韓國컴퓨터情報學會 論文誌
第15卷 第11號, 2010. 11.

2010-15-11-1-2

지역 질의 생성기반 전역 XQuery 질의 처리 기법

박 종 현*, 박 원 익**, 김 영 국***, 강 지 훈***

A Global XQuery Query Processing based on Local XQuery Query Generation

Jong-Hyun Park*, Won-Ik Park**, Young-Kuk Kim***, Ji-Hoon Kang***

요 약

XML 뷰는 분산 환경에서 이종 데이터들을 XML 데이터처럼 바라보며 XML로 통합하기 위한 방법으로 제안되었으며, 전역 XML 뷰는 분산되어 있는 다양한 형태의 이종 데이터들을 단일의 XML 데이터처럼 바라보며 질의할 수 있도록 한다. 이때 사용자가 사용하는 표준 질의어는 XQuery이며, 전역 XML 뷰를 대상으로 작성된 질의가 바로 전역 XQuery 질의이다. 그러므로 이를 효과적으로 처리하기 위한 방법은 분산 환경에서 이종 데이터들 사이의 통합 및 검색을 위한 연구의 주제이다. 기존 SQL 질의 처리에서 알 수 있는 것처럼, 분산 질의 처리를 위한 방법들 가운데 가장 범용적으로 사용되는 방법 중 하나는 전역 질의를 지역 질의로 분할하여 분할된 지역 질의들을 처리하고, 그 결과를 취하여 전역 질의의 결과로 재구성하는 것이다. 그러나 XQuery는 FOR 절과 같은 SQL 질의어에서는 찾아볼 수 없는 복잡한 구조적 특성을 갖는다. 그러므로 전역 XQuery 질의의 처리를 위해 지역 질의를 생성하기 위해서는 고려해야 할 사항들이 존재한다. 본 논문에서는 전역 SQL 질의 처리를 위한 지역 질의 생성 기법을 전역 XQuery 질의의 처리를 위해 적용할 때 발생하는 문제점을 정의하고 이를 해결하기 위한 방법을 제안한다. 또한, 제안한 방법을 기반으로 하는 전역 XQuery 질의 처리기를 구현하여 그 효율성을 보인다.

Abstract

XML view is proposed to integrate between XML data and heterogeneous data over distributed environment and global XML view is used to search distributed heterogeneous data. At this time, standard query language for user is XQuery and the method for processing global XQuery queries over distributed environment is one of the new research topics. One of the basic and simple methods to process distributed SQL queries is that generates local queries for processing a global query and constructs the result of the global query from the results of the local queries. However, the syntax of XQuery differs from SQL because the XQuery contains some special expressions like FOR clauses for querying to semi-structured data, of course, FOR clauses are not used in SQL.

• 제1저자 : 박종현 교신저자 : 강지훈
• 투고일 : 2010. 07. 15, 심사일 : 2010. 08. 13, 게재확정일 : 2010. 08. 20.
* 큐슈대학교 정보기반연구센터 방문연구원 ** 충남대학교 컴퓨터공학과 박사과정 *** 충남대학교 전기정보통신공학부 교수

Therefore, there are some problems to adopt the method for processing global SQL queries for generating local XQuery queries. This paper defines some problems when generates local XQuery queries for processing global XQuery queries and proposes a method for generating local XQuery queries considered these problems. Also we implement and evaluate a Global XQuery Processor which uses our method.

▸ Keyword : Query 분할(XQuery Decomposition), 전역 XQuery 처리(Global XQuery processing), 분산된 이종 데이터 검색(Searching of distributed heterogeneous data)

Ⅰ. 서 론

현재 인터넷 환경에는 순수한 XML 데이터도 존재하지만 기존의 관계형 데이터베이스 또는 URI로 표현되는 웹 정보 소스들과 같은 XML 이외의 데이터들이 존재하는 것이 현실이다. 그러므로 많은 응용에서는 기존의 데이터들을 XML 데이터와 함께 사용하기 위하여 다양한 방법을 시도하고 있으며, 그 중 대표적인 방법이 XML DTD, Schema, XQuery 등을 이용한 XML 뷰를 이용하여 이종의 지역 데이터들을 XML 데이터와 함께 사용하기 위한 방법이다[1, 2, 3]. 이러한 지역 XML 뷰들은 다시 전역 XML 뷰에 의해서 통합된다. 전역 XML 뷰는 지역 데이터를 표현하는 지역 XML 뷰들을 기반으로 작성되며, 사용자는 전역 XML 뷰를 이용하여 분산되어 있는 다양한 형태의 이종 데이터들을 XML 처럼 바라보며 질의할 수 있다. 이때 사용자가 사용할 수 있는 표준 질의어가 바로 XQuery 이다[4].

표준 질의어로서 XQuery는 기존 질의어들의 많은 장점을 취합한 언어라는 특성을 갖는 반면 매우 복잡하게 구성되어 있으므로, 이를 효율적으로 처리하는 것은 쉽지 않다. 특별히 분산 환경에서 전역 XQuery 질의를 처리하기 위한 방법은 현재 연구 중에 있으며, 기존에 존재하는 분산 SQL과 같은 질의어를 처리하기 위한 방법으로부터 그 해결 방법을 찾기 위한 노력들이 존재한다[5]. 전통적인 데이터베이스 질의인 SQL 질의의 경우, 전역 SQL 질의의 처리를 위해 사용되고 있는 대표적이고 기본적인 방법들 가운데 하나는 전역 질의를 분할하여 지역 시스템을 위한 지역 질의들을 생성하고, 지역 질의들의 결과로부터 전역 질의의 결과를 취하는 방법이다. 이러한 방법은 지역 시스템의 어떠한 정보도 참조할 필요 없이 전역 질의를 처리할 수 있는 가장 간단한 방법이며, 향후 동적인 질의 처리 방법으로의 확장이 용이한 방법이므로 전역 XQuery 질의의 처리를 위해서도 반드시 연구되어야 할 방법들 가운데 하나이다[5, 6, 7]. 그러나 XQuery 질의는 구조적 문서인 XML 문서의 검색을 그 목적으로 하는 질의어이므로 그 특성이 SQL과는 적지 않은 차이가 존재할 뿐만 아니라 질의어 자체의 복잡도 또한 매우 높은 언어이다[7, 8]. 그러므로 기존에 제안된 SQL 질의어의 처리 방법을 전역 XQuery 질의의 처리를 위하여 적용하기 위해서는 언어의 특성과 함께 고려해야 할 사항들이 존재한다. 예를 들어, XQuery의 FOR절에 의해 발생되는 중첩 구조들은 전역 XQuery 질의의 처리를 위하여 반드시 고려해야 할 대상이지만 SQL 질의어에서는 찾아볼 수 없는 표현들이다. 그러므로 전역 XQuery 질의 처리를 위하여 XQuery의 특성을 고려하여 전역 질의 분할하고 지역 질의를 생성하는 방법은 분산 환경에서 이종 데이터들을 통합하고 검색하기 위해서 반드시 해결되어야 할 문제이며 매우 중요한 연구이다.

