XML 정규 경로식을 위한 유연한 질의 처리 시스템
(A Flexible Query Processing System for XML Regular Path Expressions)

김 대 일 † 김 기 창 †† 김 유 성 †††
(Dae Il Kim) (Ki Chang Kim) (Yoo-Sung Kim)

요 약 반 구조적(semi-structured)이며 표현의 유연성이 높은 XML(eXtensible Markup Language)은 인터넷에서의 정보 교환의 표준으로 자리잡았다. 많은 문서들이 XML로 작성되고 있으며 이러한 문서들에 대한 효율적인 저장과 검색에 대한 요구가 증가하고 있다. XML 문서들을 저장하는 백엔드(back-end)로서 대량의 정보처리가 가능하고 회복, 동시성 제어 등의 성숙된 기술들을 가지고 있는 관계형 데이타베이스를 이용하는 많은 연구들이 있다.

기존의 관계형 데이타베이스를 이용한 질의 처리 방법들에서는 다양한 질의에 대해 동일한 구조 정보와 기본 연산을 적용하여 질의 처리를 하기 때문에 특정한 질의 형태만을 효율적으로 처리할 수 있다. 본 논문에서는 질의를 분석하여 질의에 적합한 구조 정보와 기본 연산을 적용하는 유연한 질의 처리 시스템을 제안한다. 제안된 질의 처리 시스템은 정규 경로식 XML 질의를 분석하여 부모-자식 관계 파악에는 동등(equi)-조인을 사용하고 부모-자식 관계가 아닌 조상-자손 관계에는 θ-조인을 사용하여 다양한 정규 경로식 질의를 효율적으로 처리한다. 제안된 XML 질의 처리 시스템과 기존의 관계형 데이타베이스를 이용하는 질의 처리 시스템을 비교 실험한 결과 다양한 정규 경로식 질의 처리의 성능 향상을 보였다.

키워드 : 정규 경로식, XML 질의 처리, XML-관계형 데이타베이스 매핑

Abstract The eXtensible Markup Language(XML) is emerging as a standard format of data representation and exchange on the Internet. There have been researches about storing and retrieving XML documents using the relational database which has techniques in full growth about large data processing, recovery, concurrency control and so on. Since in previous systems same structure information and fundamental operation are used for processing of various kinds of XML queries, only some specific query can be efficiently processed not all types of query. In this paper, we propose a flexible query processing system. To process query efficiently, the proposed system analyzes regular path expression queries, and uses θ-join operation using region numbering values to check ancestor-descendent relationship and equi-join operation using parent's region start value to check parent-child relationship. Thus, the proposed system processes efficiently XML regular path expressions. From the experimental results, we show that proposed XML query processing system is more efficient than previous systems.

Key words : regular path expressions, XML query processing, XML-relational database mapping

1. 서 론

반 구조적(semi-structured) 언어인 XML(eXtensible Markup Language)[1]은 사용의 편리성과 표현의 유연성의 장점으로 인해 인터넷의 정보 교환의 표준으로 자리잡게 되었다. XML에 대한 응용의 발전에 따라 이를 저장, 검색할 수 있는 XML 저장 시스템에 대한 요구 또한 증가하고 있다. XML을 정보 교환의 표준으로 사용하는 전자상거래와 같은 응용에서는 문서 저장과 관리, 검색은 이제 막연한 필요가 아닌 현실적으로 직면한 문제가 되었으며, XML과 관련된 시스템의 도입에 많은 걸림돌이 되고 있다. 현재, Tamino[2], eXcelon[3]과 같은 XML 전용 데이타베이스 제품들이 상용화되어 사용

· 본 연구는 한국과학재단 지정 인천대학교 동북아전자물류연구센터 지원에 의한 것임
† 비 회 원 : 국민은행 카드영업팀 e-Business Unit
kimi1@passmail.to
†† 비 회 원 : 인하대학교, 정보통신공학부 교수
kchang@inha.ac.kr
††† 종신회원 : 인하대학교 정보통신공학부 교수
: yskim@inha.ac.kr
논문접수 2002년 8월 2일
심사완료 2003년 7월 18일

되고 있으나 검증되지 않은 안정성과 고가의 도입 비용 등으로 널리 쓰이지 못하고 있는 상황이다[4]. XML 문서들을 저장하는 백엔드(back-end)로서 대량의 정보처리가 가능하고 회복, 동시성 제어 등의 성숙된 기술들을 가지고 있는 관계형 데이타베이스 관리 시스템을 이용하려는 많은 연구들이 있다[5-10]. 이런 연구들은 관계형 데이타베이스의 상위 계층에 XML 문서들을 처리하는 미들웨어를 구축하여 적은 비용과 노력으로 안정적인 시스템을 구축할 수 있다는 장점으로부터 출발하였다.

XML 문서들에 대한 질의는 내용에 대한 질의뿐만 아니라 구조에 대한 질의도 가능해야 한다. XML 문서에 대한 질의 언어로는 XQL, XML-QL, XQuery 등 많은 언어들이 제안되었다. 이들 질의 언어들은 계층적인 문서의 구조 검색을 위해 XPath[11]와 같은 정규 경로식(regular path expression) 형태의 질의어를 기본으로 삼고 있다. 이러한 구조 질의를 처리하기 위해서는 계층적인 구조를 가지는 XML 문서들을 평면적인 구조의 관계형 테이블에 저장할 때 문서의 내용정보뿐만 아니라 문서의 구조정보를 추출하여 관계형 테이블에 함께 저장해야 한다. 구조 질의 처리에는 저장된 구조정보를 효율적으로 활용하는 SQL문으로 변환하는 것이 중요하다. 동일한 XML 문서와 질의라 하더라도 관계형 데이타베이스의 테이블들에 저장된 구조정보의 형태와 활용 방법에 따라 질의 처리의 효율성이 다르기 때문이다. 그러나 관계형 데이타베이스를 이용하는 XML 저장 시스템에 대한 기존의 연구들에서는 모든 질의에 대해 동일한 정보와 동일한 연산을 이용하여 질의를 처리하기 때문에 다양한 질의를 효율적으로 처리하지 못한다는 문제점이 있다.

본 논문에서는 XML 문서를 관리하는 관계형 데이타베이스에서 다양한 XML 정규 경로식 질의를 효율적으로 처리하기 위한 유연 질의 처리 시스템을 제안하였다. 제안된 시스템은 경로 정보를 정규화하여 저장하고, 질의의 형태에 따라 다른 조인 연산을 이용한다. 제안된 유연 질의 처리 기법에서는 정규 경로식 질의에 나타난 경로들과 조건 노드들의 관계를 분석하여 부모-자식의 관계 조사에는 동등(equi)-조인을 사용하고 부모-자식 관계가 아닌 조상-자손의 관계 조사에는 θ-조인을 사용하는 질의로 변환하여 처리의 효율성을 증진하였다. 따라서 질의의 형태에 따라 적절한 조인 방법과 구조 정보를 이용하는 효율적인 질의 처리를 가능하게 한다. 제안된 시스템과 기존의 시스템들을 비교 실험한 결과 질의에 따라 다른 조인 연산을 이용하는 제안된 질의 처리 시스템의 질의 처리 성능이 우수함을 알 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 관계형 데이타베이스를 이용한 XML 저장 시스템의 기존 연구들에 대해서 기술한다. 3장에서는 XML 저장 시스템의 문서 저장 모델을 설명한다. 4장에서는 제안된 질의 처리 시스템에 대해서 설명한다. 5장에서는 시스템의 구현 환경을 보이고 제안된 방법과 기존의 방법들을 비교 실험하여 제안된 시스템의 우수성을 보인다. 마지막으로 6장에서는 결론 및 향후 연구 방향을 논하여 본 논문을 맺는다.

