# 정보 검색 기술을 이용한 대규모 이질적인 XML 문서에 대한 효율적인 선형 경로 질의 처리

## (Efficient Linear Path Query Processing using Information Retrieval Techniques for Large-Scale Heterogeneous XML Documents)

박 영 호 † 한 욱 신 †† 황 규 영 †††
(Young-Ho Park) (Wook-Shin Han) (Kyu-Young Whang)

**요 약** 본 논문에서는 대규모 이질 XML 문서들에 대한 부분 매치 질의를 효과적으로 처리하는 새로운 방법 XIR-Linear를 제안한다. XPath 질의는 XML 문서를 표현하는 트리 구조에 대한 경로 표현식(path expression)으로 쓰여진다. 주요한 형태의 XPath 질의는 부분 매치 질의(partial match query)이다. XIR-Linear의 목적은 이질적인 스키마들을 가진 대규모 문서들에 대한 부분 매치 질의를 효과적으로 지원하는 것이다. XIR-Linear는 관계형 테이블을 이용한 스키마-레벨 방법에 기반을 두고, 역 인덱스(inverted index) 기술을 사용하여 XPath 질의 처리의 효율성을 획기적으로 향상시킨다. 본 방법은 레이블 경로(label path)를 텍스트로 간주하고 레이블 경로 내의 레이블(label)들을 텍스트 내에 있는 키워드(keyword)로 간주한 후, 레이블들을 정보 검색 기술을 이용하여 인덱스 함으로써 전통적인 방법들에서 사용된 스트링 매치(string match) 보다 효율적인 방법으로 질의와 매치되는 레이블 경로들을 찾을 수 있도록 하였다. 성능 평가에서는 인터넷에서 수집한 XML 문서들을 사용하여 기존의 관계형 테이블을 이용하는 XRel, XParent와 비교 실험함으로써, 제안한 방법의 효율성을 입증한다. 실험을 통해 XIR-Linear가 실험 범위 내에서 XRel 이나 XParent에 비해 수십 배 이상 좋은 성능을 보이며, XML 문서 수의 증가함에 따라 더욱 우수하다는 것을 보인다.

**키워드** : XML, 부분 매치 질의, 역 인덱스, XIR-Linear, 정보검색, IR

***Abstract*** We propose XIR-Linear, a novel method for processing partial match queries on large-scale heterogeneous XML documents using information retrieval (IR) techniques. XPath queries are written in path expressions on a tree structure representing an XML document. An XPath query in its major form is a partial match query. The objective of XIR-Linear is to efficiently support this type of queries for large-scale documents of heterogeneous schemas. XIR-Linear has its basis on the schema-level methods using relational tables and drastically improves their efficiency and scalability using an inverted index technique. The method indexes the labels in label paths as keywords in texts, and allows for finding the label paths that match the queries far more efficiently than string match used in conventional methods. We demonstrate the efficiency and scalability of XIR-Linear by comparing it with XRel and XParent using XML documents crawled from the Internet. The results show that XIR-Linear is more efficient than both XRel and XParent by several orders of magnitude for linear path expressions as the number of XML documents increases.

**Key words** : XML, partial match queries, inverted indexes, XIR-Linear, Information Retrieval

· 본 연구는 첨단정보기술연구센터(AITrc)를 통하여 한국과학재단의 지원을 받았음

† 비 회 원 : 한국과학기술원 전산학과
yhpark@mozart.kaist.ac.kr
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
wshan@knu.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수
kywhang@cs.kaist.ac.kr

논문접수 : 2004년 1월 30일
심사완료 : 2004년 5월 12일

## 1. 서 론

최근, XML 문서 [30]들에 대한 질의 처리를 위해 의미있는 연구들이 진행되어 왔다. 그러나, 대부분의 XML 질의 처리 연구는 정해진 스키마를 가진 한정된 수의 문서들만을 고려하였으므로, 인터넷 검색 엔진[19, 29]과 같은 이질적인 스키마(heterogeneous schema)를

다루는 대규모 응용에 적합하지 않았다. 따라서 대규모 이질 XML 문서들을 다루는 응용들을 위해서 새로운 질의 처리 방법이 요구되었고, 본 논문에서는 이러한 요구에 초점을 맞추었다. 다음과 같은 XML 문서에 대한 예제 질의들을 생각해 보자.

아래의 Q1과 Q2는 XPath [5]에서 사용된 경로 표현식(*path expression*)으로 작성된 질의들이다. 여기서, document("*")는 모든 XML 문서들을 의미하고, 이들 질의는 경로 표현식과 매치하는(matching) 레이블 경로(*label path*)를 포함하고 있는 모든 문서들을 검출한다. 레이블 경로란 XML 문서의 스키마 정보로서 2장의 정의 1에서 정형적으로 정의한다.

- Q1:: document("*")/issue/editor/last
- Q2:: document("*")//issue//article//section/caption

이들 질의 가운데, Q1은 경로 상에 명시된 부모-자식 관계성(parent-child relationship) '/' 를 가진다. 반면, Q2는 경로 상에 명시된 '/' 이외에, 조상-후손 관계성(ancestor-descendent relationship) '//' 를 가진다. 이러한 Q\2 질의는 전체 매치 질의(*full match query*)[18]라 하는 Q1과 달리, 부분 매치 질의(*partial match query*)라 하고, 이러한 전체 매치 질의와 부분 매치 질의를 통칭하여, 선형 경로 표현식(*Linear Path Expression, LPE*) 이라 한다. 이 중, Q2와 같은 부분 매치 질의는 특히 이질적인 스키마 [1, 20]를 가진 문서들을 찾는데 유용하다. 본 논문은 참고 문헌 [6, 7]에서와 같이, 전체 매치와 부분 매치를 모두 포함하는 질의 타입인 선형 경로 표현식(LPE)의 효율적인 질의 처리 방법에 초점을 맞춘다.

전체 매치 질의는 부분 매치 질의의 특별한 형태로 볼 수 있으므로, 부분 매치 질의를 지원하는 것은 전체 매치 질의를 지원하는 것을 포함한다. 본 논문의 목적은 이질적인 스키마를 가진 대규모 문서들에 대한 부분 매치 질의를 효과적으로 지원하는 방법을 제안하는 것이다. 기존의 XML 문서에 대한 질의 처리 방법들은 (1) 관계형 테이블을 사용하는 스키마-레벨 방법들 (예, XRel [23], XParent [12, 13]), (2) 특별한 목적의 인덱스를 사용하는 스키마-레벨 방법들 (예, Index Fabric [7], APEX [6]), 그리고 (3) 인스턴스-레벨 방법들 (예, Element Numbering Scheme [17], Multi-Predicate Merge Join [24], Structural Join [2], Holistic Twig Join [4])과 같이 세 가지 타입으로 분류할 수 있다. 기존 방법들 중에서, 첫 번째 타입의 방법들은 부분 매치 질의를 위하여 사용할 수 있지만, 대규모 이질 스키마를 가진 문서들을 위한 질의 처리에 사용하기 위한 목적으로 설계 되지 않았다. 두 번째 타입은 부분 매치 질의를 제한적으로 지원한다. 세 번째 타입은 부분 매치 질의를 지원하지만 질의 처리 효율성이 떨어지기 때문에 실용적이지 않다. 보다 자세한 내용은 3장에서 소개 한다.

본 논문에서는 관계형 테이블을 사용하는 스키마-레벨 방법에 기반을 두고, 대규모 이질 스키마를 가진 문서들에 대한 질의 처리 효율성과 확장성을 크게 향상시키는 것을 목표로 한다. 특히, 정보 검색(information retrieval, IR) 분야에서 대규모 문서들을 빠르게 찾기 위해 전통적으로 사용해 온, 역 인덱스(*inverted index*) [22] 기술을 XML 문서의 구조 검색에 활용한 새로운 방법을 제안한다. 본 연구에서는 이 방법을 XIR-Linear라 부르며, "XML via Information Retrieval for Linear Path Expressions"라는 의미를 표현한 것이다. XIR-Linear는 XML 문서들의 구조 정보인 레이블 경로들에 대해 역 인덱스를 구축하고, 이 인덱스를 사용하여 LPE 질의들을 효과적으로 처리한다.

본 논문에서는 먼저 XML 문서 스키마를 저장하기 위한 관계형 테이블 구조를 설명한 다음, 역 인덱스의 구조와 이를 활용한 질의 처리 방법에 대해 논의한다. 이를 위해, LPE를 탐색 표현식 (search expression)으로 매핑하기 위한 규칙들을 제안하고, 그 LPE와 매치하는 노드들을 찾기 위한 질의 처리 방법을 제안한다. 다음으로, XRel 및 XParent와의 비교 분석을 통하여 제안한 방법의 성능에 대해 논의 하고, 웹 크로울러 (web crawler)를 써서 인터넷에서 수집한 실제 XML 문서들을 사용하여 실험함으로써 기존 방법인 XRel이나 XParent에 비해 XIR-Linear의 성능이 매우 우수함을 입증한다.

