색인된 XML 문서에서 레벨 정보를 이용한 효과적인 구조 조인 기법

(Efficient Structural Join Technique using the Level Information of Indexed XML Documents)

이 윤 호 † 최 일 환 † 김 종 익 †† 김 형 주 †††
(Yunho Lee) (Ilhwan Choi) (Jongik Kim) (Hyoung-Joo Kim)

요 약 오늘날 인터넷의 발달과 함께 XML이 많이 사용됨에 따라, XML 저장소와 질의 처리에 관한 연구가 활발하게 이뤄지고 있다. XML에서는 경로 질의를 사용하는데, 이러한 경로 질의를 효과적으로 처리하기 위하여 여러 가지 색인 기법들이 연구되었다. 최근에는 구조 조인 기법이 각광을 받고 있다. 구조 조인 기법은 엘리먼트들의 포함 관계를 살펴 질의를 처리한다. 특히 문서 전체에 대한 탐색을 하지 않고 해당하는 엘리먼트들의 역색인 리스트만을 비교하여 질의 처리를 수행하는 장점을 갖는다. 하지만 구조 조인 기법은 실제 질의 결과에 포함되지 않는 불필요한 엘리먼트들도 탐색해야 하는 비효율성을 가진다. 따라서 본 논문에서는 구조 조인 기법의 성능을 개선하는 레벨 구조 조인 기법을 제안한다. 제안하는 기법은 경로 질의 내 엘리먼트 사이의 관계 정보와 XML 문서 내의 엘리먼트들의 레벨 분포 정보를 이용한다. 이를 통해 구조 조인을 수행할 때, 비교되는 역색인 리스트 내의 모든 엘리먼트들이 아닌, 특정 레벨에 해당하는 엘리먼트들만을 이용하여 비교를 수행함으로써 질의 처리의 속도를 향상시킬 수 있다.

키워드 : XML, 구조 조인, 질의 처리, 색인

Abstract As XML is widely used with the development of internet, many researches on the XML storage and query processing have been done. Several index techniques have been proposed to efficiently process XML path queries. Recently, structural join has received much attention as a method to process the path query. Structural join technique process a path query by identifying the containment relationship of elements. Especially, it has an advantage that we can get the result set by simply comparing related elements only instead of scanning whole document. However during the comparison process, unnecessary elements that are not included in the result set can be scanned. So we propose a new technique, the level structural join. In this technique, we use both the relationship and the level distribution of elements in the path query. Using this technique, we can improve the performance of query processing only by comparing elements with specific level in the target inverted level.

Key words : XML, Structural join, Query processing, Index

1. 서 론

XML(eXtensible Markup Language)은 문서 구성 요소들 사이에 계층적인 구조를 가지고는 있으나 그 형태가 일정하게 고정된 스키마를 따를 필요가 없는 반구조적인(semistructured) 특성을 지닌다[1]. 이러한 특성을 지니는 XML 데이타의 처리를 위해 일반적으로 트리 구조 모델을 사용한다. 그림 1은 XML 문서와 트리 구조 모델로의 표현을 나타내고 있다.

최근에 인터넷의 발달과 함께 XML이 많이 사용됨에 따라, XML 저장소와 질의 처리에 관한 연구가 활발하게 이뤄지고 있다. XML 질의어로는 Lorel[2], XML-QL[3], XQuery[4], XPath[5] 등의 다양한 방식이 제안되었다. 이들 질의어들은 모두 경로 질의(path query)라는 특성을 지니고 있으며, 정규 표현식(regular expre-

· 본 연구는 BK-21 정보기술사업단과 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성지원사업(IITA-2005-C1090-0502-0016)의 연구결과로 수행되었음
† 학생회원 : 서울대학교 컴퓨터공학부
yhlee@oopsla.snu.ac.kr
ihchoi@oopsla.snu.ac.kr
†† 정 회 원 : 한국전자통신연구원 연구원
jikim@oopsla.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr
논문접수 : 2004년 12월 22일
심사완료 : 2005년 8월 19일