2. 관련 연구

인터넷의 발달과 인터넷을 통한 정보 교환의 증가와 함께 정보를 표현하는 표준의 수단에 대한 요구가 증가하였다. 일찍이 사용되던 SGML은 사용의 어려움과 제한된 응용으로 인해 널리 사용되지 못했다. XML은 SGML과 같은 구조적 특성을 가지며 문서의 스키마 정보 없이 사용될 수 있고 사용자의 편리성으로 인터넷에서의 표준이 되었다. 또한 일반적인 문서 구조와 다르게 문서 내에 의미가 있는 태그를 사용자가 정의하여 사용할 수 있으며, 계층적인 구조로 태그들을 포함하여 문서의 구조와 의미를 기계와 사람이 쉽게 이해할 수 있는 형태를 가진다. XML 문서의 예와 문서 구조는 그림 1과 같다.

계층적이고 가변적인 XML 문서에 대한 질의 언어로는 XQL, XML-QL, XQuery, XPath등이 사용되고 있으며, 모두 XPath[11]와 같은 정규 경로식 질의를 기본으로 한다. 정규 경로식 질의는 파일 시스템의 디렉토리들을 탐색하듯 문서의 계층구조를 탐색한다. 이러한 질의를 처리하기 위해 관계형 데이타베이스를 이용한 XML 저장 시스템에 대한 기존의 연구들에서는 구조정보로서 각 노드간의 연결 정보, 노드의 경로 정보 그리고 노드간의 포함 관계에 대한 영역 정보를 이용하는 방법들이 있다[6-9].

문서의 구조정보를 저장하는 방법으로 [6]에서는 노드간의 연결 정보로서 각 노드에 ID를 부여하고 부모 노드의 ID와 내용정보를 단일 테이블에 저장, 이용하였다.

```
<? xml version="1.0"?>
<PLAY play_num="S-001">
  <TITLE>The Tragedy of Hamlet, Prince of Denmark<TITLE>
  <ACT>
    <TITLE>ACT I</TITLE>
    <SCENE>
      <TITLE>SCENE I. Elsinore. A platform before the castle.</TITLE>
      <STAGEDIR>FRANCISCO at his post. Enter to him BERNARDO</STAGEDIR>
    </SCENE>
  </ACT>
  <ACT>
    <TITLE>ACT II</TITLE>
    <SCENE>
      <TITLE>SCENE I. A room in POLONIUS house.</TITLE>
      <STAGEDIR>Enter POLONIUS and REYNALDO</STAGEDIR>
    </SCENE>
  </ACT>
</PLAY>
```

그림 1 세익스피어의 희곡 '햄릿'의 XML 문서 예

질의에 표현된 노드간의 관계를 파악하기 위해 각 노드의 ID와 저장된 부모 ID를 사용한 자체-조인(self-join)을 이용한다. 위의 예제 XML 문서에 대한 "PLAY/ACT/SCENE/STAGEDIR" 질의는 3번의 자체-조인으로서 처리하게 된다. 내용 검색 및 간단한 질의에 대해서는 빠른 응답시간을 갖는다. 그러나 질의 경로가 길어지면 처리 시 늦은 응답시간을 보인다.

정규 경로식 질의 중 재귀적 내림 연산자 '//'와 확장 경로식 '*'이 포함된 질의는 높은 처리비용을 갖는다. "/PLAY/ACT//TITLE" 질의는 재귀적 내림연산자 '//'를 사용한다. 재귀적 내림연산자는 특정 노드를 찾기 위해 현 노드 하부 계층 구조 전체를 탐색하는 연산자이다. 'ACT' 노드를 찾는 경우, 앞서 설명한 바와 같이 지정된 노드들을 따라 탐사하여 쉽게 처리한다. 그러나 'TITLE' 노드를 찾기 위해서는 'ACT' 노드의 아래 레벨의 모든 노드들을 탐사해야한다. "PLAY/*/TITLE" 질의의 확장 경로식 '*'는 'PLAY'의 자식 노드 모두를 방문하여 그 노드의 자식 노드들 중에 'TITLE' 노드를 찾는다.

이러한 정규 경로식 질의 처리의 효율을 높이기 위해 [7-9]는 문서상의 경로 표현들을 문자열로 저장하여 이용하는 방법을 이용하였다. 그림 1의 XML 문서를 위한 경로 표현 저장 예는 표 1에서 볼 수 있다. 조건을 포함하지 않는 정규 경로식 질의들은 저장된 경로 표현에 대해 관계형 데이타베이스가 제공하는 패턴 매칭 연산자('LIKE')를 이용하여 적은 비용으로 처리한다. 그러나 질의 노드에 조건이 포함된 경우, 조건을 처리하기 위해 조건을 만족하는 노드를 검색하고 이들 노드와 질의 경로와의 "관계"를 계산해야 한다.

조건을 만족하는 노드와 질의 경로 사이의 관계 계산 방법으로 [6]에서는 영역 넘버링 값을 이용한 θ-조인 연산을 통해 노드간의 포함 관계를 계산한다. 예제 XML 문서(그림 1)에 대한 영역 넘버링의 예는 그림 2에서 보여진다.

조상 노드의 영역 시작 값은 자손 노드의 영역 시작 값보다 작고 조상 노드의 영역 끝 값은 자손 노드의 영역 끝 값보다 크다. "/PLAY/ACT[2]/SCENE/STAGEDIR" 질의에서 'STAGEDIR' 노드들 중에서 어떤 노드가 'ACT[2]'의 자손인지를 파악하기 위해 'ACT[2]'의 영역 넘버링 값 (22, 36)과 'STAGEDIR'의 영역 넘버링 값들 (16, 19)와 (31, 34)사이의 포함 관계를 따져보면 (31, 34)의 영역 넘버링 값을 가진 'STAGEDIR' 노드가 'ACT[2]' 노드에 포함됨을 쉽게 알 수 있다. 예제 문서는 깊이가 4에 불과하지만, XML 문서의 응용들 중에는 20~30 이상의 깊이를 가진 문서들도 많이 사용되므로 노드간의 관계 파악을 노드간의 연결 정보만으로 파악하기에는 많은 비용이 소요된다. 조상-자손 관계를 계산하기 위해 영역 넘버링 값을 이용한 θ-조인은 효과적이다. 그러나 모든 관계 계산을 관계형 데이타베이스에서 비교적 처리의 효율이 떨어지는 θ-조인만을 사용했다는 문제점이 있다.

