분산 환경에서 전역 XQuery 질의의 조인 선택치 추정 방법

(Estimating Join Selectivity of Global XQuery Queries in Distributed Environments)

박 종 현 † 강 지 훈 ††
(Jong-Hyun Park) (Ji-Hoon Kang)

요 약 분산환경에 XML 데이타들을 통합하기 위한 한가지 방법은 XML 뷰를 사용하는 것이다. 사용자는 XML을 위한 표준 질의어인 XQuery를 사용하여 분산된 XML 뷰들을 대상으로 전역 XQuery질의를 생성할 수 있다. 이렇게 생성된 전역 XQuery 질의는 분산된 이종 데이타들을 통합하고 검색하기 위하여 자연스럽게 지역 시스템들 사이의 조인 연산들을 포함한다. 그러나 조인은 비용이 많이 드는 연산자이므로 조인 연산을 효율적으로 처리하는 것은 전역 질의의 처리 성능과 직결된다. 그러므로 조인 연산을 처리하기 위한 다양한 연구들이 존재하며, 그 가운데 하나는 조인의 선택치를 추정하여 최소의 비용을 갖는 조인 순서를 선택하는 것이다. SQL 질의의 경우, 이미 전역 질의의 조인 선택치를 추정하고 이를 기반으로 그 처리 순서를 결정하기 위한 연구가 존재한다. 그러나, 테이블 구조의 데이타를 검색하기 위한 SQL 질의의 조인 선택치 추정 방법을 구조적인 XML 데이타를 검색하기 위한 XQuery질의를 위해서 그대로 사용하기에는 데이타의 구조적인 차이로 인해 문제가 있다. 그러므로 본 논문에서는 질의의 대상이 되는 XML 뷰들의 정보를 이용하여 XQuery 질의의 특성을 고려한 조인 선택치 추정 방법을 제안한다. 본 논문의 기여는 다음과 같다. 첫째, SQL 질의의 조인 선택치 추정 방법과 XQuery 질의의 방법 사이에 차이점을 분석한다. 둘째, XML 뷰를 참조하여 XQuery 질의의 처리를 위한 조인 선택치 추정 방법을 제안한다. 마지막으로, 성능 평가를 수행하여 제안하는 조인 선택치 추정 방법의 효율성을 입증한다.

키워드 : 전역 XQuery 처리, 조인 선택치 추정

Abstract One of the methods for integrating XML data in distributed environments is using XML view. User can query toward distributed local XML views by using global XQuery queries in XQuery which is a standard query language for searching XML data. The global XQuery queries naturally contain join operations because of integrating and searching distributed heterogeneous data. Since join operations are generally expensive for processing a query, its processing technique is very important for efficient processing of global XQuery queries. Therefore there are some studies on the efficient processing of join operations and one of these studies is that selects minimum join cost by estimating a join selectivity. In case of SQL, there are already some researches for estimating a join selectivity and join cost of global SQL queries. However we can not apply their methods for estimating the selectivity of join operations in SQL queries into XQuery queries because of the structural difference between relational data and XML data. Therefore this paper proposes a method for estimating a selectivity of join operations in XQuery queries using the information of XML views. Our contribution is three threefold. First, we define the difference point for estimating join selectivity between SQL and XQuery. Second, we estimate join selectivity in XQuery queries by referring XML views. Third, we evaluate our estimating method.

Key words : Global XQuery Processing, Estimating Join Selectivity

· 이 논문은 2005년도 충남대학교 학술연구비의 지원에 의하여 연구되었음

† 학생회원 : 충남대학교 컴퓨터전공
Jonghyunpark@cnu.ac.kr
†† 정 회 원 : 충남대학교 컴퓨터전공 교수
jhkang@cnu.ac.kr
(Corresponding author임)

논문접수 : 2007년 6월 18일
심사완료 : 2007년 8월 14일

정보과학회논문지: 데이타베이스 제34권 제6호(2007.12)

1. 서 론

XML은 인터넷 상에서 데이타 교환을 위한 표준이다. 그러므로 이종의 레거시 데이타를 보유하고 있는 응용들은 XML 데이타와의 통합을 위하여 다양한 접근 방법들을 찾고 있으며, 그 가운데 하나가 XML 뷰를 이용하여 이종 데이타들을 XML 데이타처럼 바라볼 수 있도록 하는 방법이다. XML 뷰는 DTD[1], XML Schema[2], XQuery[3] 등을 이용하여 기술할 수 있다[4, 5, 6]. 사용자는 분산된 XML 뷰들을 대상으로 질의할 수 있으며, 이때 검색을 위한 질의어로 XQuery를 사용하는 것은 상호 운용성 측면에서 자연스러운 선택이다.