본 논문은 대규모 이질 XML 문서들에 대한 부분 매치 질의의 효율적인 처리에 관하여 다음과 같은 공헌을 제시한다.

- 부분 매치 질의를 효과적으로 처리하기 위한 저장 구조 제안: 이를 위하여, 문서의 구조 정보에 역 인덱스 기술을 적용하기 위한 XML 문서 저장 방법과 역 인덱스의 구조를 제안한다.
- 역 인텍스 기술을 활용하기 위한 질의 처리 알고리즘 제안: 이를 위하여, LPE를 역 인덱스 검색 표현으로 변환하는 알고리즘을 소개한다.
- 선형 경로 질의 실험을 통한 XIR-Linear 성능 검증: 이를 위하여, 다양한 질의로, XRel 및 XParent 방법과 성능 평가를 수행하여 XIR-Linear가 실험 범위 내에서 수십 배 이상 좋은 성능을 보이며, XML 문서 수의 증가에 따라 더욱 우수하다는 것을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 사용하는 XML 문서 모델과, 질의 모델을 소개한다. 3장에서는 기존 XPath 질의 처리 방법들에 대하여 논의한다. 4장에서는 XML 문서 저장 구조와 역 인덱스

구조를 제안하고, 5장은 4장에서 제안한 구조를 기반으로 하는 질의 처리 알고리즘을 제안한다. 6장에서는 기존 방법인 XRel과 XParent와 비교한 XIR-Linear의 성능 평가 결과를 제시한다. 마지막으로 7장에서는 본 논문의 결론을 내린다.

## 2. 연구 배경

### 2.1 XML 문서 모델

본 논문에서 사용하는 XML 문서 모델은 Bruno 등 [4]에서 제안한 모델을 기반으로 하고 있다. 이 모델에서, XML 문서는 루트가 있고, 순서가 있으며, 레이블이 기록된 트리(rooted, ordered, labeled tree)로 표현된다. 이 트리의 노드(node)는 엘리먼트(element), 애트리뷰트(attribute), 값(value) 중 하나를 나타낸다. 또한, 이 트리의 에지(edge)는 엘리먼트-서브엘리먼트 관계성(element-subelement relationship), 엘리먼트-애트리뷰트 관계성(element-attribute relationship), 엘리먼트-값 관계성(element-value relationship), 애트리뷰트-값 관계성(attribute-value relationship) 중 하나를 나타낸다. 엘리먼트와 애트리뷰트 노드는 XML 문서의 구조를 결정하는 요소로서, 각 노드에게는 레이블(label, i.e., name)과 고유 식별자(unique identifier)를 부여한다. 그림 1은 XML 문서의 XML 트리 예를 보인다. 이 그림에서 두 개의 애트리뷰트 값(attribute value) "R"과 "T"를 나타내는 노드 번호 15와 27을 제외한, 모든 단말 노드(leaf node)들은 엘리먼트 값(element value)들을 나타내고, 애트리뷰트 @category를 나타내는 노드 번호 14와 26을 제외한, 단말 노드가 아닌 모든 노드들은 엘리먼트를 나타낸다. 이때, 애트리뷰트는 그 레이블 앞에 프리픽스 '@'를 사용함으로써 엘리먼트와 구분한다.

본 논문에서 사용하는 모델은 노드를 값(value)이 아닌 엘리먼트나 애트리뷰트를 표현하도록 참고 문헌 [4]의 모델을 수정하고, 다음 정의 1과 같이 레이블 경로의 개념을 가지도록 하였다.

**정의 1.** XML 트리 내의 레이블 경로(*label path*)는 그 트리의 루트 노드로부터 특정 노드 $p$까지의 노드 레이블 $l_1, l_2, ..., l_p$ ($p \geq 1$)들의 연속(sequence)으로 정의되고, $l_1, l_2, ..., l_p$로 표현된다. □

레이블 경로는 XML 문서 구조를 표현한다. 본 논문에서는 이를 스키마-레벨 정보(schema-level information)라 구분하고, 레이블 경로는 경로 표현식과 "매치(match)한다"고 말한다. 예를 들어, 그림 1에서, issue.editor.first는 경로 표현식 //editor//first과 매치하는 레이블 경로이다.

기존 연구에는 레이블 경로에 대해 본 논문과는 다른 정의들이 있다. 그 예는, DataGuides [11]나 XParent [12] 등에서 찾아 볼 수 있다. DataGuides는 반드시 루트 노드일 필요가 없는, 어떤 내부 노드(internal node)로 부터 어떤 단말 노드까지의 연속으로서 경로를 정의하였고, XParent는 노드 대신 에지의 연속으로서 경로를 정의하였다. 정의 1은 DataGuides의 정의와 개념적으로 다르고, XParent의 정의와는 개념적으로 동일하나 XML 모델이 다르다.

### 2.2 XML 질의 모델

XPath, XQuery, Quilt 등과 같은 주요한 XML 질의 언어들은 어떤 XML 트리 [4]에 대한 질의를 명시하기 위하여 경로 표현식을 사용한다. 관련 문헌 [1, 16, 21]의 모델을 따르는 경로 표현식은 선형 경로 표현식(linear path expression, LPE)과 분기 경로 표현식 (branch-

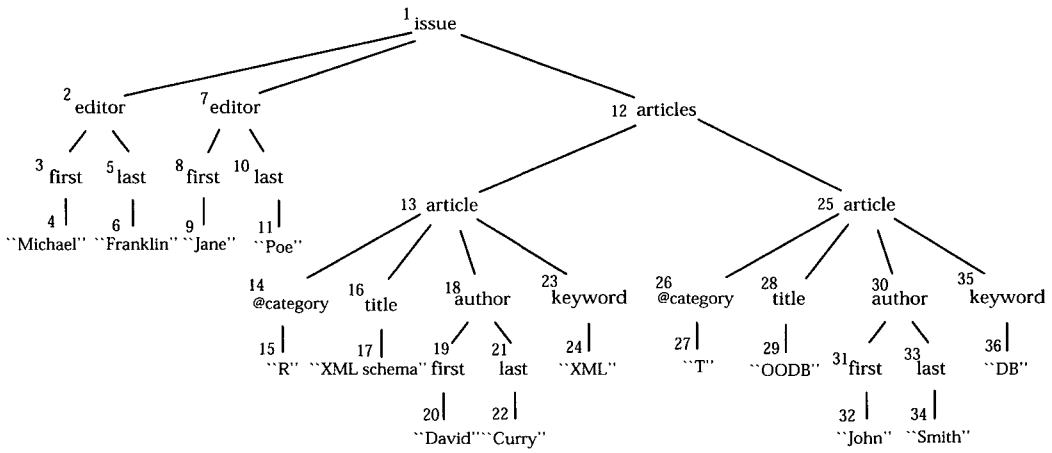



그림 1 XML 문서의 XML 트리 예

ing path expression, BPE)으로 분류된다. 본 논문에서는 이들 중 본 논문에서 다루는 선형 경로 표현식(LPE)에 대한 질의 모델을 정의한다. 다음 정의 2의 정의에 따르면, LPE는 '/'나 '//'로 연결된 레이블(label)들의 연속로서 표현된다.

**정의 2.** 선형 경로 표현식(*linear path expression*)은 $l_0\, o1\, l_1\, o_2\, l_2 \ldots o_n\, l_n$로서 정의된다. 이때, $l_i$ ($i$=1,2,⋯,$n$)은 그 경로 내에 존재하는 $i$ 번째 레이블이고, $o_j$ ($j$=1,2,⋯,$n$)는 '/'나 '//' 중 하나이며, 이들 각각은 $l_{j-1}$와 $l_j$ 간의 부모-자식 관계성(parent-child relationship)이나 조상-후손 관계성(ancestor-descendant relationship)을 나타낸다. 여기서, $l_0$는 모든 XML 문서들의 집합을 나타내고 있는 XML 트리의 루트 노드이고(즉, document("*")), 생략될 수 있다. □

다음 질의 Q3는 issue 엘리먼트의 후손인 article 엘리먼트들 중 자식으로 title 엘리먼트들을 찾는 LPE 예이다.

Q3:: /issue//article/title

### 2.3 XML 질의 패턴

선형 경로 표현식은 다음 정의 3에서 정의된 바와 같이, 선형 질의 패턴(*linear query pattern*)으로 모델된다.

**정의 3.** 정의 2의 정의 중, $l_0$가 생략된 경로 표현식 $o_1\, l_1\, o_2\, l_2 \cdots o_n\, l_n$의 경우, 경로 표현식은 최 상위 루트 노드에 연결된 허상 에지(*dangling edge*)를 가진 선형 질의 패턴(*linear query pattern*)으로 표현될 수 있으며, 이러한 선형 질의 패턴은 다음의 성질들을 가진다. □

- 노드는 경로 표현식 내의 레이블 $l_k$ ($k \in 1,2,\cdots,n$)를 나타낸다.
- 에지는 경로 표현식 내의 $o_j$ ($j \in 1,2,\cdots,n$)를 나타낸다. 에지는 만약, 경로 표현식 내의 $o_j$가 '/'일 경우는 단일 실선으로, '//'일 경우는 이중 실선으로 표현된다. 그 질의 패턴의 루트 노드에 연결된 허상 에지는 $o_1$을 나타낸다.