본 논문에서는 분산 환경에 존재하는 이종 데이터들을 XQuery를 이용하여 검색하고 통합하기 위한 방법으로, 전역 XQuery 질의로부터 지역 XQuery 질의를 생성해내는 방법과 지역 XQuery 질의의 수행 결과를 기반으로 전역 XQuery 질의의 결과를 생성하기 위한 방법을 제안한다. 이를 위하여 전역 SQL 질의의 처리를 위해 지역 질의를 생성하는 방법을 전역 XQuery를 위해 적용했을 때 발생하는 문제점을 정의한다. 또한 이러한 문제점을 해결하기위한 방법을 제안한다. 제안된 방법은 전역 XQuery 질의 처리기로 구현되고 성능 평가를 통하여 그 유효성을 보인다.

본 논문의 나머지 구성은 다음과 같다. 2절에는 전역 질의를 처리하기 위한 기존 관련 연구들을 기술하고 있으며, 3절에서는 전역 XQuery 질의와 전역 SQL 질의의 처리를 위한 지역 질의 생성 방법들 사이의 차이를 기술한다. 또한 이를 해결하기 위해서 본 논문에서 제안하고 있는 지역 XQuery 질의 생성 방법에 대해서 구체적으로 기술한다. 4절에서는 논문에서 제안하고 있는 알고리즘을 기반으로 하는 전역 XQuery 질의 처리기를 기술하며 5절에서 그 성능을 평가한다. 마지막으로 6절에서 결론과 함께 향후 연구 방향에 대해서 기술한다.

II. 관련 연구

현재 많은 응용들은 데이터베이스 시스템, 파일 시스템 그리고 웹 서비스와 같은 다양한 종류의 정보 시스템들을 통하여 정보를 접근하고 통합하고, 처리하기를 요청한다. 그러므로 이러한 통합 시스템을 가능하게 하기 위한 연구들이 진행되고 있으며, 이들 각각은 특정 질의어를 사용하여 정보를 통합하고 검색한다. 분산 환경에서 관계형 데이터베이스를 통합하는 연구들은 이미 수행된 바 있으며[9, 10], 또한 전역 관계형 데이터베이스 시스템이 반구조적 데이터(Semi-Structured Data)를 지원하도록 기능을 확장하는 연구와[11, 12] 단일 시스템에서의 XML 데이터에 대한 XQuery 질의를 효율적으로 처리하기 위한 연구도 존재한다[13, 14]. 분산 환경에 전역 질의를 처리하기 위한 방법은 다양하다. 그러나 그 방법들 가운데 가장 기본적인 방법의 하나는 지역 시스템을 위한 지역 질의들을 생성하여 처리하는 것이며, SQL 그리고 OQL 같은 질의어들로부터 지역 질의를 생성하기위한 필요성과 해결책은 이미 제안된 바 있다[5, 6, 7, 15]. 그러나 XQuery는 그 구조적인 특성으로 인하여 이미 제안된 다른 종류의 질의어들을 그대로 적용할 수는 없다.

[16, 17]에서는 다수의 지역 시스템을 대상으로 작성된 전역 XPath 질의를 지역 시스템에서 처리 가능한 지역 XPath 질의들로 분할하기 위한 방법을 제안한다. XPath가 XQuery의 부분집합(Subset)이므로 전역 XPath 질의의 분할이 전역 XQuery 질의와 유사할 수 있다. 그러나 XQuery와 XPath 사이에는 표현의 범위나 구조적인 복잡성 면에서 매우 많은 차이가 존재한다. 그러므로 XPath 질의의 처리 방법을 XQuery 질의를 위하여 참조할 수는 있지만 그대로 사용하기는 어렵다.

[18]은 분산 XQuery 질의의 처리를 위하여 지역 질의를 생성하고 있지만, 지역 XQuery 질의의 생성을 위하여 추가적인 정보를 참조한다. 그러나 본 논문에서 제안하는 지역 XQuery 질의 생성 방법은 XQuery 표준 질의 자체를 다루며, 추가적인 정보의 참조 없이 순수하게 전역 XQuery 질의만을 이용하여 지역 질의를 생성한다.

[19, 20] 역시 전역 XQuery 질의의 처리를 위하여 지역 XQuery 질의들을 생성한다, 그러나 이들은 전역 XQuery 질의의 처리를 위하여 XQuery 질의 자체를 확장하여 사용한다. 그러므로 이들이 제안한 XQuery 표현을 처리할 수 없는 응용들이 존재한다면 전역 XQuery 질의의 처리에 문제를 일으킬 수밖에 없다. 그러나 본 논문에서 제안하는 방법은 순수한 XQuery 질의 자체만을 이용하여 지역 XQuery 질의를 생성한다.

III. 지역 XQuery 질의의 생성

1. 문제 정의

전역 XQuery 질의는 그 대상이 다수의 지역 시스템들이다. 본 논문에서는 이러한 전역 XQuery 질의를 처리하기 위하여 어떻게 다수의 지역 시스템을 기반으로 작성된 전역 XQuery 질의로부터 지역 XQuery 질의들을 생성하는가에 초점을 둔다. 관계형 데이터베이스 질의어인 전역 SQL 질의의 경우, 이러한 접근 방법은 매우 보편적인 방법이다. <그림 1>은 전역 SQL 질의의 처리를 위하여 지역 SQL 질의를 생성하는 간단한 예이다.