본 논문에서는 분산된 다수의 XML 뷰들을 대상으로 작성된 XQuery 질의를 전역 XQuery 질의라 부른다. 이 질의는 여러 지역 시스템들에 저장된 데이타들을 통합하고 검색하므로 다수의 조인 연산을 포함할 수 있다. SQL 질의의 경우, 조인 연산은 질의 처리를 위하여 비용이 많이 드는 연산자들 가운데 하나로 구분된다[7]. 그러므로 조인 연산을 효율적으로 처리하는 것은 결국 질의의 처리 성능을 좌우한다. XQuery 질의 역시 이와 크게 다르지 않으므로 효율적인 질의 처리를 위하여 질의에 포함된 조인 연산을 처리하기 위한 연구는 반드시 필요하다. 특별히 전역 XQuery 질의는 분산된 다수의 지역 시스템들 사이에 조인 연산을 포함하므로 이 연산을 처리하기 위한 방법은 질의 처리 시간을 결정하는 중요한 요인이다. SQL 질의의 경우, 전역 질의를 효율적으로 처리하기 위하여 조인 선택치를 추정하고 이를 기반으로 조인 처리 순서를 결정하기 위한 연구들은 이미 많이 존재한다[8-11]. 그러나 SQL은 그 질의의 대상이 테이블 구조의 데이타인 반면 XQuery 는 반 구조적인 XML 문서이다. 그러므로 SQL 질의를 위한 조인 선택치 추정 방법을 XQuery 질의를 위해서 그대로 적용할 수 없다. 예를 들어 테이블의 각 속성들 사이의 관계는 모두 일대 일 관계인 반면 XML 문서의 각 노드들 사이의 관계는 일 대 일, 다 대 다, 다 대 일, 또는 일 대 다의 관계이다. 그러므로 본 논문에서는 XQuery 질의의 조인 선택치 추정 방법과 SQL 질의의 조인 선택치 추정 방법 사이의 차이점을 분석한다. 또한 XML 뷰를 참조하여 XQuery 질의에 포함된 조인들의 선택치를 추정하기 위한 방법을 제안하고 이를 기반으로 조인 비용을 예측한다. 마지막으로, 제안하는 선택치 추정 방법을 프로토타입으로 구현하고 그 성능을 평가한다.

논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고, 3장에서는 배경지식을 기술한다. 4장에서는 SQL 질의의 조인 선택치 추정 방법과 XQuery 질의의 조인 선택치 추정 방법 사이에 서로 다른 구조적인 차이점을 기술하고 이를 고려한 조인 선택치 추정 알고리즘을 기술한다. 5장에서는 우리의 알고리즘을 적용하여 조인 비용을 예측하고 그 결과를 평가한다. 마지막으로 6장에서는 결론을 맺고 향후 연구에 대하여 기술한다.

2. 관련 연구

분산환경에서 SQL 질의에 포함된 조인을 효율적으로 처리하기 위한 연구는 이미 많이 존재한다[10-12]. 이러한 연구들 가운데 대표적인 한가지 방법은 조인 선택치를 추정하여 비용을 예측하고, 최소 비용을 갖도록 조인의 순서를 결정하는 것이다[12]. XQuery 질의의 경우 역시 XML 전용 데이타베이스를 연구하는 곳에서 질의에 포함된 조인을 효율적으로 처리하기 위한 몇몇 연구가 존재한다[13,14]. 그러나 아직까지 분산 환경에 XQuery 질의의 조인을 처리하기 위한 연구는 많지 않다. 그러므로 XQuery 질의의 조인을 효율적으로 처리하기 위해서 SQL 질의의 조인 처리 방법을 참조하는 것은 바람직한 방법처럼 보인다.

[10-12]는 분산 환경에서 SQL질의의 조인 선택치를 추정하고 이를 기반으로 비용을 예측하기 위한 비용 모델을 제안하고 이를 기반으로 최적의 조인 순서 결정을 위한 방법을 제안하고 있다. 이들이 제안하고 있는 비용 모델은 관계형 데이타베이스를 기반으로 하고 있고, 데이타베이스가 제공하는 통계정보들을 이용하여 정확한 비용을 계산할 수 있다. 이러한 접근 방법은 XQuery 질의를 위해서 적용이 가능하다. 예를 들면 테이블의 주키를 이용하여 SQL의 조인 선택치를 구하는 방법이 이에 해당된다. XML 문서의의 ID속성이 테이블의 키와 유사한 역할을 하므로 SQL의 조인 선택치를 구하는 법을 우리의 방법에 적용할 수 있다. 그러나 이미 언급한 것처럼, SQL과 XQuery의 구조적인 특성 때문에 이들이 제안하고 있는 비용 모델을 우리의 방법에 그대로 적용하는 것은 어렵다. 그러므로 본 논문에서는 XQuery 질의의 조인 순서를 결정하기 위하여 앞선 연구들과의 차이점을 분석하고 XQuery 질의를 위한 조인 선택치 추정 방법을 제안한다, 특별히 SQL 질의의 조인 선택치 추정 방법으로부터 XQuery 질의의 조인 선택치를 추정하기 위한 ESNJ(Estimating Selectivity of Next Join) 알고리즘을 제안하고, ESNJ를 XQuery의 조인 비용 계산에 적용한다.

[13,14]는 XQuery 질의에 존재하는 조인을 효율적으로 처리하기 위한 방법을 연구하므로 본 논문의 목적과 유사하다. 그러나 이 연구들은 XQuery 질의에 포함된 조인의 처리 비용을 계산하기 위하여 조인에 포함된 XPath 표현을 처리하는 비용과 데이타의 I/O 비용, 그리고 연산자의 비교 횟수 등을 고려하여 비용을 결정한

다. 물론 이러한 요소들 역시 효율적으로 XQuery 질의를 처리하기 위해서 중요하지만 본 논문에서는 XQuery 질의에 기술된 조인 연산에 그 초점을 맞추고 있다. 또한 이들 방법은 XML 전용 데이타베이스 내부에 저장된 XML 데이타들을 위한 방법이다. 그러므로 정확한 비용을 추정하여 최적의 조인 순서를 측정할 수 있는 장점을 가지는 반면 일반적인 방법은 아니다. 다시 말하면, 이들 방법은 시스템의 내부 구조와 인덱싱 기법 등에 의존적인 방법이므로 다른 시스템에서 적용하기 어렵다. 그러나 우리의 방법은 응용에 독립적으로 XQuery 질의의 조인 선택치를 추정하고 이를 기반으로 조인 비용을 결정하므로 일반적인 시스템에서 적용할 수 있다.