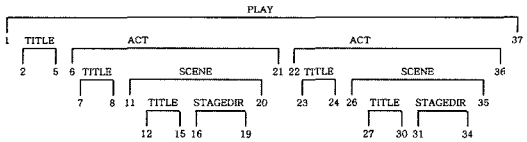

그림 2 예제 XML 문서의 영역 넘버링 예

표 1 경로 문자열 저장 예

경로 ID	경로 표현
1	"/PLAY"
2	"/PLAY/@play_num"
3	"/PLAY#/TITLE"
4	"/PLAY#/ACT"
5	"/PLAY#/ACT#/TITLE"
6	"/PLAY#/ACT#/SCENE"
7	"/PLAY#/ACT#/SCENE#/TITLE"
8	"/PLAY#/ACT#/SCENE#/STAGEDIR"

관계형 데이타베이스가 잘 지원하는 동등-조인(equi-join)만을 이용하여 노드간에 관계 파악을 위해 [7]에서는 부모-자식, 조상-자손간의 연결 정보를 별도의 테이블에 저장, 동등-조인을 이용하여 질의 처리의 효율성을 높였다. 정규 경로식 질의 처리를 위해 역시 경로 문자열을 저장하여 이용하며 부모-자식간의 연결 관계 정보는 부모-자식 테이블에 저장하고 조상-자손간의 연결 관계 정보는 조상 테이블에 저장하여 활용한다. 그러나 조상-자손 관계를 저장하기 위해 각 노드는 자기 자신을 제외한 모든 서브 노드들과의 연결 정보를 테이블에 저장해야 하며 노드들이 늘어나고 깊이가 늘어남에 따라 지수적으로 연결 정보들이 늘어난다.

정규 경로식 질의 중 재귀적 내림 연산자의 처리는 저장된 경로 문자열에 대해 "LIKE '%TITLE'"과 같은 패턴 매칭 연산을 수행하여 처리한다. 그러나 확장 경로식 '*'를 처리하기 위해서는 경로 표현에 대해 "/*/" 문자열을 중간에 포함하는 중복된 정보를 저장해야 한다. 이러한 문제점을 해결하기 위해 [9]에서는 고정된 숫자의 문자열 ID를 노드 이름에 부여하여, 경로 표현을 정규화하여 위의 문제를 해결하였다. 그러나 노드 ID간에 구분이 모호하여 잘못된 결과를 나타낼 수 있다.

기존의 연구들은 다양한 정규 경로식 질의들에 대해 동일한 정보와 동일한 계산 방법을 사용하여 처리하였다. 질의 처리시 사용되는 정보와 계산 방법에 알맞은 특정 질의에 대해서는 좋은 성능을 보이지만 질의 처리

표 2 INRIA[6], XRel[7], 그리고 PMR[8]의 특징과 장단점 비교

	특 징	장 점	단 점
INRIA[6]	노드간의 연결 관계 저장, 자체-조인을 이용	짧은 경로 질의에 대해서는 좋은 성능	질의 경로가 길어지면 처리에 많은 비용 소요됨
XRel[7]	경로 표현을 저장, 이용 영역 넘버링을 이용한 θ-조인을 이용하여 노드간의 포함관계 계산	긴 경로질의와 '//'를 쉽게 처리, 조상-자손 관계를 효과적으로 파악	노드간의 관계를 θ-조인 연산만을 이용, 질의 처리 효율이 떨어짐
PMR[8]	경로 표현을 저장, 이용 노드간의 연결 정보뿐만 아니라 조상-자손의 연결 정보 저장	동등-조인을 통해 조상-자손 관계도 효과적으로 파악	깊이가 늘어남에 따라 노드 정보가 지수적으로 늘어남

에 적합하지 않은 다른 형태의 질의에 대해서는 처리 시간이 증가하거나 저장의 비효율적인 면을 보인다. 앞에서 언급한 기존 연구에 대한 비교 설명은 표 2와 같다.

본 논문에서는 XML 문서들의 내용과 구조정보를 저장하고 다양한 정규 경로식 질의들을 효율적으로 처리하는 유연 질의 처리 시스템을 제안한다. 제안된 시스템은 정규 경로식 질의 처리시 질의의 구조를 파악하여 부모-자식 관계가 아닌 조상-자손 관계에서는 θ-조인을 사용하고 부모-자식 관계에서는 동등-조인을 사용하여 노드간의 관계를 파악한다. 그리고 정규화된 경로 문자열을 이용하여 저장 및 질의처리에 효율성을 높였다. 정규 경로식 질의 처리 시, 질의의 구조를 파악하여 질의에 알맞은 조인 연산과 구조 정보를 이용하여 효율적인 질의 처리를 한다.

3. 문서 저장 모델

3.1 경로 문자열의 정규화

각 노드의 경로를 문자열로 저장하여 활용하는 것은 XML 정규 경로식 질의를 처리하는 데 효과적이다. 즉, "PLAY//TITLE"과 같은 재귀적 내림연산자 '//'가 포함된 질의 처리를 위해 전체 노드를 탐색하지 않고, 경로 표현을 이용하여 관계형 데이타베이스가 제공하는 패턴 매칭 연산자를 사용하여 "LIKE '/PLAY#%/ TITLE'" 과 같이 간단하게 처리할 수 있다. 그러나 노드 이름을 그대로 경로 표현으로 사용할 경우 확장 경로식 '*'를 처리하기 어렵다. 이를 해결하기 위해 [9]의 방법과 유사한 방법으로 문서에 표현된 경로정보를 세 자리 문자로 정규화하여, 확장 경로식 '*'를 쉽게 처리할 수 있다. "PLAY/*/TITLE"이라는 질의에 대해 'PLAY'와 'TITLE'이 세 자리 문자 '001'과 '003'으로 정규화되면 "LIKE '/001/___/003'"과 같이 패턴 매칭 연산을 수행하여 쉽게 처리가 가능해진다. 표 3은 그림 1의 예제 XML 문서의 각 노드들에 대해 ID를 부여한 예이다.

이름이 같은 노드들에 대해서는 같은 종류의 노드로 고려하여, 하나의 ID만을 부여한다. 애트리뷰트 노드들에 대해서는 노드가 속한 엘리먼트 노드에 따라 뜻이

표 3 노드 이름에 대한 ID 부여 예

노드 이름	노드 ID	노드 이름	노드 ID
PLAY	'001'	ACT	'004'
PLAY/@play_num	'002'	SCENE	'005'
TITLE	'003'	STAGEDIR	'006'

표 4 노드 ID가 적용된 경로 문자열 예

경로 ID	경로 표현
1	"/001"
2	"/002"
3	"/001/003"
4	"/001/004"
5	"/001/004/003"
6	"/001/004/005"
7	"/001/004/005/003"
8	"/001/004/005/006"

달라지므로 이를 구별하여 ID를 부여해야 한다. 그래서 애트리뷰트가 속한 엘리먼트의 이름과 애트리뷰트를 하나의 이름으로 고려하여 ID를 부여한다. 노드 ID는 '0~9', 'a~z', 'A~Z'의 문자를 사용하여 세 자리 사용시 20만개(62*62*62) 이상의 노드들을 구별하여 표현할 수 있다. 부여된 노드 ID를 적용한 경로 표현은 표 4에서 볼 수 있다.