**정의 4.** (정의 3의) 루트 노드(*root node*)는 (모든 문서 엘리먼트 노드들을 포함하고 있는) 문서들의 집합을 표현하는 암묵적인 노드에 허상 에지인 '/'나 '//'로 연결된 엘리먼트 노드나 그것의 후손 노드로써, '/'인 경우 선형 경로 표현식 내의 레이블 $l_1$에 대응되고, '//'인 경우 $l_k$ ($k \in 1,2,\cdots,n$) 중 하나에 대응되는 노드이다. □

**정의 5.** (정의 3의) 결과 노드(result node)는 질의 패턴내의 단말로드로서, 선형 경로 표현식 내의 레이블 $l_n$에 대응되는 노드이다. □

질의 패턴에서는 결과 노드를 다른 노드들과 구분하기 위하여 회색톤으로 표현한다. 그림 2는 2.2절의 질의 Q3의 선형 질의 패턴을 보이고 있다. 이때, issue 노드는 루트 노드이며, 단말 노드인 title 노드는 결과 노드이다.

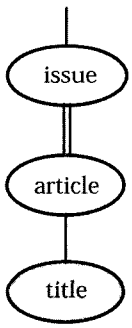



그림 2 질의 Q3의 질의 패턴

추가적으로, 본 논문에서는 질의 패턴에서 사용하는 루트 노드, 결과 노드(또는, 단말 노드)라는 용어와 경로 표현식에서 사용하는 루트 레이블, 결과 레이블(또는, 단말 레이블)이라는 용어를 상호 혼용하여 사용한다. 예를 들어, 선형 질의 패턴인 그림 2에서, 루트 노드인 issue는 질의 Q3의 루트 레이블이고, 결과 노드인 title은 질의 Q3의 결과 레이블이다.

## 3. 관련 연구

서론에서 언급한 바와 같이, 경로 표현식을 처리하는 방법은 크게 두 가지로 나누어지며, 그들은 각각 스키마-레벨 방법과 인스턴스-레벨 방법이다. 스키마-레벨 방법은 경로 표현식과 매치하는 노드들을 찾기 위해서 레이블 경로와 같이 구조적인 정보를 사용한다[12,13,23,6,7]. 반면, 인스턴스-레벨 방법은 노드의 시작(start)과 끝(end) 위치와 같이 노드 구분 정보 만을 사용한다[2,4,14,15,17,24]. 스키마-레벨 방법은 레이블 경로들이 어디에 어떻게 저장되는가에 따라, 관계형 테이블을 사용하는 방법[12,13,23]과 특별한 목적의 인덱스를 사용하는 방법[6,7]으로 구분된다. 전자의 경우, 문서 내의 모든 레이블 경로들은 질의 처리 이전에 관계형 DBMS 테이블에 저장되고 후자의 경우, 질의 처리시 저장된다.

본 절에서는 XIR-Linear 방법의 기본 구조가 되는 관계형 테이블을 이용한 스키마-레벨 방법들에 주된 초점을 맞추고 각 분류의 대표적인 방법들에 대해 논의한다.

### 3.1 관계형 테이블을 이용한 스키마-레벨 방법

XRel[23]과 XParent[12,13]는 이 분류의 대표적인 두 가지 방법으로 여겨진다. 본 절에서는 각 방법을 설명한다. 이때, 엘리먼트나 애트리뷰트는 노드(*node*)라는 용어를 사용한다. 이러한 표현은 2.1절의 XML 문서 모델

과 2.3절의 XML 질의 패턴에서 노드로 표현한 것과 동일한 의미를 가진다.

3.1.1 XRel

XRel에서, XML 트리의 구조 정보는 다음과 같이 네 개의 테이블에 저장된다.

Path (label_path_id, label_path)
Element (document_id, label_path_id, start_position, end_position, sibling_order)
Text (document_id, label_path_id, start_position, end_position, value)
Attribute (document_id, label_path_id, start_position, end_position, value)

Path 테이블은 문서들 내에 있는 모든 레이블 경로들과 그들의 식별자들을 저장한다. Element 테이블은 노드들, 즉, 엘리먼트들에 대한 정보를 저장한다. 여기서, 각 노드에 대한 정보는 그 노드를 포함하고 있는 문서의 식별자, 그 노드에서 종료되는 레이블 경로의 식별자, 한 문서 안에 있는 그 노드의 시작과 끝 위치에 대한 옵셋(offset)들, 노드의 형제(sibling)들 가운데 해당 노드의 순번(order) 등으로 구성된다. 시작 위치와 끝 위치의 조합은 노드 식별자를 대신하여 노드를 고유하게 구분하기 위한 정보로 사용된다. Text 테이블은 노드의 값들에 대한 정보를 저장한다. 여기서, value 컬럼은 텍스트 값들을 저장한다. Attribute 테이블은 value 컬럼에 텍스트 값 대신 애트리뷰트 값이 저장된다는 것 이외에 Text 테이블과 같다. 이들 두 테이블의 정보는 구현에 따라 하나의 테이블에 저장될 수도 있다.

XRel의 LPE 질의 처리는 Path 테이블에서 질의의 경로 표현식에 매치되는 레이블 경로들을 먼저 찾는다. 이때, 매치시키는 과정(matching)은 SQL의 스트링 매치 연산자인 LIKE 연산자를 사용한다. 레이블 경로에는 B+-tree 인덱스와 같은 인덱스를 사용할 수 없다. 이는 저장된 레이블 경로에 대해 부분 매치를 위한 다양한 질의 형태가 주어지기 때문이다. 그러므로, LPE 질의 처리를 위해 Path 테이블 내에 있는 레이블 경로들을 모두 읽어야 한다. 결과 노드들을 얻기 위해, XRel은 Element 테이블 내의 label_path_id 컬럼을 경유하여, 레이블 경로들과 매치하는 노드들을 조인하여 결과 노드들의 집합을 찾으며, 이것을 노드 집합(node set)이라 부른다.

**예제 1.** 다음과 같은 LPE를 생각해 보자.

Q4:: //article//author/first

그림 3은 LPE Q4의 질의 패턴을 보여준다. 그리고, 그림 4는 이 LPE Q4를 처리하기 위해 생성된 SQL문이다. 그림 4의 XRel SQL은 먼저, 스트링 매치를 통해서, LPE의 Path 튜플들의 집합을 찾는다. LIKE 절에서, '#' 심볼은 레이블 구분자이고, '%' 심볼은 한 개 이상의 문자들을 매치 시키기 위한 와일드 카드이다. 이때, Path 튜플 집합에서, 그 집합 안에 있는 경로의 결과 레이블과 매치하는 엘리먼트의 집합을 찾기 위해, Element 테이블이나 Text 테이블과 조인을 수행한다. 질의 처리 시, Element 테이블은 LPE의 결과 레이블에 대한 선택 조건(selection condition)이 없는 경우에 사용되며, Text 테이블은 결과 레이블에 대한 선택 조건이 있는 경우[1]에 사용된다. □

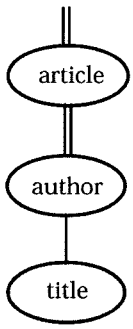



그림 3 LPE Q4의 질의 패턴

```
SELECT  distinct e1.document_id, e1.start_position, e1.end_position
FROM    Path p1, Element e1
WHERE   p1.label_path LIKE ``#%/article#%/author#/first#''
AND     e1.label_path_id = p1.label_path_id;
```

그림 4 LPE Q4를 위한 XRel SQL 문

3.1.2 XParent

XParent[12,13]는 XRel과 유사한 질의 처리 과정을 수행하지만, 다른 형태의 스키마를 가진 테이블을 사용한다. 이는 노드의 영역(즉, 시작~끝) 정보 대신, 노드의 식별자를 저장하여, BPE 처리시, XRel과 다른 포함 관계 연산을 적용하기 위함이다. XParent의 스키마는 다음과 같다.

LabelPath (label_path_id, length, label_path)
Element (document_id, label_path_id, node_id, sibling_order)
Data (document_id, label_path_id, node_id, sibling_order, value)
Ancestor (node_id, ancestor_node_id, offset_to_ancestor)

1) BPE는 분기하는 노드들과 단말 노드들까지의 LPE들로 분할하여 처리한다. XRel은 분할된 LPE들을 이용하여 각 노드 집합을 얻고, 이들 집합 간 포함 관계 조인을 수행하여 결과 노드 집합을 얻는다. 이 조인은 □-join과 equi-join을 통해 수행된다.