3. 배경 지식

XML 뷰는 지역 시스템에서 보유하고 있는 여러 종류의 데이타를 XML 데이타와 통합하기 위하여 DTD, XML Schema, XQuery 등을 이용하여 XML로 표현한다. 그러므로 XML 뷰에는 데이타의 구조 정보와 각 노드의 발생(Occurrence) 정보가 기술되어있다[1]. 이러한 정보들은 XML 데이타와 관계형 데이타 사이에 서로 다른 점이므로 반드시 XQuery 질의의 조인 선택치 추정을 위해서 고려해야 한다.

SQL의 경우, 조인 선택치를 추정하기 위해서 조인에 포함된 속성의 원소 수(Cardinality)를 기본 정보로 사용한다[7,10]. 그러므로 본 논문에서 역시 XML 뷰에 표현된 노드들의 수는 얻을 수 있다고 가정한다. 물론 XQuery Function and Operations[15]에는 해당 노드의 수를 얻을 수 있는 Count()와 같은 함수가 존재한다. 또한 이러한 함수를 사용하지 않고도 Count를 계산하기 위한 몇몇 연구가 존재한다[16,17].

그림 1은 트리로 표현된 3개의 XML 뷰고, 각 뷰는 3개의 지역 시스템 A, B, C에 저장하고 있는 지역 데이타를 표현한다. 뷰의 각 노드에 표현된 Label은 노드의 발생(Occurrence) 정보와 원소 수(Cardinality)를 표현한다. 예를 들어, B 시스템의 b1 (?,4)은 b1노드가 지역

```
For $A in doc("A.xml")/root_A/a
For $A2 in $A/a2
For $B in doc("B.xml")/root_B/b
For $B2 in $B/b1/b2
For $B3 in $B/b3
For $C in doc("C.xml")/root_C/c
For $C1 in $C/c1
Where $A2 = $B2 and $B3 = $C1
Return
   <result>{$A/a1, $C/c2}</result>
```

그림 2 전역 XQuery 질의

시스템에 4개 존재하며, b1의 부모인 b노드에 대하여 b1노드가 발생하거나 그렇지 않음을 표현한다.

사용자는 분산된 XML 뷰들을 대상으로 전역 XQuery 질의를 작성할 수 있다. 그림 2는 그림 1의 XML 뷰들을 대상으로 작성된 전역 XQuery 질의의 예이다. 이 전역 질의는 3개의 지역시스템으로 질의를 하고 지역시스템들 사이의 2개의 조인을 포함한다.

그림 3은 그림 1의 XML 뷰들을 대상으로 그림 2의 전역 질의에 기술된 조인이 어떻게 연결되었는지 보여준다. 첫 번째 조인은 A 시스템의 변수 $A와 B 시스템의 변수 $B2이고, 두 번째 조인은 B시스템의 $B3와 C 시스템의 $C1이 조인으로 연결되어 있다.

본 논문의 목적은 이와 같이 조인의 구조와 각 노드들의 연결된 관계를 분석하여 XQuery를 위한 조인 선택치를 추정하고 이를 기반으로 각 조인의 비용을 예측하는 것이다.

4. 전역 XQuery 질의의 조인 선택치 추정

4.1 문제 정의

최적의 조인 순서를 선택하기 위해서 사용되는 방법들 가운데 일반적인 한가지 방법은 수행 가능한 모든 조인 비용을 추정하여 그 합이 최소인 방법을 찾는 것이다[7,18]. 이때 조인 비용의 측정을 위한 방법으로 본 논문에서는 조인을 처리하기 위해서 필요한 값의 비교 횟수를 측정한다. 예를 들어, 그림 2에 기술된 질의의 경우, A 시스템과 B시스템 사이의 조인 연산을 먼저 수행하고, B 시스템과 C 시스템 사이의 조인 연산을 수행

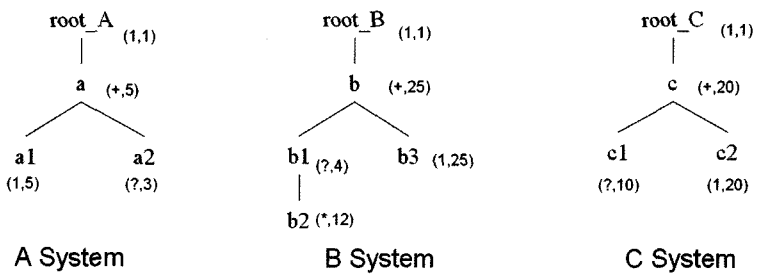

그림 1 XML 뷰들

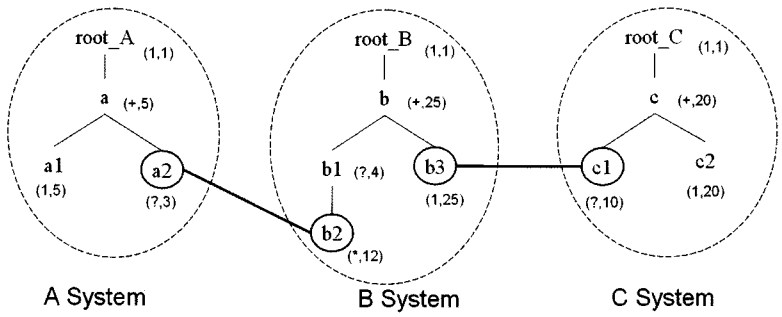

그림 3 그림 2의 XQuery 질의에 대한 조인 그래프

한다면 다음과 같이 조인 비용을 계산할 수 있다.