그러나 [9]와 같이 구분자를 포함하지 않을 경우, '001' ID를 가진 노드와 '010' ID를 가진 노드가 포함된 경로가 있다고 가정할 때 경로 문자열 '...001010...'에 대해 패턴 매칭 연산을 수행할 경우 '101' 노드가 경로 문자열에 속해 있다는 잘못된 결과를 반환하게 된다. 따라서 노드 구분자 '/'를 사용하여 이러한 오류를 방지하였다.

3.2 문서 저장 테이블 스키마

본 논문에서 제안하는 XML 저장 시스템에서는 다양한 구조의 XML 문서의 내용정보와 구조정보를 5개의 테이블을 이용하여 저장한다. XML 문서를 저장하기 위해 이용되는 테이블은 그림 3과 같다.

Document 테이블은 저장된 문서의 이름과 문서의 ID를 저장한다. 특정 문서에 대한 검색이 요구되는 질의는 docName과 docID를 이용하여 검색한다.

```
Document(docID, docName)
Element(docID, pathID, start, end, parent, index, sibling)
Attribute(docID, pathID, parent, owner, value)
Text(docID, pathID, start, end, parent, owner, value)
Path(pathID, nodeID, pathExp)
```

그림 3 XML 문서 저장 테이블 스키마

Element 테이블은 각 엘리먼트 노드의 구조정보를 저장한다. start와 end는 앞에서 설명한 엘리먼트의 영역 넘버링 값을 나타낸다. parent는 자신의 부모 엘리먼트의 시작위치를 저장한다. 엘리먼트들 사이에는 순서가 존재한다. index는 동일한 이름의 형제 엘리먼트 사이에서의 순서를 나타내며 sibling은 모든 형제 엘리먼트 사이에서의 순서를 나타낸다.

Attribute 테이블은 각 애트리뷰트 노드의 구조정보와 애트리뷰트 노드의 내용을 저장한다. 애트리뷰트들은 자식 노드가 있을 수 없으므로 영역에 대한 정보인 start와 end를 사용하지 않고 단지 부모 엘리먼트와 애트리뷰트가 속한 엘리먼트를 가르키는 parent와 owner 정보만을 포함한다. 그리고 value는 애트리뷰트의 값을 저장하는 데 이용한다.

Text 테이블은 엘리먼트의 텍스트 내용을 담고 있다. docID, pathID, start, end는 앞에서 설명한 내용과 동일하다. 그리고 start와 end 값을 이용하여 혼합 엘리먼트에 대한 구조정보를 효율적으로 표현할 수 있다. value는 엘리먼트의 내용을 담고 있는 텍스트 노드에 포함된 내용정보이다. owner는 텍스트 노드가 속한 엘리먼트의 영역 시작 값이며, parent는 텍스트 노드가 속한 엘리먼트의 부모 엘리먼트의 영역 시작 값이다.

Path 테이블은 문서상에 나타나는 경로 표현들을 문자열로 저장한다. 다양한 XML 문서들을 저장할 때 경로의 수는 무수히 증가한다. 경로 테이블 전체를 순차탐색하여 검색하는 것은 질의 처리 비용을 증가시킨다. 질의 처리 과정 중 각 정규 경로식 질의들이 검색하는 노드는 경로의 단말 노드이다. 따라서 경로의 단말 노드의 ID 값으로 해당 경로를 찾을 수 있도록 한다.

각 테이블의 기본키는 밑줄로서 표현했다. 엘리먼트 테이블과 텍스트 테이블은 docID, start, end 세 개의 필드를 키로 설정하였다. 그리고 docID, start, end의 순서로 클러스터 인덱스를 구성한다. θ-조인 사용에 사용되는 필드에 대해 클러스터 인덱스를 사용하여 θ-조인의 효율성을 높일 수 있다[10].

4. XML 정규 경로식 질의를 위한 유연한 질의 처리 시스템

계층적이며 가변적인 구조의 XML 문서에 대해 구조 질의를 하기 위한 XML 질의들은 정규 경로식 질의이다. 본 논문에서는 XML 질의들 중 가장 기본적인 형태이며 정규 경로식 질의의 형태를 잘 표현하고 있는 XPath[11]를 기본 질의로 사용한다.

4.1 정규 경로식 질의 분석

XPath는 경로 표현과 조건 그리고 관련 함수들이 포함된 질의이다. 하나의 XPath 질의는 경로 정보와 조건 술어(predicate)들 그리고 하나의 결과 노드로 구성된다. 조건 술어는 노드에 대한 조건을 명시한 것으로 조건 술어 내에는 질의 경로에서 분기되는 경로가 나타날 수 있다. 결과 노드는 질의 경로들 중 조건 술어내의 분기된 경로들을 제외한 질의 경로의 단말 노드이다.

경로 표현을 이용한 질의 처리방법에서는 조건에 의해 분기되는 경로를 처리하기 위해 질의를 서브질의(subquery)들로 나누어야 한다. 조건 술어를 포함하지 않는 질의는 중간 노드들에 대한 탐색 없이 경로 표현을 이용하여 결과 노드를 찾을 수 있다. 그러나 질의 내에 조건 술어가 포함된 경우에는 조건 술어에 포함된 분기된 경로를 처리해야 하므로 질의는 조건을 중심으로 나뉜다. 제안된 질의 처리 방법에서 XPath 질의의 분석은 조건 술어를 중심으로 질의를 서브질의들로 나누어 서브질의에 대한 조건정보를 구성하고 서브질의들 사이의 관계를 파악하는 것이다.

그림 4는 예제 질의에 대한 분석 과정인 질의 그래프와 결과물인 서브질의 집합을 보여주고 있다. 질의 분석은 질의에 표현된 경로 정보를 분석하는 것으로 시작한다. 질의를 분석하는 과정에서 조건 술어를 만나면 조건 술어 이전까지의 경로와 조건 술어 내부의 경로 그리고 조건 술어 외부의 경로로 나누어 분석하게 되며 조건 술어 이전까지의 질의 경로를 부모로 하는 질의 그래프를 생성하게 된다. 질의 분석 예에서는 Q_1과 P_1 그리고 Q_2가 이에 해당한다. 조건 술어 P1에 의해 질의는 조건 처리 이전의 Q_1과 조건 처리 이후 처리를 위한 Q_2로 나뉘게되며 Q_1은 P_1과 Q_2의 부모 질의가 된다. 나뉘어진 질의 중 조건 술어 내부에서 분기된 경로와 수식 표현을 분석하고 다시 조건 술어 밖의 다음 부분부터 분

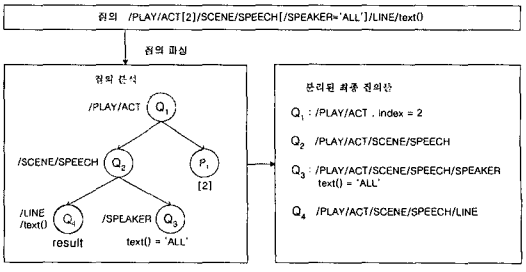

그림 4 질의 분석 과정의 예

석해 나간다. 예에서는 처음 만나는 조건 술어 P_1은 '[2]'이며 조건 술어 내에는 분기된 경로 정보가 없으므로 이는 부모 질의에 속하는 조건으로 고려할 수 있다. 조건 술어 내부에 대한 처리를 끝낸 후 조건 술어 이후에 대한 질의를 분석한다. 질의 분석 예에서는 Q_2부터 질의에 대한 분석을 해나가는 것이다. 질의가 다시 조건 술어를 만나게 되면 앞의 과정을 반복한다.