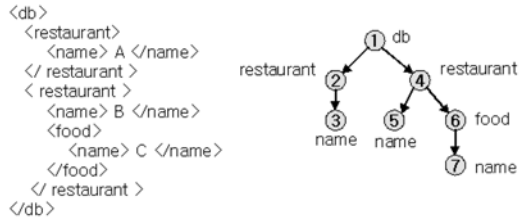


그림 1 XML 문서와 트리 구조 모델

ssion)으로 표현 가능하다. 그림 1의 문서에서 경로 질의 "/db//food/name"의 의미는 문서내의 db를 루트로 하는 모든 food의 name을 검색하는 의미이다. 이러한 XML 문서에 대한 경로 질의 처리를 효율적으로 하기 위해 색인 기법에 대한 연구도 많이 수행되었는데, 초기에는 주로 경로 색인 기법에 대한 연구가 많이 이루어졌다. 이 기법은 XML 데이타 구조에서 발생 가능한 모든 경로에 대한 색인 그래프를 별도로 구축한다. 이후 경로 질의 처리를 위해 원본 데이타 그래프의 탐색이 아니라 경로 색인 그래프를 탐색 함으로서 탐색 공간의 크기를 줄여 질의 처리 비용을 줄이는 기법이다. 최근에는 조상-후손 관계 등의 포함 질의를 효과적으로 처리할 수 있는 구조 조인 기법에 대한 연구가 많이 이루어졌다. 이 방식은 엘리먼트들의 포함 관계를 살펴 질의를 처리한다. 특히 문서 전체에 대한 탐색을 하지 않고 해당하는 엘리먼트들의 역색인 리스트만을 비교하여 질의 처리를 수행하는 장점을 갖는다. 하지만 구조 조인 기법은 실제 질의 결과에 포함되지 않는 불필요한 엘리먼트들도 탐색해야 하는 비효율성을 가진다. 이에 본 논문에서는 질의 처리시 엘리먼트 사이의 관계 정보와 XML 문서에서 각 엘리먼트들의 레벨 분포 정보를 이용하는 *레벨 구조 조인 기법*을 제안한다. 제안하는 기법에서는 비교 대상이 되는 엘리먼트의 개수를 줄이기 위해 각 엘리먼트의 특정 레벨 데이타만을 추출할 수 있도록 레벨 순으로 정렬된 역색인 리스트를 구축한다. 그래서 경로 질의를 수행할 때 XML 문서에서의 엘리먼트들의 레벨 분포 정보와 질의 내의 엘리먼트 사이의 관계 정보를 이용하여 역색인 리스트에서 특정 레벨의 엘리먼트만을 추출하여 비교 연산이 수행되도록 한다. 본 논문의 기여도는 다음과 같다. 우선 XML 문서가 트리 구조 모델로 전환될 때 지니게 되는 레벨 정보를 적극적으로 활용하여 질의 처리 결과에 포함될 수 있는 특정 레벨의 엘리먼트들 만을 추출하여 비교 연산을 수행함으로써 질의 처리 시간을 단축시켰다. 또한 가상 데이타 및 실제 데이타를 이용한 성능 평가를 통해 제안한 레벨 구조 조인 기법의 효율성을 입증하였다.

논문의 전체 구성은 다음과 같다. 2장에서는 본 논문과 관련된 배경 지식 및 선행 연구들에 대해 살펴본다. 3장에서는 경로 질의 처리를 위해 사용된 기존의 구조 조인 기법의 문제점을 지적하고 이를 보완한 레벨 구조 조인을 제안한다. 4장에서는 기존의 구조 조인 기법과 제안하는 기법과의 성능비교를 수행하고, 5장에서 결론과 향후 연구 방향에 대해 제시한다.

2. 배경 지식 및 관련 연구

XML 문서의 경로 질의 처리를 위한 연구는 활발히 진행되어 왔다. 초기에 많이 연구된 것은 경로 색인 기법이다. 이 기법은 XML 데이타 구조에서 발생 가능한 모든 경로에 대한 색인 그래프를 별도로 구축하여 경로 질의 처리를 위해 원본 데이타 그래프의 탐색이 아니라 경로 색인 그래프를 탐색함으로서 탐색 공간의 크기를 줄여 질의 처리 비용을 줄이는 기법이다. 경로 색인 기법은 Goldman and Widom[6]이 제안한 반 구조화된 데이타를 위한 스키마 추출 기법인 DataGuide를 이론적인 기반으로 하고 있다. Goldman and Widom이 제안한 DataGuide를 기반으로 하여 Milo and Suciu[7]가 XML 문서에 대한 경로 질의 처리를 위해 1-index와 2-index를 제안하였다. 1-index는 XML 데이타 트리에 대해 루트 노드로부터 시작하는 모든 독립적인 경로들을 구축하는 기법이고, 2-index는 "//element(P)"와 같은 형태의 경로 질의 처리의 효과를 극대화하기 위한 색인으로 임의의 엘리먼트 P에 도달 가능한 모든 경로가 하나로 표현되는 색인을 구축하는 기법이다. 최근에는 Kaushik et al.[8]가 1-index를 응용하여 인덱스 그래프의 크기를 줄인 A(k)-index를 제안하기도 하였다. 하지만 경로 색인 기법은 여러 문제점을 가지고 있다. 우선 새로이 구축하는 인덱스 그래프의 구조가 고정적이지 못하고 또한 원본 데이타 그래프에 비해 충분히 적은 크기를 가진다는 것을 보장할 수 없다. 또한 분기(branching)를 가지는 경로 질의를 지원할 수 없고, 질의 내의 "//" 처리시 인덱스 그래프 전체를 탐색해야 하는 비효율성을 지닌다.

이러한 경로 색인 기법이 지니는 단점으로 인해 최근에 주목 받고 있는 것은 구조 조인 기법이다. 이 기법은 경로 질의 내의 인접한 두 엘리먼트 사이에 존재하는 구조적인 포함 관계 규명을 통해 질의 처리를 수행하는 기법이다. 이들이 제안한 구조 조인 기법을 수행하기 위해서는 공통적인 선행 작업으로 XML 문서에 대한 파싱 단계에서 각 노드들에 대한 시작과 끝, 레벨에 대한 위치 코딩(Position Coding)을 해야 한다. 그런 다음 각 엘리먼트에 대해 역색인 리스트를 구축하게 된다. 그림 2는 구조 조인 방식의 처리에 앞서 생성되는 XML 문

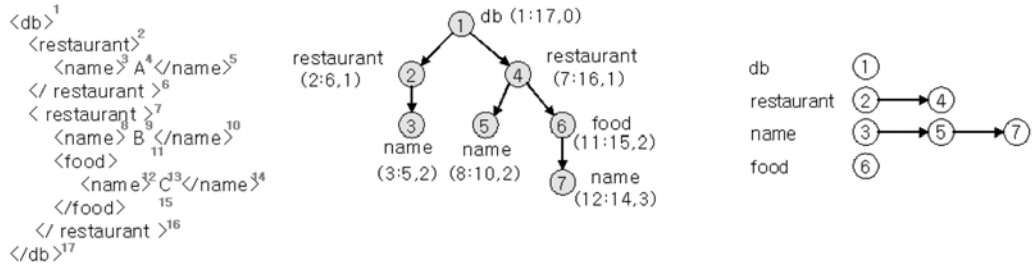


그림 2 XML 문서에 대한 Numbering Scheme과 역색인 리스트

서에 대한 Numbering Scheme과 역색인 리스트를 보여준다.