A⋈B의 Join Cost = |A.a2⋈B.b2| = |$A2⋈$B2|

B⋈C의 Join Cost = |B'.b3⋈C.c1| = |$B3⋈$C1|

위 계산식 가운데 |$A2⋈$B2|의 선택치를 구하는 방법은 SQL의 경우와 크게 다르지 않다. SQL 질의의 경우, 조인 선택치는 피연산자가 ID인 경우와 그렇지 않은 경우로 구분되어 표 1과 같은 세 가지 추정 방법에 의해서 얻어질 수 있으며[20]. XQuery 질의의 경우 역시 다르지 않다. 예를 들어, 그림 3의 경우, 첫 번째 조인인 A와 B지역 시스템 사이에 존재하는 조인 선택치는 다음과 같이 구할 수 있다.

• *Join Selectivity(A.a2, B.b2)* =
1/Max(|A.a2|,|B.b2|) = 1/|B.b2| = 1/12

그리고, 이를 기반으로 추정한 |$A2⋈$B2|의 결과는 다음과 같다.

• *|$A2⋈$B2| = Join Selectivity(A.a2, B.b2) × |A.a2| × |B.b2| = 1/12 × 3 × 12 = 3*

그러나, 그림 2의 두 번째 조인 연산인 |$B3⋈$C1|의 경우, $B3의 원소 수는 그림 3에 정의되어있는 25가 아니라 앞선 조인 연산을 만족하는 b3 노드의 개수이다. 다시 말하면 Join Selectivity(B.b3, C.c1)를 계산하기 위해서 필요한 |B.b3|는 그림 3에 정의되어있는 25가 아니고 앞선 조인을 만족하는 b2노드와 연관된 b3노드의 개수이다. 예제의 경우, 첫 번째 조인 연산을 만족하는 b2 노드는 3개로 추정되었다. 그러므로 그 선택치는 3/12이다. 테이블 구조 기반의 관계형 데이타에 질의하기 위한 SQL 질의의 경우, 앞선 조인 연산에 의해서 나온 선택치를 기반으로 다음 조인을 위한 선택치를 추정하는 것은 매우 간단하다. 만약 위 예제가 SQL 질의의 조인일 경우, 첫 번째 조인에 의해서 얻은 3개의 |B'.b2|를 기반으로 다음 조인을 위한 피 연산자인 |B'.b3| 역시 일반적으로 3개임을 알 수 있다[19]. 그러나 XQuery 질의의 경우 XML 문서의 구조적인 특징 때문에 SQL 질의의 선택치 추정 방법을 그대로 사용할 수 없다. 즉, SQL의 경우, DB의 테이블의 속성은 다른 속성들과의 관계가 일대일로 매핑되지만 XML문서에서 노드들은 일대일, 일대다, 다대일, 그리고 다대다의 경우로 매핑될 수 있다. 그림 4는 DB의 테이블과 XML 문서의 구조적인 차이를 보인다.

본 논문에서는 XML의 이러한 구조적인 특성을 고려하여 앞선 조인의 선택치로부터 다음 조인의 선택치를 추정하기 위한 방법을 제안하고 이를 기반으로 조인의 비용을 예측한다.

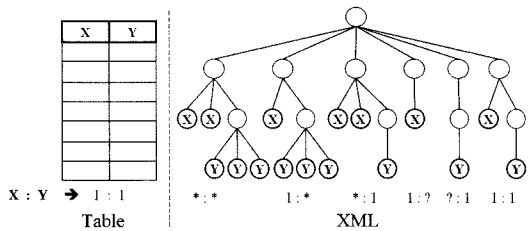

그림 4 Table과 XML 문서의 구조적인 차이

표 1 SQL과 XQuery 질의의 조인 선택치 추정 방법

Conditions	SQL	XQuery
하나의 피 연산자가 ID(key)인 경우	A.value = B.key ➔ 1/|B.key|	$A/text() = $B/@id ➔ 1/\|$B/@id|
두 개의 피 연산자가 ID인 경우	A.key = B.key ➔ 1/Max(|A.key|,|B.key|)	$A/@id = $B/@id ➔1/Max(\|$A/@id|, |$B/@id|)
두 개의 피 연산자 모두 ID가 아닌 경우	A.Value = B.Value ➔ 1/Max(|A.key|,|B.key|)	$A/text() = $B/text() ➔ 1/Max(\|$A|,|$B|)

4.2 조인 선택치의 추정

4.2.1 ESNJ 알고리즘

Estimating Selectivity for Next Join (ESNJ)는 이전 조인 노드와 연결되어 있는 다음 조인 노드의 선택치를 추정하는 알고리즘이다. ESNJ의 입력은 XML 뷰와 이전 조인 노드에 대한 선택치 그리고 찾고자 하는 노드 이름이고, 출력은 노드의 선택치이다.

그림 5는 ESNJ 알고리즘의 처리순서를 보인다.