4.2 정규 경로식 질의 변환(XPathtoSQL)

질의 분석기에 의해 분석된 서브질의 집합의 정보를 이용하여 최종적으로 관계형 데이타베이스의 질의어인 SQL로 바꾸는 작업이 필요하다. 서브질의에 대한 경로 정보와 조건 정보를 분석하여 해당 테이블을 설정하고 서브질의 사이의 연결 관계를 파악하여 SQL을 생성한다.

정규 경로 질의들은 다음과 같이 세 종류로 나눌 수 있다.

4.2.1 기본 경로 질의

기본 경로 질의는 질의 내에 조건 술어를 포함하지 않은 질의이다. 질의 분석 후 생성된 서브질의 집합에는 하나의 서브질의만이 있으며 별도의 과정 없이 SQL 질의로 변환할 수 있다. 변환 시에 질의 경로에 재귀적 내림 연산자 '//' 또는 경로 확장식 '*'가 포함될 경우는 LIKE문을 사용하는 패턴 매칭 연산을 통해 경로를 처리하며 루트로부터 결과 노드까지 경로상의 노드들이 모두 포함된 경로는 동등 연산자('=')를 사용하여 처리한다. 이러한 질의 처리는 다음에 설명하는 부모-자식 관계 질의와 조상-자손 관계 질의 처리에서 서브 경로질의 처리에도 이용된다. '*' 연산자가 포함된 질의 처리의 예는 그림 5에 나타나 있다. 서브질의 경로상의 노드들에 대해 문서 저장시 사용된 ID를 부여한다. 이때 확장 경로식 '*'는 '___'으로 변환한다. 기본 경로 질의는 서브질의가 하나밖에 없으므로 서브질의간에 관계계산이 필요 없이 해당 테이블을 설정단계와 결과 설정단계를 거쳐 SQL을 생성하게 되다. 서브질의가 하나이므로 질의 자체가 결과 질의가 된다.

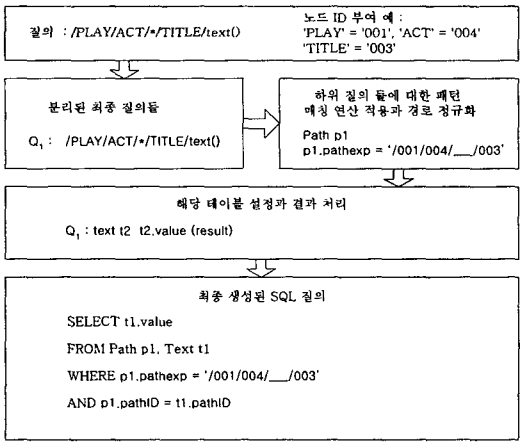

그림 5 확장 경로식 '*'이 포함된 기본 경로 질의 변환 예

4.2.2 부모-자식 관계 질의

질의 분석 단계에서 설명했듯이 질의 내에 조건 술어가 있으며 서브질의 그래프에 여러 개의 서브질의들이 포함되며 이들 서브질의들 사이의 관계가 부모-자식 관계인 질의이다. 서브 경로들 사이의 관계가 '/'로 표현되면 이를 이용하여 부모-자식 관계임을 파악할 수 있다. 두 서브질의들 사이의 관계는 부모의 start 값과 자식의 parent 값을 동등-조인에 이용하여 부모-자식 관계를 계산한다. 부모-자식 관계 질의를 SQL문으로 변환하는 과정은 그림 6에 나타나 있다. 그리고 만약 서브질의들 중에 동일한 부모 노드를 가진 서브질의 존재하면 부모 서브질의는 제외하고 서브질의들 사이의 관계만을 계산하여 질의를 최적화할 수 있다. 이러한 예는 그림 8에서 보여진다.

앞에서 설명한 기본 경로 질의에서 설명한 방법으로 각 서브질의 경로들을 처리한다. 그리고 각 서브질의가 적용될 테이블을 결정한다. Q_1과 Q_2는 엘리먼트 노드에 대한 내용이므로 엘리먼트 테이블에 적용된다. 조건 처리를 위해 Q_1에 대한 조건 처리, "index=2"를 처리하고 마지막으로 각 서브질의간의 관계를 계산한다. Q_1과 Q_2는 부모-자식 관계임을 서브질의들을 분석하여 알아낼 수 있다. 그래서 Q_1에 해당하는 엘리먼트 테이블의 docID와 start 필드와 Q_2에 해당하는 엘리먼트 테이블