<그림 1>의 상단에 기술된 테이블 A와 테이블 X는 각각 서로 다른 지역 시스템 A와 X에 존재하는 관계형 테이블이고, 그림의 하단에 기술된 전역 SQL 질의는 두 테이블을 대상으로 작성되었다. 이때 전역 SQL 질의의 처리를 위하여 <그림 1>의 하단에 기술된 것과 같은 지역 SQL 질의들을 생성할 수 있으며, 지역 질의들의 결과를 취하여 전역 SQL 질의의 결과를 생성할 수 있다. SQL 질의의 경우, 지역 질의를 생성하기 위한 기본 접근 방법은 전역 SQL 질의에 선언된 표현들 가운데 해당 지역을 대상으로 하는 표현들만을 취하여 지역 질의를 생성한다.

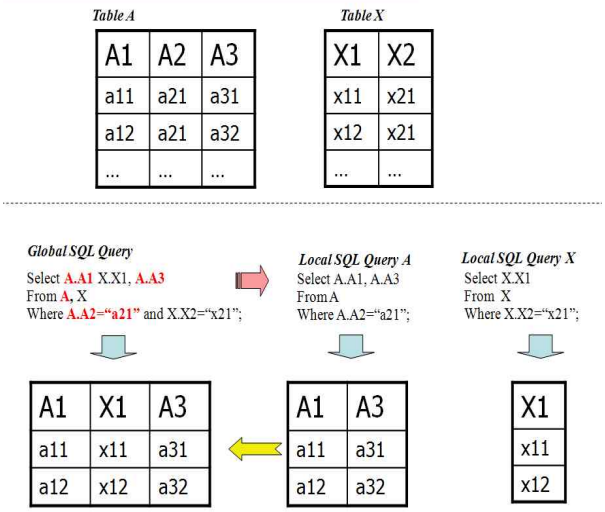

그림 1. 전역 SQL 질의 처리를 위한 지역 SQL 질의의 생성
Fig. 1. Generation of local SQL queries for processing A global SQL query

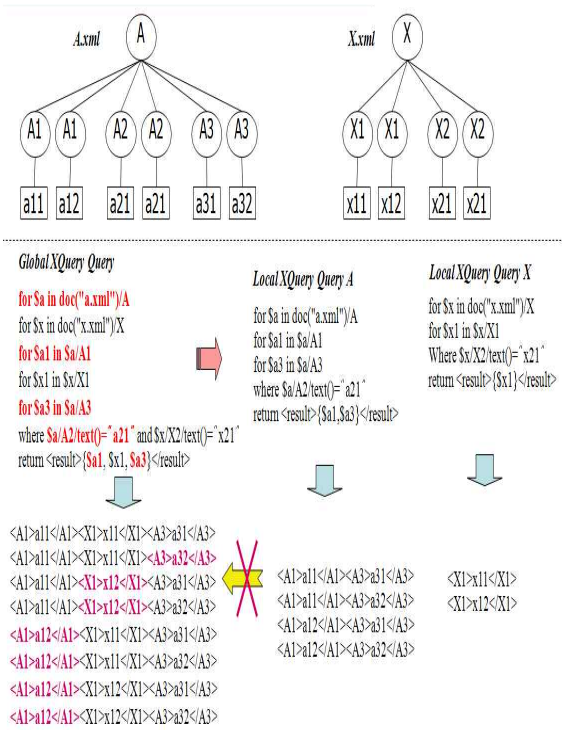

그림 2. 전역 XQuery 질의 처리를 위한 지역 XQuery 질의의 생성 시 고려사항

Fig. 2. A consideration for generating local XQuery queries for processing A global XQuery query

<그림 1>의 경우, 전역 질의에 기술된 표현들 가운데 지역 시스템 A를 대상으로 기술된 표현들(select절의 'A.A3', from 절의 'A', where절의 'A.A2="a21"')만을 취하여 지역 시스템 A를 위한 질의를 생성한다. 이렇게 생성된 지역 질의들은 각 지역 시스템에 보내져 처리된 후, 그 결과를 기반으로 전역 질의를 처리하여 최종 결과를 얻는 것이 일반적인 전역 SQL 질의 처리 방법들 가운데 하나이다. 그러나 XQuery 질의의 경우, 그 질의의 대상이 반 구조적(Semi-Structured) 문서인 XML 데이터이므로 XML 문서에 존재하는 동일 노드들의 반복을 표현하기 위하여 많은 형태의 XQuery 질의에서 FOR절을 사용한다. 그러나 이러한 FOR절은 XQuery 질의의 분할을 어렵게 만든다[7].

<그림 2>의 상단에 기술된 두 트리 구조는 각각 서로 다른 지역 시스템에 존재하는 XML 문서의 구조를 보이며, 그림의 중앙에 기술된 전역 XQuery 질의는 두 문서를 대상으로 작성되었다. 그림에 두 지역 XQuery 질의들은 지역 SQL 질의의 생성을 위한 알고리즘을 적용하여 생성한 질의들이다. 즉, 지역 XQuery 질의 A의 경우 전역 XQuery 질의에 기술된 표현들 가운데 지역 시스템 A를 대상으로 한 표현들만을 취하여 생성한 질의이다. 이러한 접근 방법이 XQuery 질의의 처리를 위해 자연스러운 접근 방법 같아 보이지만 여기에는 해결해야 할 문제가 존재한다. 그것은 바로 지역 질의의 결과가 지역 시스템에 저장된 원본 XML 문서의 일부가 아니고 전혀 새로운 구조를 따르는 XML 문서라는 것이다. <그림 2>의 하단에 기술된 지역 XQuery 질의 A의 결과는 원본 문서(A.xml)의 일부분이 아니고, 원본 문서의 특정 부분으로 재구성된 새로운 구조를 따르는 XML 문서이다. 그러므로 SQL을 위한 처리 방법을 XQuery를 위해 적용할 경우, <그림 2>의 하단에 기술된 전역 XQuery 질의의 결과를 생성할 수 없다. SQL의 경우, 모든 지역 질의의 결과는 원본 테이블의 전체 혹은 일부분이지만, XQuery의 경우, 질의에 존재하는 FOR절이나 생성자(Constructor)에 의해 전혀 새로운 XML 문서가 지역 질의의 결과로 반환된다. 그러므로 지역 XQuery 질의의 생성을 위해서는 SQL 질의를 위한 방법을 그대로 적용할 수 없으며 새로운 방법이 필요하다.