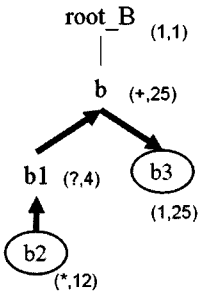

그림 5 ESNJ 알고리즘의 처리 과정

앞선 예제에서 우리는 $B2의 선택치가 3/12임을 추정했고, 이를 이용하여 $B3의 선택치를 추정하고자 한다. 그러나 b2 노드와 b3노드는 직접적인 연관관계가 존재하지 않는다. 그러므로 보다 정확한 선택치의 추정을 위하여 본 논문에서는 b2 노드로부터 b3노드까지의 최단 경로를 찾고 그 사이에 존재하는 노드들 사이의 관계를 모두 고려한다. 예를 들어 그림 5의 경우, b2로부터 b1 노드의 선택치를 추정하고 다시 b1노드로부터 b노드의 선택치를 추정한 후 마지막으로 b 노드로부터 b3노드의 선택치를 추정한다.

그림 6은 B 시스템에 저장된 XML 문서의 예를 보인다.

ESNJ알고리즘은 선택치를 적용하는 방향에 따라 두 가지 경우로 나뉜다. 첫 번째는 b2 노드로부터 b1노드의 선택치를 추정하는 방법과 같이 자식의 선택치로부터 부모의 선택치를 결정하는 경우이고, 두 번째는 b 노드의 선택치로부터 b3 노드의 선택치를 추정하는 방법과 같이 부모의 선택치로부터 자식의 선택치를 결정하는 경우이다. 표 2에 기술된 정의1과 정의2는 위의 두 가지 방법에 대해 선택치를 추정하는 방법이다.

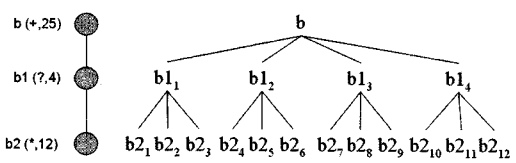

그림 6 B 시스템에 저장된 문서의 구조

표 2 ESNJ알고리즘

정의. 1. 부모 노드의 선택치를 이용하여 자식 노드의 선택치를 추정

P: 부모 노드, C: 자식 노드,

- Occurrence of P : C ➔ 1: (1 | ? | * | +) 이면 Selectivity(C) = Selectivity(P)

정의. 2. 자식 노드의 선택치를 이용하여 부모 노드의 선택치를 추정

- Occurrence of C : P➔ (1 | ?) : 1이면 Selectivity(P)= Selectivity(C)
- Occurrence of C : P➔ (+ | *) : 1 이면

$$Selectivity(P)=\sum_{i=MinP}^{MaxP} i\cdot P(i)/m$$

$$P(i)=\frac{\left({}_{i\cdot r}C_q-\sum_{j=MinP}^{i-1}E(j,i)\right)\times {}_mC_i}{{}_nC_q}$$

$$E(j,i)=\left({}_{j\cdot r}C_q-\sum_{l=MinP}^{j-1}E(l,j)\right)\times {}_iC_j$$

m : 부모 노드의 총 수
MinP : 선택 가능한 부모노드의 최소 수,
MaxP : 선택 가능한 부모노드의 최소 수,
n : 자식 노드의 총 수
r : 부모에 대한 평균 자식 수. n/m.
q : n × Selectivity(C)

정의 1의 경우, 트리 상에서 부모가 없는 자식은 존재할 수 없으므로 부모는 자식에 대하여 항상 '1'의 관계를 갖는다. 그러나 자식은 '1', '?', '+', '*'의 모든 경우가 가능하므로 네 가지의 경우의 수가 나온다. 본 논문에서는 특별한 조건이 없는 경우 부모에 대하여 자식들이 균등하게 분포되어 있다고 가정한다. 그러므로 부모 노드의 선택치를 이용하여 자식 노드의 선택치를 추정할 때 부모 노드의 선택치를 자식노드에게 그대로 적용할 수 있다. 만약 그림 5의 $b1_1$, $b1_2$, $b1_3$, $b1_4$ 노드들 가운데 $b1_1$과 $b1_2$노드가 선택되었다고 가정하면, 부모의 선택치는 2/4이므로 1/2이다. 그러므로 자식 노드인 b2노드의 선택치도 1/2로 추정할 수 있다.

정의 2의 경우, 자식 노드와 부모 노드의 발생이 각각 1:1 또는 ?:1인 경우, 자식 노드가 선택되었다면 반드시 부모 노드가 선택되어야 하므로 부모노드의 선택치는 자식 노드의 선택치와 동일하다. 그러나 노드와 부모 노드의 발생이 +:1 또는 *:1인 경우, 자식노드가 선택될 모든 확률을 계산하여 이를 기반으로 부모 노드의 선택치를 추정한다. 다시 말하면, 부모 노드가 선택될 수 있는 모든 경우의 수(i)에 그 확률(P(i))을 곱한 값의 총합을 부모노드의 개수로 나누어 선택치를 추정할 수 있다. 예를 들어, 그림 6의 경우, 앞선 조인에 의해서 12개의 b2노드들 중에서 3개의 노드가 선택되었을 때, 만약 선택된 노드가 $b2_1$, $b2_2$, $b2_3$의 3개의 노드라면, 부모

노드는 $b1_1$하나만 선택이 되므로 그 선택치는 1/4가 된다. 그러나 만약, 선택된 세 개의 노드가 $b2_1$, $b2_2$, $b2_4$ 노드라면, 부모 노드는 $b1_1$, $b1_2$ 노드가 선택되어 선택치는 1/2가 될 것이고, $b2_1$, $b2_4$, $b2_7$ 노드가 선택되었다면, 부모 노드는 $b1_1$, $b1_2$, $b1_3$ 노드가 선택될 것이고 선택치는 3/4이다. 이처럼 자식노드들이 선택되는 조합에 따라 선택되는 부모 노드의 수가 달라지고 그 선택치 또한 달라진다. 그러므로 본 논문에서는 자식 노드가 선택될 수 있는 모든 경우의 수에 대한 부모 노드가 선택될 수 있는 확률을 계산하여 부모노드의 선택치를 추정한다.