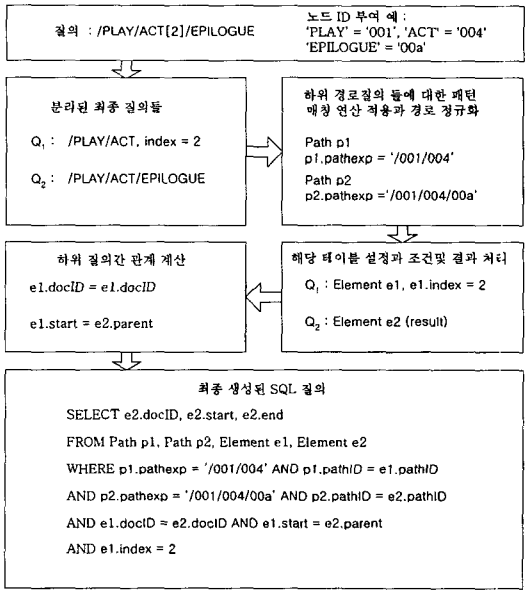

그림 6 서브질의들이 부모-자식 관계인 질의 변환 예

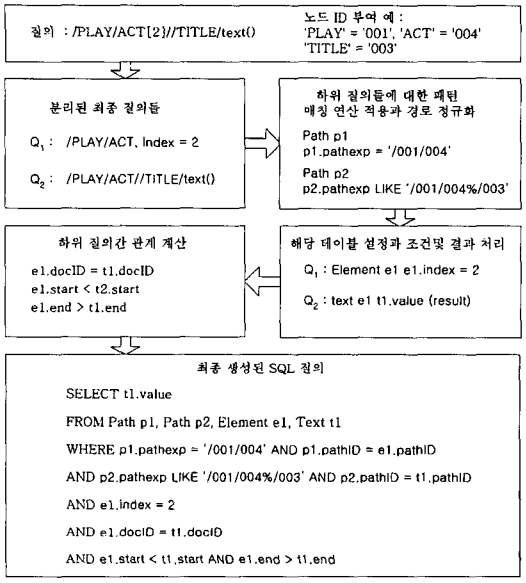

그림 7 서브질의들이 조상-자손 관계인 질의 변환 예

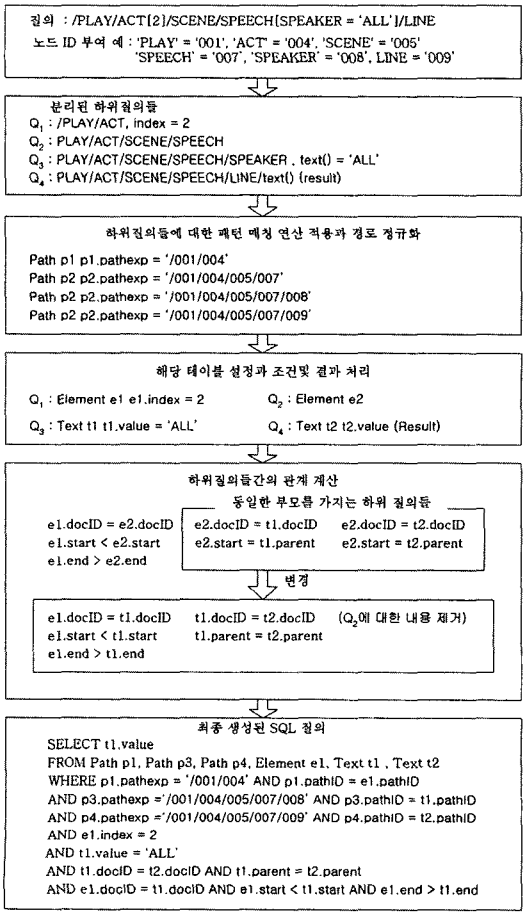

그림 8 부모-자식, 조상-자손 관계가 섞인 질의 변환 예

의 docID와 parent 필드를 동등-조인하여 서브질의간에 연결 관계를 계산한다.

4.2.3 조상-자손 관계 질의

질의 내의 조건 술어에 의해 나뉜 서브질의들의 관계가 조상-자손 관계인 질의이다. 서브질의들 사이의 관계가 재귀적 내림 연산자 '//'로 구분된 질의 혹은 재귀적 내림 연산자를 사용하지 않지만 서브질의들의 관계 계산에 다수의 노드들을 탐색해야 하는 경우이다. 이러한 경우 영역 넘버링 값(start, end)의 포함관계를 이용한 θ-조인으로 관계를 파악할 수 있다. 이러한 질의 예는 그림 7에서 보여진다.

복잡한 질의의 경우 서브질의들의 관계는 부모-조상 관계와 조상-자손 관계가 섞여 있을 수 있다. 이러한 복잡한 질의의 경우 각 서브질의에 대해 위의 내용들 모두를 적용하여 처리해야 한다. 또한 서브질의들 중 동일한 부모를 가지는 질의는 부모 노드와 조인 연산을 하는 것이 아니라 형제 질의들 간에 동등-조인을 사용하여 조인 수와 조인 연산에 참여하는 테이블의 수를 줄여 질의 처리의 효율을 높일 수 있다. 이러한 예는 그림 8에 나타나 있다.

서브질의간 관계 계산에서 서브질의 Q_3와 Q_4는 Q_2와 부모-자식의 관계이며 Q_2는 Q_1과 조상-자손의 관계이다. 질의 변환 시 Q_1과 Q_2 사이의 관계 계산은 θ-조인을 이용하여 파악하고 Q_2와 Q_3 그리고 Q_2와 Q_4는 동등-조인을 사용하여 질의 처리를 한다. 그러나 Q_3와 Q_4는 동일한 부모 노드를 가지므로 Q_3와 Q_4의 부모 정보를 이용하여 두 질의 사이에 관계를 계산한다. 그리고 Q_3 또는 Q_4와 Q_1 사이에 관계를 계산하여 질의 처리를 할 수 있다. 즉 질의 Q_2에 관련된 정보를 제외하고 Q_3와 Q_4 사이의 동등-조인과 Q_3와 Q_1 사이의 θ-조인으로 서브질의간의 관계 계산을 한다. 이러한 질의 변환(X-PathtoSQL)에 대한 알고리즘은 알고리즘 1에서 볼 수 있다.

질의 분석 단계에서 분석된 서브질의들의 정보를 이용하여 서브질의간 관계를 파악하여 적절한 연산을 유연하게 적용시킨다. 서브질의간의 관계가 부모-자식일 경우에는 동등-조인을 조상-자손 관계일 경우에는 θ-조인을 사용하는 SQL을 생성하여 질의 처리를 하게된다. 또 동일 부모 질의를 가지는 서브질의들은 부모 서브질의에 대한 연산 없이 처리하므로 질의를 더욱 최적화시킬 수 있다.

5. 시스템의 구현 및 실험

본 장에서는 XML 저장, 질의처리 시스템의 구현 환경과 구현에 대해 설명하고 질의 처리 모델의 성능을 기존의 방법과 비교 실험하여 질의 처리의 효율성을 증

```
입력 : 서브질의 집합과 관련 정보 subQuery
출력 : SQL 질의문 querySQL

서브질의 관련 정보 subInfo[]
for i=1 to subQuery.size() do
        subInfo[i] = 서브질의 경로 변환(subQuery[i])
        subInfo[i] = 관련 데이블 결정(subQuery[i])
        subInfo[i] = 조건 처리(subQuery[i])
        for j=i to subQuery.size() do
                subInfo[i] = 질의간 관계 계산(subQuery[i],subQuery[j])
        end for-loop
end for-loop
if ( 서브 질의간에 동일 부모 질의 in subQuery[] ) then
        관계 상속 from subInfo[부모 질의]
        부모 질의 정보 제거 from subInfo[]
end if
querySQL = 질의 SQL 출력 {
        SELECT 결과 질의 in subInfo[]
        FROM 관련 데이블 in subInfo[]
        WHERE
                for i=0 to subInfo.size() do
                        서브질의 경로 출력 in subInfo[i]
                end for-loop
        AND
                for i=0 to subInfo.size() do
                        질의간 관계 설정 출력 in subInfo[i]
                end for-loop
        AND
                for i=0 to subInfo.size() do
                        질의내 조건 처리 출력 in subInfo[i]
                end for-loop
}
```

알고리즘 1 서브질의간 관계 계산 알고리즘

명한다. 본 논문에서 제안한 XML 저장 시스템은 windows 2000 서버를 운영체제로 하며, Pentium-Ⅲ 1GHz, 256MB RAM에서 구현하였다. 그리고 관계형 데이타베이스는 MS-SQL2000([13])을 사용하고 운영체제에 독립적인 시스템을 구축하기 위해 개발언어로 자바(JAVA) jdk1.3.1([14])를 사용한다. 문서 저장기 구현에는 XML 문서의 적합성 분석과 구조정보 및 내용정보를 추출하기 위해 IBM XML4J([15]) DOM 파서를 사용한다.