2. 지역 XQuery 질의 생성 알고리즘

지역 SQL 질의 생성 방법의 기본 접근 방법은 각 지역 시스템으로부터 전역 SQL 질의의 처리를 위한 최소의 부분을 취하는 것이다. 그러므로 XQuery 질의 역시 이러한 방법으로 접근한다. 즉, 지역 시스템으로부터 전역 XQuery 질의의 처리를 위해서 필요한 최소한의 XML 문서 일부분을 구조적인 변경 없이 얻기 위한 지역 XQuery 질의들을 생성하는 것이다. 이를 위하여 논문에서는 먼저 전역 XQuery 질의의 처리를 위해서 필요한 XML 데이터가 지역 시스템에 어떻게 저장되어 있는지 분석하고, 이를 추출하기위한 지역 XQuery 질의들을 생성한다.

전역 XQuery 질의에 기술된 XPath 표현들은 각 지역 시스템에 저장된 데이터들 가운데 전역 질의의 처리를 위해 필요한 데이터들을 지정(addressing)하고 있다. 그러므로 우리는 전역 질의에 기술된 XPath 표현들을 이용하여 지역 시스템으로부터 추출해야 할 XML 데이터의 구조를 알 수 있고, 데이터를 추출하기 위한 지역 XQuery 질의를 생성 수 있다. 즉, 전역 XQuery 질의로부터 지역 시스템에 저장된 XML 문서의 전체 구조를 알 수는 없지만, 전역 질의의 처리를 위해서 필요한 최소한의 노드 구조는 알 수 있다. 그러므로 이러한 노드들을 구조적 변형 없이 추출하기 위한 XQuery 질의가 바로 전역 XQuery 질의 처리를 위한 지역 XQuery 질의이다.

표 1. “item.xml”, “seller.xml” 문서와 이를 대상으로 작성된 전역 XQuery 질의
Table 1. “item.xml” and “seller.xml” documents and an example XQuery query

Global XQuery query (GQ)
for $item in doc("item.xml")/items/item *for $seller in doc("seller.xml")/sellers/seller* for $item_URI in $item//URI for $item_feature in $item/feature *for $seller_address in $seller/address* for $item_des in $item_feature/description for $item_color in $item_feature/color where $item_color="red" and $item/owner= *$seller/name* return <result item_id="{$item/@id}"> {$item_URI,*$seller_address*, $item_des} </result>
item.xml
<items><item id="item0"><feature> <description>A is~~~</description> <color>red</color> </feature> <URI>URI1</URI><URI>URI2</URI> <owner>Park</owner> </item> <item id="item1"><feature> <description>B is~~~</description> <color>red</color></feature> <URI>URI3</URI><owner>Park</owner> </item> <item id="item2"><feature> <description>C is~~~</description> <color>Blue</color></feature> <URI>URI4</URI><owner>Lee</owner> </item></items>
seller.xml
<sellers> <seller id="person0"> <name>Park</name> <address>Daejeon</address> <address>Seoul</address> </seller> <seller id="person1"> <address>Incheon</address> <name>Kim</name> </seller> </sellers>

<표 1>은 견본 전역 XQuery 질의(GQ)와 질의의 대상이 되는 두 지역 XML 문서를 보인다. <그림 3>은 <표 1>의 전역 질의 GQ의 처리를 위해서 필요한 정보들을 표현하는 트리로, 왼쪽에 기술된 트리는 지역 시스템에 저장된 "item.xml" 문서로부터 GQ 처리를 위해서 필요한 노드들의 구조를 나타내고 있는 트리고, 오른쪽에 기술된 트리는 또 다른 지역 시스템에 저장된 “seller.xml” 문서로부터 GQ 처리를 위해 필요한 노드의 구성을 보이는 트리다. 트리에 각 노드들은 GQ의 XPath 표현으로부터 추출된다. 예를 들어, 우리는 GQ의 첫 번째 선언된 FOR절의 XPath 표현인 ”doc("item.xml")/items/item “로부터 ”item.xml“문서의 최상위 노드로 ”items" 노드가 존재하며 그 하위 자식 노드로 “item"노드가 존재함을 알 수 있다. 트리의 노드들 가운데 이중 원으로 기술된“URI", "description" 그리고” address" 노드들은 GQ의 RETURN절에 기술된 노드들을 나타내고 점선으로 기술된 노드들은 WHERE절에 조건으로 사용된 노드들을 나타낸다. 노드들을 연결하는 연결선들 가운데 실선은 부모-자식 관계를 나타내고 있으며, 이중선으로 기술된 연결선은 조상-자손의 관계를 표현한다. 그러므로 결국 GQ의 처리를 위해 각 지역 시스템들로부터 <그림 3>에 기술된 트리와 같은 구조의 문서들을 취해야하며, 이를 지역 XQuery 질의를 생성해야한다.