이후 구조 조인 기법은 질의 내의 각 엘리먼트들 사이의 구조적인 포함관계를 규명하기 위해 역색인 리스트에서 추출된 데이타들 사이의 (시작:끝)의 위치 정보를 서로 비교한다. 조상-후손 관계의 판별을 위해서는 조상에 해당하는 엘리먼트의 (시작:끝) 위치가 후손의 (시작:끝)을 포함하는지를 평가한다. 부모-자식 관계의 판별을 위해서는 두 엘리먼트가 조상-후손 관계이면서 동시에 레벨의 차이가 1인지를 추가로 확인한다. 예를 들어 그림 2의 XML 문서에 대해 경로 질의 "//food//name"를 구조 조인 기법으로 처리하면 food와 name에 해당하는 데이타를 역색인 리스트에서 추출한다. food와 name은 조상-후손 관계이므로 추출된 6번과 3,5,7번 노드의 위치 정보인 (11:15)와 (3:5),(8:10),(12:14)를 서로 비교하여 포함관계를 규명하는 방식으로 진행된다. 구조 조인 기법과 관련된 연구들로 Chun Zhang[9]은 전통적인 Merge-Join 알고리즘을 확장한 MPMGJN(Multi Predicate Merge Join) 알고리즘을 제안하였다. Li and Moon[10]은 XML 문서의 잦은 변경에도 잘 대처할 수 있는 유연한 위치 코딩 기법을 추가적으로 제안하였으며 Al-Khalifa, et al.[11]은 MPMGJN의 결과 리스트의 정렬 기법을 결정할 수 있는 tree-merge 조인과 스택 자료 구조를 이용한 조인 연산의 성능을 개선하였다. 최근에 Jiang et al.[12]은 XML 데이타 트리를 몇 개의 서브 트리로 분할하여 고유한 그룹 ID를 부여하는 기법으로 같은 그룹 ID를 지닌 엘리먼트의 데이타들간에 비교 연산이 수행되도록 하는 기법을 제안하였다. 하지만 지금까지 진행된 구조 조인 기법의 연구들은 XML 문서내의 각 엘리먼트들의 레벨 분포 정보와 경로 질의 내의 엘리먼트들 사이의 관계 정보를 고려하지 않았다. 그래서 질의의 결과에 무관한 레벨의 데이타까지 모두 비교 연산의 입력에 포함되어 불필요한 연산이 수행된다는 단점을 지닌다.

3. 레벨 구조 조인 기법

본 장에서는 레벨 구조 조인 기법을 제안하게 된 동기와 알고리즘을 소개한다. 또한 경로 질의 처리시 레벨 구조 조인 기법의 동작 과정을 살펴보고, 레벨 구조 조인이 가질 수 있는 문제점을 논의한다.

3.1 동기

우리는 많은 XML 문서에서 같은 이름의 엘리먼트들이 여러 레벨에 분산되어 분포되는 경향이 있다는 것을 확인하기 위해 Web상에서 임의로 1000개의 XML 문서를 수집하여 실험과 분석을 통해 약 70%에 해당하는 XML 문서들이 그림 3과 같은 특성을 지님을 알아냈다. 즉 많은 XML 문서에서 같은 이름의 엘리먼트들이 여러 레벨에 분산되어 분포한다.

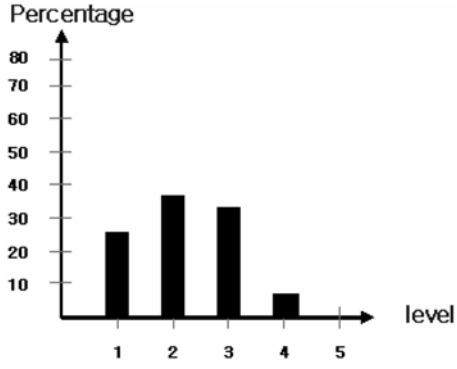


그림 3 XML 문서에서 동일 이름 엘리먼트의 레벨 분포도

이와 같은 XML 문서의 엘리먼트 레벨 분포 정보를 적극적으로 이용하면 보다 효과적인 경로 질의 처리를 수행할 수 있다. 즉, 기존의 경로 질의 처리에 사용된 구조 조인 기법은 동일한 태그명의 모든 엘리먼트들에 대하여 비교 연산을 수행하였다. 하지만 위와 같은 XML 문서가 지니는 엘리먼트의 레벨 분포 정보와 경

로 질의 내 각 엘리먼트 사이의 관계 정보를 이용하면 결과에 포함될 수 있는 레벨의 엘리먼트만 추출하여 비교 연산을 수행함으로써 조인 연산의 횟수를 감소시킬 수 있다.

3.2 알고리즘

그림 4는 제안하는 레벨 구조 조인 기법의 알고리즘을 보여준다.

```
Algorithm Level-Structural-Join
1:  /* A is the ancestor element, D is the descendant element */
2:  /* AList is the list of potential ancestors, in sorted order of levelNum */
3:  /* DList is the list of potential descendants, in sorted order of levelNum */
4:  /* LevelInfoTable is the table of all element's level information */
5:  A-level-list ← extractLevel(A) from LevelInfoTable;
6:  D-level-list ← extractLevel(D) from LevelInfoTable;
7:  if(isParentChildRelationship(A,D)){
8:    for(int i=0; i<A-level-list.size(); i++) {
9:      for(int j=0; j<D-level-list.size(); j++) {
10:       if(A-level-list.get(i)=D-level-list.get(j)-1) {
11:         Result-A-level-list ← A-level-list.get(i);
12:         Result-D-level-list ← D-level-list.get(j); } } }
13: } //end if
14: else{   //isAnscestorDescendantRelationship(A,D)
15:   for(int i=0; i<A-level-list.size(); i++) {
16:     for(int j=0; j<D-level-list.size(); j++) {
17:       if(A-level-list.get(i)<D-level-list.get(j)) {
18:         Result-A-level-list ← A-level-list.get(i);
19:         Result-D-level-list ← D-level-list.get(j); } } }
20: }//end else
21: Result-AList ← extractList(AList, Result-A-level-list);
22: Result-DList ← extractList(DList, Result-D-level-list);
23: Structural Join(Result-AList, Result-DList);
```