DataPath (parent_node_id, child_node_id)

LabelPath 테이블은 XRel의 Path 테이블과 같다. Element와 Data 테이블은 영역 정보 대신, 노드 식별자를 사용한다는 점을 제외하면 XRel의 Element와 Text 테이블과 같다. Jiang 등[12,13]의 연구는 Ancestor 테이블은 노드들 간의 조상-후손 관계성과 그들 간의 레벨 수를 유지하는데 사용하며, 이와 선택적으로 사용할 수 있는 DataPath 테이블(소위, Parent 테이블)은 노드들 간의 부모-자식 관계성을 유지하는데 사용하였다. 그러나, 부분 매치 질의에 대해서는 DataPath 테이블의 사용이 적합하지 않으므로 Ancestor 테이블을 사용한다[12]. 질의 처리시 XParent는 XRel과 같은 방법으로 LPE를 처리한다.

**예제 2.** LPE Q4를 생각해 보자. 그림 5는 LPE Q4에 대한 XParent SQL 문을 보인다. 그림 4와 달리, LPE들로 부터 얻은 노드 집합들이 노드 영역 정보가 아닌, 노드 식별자를 사용한다. 그러나, 그 밖의 다른 부분의 XParent SQL 문은 XRel SQL 문과 매우 유사[2] 하다. □

```
SELECT   distinct e1.document_id, e1.node_id
FROM     LabelPath lp1, Element e1
WHERE    e1.label_path_id = lp1.label_path_id
AND      lp1.label_path LIKE '%.%/article.%/author./first.%';
```

그림 5 LPE Q4에 대한 XParent SQL 문

### 3.2 특별한 목적의 인덱스를 사용하는 스키마-레벨 방법

Index Fabric [7]과 APEX [6]는 이 분류에서 두 개의 대표적인 방법들이다. Index Fabric은 질의에서 발생한 레이블 경로들과 값들을 인덱스하기 위해서 패트리시아 트라이(Patricia trie)를 사용하고, APEX는 자주 발생하는 질의에서 나타나는 레이블 경로들에 속하는 노드들에 대하여, 효과적인 검색을 수행하기 위해 해쉬 테이블을 사용한다. 이 두 가지 방법은 인덱스에 등록된 레이블 경로들에 대해 매우 효과적으로 질의를 수행할 수 있다. 그러나, 이 방법들은 레이블 경로의 수가 증가하는 경우에 효율성이 떨어진다. 왜냐하면, 어떤 질의 내의 레이블 경로는 그 인덱스에서 쉽게 발견되지 않을 것이기 때문이다. 따라서, 거대한 수의 고유한 레이블 경로들이 발생하는 인터넷과 같은 이질적인 환경에는 적합하지 않다. 더욱이, 인덱스 내에 등록되지 않은 레이블 경로들을 요구하는 질의들의 경우, 그 처리 비용 부담은 매우 심각해질 것이다. 예를 들어, Index Fabric 방법에서는 트라이 안에 있는 모든 노드들을 검색해야 한다. 또한, APEX 방법에서는 해쉬 테이블에 없는 레이블 경로들은 해쉬 테이블에 새로이 삽입해야 하고, 결과적으로 XML 문서 구조들을 요약하는 그래프에도 적용해야 한다.

뿐만 아니라, 이들 방법들은 부분 매치 질의를 효과적으로 지원하지 못한다. Index Fabric은 부분 매치 질의를 지원하지 않으며, APEX는 어떤 경로의 시작 부분에서 오로지 한개의 "//"를 가진 경로 형태는 다룰 수 있으나, 그 경로 상에 다수 개의 "//"들이 나타나는 경우(예, //.../B//C...)에는 질의를 효과적으로 다룰 수 없다. 이 경우, 그 질의는 질의 내 "//"를 "/"들의 연속인 형태로 대치하기 위해 질의를 재 작성(query rewrite)해야 한다(예, //.../B/C..., //.../B/B1/C..., //.../B/B2/B3/B4/C...). 그러나, APEX는 인트라넷(Intranet) 환경을 위해서만 디자인된 질의 재 작성 알고리즘[9]을 사용하였다. 여기서, 그러한 질의 재 작성이 가능한 이유는, 인트라넷 환경은 그 스키마가 동질성(homogeneous)을 가지고 있고, 구분된 레이블 경로의 수가 한정되기 때문이다. 뿐만 아니라, APEX에서 사용된 재 작성 알고리즘 [9]의 실행 시간(run-time)은 "//"들의 수에 대해 지수적(exponential)이다. 따라서, 이러한 점들은 이 방법들을 대규모 이질 XML 문서들에 대한 질의 처리 환경에 적합하지 않다.

### 3.3 인스턴스-레벨 방법

인스턴스-레벨 방법에서, 질의는 XML 트리 내의 각 노드를 위해 저장된 간격(즉, 시작과 끝 위치) 정보를 비교함으로 처리된다. 특히, 각 노드는 부모 노드의 간격이 자식 노드 각각의 간격을 포함하도록 할당된 간격 정보를 가지고 있으며, 부모-자식이나 조상-후손 관계성을 판별하기 위한 포함 관계 조인은 이 노드 간격 정보를 사용한다.

이러한 노드 간격 정보를 사용하는 기존 방법들로는 Element Numbering Scheme[17], Multi-Predicate Merge Join[24], Structural Join[2], Holistic Twig Join[4], 인덱스를 가진 Holistic Twig Join[15], XR-tree[14]와 결합된 Holistic Twig Join의 변형[4] 등이 있다.

이 방법들은 스키마 정보 없이 질의를 처리할 수 있다는 장점이 있다. 그러나, 질의 처리 비용은 스키마-레벨 방법보다 훨씬 높다. 왜냐하면, 스키마-레벨 방법은 스키마 정보를 먼저 이용함으로써, 검색 결과에 근접한 인스턴스들로 한정하는 반면, 인스턴스-레벨 방법은 스키마-레벨 정보를 사용할 수 없기 때문이다. 그러므로, 인스턴스-레벨 방법의 질의 처리 비용은 문서 수의 증

2) BPE는 단말 노드들까지의 LPE들로 분할하여 처리한다. 그러므로, XRel 방법 보다 작은 수의 LPE들로 분할 처리된다. XPaprent는 분할된 LPE들을 이용하여 각 노드 집합을 얻고, 이들 집합 간 포함 관계 조인을 수행하여 결과 노드 집합을 얻는다. 이 조인은 equi-join 만을 통해 수행된다.

가에 따라 매우 빠른 속도로 높아지는 단점이 있다. 따라서, 인스턴스-레벨 방법은 스키마-레벨 방법을 보조하는 방법으로 사용될 수 있는 독립된 방법이다.

## 4. XML 저장 구조

본 장에서는 XIR-Linear 방법에서 사용하는 XML 문서의 저장 구조를 제안한다. XML 문서는 스키마-레벨과 인스턴스-레벨 정보로 분리되어, 질의 처리에 필요한 정보의 형태로 관계형 데이타베이스 테이블에 저장된다. 본 논문에서는 XML 트리에서 발생하는 레이블 경로들과 그 레이블 경로들에 대한 역 인덱스(inverted index)로 구성되는 정보를 스키마-레벨 정보라 하며, XML 트리 내에 있는 모든 노드(노드 인스턴스)들로 구성되는 정보를 인스턴스-레벨 정보라 한다. XIR-Linear는 스키마-레벨 정보에 초점을 맞추어 이를 LabelPath 테이블과 LabelPath 역 인덱스(inverted index)에 저장한다. 그림 6은 그림 1의 XML 트리에 대한 스키마-레벨 정보의 예를 보여준다.

| pid | labelpath |
|---|---|
| 1 | $issue.issue.&issue |
| 2 | $issue.issue.editor.&editor |
| 3 | $issue.issue.editor.first.&first |
| 4 | $issue.issue.editor.last.&last |
| 5 | $issue.issue.articles.&articles |
| 6 | $issue.issue.articles.article.&article |
| 7 | $issue.issue.articles.article.@category.&@category |
| 8 | $issue.issue.articles.article.title.&title |
| 9 | $issue.issue.articles.article.author.&author |
| 10 | $issue.issue.articles.article.author.first.&first |
| 11 | $issue.issue.articles.article.author.last.&last |
| 12 | $issue.issue.articles.article.keyword.&keyword |