질의 처리 시스템의 성능을 평가하기 위해 기존의 시스템과 비교 실험하였다. 실험 데이타와 질의는 [7]에서 INRIA([6])와 XRel([7])의 비교 실험에 사용된 셰익스피어의 희곡(Bosak Shakespeare collection)과 질의들을 사용하였다. 실험 데이타에 상세 내용은 표 5에서 볼 수 있다. 그리고 동일한 엘리먼트 이름에 대한 노드의 개수와 전체 노드에 대한 비율은 표 6에서 볼 수 있다.

비교 대상은 INRIA([6])의 Single Inlining Edge 테이블과 XRel([7])로 하였다. PMR([8])의 경우 저장의 비효율성으로 비교 대상에서 제외 시켰다. 실험 대상의 테이블 및 인덱싱의 구성은 각각 [6]과 [7]에 나온 내용을 참고로 하였다. 표 2에 기술되었듯이, XRel의 경우 '*' 연산자를 지원하지 않아 사용자의 부가적인 정보 입력이 필요하기 때문에 실험에 사용된 질의에서는 이를 생략하였다. INRIA의 경우 '//' 연산자의 처리가 기본 SQL 만으로는 구성할 수 없고 별도의 프로그래밍 방법을 사용해야 한다. 따라서 질의 제어 언어인 T-SQL문 [13]을 사용하여 저장 프로시져를 구성하여 질의를 처리하였다. 동일한 조건을 구성하기 위해 다른 시스템에 사용되는 해당 실험 질의 역시 저장 프로시져를 이용하였다. 실험에 사용된 질의는 표 7에 나타나 있다. 질의 Q_1 번부터 질의 Q_6번까지 질의의 복잡도가 높아지도록 구성하였다.

표 5 사용된 XML 데이타에 대한 상세 정보들

항 목	설정치
문서의 총 크기(MB)	6.91
문서 개수	35
문서 평균 크기(KB)	197.4
엘리먼트 노드의 수	170,728
텍스트 노드의 수	140,018
경로의 개 수	57
문서의 깊이	6
문서당 평균 엘리먼트 노드 수	4,878
문서당 평균 애트리뷰트 노드 수	4,000

표 6 실험 문서의 각 노드의 개수와 전체 노드에서의 비율

엘리먼트	개 수	비 율(%)
ACT	178	0.10
TITLE	982	0.57
SCENE	715	0.41
STAGEDIR	5,974	3.49
SPEECH	29,544	17.30
SPEAKER	29,954	17.54
LINE	102,294	59.91

표 7 실험에 사용된 질의

	질 의	결과 튜플 수
Q_1	/PLAY/ACT	175
Q_2	/PLAY/ACT/SCENE/SPEECH	713
Q_3	//ACT[2]/TITLE	35
Q_4	//ACT[2]//TITLE	181
Q_5	/PLAY/ACT/SCENE/SPEECH[SPEAKER='CURIO']	4
Q_6	/PLAY/ACT[2]/SCENE/SPEECH[SPEAKER = 'ALL']/LINE/text()	8

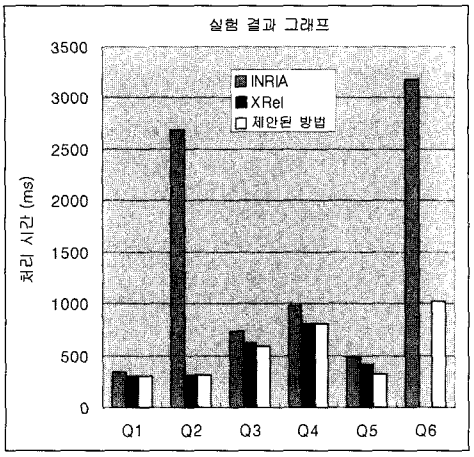

그림 9 실험 결과 및 비교 그래프

제안된 질의 처리 방법에 대한 성능 평가는 변환된 질의 SQL문을 데이타베이스에 입력한 시점에서 질의 결과가 출력되는 시점까지의 시간을 측정하였다. 각 질의에 대해서 10번씩 반복 수행하여 평균값을 이용하였다. 실험에 대한 결과는 그림 9에 나타나 있다. Q_6의 경우 XRel의 질의 처리 시간이 너무 길어(27,000ms) 실험 결과 그래프에서 제외하였다.

간단한 질의 Q_1의 경우 세 방법 모두 비슷한 성능을 보였다. INRIA의 경우 한번의 조인연산을 사용하고 XRel과 제안 모델 역시 한번의 조인 연산을 하므로 비슷한 성능을 보인다.

Q_2의 경우 조건이 포함되지 않은 단순한 경로 질의임에 불구하고 INRIA의 방법은 많은 질의 처리시간이 요구되었다. INRIA는 경로에 대한 정보를 포함하지 않으므로 루트로부터 해당 노드까지 중간 노드들을 모두 탐사해야 하며, Q_2의 해당 결과 집합의 수가 크기 때문이다. 즉, INRIA에서 Q_2에 비해 Q_5의 처리 시간이 월등히 많이 걸리는 이유는 해당 결과 집합의 수 차이가 크기 때문이다. 결과적으로 XRel과 제안한 방법은 Q_1과 마찬가지로 한번의 조인 연산으로 처리 가능하지만 INRIA의 경우 세 번의 조인을 해야 한다.

Q_3, Q_5의 경우 조건이 포함된 질의 형태이며 이들 조건을 처리하기 위해 XRel과 제안된 방법 모두 별도의 조인 연산이 필요하다. 서브 질의간의 관계 계산에 XRel은 θ-조인을 사용하고 제안된 방법에서는 동등-조인이 사용된다. 실험 결과 값을 비교하면 부모-자식 관계에는 동등-조인을 사용하는 제안된 방법이 우수함을 볼 수 있다.

Q_4의 경우 조건과 재귀적 내림연산자 '//'를 포함하는 질의로서 XRel과 제안된 방법 모두 θ-조인을 사용하여 처리하므로 질의 처리 시간은 비슷하다. INRIA의 경우 제안된 방법과 많은 차이를 보이지 않지만 질의 처리를 위해서는 기본 SQL문만으로는 처리가 불가능하며 문서의 깊이가 늘어나거나 또는 초기 검색 노드의 양이 많아질 경우 처리 시간은 더욱 늘어날 것이다.