그림 4 레벨 구조 조인 기법 알고리즘

레벨 구조 조인 기법의 알고리즘은 크게 전처리 단계(1~4)와 질의 처리 단계(5~23)로 구분하여 설명할 수 있다. 전처리 단계는 실제 질의 처리에 필요한 기반 자료 구조를 생성하는 단계로써 기존의 구조 조인 기법과 유사하지만, XML 문서의 모든 노드들을 깊이 우선 탐색 기법으로 위치 코딩 후 각 엘리먼트들에 대해 레벨 순으로 정렬된 레벨 역색인 리스트를 생성한다(2~3). 물론 같은 레벨 내의 엘리먼트들은 시작 위치 순으로 정렬된다. 그런 다음 각 엘리먼트들의 레벨 정보를 함축하는 새로운 자료 구조인 레벨 정보 테이블을 생성한다(4).

실제 질의 처리 단계에서는 우선 경로 질의 내의 각 엘리먼트들에 대한 레벨을 레벨 정보 테이블에서 추출한다(5~6). 그런 다음 각 엘리먼트 사이의 구조적인 포함 관계 정보를 이용하여 불필요한 레벨을 제거한다(7~20). 최종적으로 제거되고 남은 특정 레벨의 데이타들만을 역색인 리스트에서 추출하여 구조 조인 연산을 수행한다(21~23).

3.3 경로 질의에 대한 레벨 구조 조인 기법의 처리 절차

이 절에서는 레벨 구조 조인의 처리 과정을 살펴 보기로 한다.

3.3.1 부모-자식 관계

부모-자식 관계에 대한 질의 처리 경우에는 결과에 포함될 각 엘리먼트들의 정확한 레벨을 결정지을 수 있다. 만약 레벨 정보 테이블로부터 부모 엘리먼트의 레벨이 L임을 확인하면 부모 엘리먼트의 데이타 리스트와 조인될 수 있는 자식 엘리먼트의 레벨은 L+1임을 결정지을 수 있다. 예를 들어 그림 5와 그림 6에는 임의의 XML 문서와 이에 대해 레벨 구조 조인의 전처리 과정을 수행해 얻은 레벨 역색인 리스트와 레벨 정보 테이블이 있다. 여기에 경로 질의 "//restaurant/name"을 수행한다고 하자. 이를 위해 우선 restaurant과 name의 레벨을 레벨 정보 테이블에서 추출한다. restaurant의 레벨은 1, name의 레벨은 1,2,3,4,5가 추출된다. 질의는 부모-자식 관계를 가지는 restaurant와 name을 찾는 것이므로, restaurant의 레벨이 1의 엘리먼트들에 대해 name의 레벨 2의 엘리먼트들 만이 비교의 대상이 된다. 그림 6의 레벨 역색인 리스트에서 restaurant의 레벨 1에 해당되는 (5:41,1), (42:61,1)와 name의 레벨 2에 해당되는 (6:8,2), (43:45,2) 만을 추출하여 시작과 끝 위치의 포함관계 여부를 비교하게 된다. 만일 기존의 구조 조인 기법을 사용한다면 restaurant과 name에 해당되는 역색인 리스트인 (5:41,1), (42:61,1)와 (2:4,1), (6:8,2), (11:13,4), (15:17,5), (20:22,5), (26:28,4), (30:32,5), (35:37,5) (43:45,2), (47:49,3), (52:54,3), (57:59,3)를 추출하여 포함 관계를 비교하게 된다. 결과적으로 레벨 구조 조인 기법은 경로 질의 내의 각 엘리먼트들 사이의 포함관계 정보와 XML 문서 내 각 엘리먼트들의 레벨 분포 정보를 기반으로 하는 레벨 제거 절차를 통해 필요한 레벨의 엘리먼트만을 추출하여 비교 연산을 수행함으로써 기존의 구조 조인 기법에 비해 조인 연산의 횟수가 감소하는 효과를 가져온다.

3.3.2 조상-후손 관계

두 엘리먼트들 간의 구조적인 포함 관계가 조상-후손인 경우에는 결과에 포함될 조상 혹은 후손 엘리먼트의 레벨 범위를 결정지을 수 있다. 만약 레벨 정보 테이블로부터 조상 엘리먼트의 레벨이 L임을 확인하면 조상 엘리먼트의 데이타 리스트와 조인될 수 있는 후손 엘리먼트의 레벨은 L 보다 큰 것임을 결정지을 수 있다. 그런데 조상-후손 관계에서는 부모-자식 관계와는 달리 엘리먼트간의 비교 대상이 될 수 있는 레벨 범위 만을 결정 지을 수 있어 경우에 따라서는 레벨 제거 후 다수의 레벨이 잔류할 수도 있다. 따라서 좀더 세분화 해서 살펴보도록 하자.

3.3.2.1 중복 탐색이 발생하지 않는 경우

조상의 특정 레벨에 대해 후손의 레벨 제거 후 하나의 레벨 만이 남는다면 위의 부모-자식 관계와 동일하다.