(a) LabelPath table

| keyword | posting list |
|---|---|
| $issue | : <1, 1, {1}, 3> <2, 1, {1}, 4> <3, 1, {1}, 5> ... |
| issue | : <1, 1, {2}, 3> <2, 1, {2}, 4> <3, 1, {2}, 5> ... |
| &issue | : <1, 1, {3}, 3> |
| article | : <6, 1, {4}, 5> <7, 1, {4}, 6> <8, 1, {4}, 6> ... |
| &article | : <6, 1, {5}, 5> |
| articles | : <5, 1, {3}, 4> <6, 1, {3}, 5> <7, 1, {3}, 6> ... |
| &articles | : <5, 1, {4}, 4> |
| editor | : <2, 1, {3}, 4> <3, 1, {3}, 5> <4, 1, {3}, 5> |
| &editor | : <2, 1, {4}, 4> |
| author | : <9, 1, {5}, 6> <10, 1, {5}, 7> <11, 1, {5}, 7> |
| &author | : <9, 1, {6}, 6> |
| first | : <3, 1, {4}, 5> <10, 1, {6}, 7> |
| &first | : <10, 1, {7}, 7> |
| last | : <4, 1, {4}, 5> <11, 1, {6}, 7> |
| &last | : <11, 1, {7}, 7> |
| title | : <8, 1, {5}, 6> |
| &title | : <8, 1, {6}, 6> |
| keyword | : <12, 1, {5}, 6> |
| &keyword | : <12, 1, {6}, 6> |
| @category | : <7, 1, {5}, 6> |
| &@category | : <7, 1, {6}, 6> |

(b) LabelPath Inverted Index

그림 6 LabelPath 테이블과 역 인덱스의 예

LabelPath 테이블은 XML 문서들에서 발생하는 중복이 배제된 모든 레이블 경로들과 그들의 경로 식별자(path identifier, pid)를 저장한다. 프리픽스(prefix)로써 '$'와 '&'가 덧 붙여진 레이블들은 각 레이블 경로의 첫 레이블과 마지막 레이블이라는 것을 표시한다. 이러한 덧붙임은 경로 표현식의 루트 레이블과, 결과 레이블에 매치 시키기 위해 필요한 것이다(자세한 내용은 5.1절에서 설명한다).

LabelPath 역 인덱스는 LabelPath 테이블의 label-path 필드에 대해서 구축된다. 이때, XIR-Linear는 각 레이블 경로를 텍스트 문서로 간주하고, 이들 레이블 경로 안에 있는 각 레이블을 키워드로 간주한다. 전통적인 역 인덱스[22]와 같이, LabelPath 역 인덱스 구조는 키워드(즉, 레이블)와 포스팅 리스트의 쌍들로 구성된다. 포스팅 리스트 내의 각 포스팅은 pid, occurrence_count, offsets, label_path_length와 같은 필드를 가진다. 여기서, pid는 레이블이 존재하는 레이블 경로의 식별자이고, occurrence_count는 레이블 경로 내에서 그 레이블이 나타난 개수이며, offsets는 레이블 경로가 시작하는 위치로부터 해당 레이블들의 위치까지의 거리들의 집합(즉, 옵셋의 집합)이다. label_path_length는 레이블 경로 내에 존재하는 레이블들의 개수이다. 예를 들어, 레이블 경로 $chapter.chapter.section.section.section.paragraph.&paragraph에서, section의 occurrence_count, section의 offsets, label_path_length는 각각 3, {3, 4, 5}, 7이다.

XIR-Linear는 XML 문서들 내에 있는 모든 노드들을 고유하게 식별하기 위하여 인스턴스-레벨 정보를 유지한다. XIR-Linear의 인스턴스-레벨 저장 구조는 XRel이나 XParent에서 사용한 저장 구조들을 선택적으로 사용할 수 있다. 그러나, 본 논문에서는 설명의 편의를 위하여, XParent의 인스턴스-레벨 저장 구조인 Element와 Data 테이블 구조를 XIR-Linear의 인스턴스-레벨 저장 구조로 사용한다.

## 5. XIR 질의 처리 알고리즘

본 절은 4장에서 설명한 XIR-Linear의 저장 구조를 기반으로 LPE를 처리하기 위한 알고리즘을 제안하고, 성능과 관련된 항목들에 초점을 맞추면서 XRel, XParent, XIR-Linear를 분석적으로 비교한다.

### 5.1 LPE 처리 알고리즘

그림 7은 LPE를 처리하는 알고리즘이다. 먼저, Label-Path 역 인덱스를 사용하여 LabelPath 테이블 내에 매치하는 레이블 경로를 찾고, 이로서 얻어진 레이블 경로들의 집합과 Element 테이블 간에 pid 컬럼을 이용하여 equi-join을 수행한다. 이때, 매치하는 노드들이 질의의 결과가 된다. XIR-Linear 방법은 기존 방법의 LPE 처

리 부분을 역 인덱스 검색으로 대치한 방법으로, 비용이 큰 스트링 매치를 사용하지 않는다는 장점이 있다.

```
Algorithm XIR-Linear_LPE_evaluation
Input: LPE P, LabelPath inverted index, Element table (in XParent)
Output: set of nodes matching the LPE
begin
Continue until the P is processed as follows,
1. Convert the input LPE P to an IR expression E using the syntactic
   mapping rule LPE-to-IRExp (Rule 1).
2. Find the set of pids (pidSet) using the LabelPath inverted index
   given the IR expression E.
3. Find the set of nodes(Nset) through an equi-join between
   pidSet and the Element table.
4. Return Nset.
end
```

그림 7 XIR-Linear LPE 처리 알고리즘

LPE의 처리를 정형적으로 표현하면 식 (1)과 같다.

$$\Pi\ nodepath\ (\sigma_{\text{MATCH (labelpath, LPE)}} LabelPath \bowtie_{pid=pid} Element) \quad (1)$$

식 (1)의 선택문 $\sigma_{\text{MATCH (labelpath, LPE)}}$ *LabelPath*는 labelpath 컬럼에 대한 텍스트 검색으로서 구현되기 때문에 먼저, LPE를 키워드 기반의 텍스트 검색 조건으로 변환해야 한다. 본 논문에서는 이 검색 조건을 정보 검색 표현식(*information retrieval expression, IRExp*)이라 한다. 다음 *Rule 1*은 이러한 변환을 완성하는 방법을 명시하고 있다.

**Rule 1** LPE $o_1\ l_1\ o_2\ l_2\ \cdots\ o_p\ l_p$ (여기서 $i=1,2,\cdots,p$)에 대한 $o_i \in \{$ '/', '//' $\}$는 다음 rule을 사용하여 IRExp로 변환된다.

$$o_1 l_1 \Longrightarrow \begin{cases} l_1 & \text{if } o_1 = \text{'//'} \\ \$l_1\ near(1)\ l_1 & \text{if } o_1 = \text{'/'} \end{cases}$$

$$l_i o_{i+1} l_{i+1} \Longrightarrow \begin{cases} l_i\ near(\infty)\ l_{i+1} & \text{if } o_{i+1} = \text{'//'} \\ l_i\ near(1)\ l_{i+1} & \text{if } o_{i+1} = \text{'/'} \end{cases} \quad \text{for } i = 1, 2, \ldots, p\text{-}1$$

$$l_p \Longrightarrow l_p\ near(1)\ \&l_p$$

여기서, *near(w)*는 최대 $w$개의 키워드들만큼 떨어진 두 개의 대상 키워드(operand keyword)들을 가진 문서들을 검색하기 위해 사용되는 근접 연산자(*proximity operator*)이다. □

이때, $l_1$과 $l_p$는 각각 LPE를 표현하는 선형 질의 패턴의 루트 노드 (즉, 첫 노드)와 결과 노드 (즉, 마지막 노드)가 됨에 주목해야 한다. 예를 들어, LPE //article//author/last는 IRExp article near(∞) author near(1) last near(1) &last으로 변환된다. 또한, LPE /issue/articles//author는 IRExp $issue near(1) issue near(1) articles near(∞) author near(1) &author로 변환된다. 이때, $issue는 issue가 그 문서의 루트 노드 임을 나타내기 위해 사용된다.

**예제 3.** LPE Q4를 생각해 보자. XIR-Linear는 Q4에 LPE-to-IRExp 규칙을 적용하여, LPE인//article//author/first를 IRExp인 article near(∞) author near(1) first near(1) &first로 변환한다. 그리고, 그림 6(b)의 LabelPath 역 인덱스에 대한 검색을 통해 *pidSet* {10}를 반환하고, 이 집합을 Element 테이블과 조인하여 *NPset* {19, 31}을 반환한다. 이러한 질의 처리 결과는 그림 1을 통해 검증될 수 있다. 그림 8은 LPE Q4를 처리하기 위해 생성된 XIR-Linear SQL 문을 보인다. WHERE 절에 있는 MATCH는 LabelPath 역 인덱스를 사용하게 하는 연산자이다. MAXINT 심볼은 시스템에 정의된 최대 정수 값이다. □

```
SELECT   DISTINCT n1.docid, n1.nodepath
FROM     LabelPath p1, NodePath n1
WHERE    p1.pid = n1.pid
AND      MATCH(p1.labelpath, 'article' NEAR(MAXINT) 'author'
                             NEAR(1) 'first' NEAR(1) '&first');
```

그림 8 LPE Q4를 위한 SQL 문

XIR-Linear의 LPE 처리 알고리즘은 *Rule 1*을 통해 주어진 LPE를 *IRExp*로 변환하고, 이를 통해 그림 6(b)에서 보인 역 인덱스를 검색함으로써(XML 구조 정보인) 레이블 경로를 빠르게 찾을 수 있는 방법이다. 그러므로, 4장에서 설명한 새로운 저장 구조와 본 장의 그림 7에서 보인 질의 처리 알고리즘을 통해 매우 우수한 선형 질의 처리 성능을 얻을 수 있다.

