보 테이블의 title에서 값이 'reference'인 엔트리들을 추출한다. 추출된 엔트리들에 대해 fromName, fromID과 toName, toID 필드의 값을 최종 결과 집합으로 반환한다. 여기서는 식별자가 <1,5>인 'Sec2'인 엘리먼트와 식별자가 <2,4>인 'Par1' 엘리먼트가 검색된다.

5. 결론

효율적인 XML 문서의 저장과 검색을 위해서는 XML 문서의 특성을 고려하면서 XML 문서가 내포하는 의미 정보는 물론 구조적 정보를 완벽하게 표현할 수 있는 XML 용 저장 시스템의 개발이 요구된다. 특히 문서 중심 XML의 경우에는 그 중요성이 더욱 중요하다. 따라서 본 논문에서는 문서 중심 XML의 특성을 충분히 표현하기 위한 데이터 모델을 정의하고, 제안한 데이터 모델을 이용하여 주기억장치를 기반으로 하는 XML 전용 저장 시스템을 설계하였다.

또한, 엘리먼트 타입 정보를 포함한 사용자 질의를 적절한 스트링 형태로 변환 후, 경로 정보 테이블과의 스트링 매칭을 통해, 후보 엘리먼트 타입을 추출하여 구조적 검색 질의에 대한 처리 영역을 줄일 수 있는 방법을 제안하였다. 특히, 후손 검색의 경우 여러 개의 하향 경로를 반복적 혹은 재귀적으로 탐색하지 않고 목표 엘리먼트 들만을 접근하여 결과를 구하므로 처리시간을 단축할 수 있다. 본 논문에서 사용된 후보 엘리먼트 추출 접근 방식은 목표 엘리먼트 타입을 기반으로 한 내용 기반 질의 처리 시, 질의 처리 영역을 줄이기 위한 필터링 과정에서 목표 엘리먼트 타입 정보를 구하는 구체적인 대안으로 활용될 수 있다.

본 논문에서는 링크 정보를 가지고 있는 XML 문서의 데이터 모델을 정의하고, 문서 간 링크 정보가 가져야 할 참조 무결성을 제안하였다. 또한, 기존의 링크 정보를 이용한 인덱싱 기법이 가지는 최신 XLink의 표준의 특징을 모두 지원하지 않는 문제점을 해결하도록 링크 정보만을 따로 저장하고 있는 링크 정보 테이블 제안하였다. 또한, 대표적인 몇 가지 예제를 통해 링크 정보를 포함한 질의 처리 과정을 보였다.

참고 문헌

[1] T. Bray et al., "Extensible Markup Language (XML) 1.0 (Second Edition)", http://www.w3.org/TR/REC-xml, 2000

[2] Dongwook Shin, et al., "BUS: An Effective Indexing and Retrieval Scheme in Structured Documents", Proc.of the 3rd ACM Int'l Conference on Digital Libraries, 1998

[3] Q. Li, B. Moon, "Indexing and Querying XML Data for Regular Path Expressions", VLDB 2001

[4] Stefano Ceri, Piero fraternali and Stefano Paraboschi, "XML: Current Developments and Future Challenges for the Database Community", Proc. of the 7 th Int. Conf. on EDBT, pp. 3-17, March, 2000.

[5] Ronald Bourret, "XML and Databases", 2001.

[6] Alin Deutsch, Mary Fernandez, Dan Suciu, "Storing Semistructured Data With STORED", In Proceedings ACM SIGMOD International Conference on Management of Data, pp431-442, Philadephia, USA, June, 1999.

[7] Carl-Christian Kanne, Guido Moerkotte, "Efficient storage of XML data", Technical Report, University of Mannheim, 1999.

[8] R. Goldman, J. McHugh, and J. Widom, "Lore: A Database Management System for XML", Dr. Dobb's Journal, April, 2000.

[9] J. McHugh, J. Widom, Query Optimization for XML, VLDB 1999

[10] S.W. Kim, et al, "Indexing and Retrieval of XML-encoded Structured Documents in Dynamic Environment", Lecture Notes in Computer Science (LNCS) Vol.2480, 2002

[11] S. DeRose et al, "XML Linking Language (XLink) Version 1.0", http://www.w3.org/TR/xlink/, 2001

[12] J. Shanmugasundaram et al., Efficiently Publishing Relational Data as XML Document, VLDB, 2000.

[13]M. Klettke, H. Meyer, Managing XML Documents in Orient-Relational Databases, XML , 2000.

[14] Harald Schoning, "Tamino - A DBMS Designed for XML", Proc. of the 17 th International Conference on Data Engineering, 2000.

[15].Igor Tatarinov et al., Updating XML, ACM SIGMOD 2001

[16]김은정, 배종민, "XML 링크정보를 이용한 정보 검색 색인 기법의 설계", 한국정보처리학회 논문지, 제7권 제7호, 2000.7

[17]김용훈, 이강찬, 이규철, "링크 검색을 지원하는 XML 문서 질의 언어(XQL)의 설계", 한국정보과학회 가을 학술발표논문집, Vol.25.No.2, 1998

[18]Timothy Arnold-Moore, Ron Sacks-Davis, "Models for Structured Document Database Systems", Markup Technologies, 1998

신 판 섭

1992년 홍익대학교 전자계산학과 (이학사)

1994년 홍익대학교 전자계산학과 (이학석사)

2000년 홍익대학교 전자계산학과 (이학박사)

2002년 ~ 현재 : 대진대학교 컴퓨터공학과 전임강사

관심분야 : 분산 · 객체 데이터베이스, XML, 멀티미디어 시스템

E-Mail : psshin@daejin.ac.kr