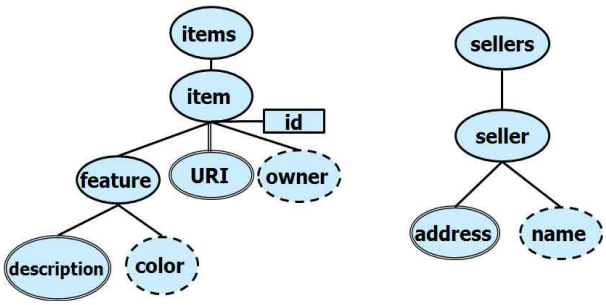

그림 3. <표 1>에 기술된 GQ의 처리를 위하여 두 지역 시스템에 저장된 정보를 표현하는 XML 트리들
Fig. 3. XML Trees which describe the information stored in local systems for processing the GQ in <Table 1>

지역 시스템으로부터 취해야할 XML 문서의 구조를 추출하면 이를 기반으로 지역 XQuery 질의를 생성하는 것은 그리 어렵지 않다. <표 2>는 지역 XQuery 질의들을 생성하기 위한 기본 알고리즘이다. 알고리즘의 기본 접근 방법은 지역 시스템으로부터 취해야하는 문서의 모든 노드들을 위에서부터 아래로 각각 For절로 지정하여 경로를 검증하면서 문서의 일부를 취하는 방법이다.

표 2. 지역 Xquery 질의 생성 알고리즘
Table 2. An algorithm for generating local XQuery queries

▪PATHexp : XPath 표현들의 집합
▪FORClauses : FOR절의 집합
▪LocalXQueryQueries : 지역 XQuery 질의의 집합

TempLXQ = seperateGlobalXQuery(GXQuery);
//전역 XQuery 질의에 표현들 가운데 동일 지역 시스템을 대상으로 하는 표현들만 취합하여 임시 지역 XQuery 질의들을 생성

```
While(TempLXQ!= null){
tlxq = TempLXQ.getLocalQuery();
//임시 생성된 지역 XQuery 질의를 하나씩 처리
PATHexp=getPATHexp(tlxq);
//지역 XQuery 질의에서 모든 XPath 표현을 취함
LocalXMLTree = makeLocalXMLTree(Pathexp);
//지역 시스템으로부터 취해야할 지역 XML 문서 트리를 생성
for(each node in LocalXMLTree){
//트리를 preorder 순회하면서 각 node마다 내포질의를 생성
makeFORclause(node);
//각 node를 지정하기위한 For절을 생성
innerQuery = Wrapper(FORclauses);
//각 For절을 내포하는 내포 XQuery 질의 생성
LocalXQuery.append(innerQuery);
//지역 질의에 내포 XQuery 질의 추가
}
createWHERERETURN(LocalXQuery);
//지역질의의 Where절과 Return절 생성
LocalXQueryQueries.add(LocalXQuery);
TempLXQ.remove(tlxq);
}
```

알고리즘의 첫 번째 과정은 전역 XQuery 질의의 표현들 중 그 대상이 동일한 지역 시스템인 표현들만을 취하여 임시 지역 XQuery 질의를 생성하는 것이다. 두 번째 단계는 임시 지역 XQuery 질의에 기술된 XPath 표현들로부터 <그림 3>의 트리와 같은 지역 시스템으로부터 취해야할 XML 문서 트리를 생성하는 것이다. 알고리즘은 이렇게 생성된 트리를 전위 순회(Preorder)하면서 각 노드를 지정(Addressing)하기 위한 내포 XQuery 질의를 생성하면서 지역 XQuery 질의를 생성해 나간다. <표 3>은 <표1>의 GQ를 대상으로 본 논문에서 제안한 알고리즘을 적용하여 생성된 지역 XQuery 질의들을 보인다. 지역 질의의 내포 질의들은 각 노드를 지정한다. 그러므로 결국 내포 질의의 개수는 지역 시스템으로부터 취해야할 트리의 노드 개수와 동일하다.

표 3. <표 2>의 지역 질의 생성 알고리즘에 의해서 생성된 두 지역 XQuery 질의

Table 3. Two Local XQuery queries generated by the algorithm in <Table 2>

Local XQuery Query for 'item.xml'

```
for $foritems in doc("item.xml")/items
let $Wrapper0 :=
for $foritem in $foritems/item
let $Wrapper1 :=
for $forfeature in $foritem/feature
let $Wrapper2 :=
for $fordescription in $forfeature/description
return $fordescription
let $Wrapper3 :=
for $forcolor in $forfeature/color
where $forcolor = "red"
return $forcolor
where exists($Wrapper2) and exists($Wrapper3)
return <feature>{$Wrapper2, $Wrapper3}</feature>
let $Wrapper4 :=
for $forURI in $foritem//URI
return $forURI
let $Wrapper5 :=
for $forowner in $foritem/owner
return $forowner
where exists($Wrapper1) and exists($Wrapper4) and
exists($Wrapper5)
return <item id="{$foritem/@id}">
{$Wrapper1, $Wrapper4, $Wrapper5}</item>
return <items>{$Wrapper0}</items>
```

Local XQuery Query for 'seller.xml'

```
for $forsellers in doc("seller.xml")/sellers
let $Wrapper0 :=
for $forseller in $forsellers/seller
let $Wrapper1 :=
for $seller_address in $forseller/address
return $seller_address
let $Wrapper2 :=
for $innerVariable1 in $forseller/name
return $innerVariable1
where exists($Wrapper1) and exists($Wrapper2)
return <seller>{$Wrapper1, $Wrapper2}</seller>
where exists($Wrapper0)
return
<sellers>{$Wrapper0}</sellers>
```

본 논문에서는 전역 XQuery 질의의 조건을 처리하기 위해서 두 가지 형태의 조건으로 분류하여 처리한다. 하나는 지역 시스템 내부에서 단독으로 처리가 가능한 조건이고, 다른 하나는 다른 지역 시스템과의 조인이 존재하는 경우이다. <표 1>에 기술된 GQ의 Where절에는 두 개의 조건이 기술되어있다. 첫 번째는 '$item_color="red"'이고 두 번째는 '$item/owner=$seller/name'이다. 첫 번째 조건은 단일 시스템에서 처리할 수 있는 조건이고, 두 번째 조건은 서로 다른 두 시스템 사이에 조인이 필요한 조건이다. 이 경우 첫 번째 조건과 같이 단일 시스템에서 처리 가능한 조건은 지역 질의에서 처리한 후 그 결과를 취하고, 다른 시스템과의 조인이 필요한 경우에는 지역 시스템으로부터 조건 처리를 위해 필요한 모든 정보를 취한다. 그러므로 첫 번째 조건은 <표 3>의 상위에 기술된 지역 XQuery 질의에서 Where절의 조건으로 처리되며, 두 번째 조건은 조건으로 처리되지 못하고 Return절에 기술되어 되어 각 지역 시스템들로부터 그 값을 취한다.