### 5.2 XRel 및 XParent와의 질의 처리 방법 비교

그림 9는 XRel, XParent와 제안된 XIR-Linear의 구

| 성능 관련 요소들 | XRel | XParent | XIR-Linear |
|---|---|---|---|
| 레이블 경로 매치 방법 | 스트링 매치 | 스트링 매치 | 역 인덱스 검색 |
| 엘리먼트 식별 방법 | 간격 (interval) | 노드 식별자 (node id) | XRel, XParent 방법 사용 |
| 레이블 경로 매치 시간에 영향을 미치는 요소 | 레이블 경로 테이블 내에 존재하는 레이블 경로의 개수 | 레이블 경로 테이블 내에 존재하는 레이블 경로의 개수 | LPE에 주어진 레이블들의 개수와 이들 레이블들과 연관된 포스팅 리스트들의 길이 |
| n 레이블 경로 탐색 비용 | *O (n)* | *O (n)* | *O (log n)* |

그림 9 XRel, XParent, XIR-Linear의 비교

표 1 질의들

| 그룹 | LPE 구분 | XPath 질의 | 선택률 |
|---|---|---|---|
| 그룹 1 (issue 관련 LPE) | LPE1 | /issue/editor | $10^{-7} \sim 10^{-6}$ |
| | LPE2 | //issue//first | $10^{-4} \sim 10^{-3}$ |
| | LPE3 | //issue//author/first | $10^{-5} \sim 10^{-4}$ |
| | LPE4 | //issue//article//author/first | $10^{-5} \sim 10^{-4}$ |
| 그룹 2 (movie 관련 LPE) | LPE5 | /movie/cast | $10^{-7} \sim 10^{-6}$ |
| | LPE6 | //movie//first | $10^{-6} \sim 10^{-5}$ |
| | LPE7 | //movie//actor//first | $10^{-6} \sim 10^{-5}$ |
| | LPE8 | //movie/cast//actor//first | $10^{-6} \sim 10^{-5}$ |

현 방법과 성능을 비교한 표이다.

XIR-Linear의 성능이 XRel이나 XParent에 비해 우수한 이유는, LPE에 매치되는 레이블 경로들을 찾기 위해 스트링 매치가 아닌, 역 인덱스를 사용한 검색을 수행하기 때문이다. XIR-Linear의 레이블 경로 매치 시간은 그 LPE 내에 있는 레이블들의 수와 역 인덱스에서 이들 레이블들에 대응되는 포스팅 리스트들의 길이에 의해서 결정된다. 반면, XRel과 XParent의 레이블 경로 매치 시간은 Path나 Labelpath 테이블 안에 저장된 레이블 경로들의 수에 비례하여 결정된다. 그러므로, 레이블 경로 테이블에 저장된 레이블 경로의 개수를 $n$ 이라 할 때, XRel이나 XParent의 수행 시간은 $O(n)$이나, XIR-Linear의 경우에는 그 수행 시간을 $O(\log n)$으로 줄일 수 있다.

## 6. 성능 평가

본 장에서는 XRel 및 XParent와 XIR-Linear의 질의 처리 성능을 비교한다. 이를 위해, 두 종류의 실험을 수행하였다. 그 첫번째는 데이타베이스 내에 있는 구분된 레이블 경로의 수를 기준으로 이들 방법의 효율성 (*efficiency*)을 비교한다. 두번째는 제안하는 방법이 큰 데이타베이스를 수용할 능력이 있는지에 대해 살펴보기 위해 확장성(*scalability*)을 보인다. 실험 결과는 XIR- Linear가 XRel이나 XParent에 비해 보다 큰 효율성을 가지고 있을 뿐 아니라 확장성도 가지고 있음을 나타낸다.

### 6.1 실험 설정

6.1.1 데이타베이스

본 실험에서는 두 개의 웹 크로울러(web crawler)인 Teleport Pro Version 1.29.1959[27]와 ReGet Deluxe 3.3 Beta (build 173) [26]을 사용하여 인터넷으로 부터 10008개의 XML 문서를 수집했다; 본 실험에서는 XMark 벤치마크 데이타(benchmark data)[28]와 같은 가공된 데이타 집합을 사용하지 않았다. 이는, 가공된 데이타가 특정한 도메인(domain)에 한정되어 있기 때문에 이로 인해, 충분히 이질적인 구조들을 얻어 낼 수 없기 때문이다.

수집된 XML 문서들은 크기에 따라 서로 구별되는 다섯 개의 데이타 화일 집합으로 나뉘어 저장되었다. 각 집합은 구분된 레이블 경로(distinct label path)들을 기준으로 대략 5000, 10000, 20000, 40000, 80000개를 포함하고 있다. 이들 중 마지막 집합은 1460000개의 노드를 가진다. 더 큰 집합은 더 작은 집합이 가진 레이블 경로들을 모두 가지고 있다. 즉, 더 큰 집합은 더 작은 집합의 수퍼셋(superset)이다. 각 집합은 세개의 데이타베이스들, 즉, XRel, XParent, XIR-Linear에 의해 사용되는 각각의 테이블들을 포함하고 있는 서로 다른 데이타베이스들로 로드된다. 그러므로, 생성되는 데이타베이스의 총 개수는 15개이다.

본 논문에서는 XRel과 XParent의 원 논문[12,23]에서 사용한 것과 같은 데이타베이스 스키마와 인덱스들을 사용하였다. XIR-Linear는 XRel이나 XParent에서와 같이, 각 테이블의 pid와 docid 컬럼에 B+-tree 인덱스를 구축한 LabelPath 테이블로 데이타 화일들을 로드했고, Labelpath 테이블의 labelpath 컬럼에 역 인덱스를 구축했다.[3)]

6.1.2 질의

Table 1은 XPath LPE의 두 가지 그룹이다. 그룹 1은 issue 문서들에 관한 것이고, 그룹 2는 movie 문서들에 관한 것이다. 전자는 후자보다 많은 문서 인스턴스들을 가진다. 표 1의 선택률은 데이타베이스 내의 전체 노드 개수 중 해당 LPE로 인해 얻어진 결과 노드의 개수의 비율이며, 선택률이 구간으로 표시된 이유는 크로울된 XML 문서로 작성된 데이타베이스 크기(레이블 경로 기준 5000~80000)에 따른 선택률의 가변 범위를 표시하기 위함이다.

6.1.3 수행 환경

실험은 512~Mbyte RAM을 가진 SUN Ultra~60

3) [3, 8, 10]과 같은 문헌에서 볼 수 있는 다른 연구들은 labelpath 컬럼이 아닌 'value' 컬럼에 역 인덱스를 사용한다. 그러나, 그들의 동작 체계와 목적 등은 XIR-Linear의 것과 상당히 다르다.

워크스테이션 상에서 문서 검색 엔진(text search engine)이 필요로 하는 연산들을 지원하는 오디세우스 객체 관계형 데이타베이스 관리 시스템 [25][4]을 사용하여 수행하였다. 운영 체제의 예상치 못한 버퍼링 효과를 제거하기 위하여 원시 디스크 장치(raw disk device)를 사용하였다. 또한, 질의 수행 결과가 바로 이전에 수행된 결과로 인해 영향 받지 않게 하기 위하여, 각 질의 수행 이후에 DBMS 버퍼를 초기화하였다. 사용된 비용 측정 기준은 수행 시간(elapsed time)과 디스크 입출력 횟수(the number of disk I/O's)이다. 데이타베이스 버퍼의 크기는 불충분한 버퍼 크기로 인해 발생하는 추가적인 디스크 입출력 횟수를 제거하기 위해 충분한 크기인 200 페이지로 설정하였다.

### 6.2 실험 결과

6.2.1 실험 1: 효율성

크로울러들은 인터넷으로부터 임의의 문서들을 수집하기 때문에, 크로울러에 의해 새로운 문서들이 더해질 때마다 새로운 레이블 경로들이 추가된다. 그림 10은 수집된 XML 문서들로부터 구분되는 레이블 경로들을 추출한 결과이다.