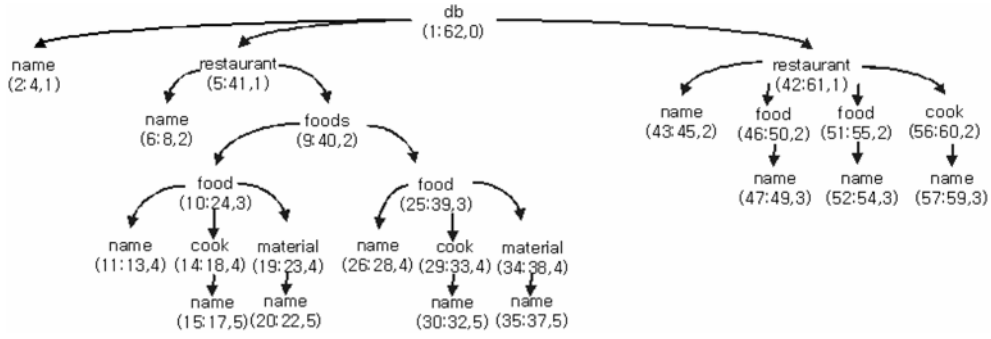


그림 5 레벨 구조 조인 처리 대상 XML 문서의 트리 모델

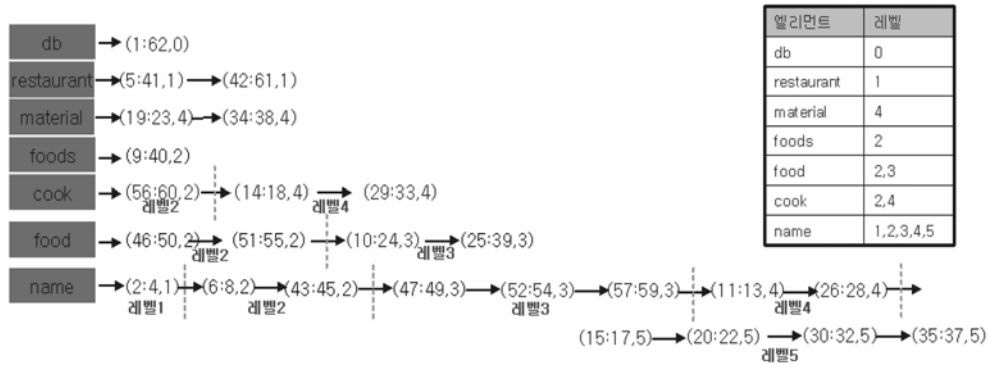


그림 6 레벨 역색인 리스트와 레벨 정보 테이블

3.3.2.2 중복 탐색이 발생하는 경우

레벨 제거 작업 후 조상의 특정 레벨에 대해 두 개 이상의 후손 레벨이 비교 대상에 포함되면 후손 레벨의 수만큼 조상의 비교 대상 레벨의 엘리먼트들이 중복하여 비교되는 문제가 발생한다. 그림 5의 XML 문서에 대해 경로 질의 "//restaurant//name"를 레벨 구조 조인 기법으로 처리하는 과정은 다음과 같다. 우선 그림 6에서 restaurant과 name의 레벨을 레벨 정보 테이블에서 추출한다. restaurant의 레벨은 1, name의 레벨은 1,2,3,4,5가 추출된다. 질의는 조상-후손 관계를 가지는 restaurant와 name을 찾는 것이므로, restaurant의 레벨이 1의 엘리먼트들에 대해 name의 레벨 2,3,4,5의 엘리먼트들이 비교의 대상이 된다. 그림 6의 레벨 역색인 리스트에서 restaurant의 레벨 1에 해당되는 (5:41,1), (42:61,1)와 name의 레벨 2에 해당되는 (6:8,2), (43:45,2), 레벨 3에 해당되는 (47:49,3), (52:54,3), (57:59,3), 레벨 4에 해당되는 (11:13,4), (26:28,4), 레벨 5에 해당되는 (15:17,5), (20:22,5), (30:32,5), (35:37,5)를 각각 추출하여 restaurant의 레벨 1에 해당되는 (5:41,1), (42:61,1)와 중복 비교되어 기존의 구조 조인 기법 보다 오히려 조인 연산의 횟수가 증가하는 문제가 발생한다.

3.4 레벨 구조 조인 기법의 분석

지금까지 살펴본 것처럼 동일한 이름의 엘리먼트가 여러 레벨에 분포되는 특성을 지니는 XML 문서에 대해 부모-자식 관계의 경로 질의에서는 조인 연산의 횟수가 줄어드는 효과가 발생하는 것을 확인하였다. 반대로 조상-후손 관계의 경로 질의에 있어서는 경우에 따라 조인 연산의 횟수가 오히려 증가하는 문제를 발생시킬 수도 있음을 확인하였다. 제안하는 레벨 구조 조인의 문제를 발생시킬 수 있는 경우에 대해 좀더 구체적으로 살펴보자.

경로 질의 처리에 있어서 제안하는 레벨 구조 조인 기법이 기존의 구조 조인 보다 성능이 나빠질 수 있는 경우는 질의가 조상-후손 관계에 대한 것이고, 조상과 후손에 해당되는 각 엘리먼트가 2개 이상의 레벨에 분산 분포하며 레벨 제거 후에도 2개 이상의 레벨이 존재하는 상황에서만 발생한다. 하지만 조상-후손 관계에 의해 비록 소수의 후손 레벨만이 제거 되더라도 제거되는 레벨에 집중적으로 많은 데이타가 모여 있다면 성능 효과를 볼 수 있다. 또한 N. Zhang[14]은 사용자가 XML 문서에서 특정 정보를 찾기 위해 사용하는 경로 질의의