표 4. 주 XQuery 질의
Table 4. Main XQuery query

```
for $item in doc("item.xml")/items/item
for $seller in doc("seller.xml")/sellers/seller
for $item_URI in $item//URI
for $item_feature in $item/feature
for $seller_address in $seller/address
for $item_des in $item_feature/description
for $item_color in $item_feature/color
where $item/owner= $seller/name
return <result item_id="{$item/@id}">
{$item_URI, $seller_address, $item_des}</result>
```

지역 시스템으로부터 전역 질의의 처리를 위한 모든 정보를 취하면 전역 XQuery 질의의 결과를 생성한다. 이를 위하여 초기 전역 XQuery 질의를 그대로 사용할 수는 없고, 이를 조금 변형해서 사용한다. 본 논문에서는 이를 주 XQuery 질의라고 표현한다. 주 XQuery 질의의 생성은 매우 간단하다. 주 XQuery 질의의 Where절을 제외한 모든 표현들은 전역 XQuery 질의의 표현과 동일하다. 그러나 주 XQuery 질의의 Where절에는 전역 질의의 Where절에 기술된 조건들 가운데 서로 다른 지역 시스템들 사이의 조인 연산만을 포함한다. 그 이유는 지역 XQuery 질의의 처리 시, 이미 지역 시스템에서 독립적으로 처리 가능한 조건들은 처리되었기 때문이다.

IV. 지역 질의의 생성을 기반으로 하는 전역 XQuery 질의 처리 시스템

<그림 4>는 질의 분할 기반의 전역 XQuery 질의를 처리하기 위한 시스템의 모델이다.

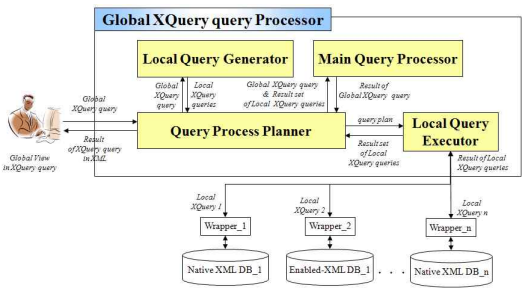

그림 4. 전역 XQuery 질의 처리기
Fig. 4. Global XQuery query Processor

사용자는 전역 View를 기반으로 분산된 지역 시스템들을 대상으로 하는 전역 XQuery 질의를 작성한다. Query Process Planner는 지역 시스템들을 대상으로 전역 XQuery 질의를 처리하기 위한 질의 처리 계획을 생성한다. 이때 가장 먼저 전역 XQuery 질의의 정제와 최적화를 수행한다. 이 과정은 본 논문에 앞서 [21]에서 이미 제안하고 검증한 바 있다. Local Query Generator는 최적화된 전역 질의를 입력으로 지역 XQuery 질의들을 생성하고, Query Process Planner는 이를 기반으로 지역 질의들의 처리 계획을 생성한다. 이렇게 생성된 질의 처리 계획은 Local Query Executor에 의해서 순차 혹은 병렬로 처리되며, 그 결과들은 취합되어 다시 Query Process Planner에게 반환된다. Main Query Processor는 Query Process Planner로부터 지역 질의의 결과와 전역 XQuery 질의를 전달받아 주 XQuery 질의를 생성하고 처리하여 최종 결과를 생성한다. 이렇게 얻어진 전역 XQuery 질의의 최종 결과는 다시 Query Process Planner가 사용자에게 반환한다.

V. 성능 평가

본 논문에서는 성능 평가를 위하여 AMD Phenom(tm) II X4 920 Processor 2.81 GHz CPU와 3.25GB 메모리의 개인용 컴퓨터 다섯 대를 단일 허브에 연결하여 사용하였다. 한 대는 전역 XQuery 질의를 처리하기 위해서 전역 XQuery 질의 처리기가 설치되어있으며, 다른 네 대는 각각 지역 XQuery 질의의 처리를 위한 지역 시스템으로 사용된다. 또한, XQuery 질의 처리를 위하여 Saxon-PE 9.2 XQuery processor[23]를 사용하였다.

표 5. 성능평가를 위한 전역 XQuery 질의
Table 5. Global XQuery query for performance evaluation

```
for $people in doc("people.xml")/people/person
for $open in
doc("open_auctions.xml")/open_auctions/open_auction
for $item in doc("Items.xml")/regions//item
for $closed in
doc("closed_auctions.xml")/closed_auctions/closed_auction
let $clD := $closed/seller/@person
let $plD := $people/@id
let $olD := $open/bidder/personref/@person
let $oItem := $open/itemref/@item
let $iItem := $item/@id
where $people/address/country="United States" and
$plD=$olD and $oItem=$iItem and $plD=$clD and
$item/payment="Creditcard"
return
<result>{ $people/name, $item/name, $closed/itemref,
$open/itemref }</result>
```

<표 5>는 성능 평가를 위한 전역 XQuery 질의이다. 성능 평가를 위한 질의 처리 방법은 세 가지 방법을 사용하였다. 첫 번째는 지역 XQuery 질의의 생성 없이 전역 XQuery 질의를 처리하기위하여, 모든 지역 XML 문서들을 하나의 시스템에 저장하고 한 시스템에서 전역 XQuery 질의를 처리하는 방법이다. 그리고 두 번째 방법은 <그림 5>의 위 쪽 그림 (A)와 같이 지역 XQuery 질의를 생성하고 이를 병렬로 처리하는 방법이다. 각 지역 XQuery 질의들은 각각 해당 지역 시스템에 분배되어 처리된 후, 그 결과를 전역 XQuery 질의 처리기에서 취합하여 최종 결과를 생성한다. 마지막 세 번째 방법은 <그림 5>의 아래 쪽 그림 (B)와 같이 지역 XQuery 질의들을 순차적으로 처리하는 방법이다.