그림 11과 그림 12는 issue 그룹에 대한 효율성 비교 실험 결과이고, 그림 13과 그림 14는 movie 그룹에 대한 효율성 비교 실험 결과이다. 이 실험 결과들은 XRel, XParent, XIR-Linear에 대해 레이블 경로 개수의 증가에 따른 질의 유형 LPE1, LPE3, LPE6, LPE8의 질의 처리 비용을 보인 것이다. 실험 결과들을 통해, XRel이나 XParent의 성능은 질의 유형 보다 경로 테이블에 저장된 레이블 경로의 개수에 비례한다는 것을 보이는 반면, XIR-Linear의 성능은 LPE에 주어진 레이블의 개수와 각 레이블의 포스팅 리스트 길이(즉, 포스팅의 레이블 경로의 개수에 비해 매우 작기 때문에 방법의

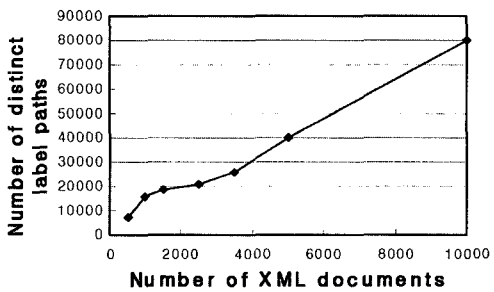
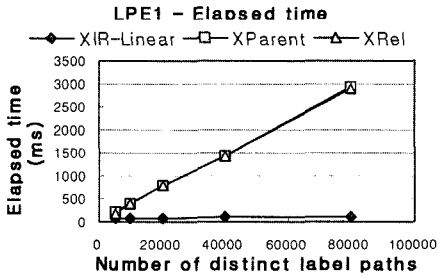



그림 10 XML 문서의 수가 증가함에 따라 변화하는 구분된 레이블 경로의 수



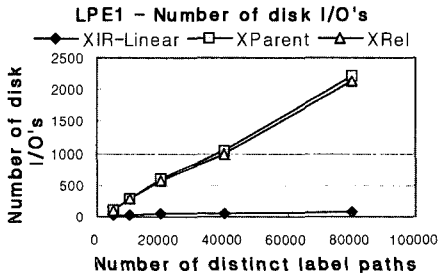
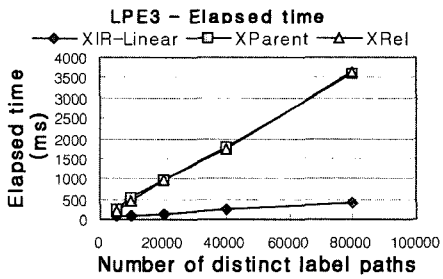




그림 11 LPE1을 위한 XRel, XParent, XIR-Linear의 질의 비용 (버퍼 크기 = 200 pages)



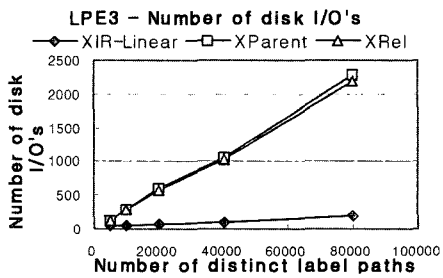




그림 12 LPE3을 위한 XRel, XParent, XIR-Linear의 질의 비용 (버퍼 크기 = 200 pages)

개수)에 비례한다는 것을 보인다. 그러나, 이 수는 전체 복잡도에 영향을 미치지 않는다.

그림 15는 레이블 경로 개수 증가에 따른 레이블 별 레이블 빈도수 (포스팅의 개수)를 측정한 결과이다. 레이블들 중 article, issue, author, first 등은 XML 문서

4) 본 시스템은 한국과학기술원(KAIST) 첨단정보기술연구센터에서 개발하였다. 본 시스템의 아키텍처는 텍스트 검색 엔진에 의해 요구되는 연산들을 지원한다.

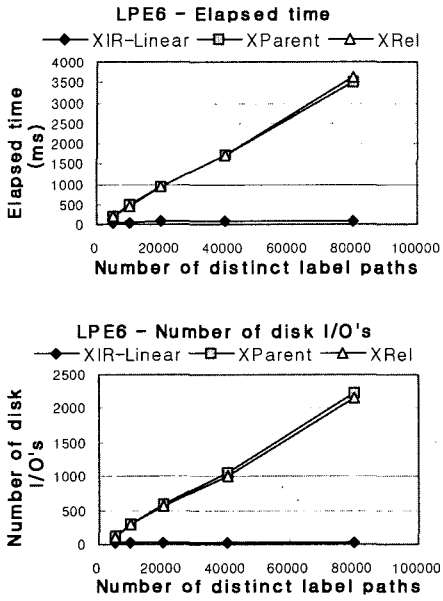



그림 13 LPE6을 위한 XRel, XParent, XIR-Linear의 질의 비용 (버퍼 크기 = 200 pages)

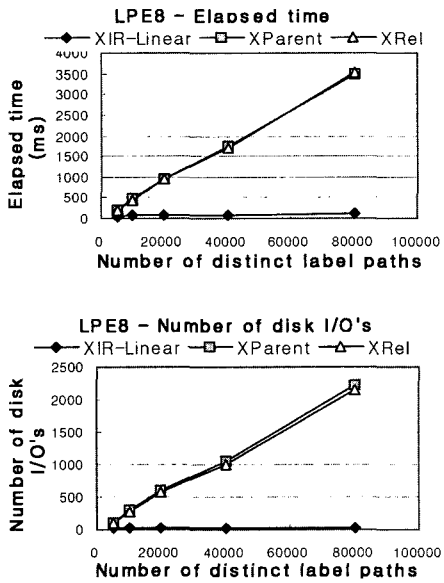



그림 14 LPE8을 위한 XRel, XParent, XIR-Linear의 질의 비용 (버퍼 크기 = 200 pages)

들 중 빈도수가 매우 높은 레이블들이다. 이들의 레이블 빈도수는 레이블 경로 개수가 80000개일 경우, 레이블 빈도수가 가장 높은 article로 부터 first까지 차례로 5288, 3690, 2624, 1286임을 보인다. 그러나, 대부분의 레이블들에 대한 포스팅 개수는 first 보다도 작다. 이러

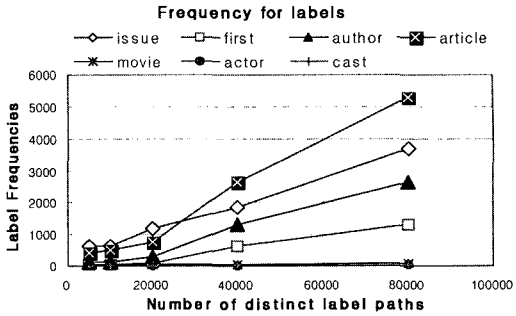



그림 15 레이블 경로 증가에 따른 레이블 별 레이블 빈도수

한 레이블에 대한 포스팅의 개수는 레이블 경로의 개수에 비해 상대적으로 매우 작기 때문에 거의 모든 부분 매치 질의에 대해 우수한 성능 결과를 보이게 된다. 그림 12의 LPE3에 대한 결과는 XIR-Linear가 다른 질의 유형들에 비해 질의 처리 비용이 상대적으로 크다는 것을 보여주고 있다. 여기서, LPE3 내 각 레이블들(issue, author, first)은 문서 내 빈도수가 높은 레이블들이므로, 역 인덱스의 각 포스팅 리스트가 매우 길어져 질의 처리 비용이 다른 질의들에 비해 상대적으로 크게 나타난다. 그러나, 이러한 경우에도 다른 방법들에 비해 월등히 좋은 성능을 보인다. 그러므로, XIR-Linear는 레이블 경로의 개수가 커질 수 있는 인터넷 환경에서 사용할 수 있는 선형 경로 질의 처리 방법이라 할 수 있다. 다른 질의 유형들에 대한 결과는 유사한 경향을 보이므로 본 논문에서는 생략한다.

그림 11~14를 통해, XIR-Linear가 XRel이나 XParent 보다 효율적으로 질의를 처리함을 알 수 있다. XRel과 XParent는 레이블 경로의 개수 증가에 따라 질의 처리 시간과 디스크 입출력 횟수가 거의 선형적(linear)으로 증가하는 반면, XIR-Linear의 경우는 서브-선형적(sublinear)으로, 거의 상수적(nearly constant)으로 증가한다. 이러한 결과는 매치하는 레이블 경로를 찾기 위한 방법들인, 스트링 매치와 역 인덱스 검색 간의 차이가 가져오는 영향이 반영된 것이다. XIR-Linear는 실험 범위 내에서 기존 방법들에 비해 수십 배 이상 성능이 우수하다. 실제로, 질의 대상 문서 수가 증가함에 따라 그 성능의 차이는 점차 커지므로 점차 더 좋은 질의 처리 능력을 보인다.

6.2.2 실험 2: 확장성

실험 데이타베이스의 크기는 고정되어 있으므로, 대신 버퍼 크기를 줄여가면서 데이타베이스 크기가 커지는 효과를 모의 실험하였다. 그림 16은 데이타베이스 크기 (= 114000 페이지)의 0.05% 로 부터 0.005% (= 6 페이

지)까지 점차 작아지는 버퍼 크기에 대하여, 변화하는 질의 처리 비용을 로그 스케일로 보인다. 그림 16에서, 버퍼 크기가 점차 작아지게 될 때, XRel과 XParent는 그 처리 비용이 상당량 증가하는 것에 비해, XIR-Linear는 약간만 증가하는 것을 볼 수 있다. 이것은 데이타베이스 크기 관점에서 XRel이나 XParent 보다 XIR-Linear가 좋은 확장성을 가졌다는 것을 보여주는 결과이다.