패턴을 분석하여 부모-자식 관계를 나타내는 축 "/"가 전체 경로 질의에서 70% 이상 차지한다는 사실을 실험을 통해 밝혔다. 즉 전체 경로 질의에서 분해된 특정 이항 관계 사이에 조상-후손 관계가 발생하여 성능을 떨어뜨리더라도 다른 부모-자식 관계에서 성능 향상을 가져오기 때문에 전체적인 경로질의 처리에서는 성능 향상을 가져오게 된다. 그리고 Laurent Mignet[13]은 웹상에서 임의로 10만개 가량의 XML 문서를 수집하여 실험한 결과 99% 해당하는 대부분의 문서의 레벨 깊이가 8을 넘지 않는다는 것을 밝혔다. 즉 대부분의 XML 문서는 비교적 레벨이 깊지 않은 구조를 유지하고 또한 일반적으로 사용자는 질의 대상 XML 문서에 대해 개략적인 스키마 정보를 알고 있기 때문에 경로 질의를 할 때 조상-후손 관계 보다는 부모-자식 관계의 패턴을 보다 많이 사용하게 된다.

4. 성능 평가

4.1 실험 환경

본 실험은 MS-Windows XP 환경의 512 MByte 메인 메모리를 가진 Pentium 4-2.4GHz 와 데이타 베이스 시스템으로는 MySQL 4.1을 사용하였다. 경로 질의에 대한 구조 조인 기법의 효율성을 측정하기 위하여 여러 가지 형태의 경로 질의를 사용하였다. 또한 실험을 위한 데이타로는 실제 XML 데이타와 특정 DTD에 맞도록 가상으로 생성한 XML 데이타를 이용하였으며, 이는 다음과 같다.

(1) 실험 데이타: IBM 사에서 제공하는 XML Generator [15]를 이용하여 생성

(2) 실제 데이타: 셰익스피어 희곡들을 XML 형태로 변환하여 만든 XML 문서[출처: http://www.cs.wisc.edu/niagara/data/plays/]

4.2 실험 방식

MySQL 4.1을 사용하여 레벨 구조 조인 기법을 구현하기 위해서는 XML 문서를 파싱하는 전처리 단계에서 엘리먼트 테이블과 레벨 정보 테이블을 구축한다. 예를 들어 그림 5의 XML 문서를 이용하여 엘리먼트 테이블과 레벨 정보 테이블을 구축하면 표 1의 형식과 같다. 이때 각각의 엘리먼트 테이블은 레벨별로 구분하여 저장된다. 즉 분할된 엘리먼트 테이블 명칭은 각각 "Element Name+Level" 형식을 가진다.

위와 같이 전처리 단계에서 생성된 엘리먼트 테이블과 레벨 정보 테이블을 이용하여 경로질의를 처리하는 방식은 다음과 같다. 예를 들어 경로 질의 "//restaurant/food"을 처리하기 위해서 우선 각 엘리먼트 restaurant와 food의 레벨을 추출하기 위해 "select level from LevelInfoTable where Element = 'Element Name'"와 같은 패턴의 SQL문으로 변환하여 사용한다. 즉 "select level from LevelInfoTable where Element='restaurant'"와 "select level from LevelInfoTable where Element='restaurant'"을 사용하여 각각의 엘리먼트 레벨 정보를 추출한다. 추출된 restaurant와 food의 레벨 1과 2,3을 두 엘리먼트 사이의 관계 정보를 이용하여 불필요한 레벨을 제거한다. 즉 두 엘리먼트 사이의 관계가 부모-자식 관계이므로 각각 레벨 1과 2가 남게 된다. 그런 다음 각각의 엘리먼트 데이타 중에서 비교 대상이 되는 특정 레벨의 데이타들만을 추출하기 위해 "select * from 'Element Name+Level'"와 같은 패턴의 SQL문을 사용한다. 즉 "select * from restaurant1"와 "select * from food2"을 사용하여 각 엘리먼트들의 특정 레벨 데이타를 추출하게 된다. 최종

표 1 엘리먼트들 테이블과 레벨 정보 테이블

< restaurant 1 테이블 >

노드 ID	시작 위치	끝 위치
3	5	41
18	42	61

< food2 테이블>

노드 ID	시작 위치	끝 위치
20	46	50
22	51	55

< food3 테이블 >

노드 ID	시작 위치	끝 위치
6	10	24
12	25	39

< LevelInfoTable >

엘리먼트	레벨
db	0
restaurant	1
material	4
foods	2
food	2
food	3
cook	2
cook	4
name	1
name	2
name	3
name	4
name	5

적으로 추출된 각 엘리먼트들의 (노드 ID,시작 위치,끝 위치) 데이타를 이용하여 각각 시작과 끝 위치가 포함 관계에 있는지 여부를 비교하여 결과에 해당되는 노드 ID 쌍들을 결과로 반환하게 된다.

4.3 실험 결과

그림 7은 XML Generator를 이용하여 생성할 실험용 가상 XML 데이타의 DTD 구조를 나타내는 그래프이다. 표 2는 XML Generator를 이용하여 생성한 가상 XML 데이타의 특징과 성능 평가를 위해 사용한 경로 질의를 나타내는 표이다. 그림 8은 표 2의 경로 질의를 이용하여 구조 조인 기법과 레벨 구조 조인 기법으로 각각 처리하는데 걸리는 시간을 보여 준다.