그림 6에 기술된 문서를 대상으로 b2 노드의 선택치(3/12)로부터 b1 노드의 선택치를 추정하는 과정은 다음과 같다. 먼저 3개의 자식 노드가 선택되었으므로 선택 될 수 있는 부모노드의 범위는 1~3개이고, 총 경우의 수는 $_{12}C_3$이다. 이 중에서 부모가 1개 선택되는 경우를 찾아보면, ('$b2_1$', '$b2_2$', '$b2_3$'), ('$b2_4$', '$b2_5$', '$b2_6$'), ('$b2_7$', '$b2_8$', '$b2_9$'), ('$b2_{10}$', '$b2_{11}$', '$b2_{12}$')의 4가지 경우가 된다. 또한 $b1_1$의 자식인 ('$b2_1$', '$b2_2$', '$b2_3$')중에서 3개의 노드를 선택할 수 있는 경우의 수는 $_3C_3$이고, 이 경우의 수는 다른 부모들인 $b1_2$, $b1_3$, $b1_4$에도 적용되므로, 전체 부모 수에서 하나의 부모를 선택하는 조합인 $_4C_1$을 곱해야 한다. 따라서 부모가 1개 선택될 경우의 수는 $_3C_3 \times {}_4C_1$로 구할 수 있고, P(1)은 $_3C_3 \times {}_4C_1 / {}_{12}C_3$이다. 다음으로 부모가 2개 선택되는 경우를 고려해보자. 만약 부모 노드 $b1_1$, $b1_2$가 선택되려면 $b2_1$~$b2_6$의 노드들 가운데 3개가 선택이 되어야 하며, 그 경우의 수는 $_6C_3$이다. 그러나 이 경우의 수에는 부모가 1개 선택되는 경우인, ('$b2_1$', '$b2_2$', '$b2_3$'), ('$b2_4$', '$b2_5$', '$b2_6$')도 포함되어 있기 때문에, 이 경우의 수를 제외시켜야 한다. 그러므로 부모 노드 $b1_1$, $b1_2$가 선택될 경우의 수는 $_6C_3 - {}_3C_3 \times {}_2C_1$이 된다. 이 경우의 수는 $b1_1$~$b1_4$의 4개의 부모 노드들 중에서 2개를 선택하는 조합만큼 적용되므로 그 경우의 수는 $(_6C_3 - {}_3C_3 \times {}_2C_1) \times {}_4C_2$가 되며, 결국 P(2)는 $(_6C_3 - {}_3C_3 \times {}_2C_1) \times {}_4C_2 / {}_{12}C_3$이 된다. 마지막으로 3개의 부모가 선택될 확률 역시 앞선 경우들과 유사하며 위의 예를 계산한 식은 다음과 같다.

$$P(1) = {}_3C_3 \times {}_4C_1 / {}_{12}C_3 = 4 / 220 = 1/55$$

$$P(2) = (_6C_3 - {}_3C_3 \times {}_2C_1) \times {}_4C_2 = (_6C_3 - E(1,2)) \times {}_4C_2 / {}_{12}C_3 = 108 / 220 = 27/55$$

$$P(3) = (_9C_3 - (_3C_3 \times {}_3C_1 + (_6C_3 - {}_3C_3 \times {}_2C_1) \times {}_3C_2)) \times {}_4C_3 / {}_{12}C_3 = (_9C_3 - (E(1,3) + E(2,3))) \times {}_4C_3 / {}_{12}C_3 = 108 / 220 = 27/55$$

결국 b2 노드의 선택치로부터 추정된 b1 노드가 선택될 노드의 개수는 약 2.5개이며 그 선택치는 다음과 같다.

$$selectivity(P) = \sum_{i=MinP}^{MaxP} i \cdot P(i)/4 = \left(1\times\frac{1}{55}+2\times\frac{27}{55}+3\times\frac{27}{55}\right)/4 \approx 2.5/4$$

5. 성능 평가

논문에서 제안하고 있는 XQuery 조인의 선택치 추정 방법의 평가를 위하여 네 개의 XML 문서를 사용하였다. 네 문서는 모두 XMark의 Auction 데이타를 분할하여 생성하였으며, 그림 7과 같은 뷰로 표현된다.

평가를 위한 예제 XQuery 질의는 표 3과 같다.

Q1은 3개의 XML 문서를 대상으로 2개의 조인을 포함하고 있는 XQuery 질의이며, Q2는 4개의 XML 문서를 대상으로 3개의 조인을 포함하고 있는 XQuery 질의이다. 평가의 방법은 본 논문에서 제안한 선택치 추정 알고리즘을 이용하여 예제 질의에 존재하는 조인 연산을 수행하기 위해서 필요한 노드의 비교 횟수를 추정한 결과와 실제 질의를 수행시켰을 경우 수행되는 노드의 비교 횟수를 계산한다.

본 논문에서는 추정해야 할 노드의 개수를 증가시키기 위해서 XMark의 임의 문서 생성 프로그램을 이용하여 XML 데이타의 크기를 1MB에서 5MB까지 1MB단위로 증가시키며 비교 횟수를 측정하였다.