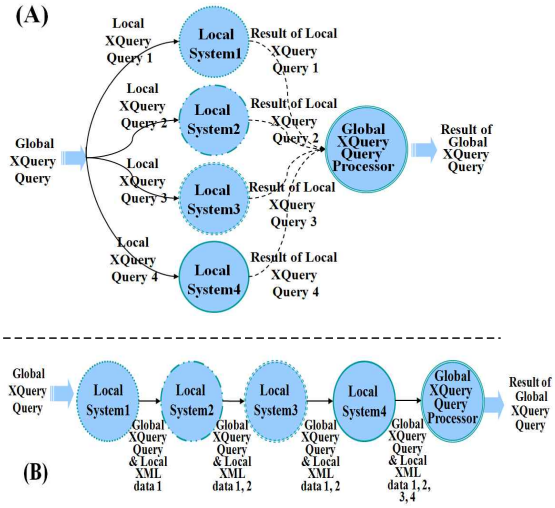

그림 5. 성능평가를 위한 전역 XQuery 질의 처리 방법들
Fig. 5. two processing methods of a Global XQuery query for performance evaluation

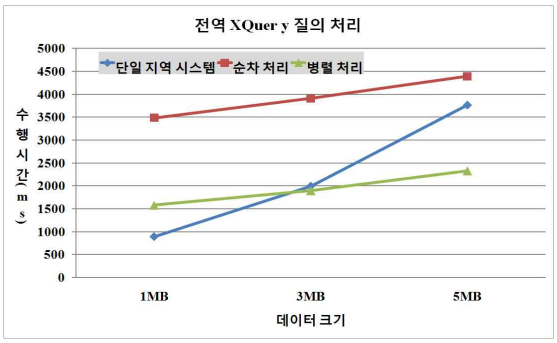

그림 6. 전역 XQuery 질의 처리 시간
Fig. 6. Processing Times of a Global XQuery query

<그림 6>은 <표 5>에 기술된 전역 XQuery 질의의 처리 시간을 보인다. 각 지역 시스템에 저장된 문서로는 XMark[22]의 Auction XML 문서를 각각 1MB, 3MB, 5MB씩 자동 생성한 후, 문서의 차 상위 노드인 'people', 'open auction', 'closed auction', 그리고 'item' 노드를 최상위 노드로 하는 네 개의 지역 XML 문서를 생성하여 사용하였다. 분산 환경에 전역 질의 처리의 성능에 영향을 미치는 또 다른 요소인 네트워크의 성능은 본 논문의 주제 밖에 있으므로 처리 시간 측정에서 제외하였다.

평가의 결과에서 볼 수 있는 것처럼, 데이터의 크기가 작은 경우 질의의 분할 없이 단일 시스템에서 전역 XQuery 질의를 처리하는 시간이 가장 좋게 나타났다. 그러나 데이터의 크기가 커질 경우 병렬 처리하는 방법이 좋은 성능을 보였다. 병렬 처리가 좋은 성능을 보인 이유는, 전역 XQuery 질의에 존재하는 선택 연산(Selection)을 지역 시스템에서 먼저 수행한 후 그 결과를 반환하였으므로 주 XQuery 질의의 처리 시간이 줄었기 때문이다. 순차 처리의 경우, 다른 질의 처리 방법에 비해 전반적으로 좋지 않은 결과를 보였다. 이는 지역 시스템에서 질의의 처리 시간이 많이 소요되었다기보다는 다섯 시스템에서 모두 XQuery 엔진을 구동하여 각각의 지역 XQuery 질의를 순차적으로 처리하였기 때문이다. 물론 이러한 실험의 결과는 전역 XQuery 질의에 존재하는 조인과 같은 질의의 특성, 분할된 질의의 처리 전략, 또는 질의 최적화 등에 의존적이다. 예를 들어 <표 5>의 전역 XQuery 질의에 선택 연산(Selection)이 존재하지 않고 지역 시스템들 사이에 조인 연산(Join)만 존재할 경우, 순차 처리가 가장 좋은 성능을 보였다. 그 이유는 조인 연산의 수가 너무 많아지므로 병렬 처리나 단일 지역 시스템에서의 경우, 모든 조인을 한 시스템에서 처리해야하기 때문이다. 그러나 순차 처리의 경우, 지역 시스템에 접근하는 횟수는 많지만 조인 연산을 분산하여 처리하므로 가장 좋은 성능을 보였다. 이러한 연구는 이미 전역 SQL 질의 처리를 위해 이미 연구된바 있다. 그러나 병렬 처리나 순차 처리 어떠한 경우에도 전역 XQuery 질의를 처리하기 위해서는 지역 XQuery 질의 생성 알고리즘이 반드시 필요하다.

VI. 결론

본 논문에서는 분산 환경에서 XQuery를 이용하여 이종 데이터들을 통합하고 검색하기위해서 사용되는 전역 XQuery 질의를 처리하기위한 방법을 제안하였다. 특별히 본 논문에서는 지역 XQuery 질의를 생성하여 전역 XQuery 질의를 처

리하기 위한 방법을 제안하였다 이를 위하여 이미 기존에 존재하는 지역 SQL 질의 생성 방법과 비교하여 그 문제를 정의하였다. 또한 우리는 지역 XQuery 질의의 결과로부터 통합 XQuery 질의의 올바른 결과를 재구성하기 위한 주 XQuery 질의를 제안하였다. 논문에서는 제안한 방법을 기반으로 프로토타입 시스템을 구형하고 그 유효성을 보였다.

논문에서 제안한 지역 XQuery 질의 생성 방법은 알고리즘을 수행하기 위하여 전역 XQuery 질의 이외의 어떠한 정보도 참조하지 않으므로, XQuery 질의 처리를 위한 다양한 응용에서 특별한 제약 없이 사용할 수 있을 것으로 기대된다. 현재 논문에서 제안하고 있는 질의 생성 방법은 정적으로 수행되고 있지만, 향후 각 응용들의 특성에 맞는 동적인 질의 생성 방법으로 확장이 가능할 것으로 사료된다.