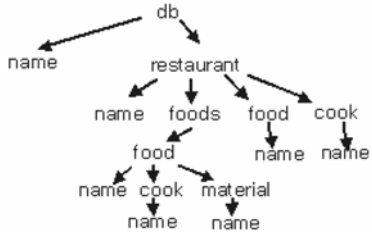


그림 7 가상 DTD 그래프

그림 8에서 보듯 대부분의 경로 질의 처리에 있어 레벨 구조 조인 기법이 기존의 구조 조인 기법에 비해 성능이 뛰어나다는 것을 확인할 수 있다. 하지만 QS4(//restaurant//food)와 QS10(//cook//name)은 레벨 구조 조인 기법으로 경로 질의를 처리하는 것이 오히려 성능이 나빠진다는 사실을 확인할 수 있다. 즉 후손의 여러 레벨이 전혀 제거 되지 않거나 혹은 데이타의 집중이 적은 일부 레벨만이 제거되어 조상의 데이타를 두 번 이상 중복 비교함으로써 발생되는 단점이다. 추가적으로 주목할 만한 질의로 QS2(//restaurant//material)는 두 엘리먼트 사이의 관계가 조상-후손 관계이면서 후손의 레벨 제거가 발생하지 않았음에도 불구하고 두 기법이 거의 비슷한 결과를 보인다. 이 결과에서 비록 레벨 제거의 효과를 볼 수는 없었지만 조상과 후손의 레벨이 모두 동일 레벨에만 분포하는 단순 구조의 XML 문서에서는 기존의 구조 조인 기법과 레벨 구조 조인 기법이 동일한 성능을 보인다는 사실을 추가적으

표 2 가상 데이타의 특징과 경로 질의

엘리먼트	개수	레벨
db	1	0
restaurant	9,720	1
material	90,888	4
foods	2,164	2
food	120,932	2,3
cook	68,040	2,4
name	347,901	1,2,3,4,5

문서 크기 : 20 Mbyte

질의	경로 질의 표현
QS1	//restaurant/material
QS2	//restaurant//material
QS3	//restaurant/food
QS4	//restaurant//food
QS5	//food/cook
QS6	//food//cook
QS7	//material/name
QS8	//material//name
QS9	//cook/name
QS10	//cook//name
QC1	//restaurant//food/material
QC2	//restaurant//food/cook/name

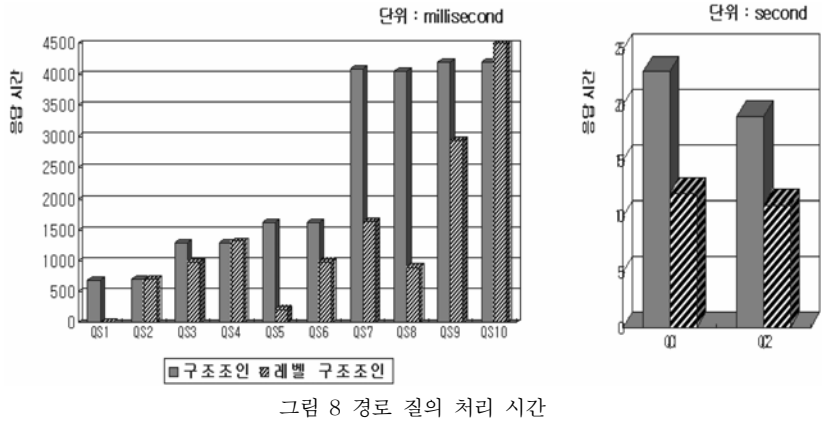


그림 8 경로 질의 처리 시간

로 확인할 수 있다.

그림 9는 실제 셰익스피어의 희곡 XML 데이타의 DTD 그래프이다. 표 3은 이 XML 데이타의 특징과 성능 평가를 위해 사용한 경로 질의를 나타내는 표이다. 그림 10은 표 3의 경로 질의를 이용하여 구조 조인 기법과 레벨 구조 조인 기법으로 각각 처리하는데 걸리는 시간을 보여 준다.

그림 10에서 보듯이 실제 XML 문서에서도 앞서 실험한 가상 XML 문서에 대해 얻은 결과와 비슷한 결과를 얻을 수 있다. 즉 대부분의 경로 질의 처리에 있어 레벨 구조 조인 기법이 기존의 구조 조인 기법 보다 효과적이라는 사실을 실험을 통해 확인하였다.

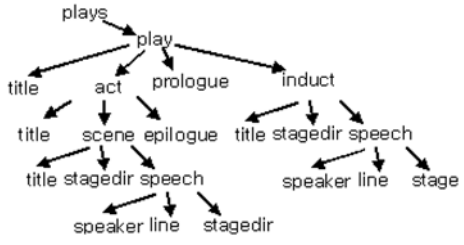


그림 9 셰익스피어의 희곡 XML 데이타의 DTD 그래프

5. 결론 및 향후 연구과제

우리는 본 연구에서 경로 질의를 효율적으로 처리하기 위한 새로운 레벨 구조 조인 기법을 제안하였다. 이 기법은 경로 질의 처리를 위해 XML 문서의 역색인 리스트를 각 엘리먼트들의 레벨 순으로 구성하고 또한 모든 엘리먼트들의 레벨 정보를 함축하는 새로운 자료 구조를 구축한다. 그런 다음 경로 질의 처리시 질의 내의 각 엘리먼트들의 레벨 정보와 엘리먼트들 사이의 관계 정보를 이용하여 결과에 포함될 수 있는 레벨의 데이타만을 추출하여 조인 연산을 수행함으로써 불필요한 비교를 줄일 수 있는 장점을 지니는 사실을 확인하였다. 물론 역색인 리스트를 엘리먼트의 시작 위치 순으로 정렬하는 것을 깨고 레벨 순으로 정렬함으로 인해 기존의 구조 조인 기법이 지녔던 각 엘리먼트들의 데이타를 한 번씩만 비교하여 포함 관계를 규명할 수 있었던 장점을 없애버렸다. 하지만 앞장의 실험을 통해 살펴보았듯이 단순 구조의 XML 문서에서는 레벨 구조 조인과 구조 조인 기법이 동등한 성능을 보이게 되고 같은 이름의