그림 8은 본 논문에서 제안한 조인 선택치 추정 알고리즘을 기반으로 Q1과 Q2에 존재하는 조인 연산을 수

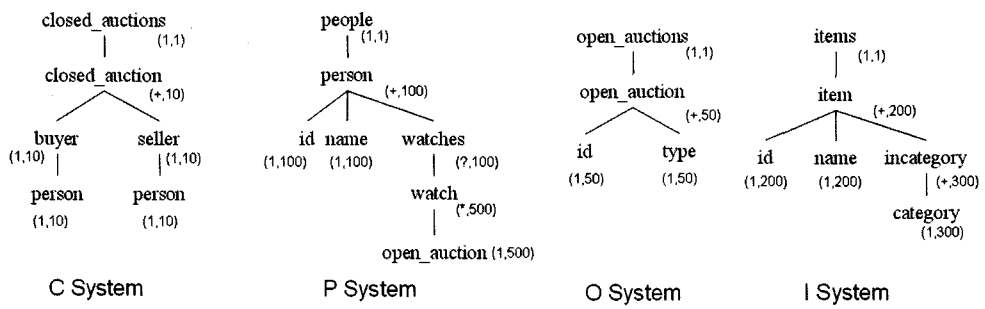

그림 7 성능 평가를 위한 XML 뷰들

표 3 성능평가를 위한 예제 질의

Q1	<results>{ for $closedAuction in doc("closed_auctions.xml")/closed_auctions/closed_auction for $closedAuctionSeller in $closedAuction/seller/@person for $people in doc("people.xml")/people/person for $peopleID in $people/@id for $peopleWatches in $people/watches/watch for $peopleWatch in $peopleWatches/@open_auction for $openAuction in doc("open_auctions.xml")/open_auctions/open_auction for $openAuctionID in $openAuction/@id where $closedAuctionSeller=$peopleID and $peopleWatch=$openAuctionID return <result>{ $people/name, $closedAuction/buyer, $openAuction/type }</result> }</results>

Q2	<results>{
	for $closedAuction in
	doc("closed_auctions.xml")/closed_auctions/closed_auction
	for $closedAuctionSeller in $closedAuction/seller/@person
	for $people in doc("people.xml")/people/person
	for $peopleID in $people/@id
	for $peopleWatches in $people/watches/watch
	for $peopleWatch in $peopleWatches/@open_auction
	for $openAuction in doc("open_auctions.xml")/open_auctions/open_auction
	for $openAuctionID in $openAuction/@id
	for $openAuctionItemref in $openAuction/itemref/@item
	for $items in doc("ItemsOfNamerica.xml")//item
	for $itemsID in $items/@id
	where $closedAuctionSeller=$peopleID and $peopleWatch=$openAuctionID and
	$openAuctionItemref = $itemsID
	return <result>{$people/name, $closedAuction/buyer, $openAuction/type, $items/name}</result>
	}</results>

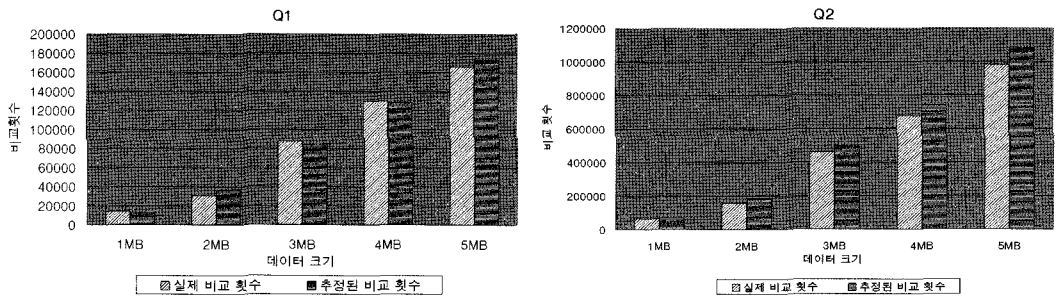

그림 8 조인 선택치 추정 알고리즘에 의해서 추정된 조인 연산의 노드 비교 횟수와 실제 조인 시 수행되는 노드 비교 횟수

행하기 위하여 필요한 노드의 비교 횟수를 추정한 결과와 실제로 두 질의를 수행했을 경우 수행되는 노드의 비교횟수를 보인다. 그림에서 볼 수 있는 것처럼 본 논문에서 제안한 알고리즘을 기반으로 조인 비용을 추정한 방법은 실제 조인의 수행 시 발생하는 비용과 유사하다는 것을 알 수 있다. 이러한 결과는 본 논문에서 제안하고 있는 XQuery 조인 선택치 추정 방법이 실제 조인 연산을 수행했을 경우 발생하는 노드의 선택치와 크게 다르지 않는다는 것을 입증한다.

6. 결 론

본 논문에서는 XQuery의 조인 선택치를 추정하기 위한 방법을 제안하였다. 이를 위하여 SQL 질의의 조인 선택치 추정 방법과 XQuery 질의의 방법 사이에 차이점을 분석하였고, XML 문서의 구조적인 특성을 고려한 XQuery 조인 선택치 추정 알고리즘을 제안하였다. 또한 본 논문에서는 추정된 선택치를 기반으로 조인 비용을 예측하고 이를 실제 수행된 조인 연산의 비용과 비교 평가하여 우리 알고리즘의 효율성을 입증하였다.