Q_6의 경우 서브 질의가 부모-자식, 조상-자손을 모두 포함하는 복잡한 형태의 질의이며 이를 처리하기 위해서는 많은 조인 연산이 필요하다. INRIA의 경우 이러한 질의를 많은 시간이 걸리더라도 처리가 가능하지만 XRel의 경우 많은 θ-조인 연산으로 오랜 질의 처리 시간이 소요되는 단점을 보였다. 특히 제안된 방법에서는 부모 노드에 대한 정보를 이용하여 질의 처리에 참여하는 노드의 수를 줄이며, 전체 노드 정보에 대한 조인 대신에 일부는 경로 테이블과 조인함으로써 질의 처리 시간을 감소시킨다. XRel의 경우 전체 약 95%의 노드들을 3번의 θ-조인으로 처리하지만, 제안된 방법에서는 앞에서 설명한 방법으로 전체 77%의 노드들을 1번의 동등-조인과 1번의 θ-조인으로 처리하므로 질의 처리에 효율을 높였음을 실험 결과를 통해 알 수 있다.

모든 질의에 대해 동일한 정보와 동일한 연산만을 이용하여 처리하는 기존의 방법들보다 질의를 분석하여 질의에 따라 적절한 조인 방법을 적용하는 제안된 방법이 좋은 성능을 보임을 실험을 통해 알 수 있다. 특히 부모-자식 관계와 조상-자손 관계가 섞이고 질의의 경로가 긴 질의일수록 좋은 성능을 보임을 알 수 있다. 그러나, 문서 저장 시간에 대해서는 단일의 테이블을 사용하는 INRIA에 비해 여러 테이블을 사용하는 XRel과 제안된 시스템이 관계형 데이타베이스 입력시간에 있어서 성능이 떨어진다. 그러나 한번의 문서 입력과 계속되는 질의 처리의 비율을 볼 때 이는 큰 문제가 되지 않는다.

6. 결론 및 향후 연구방향

관계형 데이타베이스를 이용한 XML 문서 저장 시스템은 대량의 정보처리, 회복, 동시성 제어 등의 성숙된 기술을 그대로 활용할 수 있다는 장점이 있다. 또한 기존의 시스템들을 상위 계층에 미들웨어를 구성하여 활용할 수 있으므로 별도의 시스템을 구입, 구축하는 어려움을 해결할 수 있다. 그러나 평면구조의 관계형 데이타베이스에 계층 구조의 XML 문서를 저장하기 위해서는 문서의 내용정보뿐만 아니라 구조정보도 함께 추출하여 저장해야 한다. 구조정보의 표현 방법과 이용은 관계형 데이타베이스에서 XML 질의를 처리하는데 많은 영향을 끼친다.

정규 경로식 질의인 XML 질의들은 XML 문서의 가변적인 구조에 대해 다양한 구조 질의가 가능하다. 기존의 연구들에서는 모든 질의에 대해 동일한 구조정보와

방법을 적용하여 질의 처리를 하였다. 그러나 질의에 따라 효율성을 보이는 연산 방법이 다르므로 이를 구별하여 적용해야 한다. 다양한 정규 경로식 질의의 특성에 맞는 질의 처리를 위해서는 질의의 구조를 파악하여 질의에 알맞은 연산과 구조정보를 이용해야 한다.

본 논문에서는 문서로부터 영역 넘버링 값을 추출하여 노드간의 포함 관계와 연결 관계에 대한 구조정보로서 저장하였다. XML 정규 경로식 질의들을 효율적으로 처리하기 위해 질의를 분석하여, 질의에 포함된 노드간의 관계를 파악하여 부모-자식 관계의 질의에는 동등-조인을 사용하고 부모-자식 관계가 아닌 조상-자손 관계의 질의에는 θ-조인을 사용하는 유연성 있는 질의 처리 방법을 사용하였다. 실험을 통해 다양한 정규 경로식 질의 처리에 기존의 질의 처리방법에 비해 빠른 응답시간을 보임을 확인했다. 또한 문서의 경로 정보를 정규화하여 저장, 이용하여 확장 경로식 '*'의 처리가 가능하게 하였다.

향후 연구로 결과 생성기의 구현과 구현된 문서 저장기와 질의 처리기와 통합하여 사용자 인터페이스로서 사용될 API를 구성하는 것이다. 또한 제안된 모델은 검색뿐만 아니라 갱신 연산에 따른 개선 연구가 필요하며 기반 데이타베이스의 최적화에 대한 연구가 필요하다.

참 고 문 헌

[1] World Wide Web Consortium(W3C), "Extensible Markup Language (XML) 1.0 (Second Edition)," *http://www.w3c.org*, 2000.

[2] Tamino, *http://www.softwareag.com/tamino/*

[3] eXcelon, *http://www.exceloncorp.com/*

[4] 박상원, 정재목, 정태산, 김형주, "XML과 데이타베이스," *정보과학회지 제1권 제1호*, 2001.

[5] Jayavel Shanmugasundaram, et al, "Relational Databases for Querying XML Documents: Limitations and Opportunities," *VLDB*, 1999.

[6] D. Floresc and D. Kossman, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," *Technical report, INRIA*, 1999.

[7] M. Yoshikawa, O. Amagasa, T. Shimura, and S. Uemura, "XRel: A Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases," *ACM TOIT 1(1)*, 2001.

[8] H. Jiang, et al, "Path Materialization Revisited: An Efficient Storage Model for XML Data," *Proc. of ADC*, 2002.

[9] 이명철, "효율적인 구조검색을 지원하는 XQL 질의 처리기의 설계 및 구현", *충남대학교 컴퓨터공학과 석사 학위논문*, 2001.

[10] Chun Zhang, Jeffrey F. Naughton, David J. DeWitt, Qiong Luo, and Guy Lohman, "On Supporting Containment Queries in Relational Database Management Systems," *Proc. of the SIGMOD*, 2001.

[11] World Wide Web Consortium(W3C), "XML Path Language(XPath)," *http://www.w3.org/TR/xpath*

[12] World Wide Web Consortium(W3C), "Document Object Model(DOM)," *http://www.w3.org/DOM/*

[13] 권병희, SQL Server Bible, *영진 출판사*, 1999.

[14] SUN, Java jdk1.3.1, *http://java.sun.com/j2se/1.5/docs/api/index.html*

[15] IBM, XML4J, *http://www.alphaworks.ibm.com/tech/xml4j*

김 대 일

2000년 2월 인하대학교 전자계산공학과(공학사). 2002년 7월 인하대학교 전자계산공학과(공학석사). 2002년 7월~현재 국민은행 카드영업팀 e-Business Unit. 관심분야는 데이타베이스, XML, 정보검색

김 기 창

1984년 6월 California State Polytechnic University at Pomona, 전산학, 학사. 1988년 6월 University of California at Irvine, 전산학, 석사. 1992년 3월 University of California at Irvine, 전산학, 박사. 1992년 4월~1994년 8월 IBM T.J. Watson 연구소, 연구원. 1994년 9월~현재 인하대학교, 정보통신공학부 부교수. 관심분야는 컴퓨터 시스템 보안, 유무선 네트워크 보안, 실시간 운영체제, 병렬화 컴파일러

김 유 성

정보과학회논문지 : 데이타베이스
제 30 권 제 5 호 참조