표 3 셰익스피어의 희곡 XML 데이타의 특징과 경로 질의

엘리먼트	개수	레벨
induct	97	2
scene	514	3
act	120	2
speech	20,159	3,4
speaker	20,201	4,5
line	70,723	4,5
title	620	2,3,4
stagedir	3,682	3,4,5,6

문서 크기 : 7.5 Mbyte

질의	경로 질의 표현
QS1	//induct/speech
QS2	//induct//speech
QS3	//induct/stagedir
QS4	//induct//stagedir
QS5	//scene/title
QS6	//scene//title
QC1	//Induct/speech/line
QC2	//prologue//speech/speaker
QC3	//prologue//speech/speaker

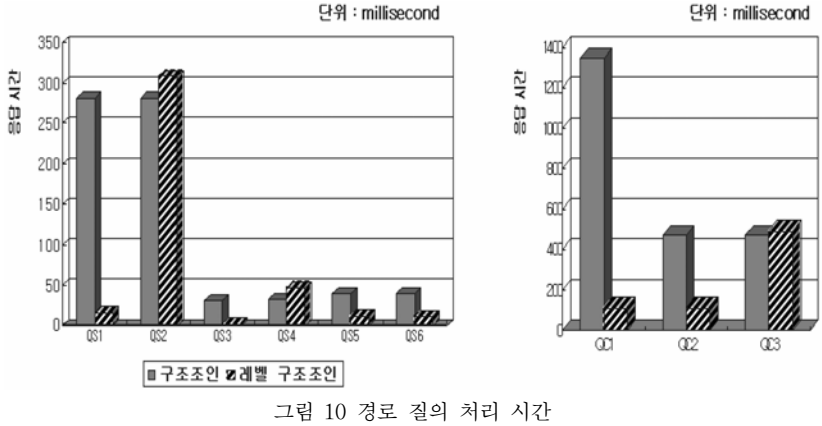


그림 10 경로 질의 처리 시간

엘리먼트가 여러 레벨에 분산되어 분포하는 복잡한 구조의 XML 문서에서도 대부분 레벨 구조 조인 기법이 기존의 구조 조인 기법 보다 성능이 뛰어나다는 사실을 확인하였다.

향후에 우리는 경로 질의를 처리할 때 XML 문서 내 엘리먼트들의 레벨 분포도를 분석하여 레벨 구조 조인 기법 혹은 구조 조인 기법 중에 최적의 기법을 동적으로 제공하는 질의 최적기에 대한 연구를 진행할 계획이다.

참 고 문 헌

[1] B. Bray, J. T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler, "Extensible Markup Language(XML) 1.0," W3C Recommendation, http://www.w3.org/TR/REC-xml, 2004.

[2] S. Abiteboul, et. al., "The Lorel query language for semistructured data," International Journal on Digital Libraries, Vol. 1, No. 1, pp.68-88, 1997.

[3] Alin Deutsch, Mary F. Fernandez, Daniela Florescu, Alon Y. Levy, Dan Suciu, XML-QL, QL, 1997.

[4] XQuery 1.0: An XML Query Language W3C Working Draft, http://www.w3.org/TR/xquery/, 2004.

[5] XML Path Language(XPath) 2.0 W3C Working Draft, http://www.w3.org/TR/2004/ WD-xpath20-20040723/, 2004.

[6] Goldman, R. and Widom, J. Dataguides: enabling query formulation and optimization in semistructured databases. In Proceedings of the Conference on Very Large Data Bases, 1997.

[7] Milo, T. and Suciu, D., Index structures for path expressions, In Proceedings of the International Conference on Database Theory, 1999.

[8] Kaushik, R., Shenoy, P., Bohannon, P., and Gudes, E., Exploiting local similarity for indexing paths in graph-structured data. In IEEE International Conference on Data Engineering, 2002.

[9] Chun Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational database Management Systems, SIGMOD, 2001.

[10] Li,Q. and Moon, B., Indexing and querying XML data for regular path expressions., In Proceeding of the Conference on Very Large Data Bases, 2001.

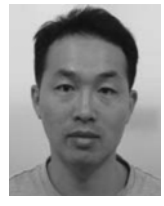
[11] Shurug Al-Khalifa, H.V. Jagadish, Nick Koudas, Jignesh M.Patel, Divesh M.Patel, Divesh Srivastava, Yuqing Wu, Structural Joins: A Primitive for Efficient XML Query Pattern Matching, In IEEE International Conference on Data Engineering, 2002.

[12] Jiang,H.,Lu, H.,Wang, W., and Ooi, B.C., XR-Tree : Indexing XML Data for Efficient Structural Joins, In IEEE International Conference on Data Engineering, 2003.

[13] Laurent Mignet, Denilson Barbosa, The XML Web: a First Study, World Wide Web Conference, 2003.

[14] N. Zhang, V. Kacholia, A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML, ICDE, 2004.

[15] IBM, http://www.alphaworks.ibm.com/tech/xmlgenerator, 2003.



이 윤 호

1999년 육군사관학교 전자공학부 학사
2005년 서울대학교 컴퓨터공학부 석사
관심분야는 XML, 질의처리, 데이타베이스

최 일 환

1996년 서울대학교 컴퓨터공학과 학사
1998년 서울대학교 컴퓨터공학과 석사
1998년~현재 컴퓨터공학과 박사과정. 관심분야는 데이타베이스, XML, 질의처리

김 종 익

1998년 KAIST 컴퓨터공학부 학사. 2000년 KAIST 컴퓨터공학부 석사. 2004년 서울대학교 컴퓨터공학부 박사. 2004년~현재 ETRI Telematics Division 연구원. 관심분야는 데이타베이스, 객체지향 시스템, 질의처리, XML

김 형 주

정보과학회논문지 : 데이타베이스
제 32 권 제 5 호 참조