본 논문에서는 선택치 추정 알고리즘을 적용하기 위하여 부모 노드가 균등한 분포로 자식 노드를 포함하고 있다는 가정을 했고, 성능평가를 위한 견본 데이타들 역시 문서 생성기로부터 자동 생성된 문서이므로 이 가정을 크게 벗어나지 않는다. 그러나 최악의 경우 자식 노드가 특정 부모 노드에 편중되어 존재한다면, 우리의 알고리즘에 의해 추정된 선택치는 많은 오차가 존재할 것이다. 그러므로 향후 이를 고려한 방법이 연구되어야 할 것으로 사료된다.

참 고 문 헌

[1] W3C, Extensible Markup Language (XML) 1.1 (Second Edition) W3C Recommendation 16 August 2006, (http://www.w3.org/TR/2006/REC-x ml11-20060816).

[2] W3C, XML Schema Part 0: Primer Second Edition W3C Recommendation 28 October 2004, (http://www.w3.org/TR/2004/REC-xmlschema -0-20041028).

[3] W3C, XQuery 1.0: An XML Query Language, W3C Recommendation 23 January 2007, (http://www.w3.org/TR/2007/REC-xquery-20070123).

[4] S. Groppe, & S. Bottcher, "Schema-based Query Optimization for XQuery Queries," Proc. ADBIS 2005, Tallinn, Estonia, September, 2005.

[5] I. Manolescu, D. Florescu & D Kossmann, "Answering XML Queries over Heterogeneous Data Sources," Proc. 27th International Conference on Very Large Data Bases, Roma, Italy, pp. 241-250, September 11-14, 2001.

[6] L. Wang, M. Mulchandani & E. A. Rundensteiner, "Updating XQuery Views Published over Relational Data : A Roundtrip Case Study," Xsym 2003, pp. 223-237, Berlin, Germany, September 8, 2003.

[7] M. Steinbrunn, G. Moerkotte, & A. Kemper, "Optimizing Join Orders," Technical Report MIP-9307, Faculty of Mathematic, Univ. of Passau, Passau, Germany, 1993.

[8] C. Shahabi, L. Khan, & D. McLeod, "A Probe-Based Technique to Optimize Join Queries in Distributed Internet Databases," Knowledge and Information Systems, pp. 373-385, 2001.

[9] X. Lin & M. E. Orlowska, "An Efficient Processing of a Chain Join with the Minimum Communication Cost in Distributed Database Systems," Distributed and Parallel Databases 3, pp. 69-83, 1995.

[10] M. J. Yu & P. C.-Y. Sheu, "Adaptive Join Algorithms in Dynamic Distributed Databases," Distrib. Parallel Databases, Vol.5, No.1, pp. 5-30, January, 1997.

[11] L. Liu, C. Pu & K. Richine, "Distributed Query Scheduling Service: An Architecture and Its Implementation," IJCIS, Vol.7, No.2-3, pp. 123-166, 1998.

[12] I. Eldosouky, H. Arafat, & A. A. Eldin, "New Heuristic Approaches for Improving Dis-tributed Query Processing based on The Enhancement of Semi-Join Strategies," Proc. the International Conference on Statistics, Computer Science, and Operational Research, Egypt Dec, 2001.

[13] A. Halverson, J. Burger, L. Galanis, A. Kini, R. Krishnamurthy, A. N. Rao, F. Tian, S. D. Viglas, Y. Wang, J. F. Naughton, & D. J. DeWitt, "Mixed Mode XML Query Processing," VLDB 2003.

[14] N. May, S. Helmer, C. C. Kanne, G. Moerkotte, "XQuery Processing in Natix with an Emphasis on Join Ordering," Proc. 1st Int. Workshop on XIME-P 2004, Paris, France, June 17-18, 2004.

[15] W3C, XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Recommendation 23 January 2007, (http://www.w3.org/TR/2007/REC-xpath-functions-20070123/).

[16] A. Aboulnaga, A. R. Alameldeen, & J. F. Naughton, "Estimating the selectivity of XML path expressions for internet scale applications," Proc. 27th VLDB, Roma, Italy, 2001.

[17] J. Freire, J. Haritsa, M. Ramanath, P. Roy, & J. Simeon, "Statix: Making XML count," Proc. of ACM SIGMOD Intl. Conf. on Management of Data, pp. 181-191, 2002.

[18] M. Steinbrunn, G. Moerkotte & A. Kemper, "Heuristic and Randomized Optimization for the Join Ordering Problem," VLDB Journal, 6(3), pp. 191-208, 1997.

[19] A. N. Swami & K. B. Schiefer, "On the Estimation of Join Result Sizes," Proc. EDBT 1994, pp. 287-300, Cambridge, March 28-31, 1994.

[20] S. B. Navathe & R. Elmasri, "Fundamentals of Database Systems," fourth edition, Addi-son Wesley, 2003.

박 종 현

2002년 충남대학교 컴퓨터과학과 석사 2002년~2007년 현재 충남대학교 컴퓨터과학과 박사. 관심분야는 XML, XQuery, 웹 정보 시스템, XML 데이타베이스, 분산 웹 데이타베이스, Semantic Web.

강 지 훈

1981년 한국과학기술원 전산학과 석사 1996년 한국과학기술원 전산학과 박사 1996년~1998년 미국 버지니아대학교 컴퓨터과학과 방문교수. 2000년~2002년 충남대학교 정보 통신 원장. 1985년~현재 충남대학교 정보통신공학부 교수. 관심분야는 XML, 디지털 도서관, Semantic Web, 데이타베이스 시스템, 웹 정보 시스템