참고문헌

[1] M. Mani, "*XML Views*," Encyclopedia of Database Systems, pp.3656-3659, Springer, 2009.

[2] V. M. P. Vidal, F. C. Lemos, V. d. S. Araujo and M. A. Casanova, "A Mapping-Driven Approach for SQL/XML View Maintenance," Proc. ICEIS 2008, Spain, pp.65-73, June 2008.

[3] I. Manolescu, D. Florescu & D. Kossmann "Answering XML Queries over Heterogeneous Data Sources," Proc. VLDB, Roma, Italy, September 2001.

[4] XQuery 1.0: An XML Query Language, *"http://www.w3.org/TR/2005/WD-xquery-20050404/,"* April 2005.

[5] H. Kozankiewicz, K. Stencel, K. Subieta, "Distributed Query Optimization in the Stack-Based Approach," Proc. HPCC 2005, Sorrento, Italy, September 2005.

[6] V. Josifovski & T. Risch, "Query Decomposition for a Distributed Object-Oriented Mediator System," Distributed and Parallel Databases, Vol. 11, No. 3, pp. 307-336, May 2002.

[7] D. Suciu, "Query Decomposition and View Maintenance for Query Languages for Unstructured Data," Proc. VLDB 1996, Mumbai (Bombay), India, September, 1996.

[8] M. Smiljanic, L. Feng & W. Jonker, "Web-Based Distributed XML Query Processing," Proc. Intelligent Search on XML Data 2003, September 2003.

[9] D. Kossmann "The state of the art in distributed query processing," ACM Computing Surveys. vol 32(4), pp 422-469, 2000.

[10] D. Suciu "Distributed query evaluation on semistructured data," ACM Transaction. Database System. Vol 27, No.1, pp 1-62, 2002.

[11] X. Zhang, B. Pielech & E. A. Rundensteiner "Honey, I shrunk the XQuery!: an XML algebra optimization approach," Proc. WIDM 2002, McLean, Virginia, USA, November 2002.

[12] H. Su, E. A. Rundensteiner & Murali Mani "Semantic Query Optimization in an Automata-Algebra Combined XQuery Engine over XML Streams," Proc. VLDB 2004, Toronto, Canada, September 2004.

[13] C. Koch, S. Scherzinger, N. Schweikardt & B. Stegmaier "FluXQuery: An Optimizing XQuery Processor for Streaming XML Data," Proc. VLDB 2004, Toronto, Canada, September 2004.

[14] L. H. Yang, M. L. Lee & W. Hsu "Finding hot query patterns over an XQuery stream," VLDB Journal. Vol. 13(4), pp 318-332, 2004.

[15] V. Josifovski and T. Risch, "Query Decomposition for a Distributed Object-Oriented Mediator System," Distributed and Parallel Databases, Vol.11, no.3, pp.307-336, 2002.

[16] L. T. T. Thuy, D. D. Duong, V. C. Bhavsar and H. Boley, "A Bottom-up Strategy for Query Decomposition," Proc. ICDIM 2006, India, pp.215-221, 2006.

[17] G. Lausen and P. J. Marron, "Adaptive Evaluation Techniques for Querying XML-based E-Catalogs," Proc. of RIDE 2002, USA, pp.19-28, 2002.

[18] L. T. T. Thuy and D. D. Duong, "Query Decomposition Using the XML Declarative Description Language," Proc. ICCSA 2005, Singapore, pp.1066-1075, 2005.

[19] Y. Zhang, N. Tang and P. A. Boncz, "Efficient Distribution of Full-Fledged XQuery," Proc.

ICDE 2009, China, pp.565-576, 2009.
[20] Y. Zhang and P. A. Boncz, "XRPC: distributed XQuery and update processing with heterogeneous XQuery engines," Proc. SIGMOD2008, Canada, pp.1331-133, 62008.
[21] 박종현, 강지훈 "분산 환경에 질의 최적화를 위한 XQuery 질의 재작성," 한국컴퓨터정보학회논문지 제 14권, 제 3호, 1-11쪽, 2009년. 3월.
[22] A. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse "XMark: A Benchmark for XML Data Management," Proceeding of VLDB 2001, Hong Kong, China, August, 2001.
[23] SAXON, The XSLT and XQuery Processor, *"http://saxon.sourceforge.net/"*

저 자 소 개

박 종 현

2002년 : 충남대학교 컴퓨터과학과 석사 졸업
2007년 : 충남대학교 컴퓨터과학과 박사 졸업
2007년~2008년 : 충남대학교 소프트웨어연구소 연구원
2009년~2009년 : 거제대학 조선정보계열 초빙교수
2009년~현재 : 큐슈대학교 정보기반 연구센터 방문연구원
관심분야 : XML, XQuery, 상황인지 추론시스템 Ontology, 분산 데이터베이스, 유비쿼터스 컴퓨팅, 웹정보시스템

박 원 익

2004년 : 충남대학교 컴퓨터과학과 학사 졸업
2006년 : 충남대학교 컴퓨터공학과 석사 졸업
2007년~현재 : 충남대학교 컴퓨터 공학과 박사 과정
관심분야 : 지능형 추천 알고리즘, e-비즈니스, 유비쿼터스 컴퓨팅

김 영 국

1985년 : 서울대학교 계산통계학과 학사 졸업
1987년 : 서울대학교 계산통계학과 석사 졸업
1995년 : 버지니아 대 컴퓨터과학과 박사 졸업
1995년~1996년 : 핀란드 VIT, 노르웨이 SINTEF DELAB 방문연구원
1996년~현재 : 충남대학교 컴퓨터 공학과 교수
관심분야 : 실시간데이터베이스, 모바일정보시스템, 전자상거래시스템

강 지 훈

1981년 : 한국과학기술원 전산학과 석사 졸업
1996년 : 한국과학기술원 전산학과 박사 졸업
2000년~2002년 : 충남대학교 정보통신원장
1985년~현재 : 충남대학교 전기정보 통신공학부 교수
관심분야 : 시멘틱웹, 추론, XML, XQuery, 데이터베이스 시스템, 웹정보시스템