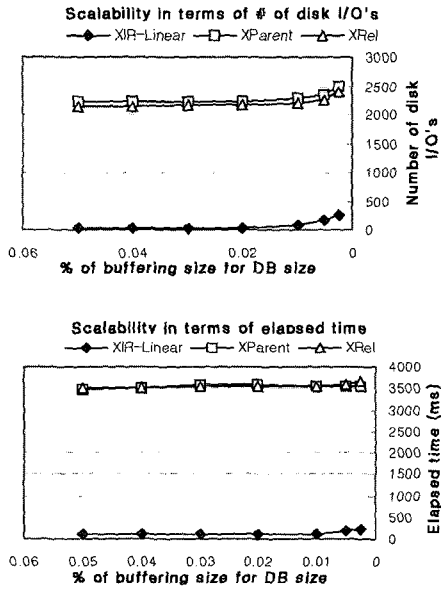



그림 16 버퍼 크기 변화에 따른 LPE8 질의 비용(구분된 레이블 경로의 수 = 80000)

## 7. 결 론

본 연구는 인터넷 환경에서 전형적인 형태로 존재하는, 이질적이고 방대한 양의 XML 문서들과 이에 주어질 질의를 처리하기 위하여, XIR-Linear라는 새로운 질의 처리 방법을 소개하였다. XIR-Linear가 사용하는 주요 기술은 XML 문서에서 발생하는 레이블 경로들을 텍스트들로 간주하고, 그들 위에 역 인덱스를 구축하는 것이다. 이 역 인덱스는 선형 경로 표현식 처리시, XRel이나 XParent의 스트링 매치 보다 훨씬 빠른 부분 매치를 지원한다.

본 논문에서는 XML 문서의 모든 레이블 경로를 저장하는 테이블과 역 인덱스를 포함하는 저장 구조와 선형 경로 표현식을 정보 검색 표현식(information retrieval expression, IRExp)으로 변환하는 방법을 제안하였다. 그리고, 실험을 통하여 XIR-Linear의 성능을 XRel, XParent와 비교하였다. 실험 데이타로는, 크로울러를 사용하여 인터넷으로 부터 수집한 실제 XML 문서들을 사용하였다. 실험 결과는 비교하는 XRel이나 XParent에 비해, XIR-Linear이 수배에서 수십(백)배 이상 효과적이며, 데이타베이스 크기에 대해 확장성을 가졌음을 보였다.

## 참 고 문 헌

[1] A. Aboulnaga, A. R. Alameldeen, and J. Naughton, "Estimating the Selectivity of XML Path Expressions for Internet Scale Applications," In *Proc. the 27th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 591-600, Rome, Italy, Sept. 11-14, 2001.

[2] S. Al-Khalifa, H. V. Jagadish, N. Koudas, and J. M. Patel, "Structural Joins: A Primitive for Efficient XML Query Pattern Matching," In *Proc. the 18th Int'l Conf. on Data Engineering (ICDE)*, pp. 141-152, San Jose, California, Feb. 26 Mar. 1, 2002.

[3] Jan-Marco Bremer and Michael Gertz, "XQuery/IR: Integrating XML Document and Data Retrieval," In *Proc. the Fifth Int'l Workshop on the Web and Databases (WebDB 2002)*, pp. 1-6, Madison, Wisconsin, 2002.

[4] N. Bruno, N. Koudas, and D. Srivastava, "Holistic Twig Joins: Optimal XML Pattern Matching," In *Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 310-321, Madison, Wisconsin, June 3-6, 2002.

[5] J. Clark and S. DeRose, XML Path Language (XPath), W3C Recommendation, *http://www.w3.org/TR/xpath*, Nov. 1999.

[6] C. Chung, J. Min, and K. Shim, "APEX: An Adaptive Path Index for XML Data," In *Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 121-132, Madison, Wisconsin, June 3-6, 2002.

[7] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon, "A Fast Index for Semistructured Data," In *Proc. the 27th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 341-350, Rome, Italy, Sept. 11-14, 2001.

[8] Daniela Florescu, Donald Kossmann, and Ioana Manolescu, "Integrating Keyword Search into XML Query Processing," In *Proc. the 9th WWW Conference/Computer Networks*, pp. 119-135, Amsterdam, NL, May 2000.

[9] M. F. Fernandez and D. Suciu, "Optimizing Regular Path Expressions using Graph Schemas," In *Proc. the 14th Int'l Conf. on Data Engineering (ICDE)*, pp. 14-23, Orlando, Florida, USA, Feb. 23-27, 1998.

[10] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel

Shanmugasundaram, "XRANK: Ranked Keyword Search over XML Documents," In *Proc. 2003 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 16-27, San Diego, California, June 9-12, 2003.

[11] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semi-structured Databases," In *Proc. the 23th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 436-445, Athens, Greece, Aug. 26-29, 1997.

[12] H. Jiang, H. Lu, W. Wang, and J. Xu Yu, "Path Materialization Revisited: An Efficient Storage Model for XML Data," In *Proc. the 13th Australasian Database Conference (ADC)*, pp. 85-94, Melbourne, Australia, Jan. 28 - Feb. 1, 2002.

[13] H. Jiang, H. Lu, W. Wang, and J. Xu Yu, "XParent: An Efficient RDBMS-Based XML Database System," In *Proc. the 18th Int'l Conf. on Data Engineering (ICDE)*, pp. 335-336, San Jose, California, Feb. 26 - Mar. 1, 2002.

[14] H. Jiang, H, Lu, W, Wang, and B. C. Ooi, "XR-Tree: Indexing XML Data for Efficient Structural Joins," In *Proc. the 19th Int'l Conf. on Data Engineering (ICDE)*, pp. 253-264, Bangalore, India, Mar. 5-8, 2003.

[15] H. Jiang, W. Wang, H. Lu, and J. X. Yu, "Holistic Twig Joins on Indexed XML Documents," In *Proc. the 29th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 273-284, Berlin, Germany, Sept. 9-12, 2003.

[16] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth, "Covering Indexes for Branching Path Queries," In *Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 133-144, Madison, Wisconsin, June 3-6, 2002.

[17] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expressions," In *Proc. the 27th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 361-370, Rome, Italy, Sept. 11-14, 2001.

[18] F. Mandreoli, R. Martoglia, P. Tiberio, "Searching Similar (Sub)Sentences for Example-Based Machine Translation," In *Proc. SEBD'02*, Isola d'Elba, Italy, June 2002.

[19] J. Naughton et al., "The Niagara Internet Query System," *IEEE Data Engineering Bulletin*, Vol. 24, No. 2, pp. 27-33, June, 2001.

[20] C. Petrou, S. Hadjiefthymiades, and D. Martakos, "An XML-based, 3-tier Scheme for Integrating Heterogeneous Information Sources to the WWW," In *Proc. the 10th Int'l Workshop on Database and Expert Systems Applications*, pp. 706-710, Florence, Italy, Sept.1-3, 1999.

[21] N. Polyzotis and M. Garofalakis, "Statistical Synopses for Graph-structured XML Databases," In *Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 358-369, Madison, Wisconsin, June 3-6, 2002.

[22] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York 1983.

[23] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, "XRel: A Path-based Approach to Storage and Retrieval of XML Documents using Relational Databases," *ACM Transactions on Internet Technology(TOIT)*, Vol. 1, No. 1, pp. 110-141, 2001.

[24] C. Zhang, J. F. Naughton, D. J. DeWitt, Q. Luo, and G. M. Lohmann, "On Supporting Containment Queries in Relational Database Management Systems," In *Proc. 2001 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 425-436, Santa Barbara, California, May 21-24, 2001.

[25] Odysseus Object-Relational Database Management System, *http://odysseus.kaist.ac.kr/*.

[26] ReGet Deluxe 3.3 Beta (build 173), *http://deluxe.reget.com/en/*.

[27] Teleport Pro Version 1.29, *http://www.tenmax.com/teleport/pro/home.htm*.

[28] XMark-An XML Benchmark Project, *http://monetdb.cwi.nl/xml/*.

[29] Xyleme, *http://www.xyleme.com*.

[30] eXtensible Markup Language(XML), *http://www.w3.org/XML/*.

박 영 호

1990년 2월 동국대학교 컴퓨터공학과 학사. 1992년 2월 동국대학교 컴퓨터공학과 석사. 1999년 2월 한국전자통신연구원 선임연구원. 1999년 3월~현재 한국과학기술원 전산학과 박사과정. 관심분야는 XML, Web DBMS, 정보 검색, 운영체제, 임베디드 시스템

한 욱 신

정보과학회논문지 : 데이타베이스 제 31 권 제 4 호 참조

황 규 영

정보과학회논문지 : 데이타베이스 제 31 권 제 4 호 참조