메타데이타 인터페이스를 이용한 분산된 반구조적 문서 검색을 위한 질의처리 알고리즘 설계 및 구현

(Design and Implementation of a Query Processing Algorithm for Distributed Semistructured Documents Retrieval with Metadata Interface)

최 귀 자 † 남 영 광 ††
(Guija Choe) (Young-Kwang Nam)

요 약 반구조적 분산 문서에서는 구조 정보가 제공되지 않고, 자료 구조에 대한 엄격한 형식이 없기 때문에 질의 처리 시스템을 정형화하고 구현하기 어렵다. 이질적이고 이구조적인 반구조적 문서의 요소를 정확하게 검색하기 위해서는 한 요소가 1:1, 1:N, N:1과 같이 서로 다른 매핑 형태를 취하면서 동시에 여러 요소에 매핑되는 다중 매핑을 처리할 수 있어야 하며, 지역문서의 태그를 파싱하여 구조적인 정보를 얻고 경로 트리를 생성해야 한다.

본 논문에서는 분산된 시스템에 존재하는 이질적인 반구조적 자료나 문서에 대한 동시 다중 매핑을 완벽히 지원하고, 문서 자체를 파싱하여 구조적 정보를 얻을 수 있도록 통합 질의와 검색을 수행하기 위한 추상적인 질의 처리 알고리즘을 설계하고 메타데이타 인터페이스를 이용하여 구현하였다. 이 알고리즘은 전역질의를 기반으로 지역질의를 생성하기 위해서 메타데이타 정보를 이용하여 노드들 사이의 매핑, 매핑 종류에 따른 데이타의 변환, 경로교체 및 노드 사이의 이질성을 해결하기 위한 알고리즘으로 제시하였다.

전역스키마와 지역스키마에 대한 매핑과 함수에 의한 데이타의 변환 및 경로교체는 사용자에 의해 구축된 메타데이타 인터페이스인 DDXMI(for Distributed Documents XML Metadata Interface)를 기반으로 하여 구현되었으며, 같은 이름을 갖지만 다른 의미를 갖는 자료나 노드에 대한 검색은 노드를 구분할 수 있는 노드가 가지고 있는 자식정보를 이용하여 노드 구분 조건절을 생성하여 구현하였다. XML 질의 언어로 Quilt를 사용하였으며, OEM 모델로 제시한 세 개의 서로 다른 반구조적 레스토랑 안내 문서에서 구현한 결과를 보였다. 프로토타입 시스템은 윈도우즈 환경에서 Java와 JavaCC 컴파일러를 이용하여 개발하였다.

키워드 : 반구조적, XML, 정보검색, 메타데이타

Abstract In the semistructured distributed documents, it is very difficult to formalize and implement the query processing system due to the lack of structure and rule of the data. In order to precisely retrieve and process the heterogeneous semistructured documents, it is required to handle multiple mappings such as 1:1, 1:N and N:1 on an element simultaneously and to generate the schema from the distributed documents.

In this paper, we have proposed an abstract query processing algorithm for querying and answering on the heterogeneous semistructured data or documents over distributed systems and implemented with a metadata interface. The algorithm for generating local queries from the global query consists of mapping between global and local nodes, data transformation according to the mapping types, path substitution, and resolving the heterogeneity among nodes on a global input query with metadata information.

The mapping, transformation, and path substitution algorithms between the global schema and the local schemas have been implemented the metadata interface called DDXMI (for Distributed Documents XML Metadata Interface). The nodes with the same node name and different mapping or

† 정 회 원 : 연세대학교 전산학과
gjchoe@hosu.yonsei.ac.kr
†† 종신회원 : 연세대학교 전산학과 교수
yknam@dragon.yonsei.ac.kr
논문접수 : 2004년 11월 10일
심사완료 : 2005년 4월 18일

meanings is resolved by automatically extracting node identification information from the local schema automatically. The system uses Quilt as its XML query language. An experiment testing is reported over 3 different OEM model semistructured restaurant documents. The prototype system is developed under Windows system with Java and JavaCC compiler.

Key words : Semistructured, XML, Information Retrieval, Metadata

1. 서 론

XML DTD(Document Type Definition)나 스키마에 의해 구조적으로 정의된 분산 문서상의 이질성을 극복하기 위한 질의와 처리 결과에 대한 연구 활동이 다양하게 진행되고 있다. 이러한 기술들의 대부분은 구조적으로 잘 정의된 자료나 문서의 경우, 문서의 구조 정보를 DTD나 스키마로부터 얻어 가상의 전역스키마에 질의를 보내고 결과를 얻는 방법을 취한다. 이에 비해 반구조적 분산 문서는 문서에 대한 구조적 정보가 제공되지 않고, 자료 구조에 대한 엄격한 형식이 없기 때문에 시스템을 정형화하고 구현하기가 매우 어렵다.

분산시스템에서 반구조적 문서와 구조적 문서의 질의처리시 중요한 차이점은 첫째, 문서의 한 노드나 요소가 1:1, 1:N, N:1과 같이 서로 다른 매핑 형태를 취하면서 동시에 여러 노드에 매핑될 수 있으며, 둘째, DTD나 스키마와 같은 구조적 정보가 지역문서에 대해 제공되지 않기 때문에 문서의 태그를 파싱하여 구조적인 정보를 얻고 경로 트리를 생성해야 한다. 반구조적인 문서로부터 사용자가 원하는 형태의 검색 결과를 얻기 위한 몇몇 시스템들이 제안되고 있으나, 이질적이면서 이구조적인 요소가 지역문서 상에서 동시에 발생하는 경우에는 정확한 매핑을 지원하지 못한다. 전역스키마의 하나의 요소에 대응하는 지역문서 상의 여러 요소가 동일한 의미이지만 구조가 달라 다양한 형태의 매핑이 동시에 발생하는 경우에 대해 기존의 시스템들은 1:1이거나, 1:N이거나, 혹은 N:1인 경우로 선택적 매핑만을 지원한다. 또한 현재까지 조사된 바로는 분산 이구조 문서검색 시스템에서 반구조적 문서에 대한 처리 기술이 제시되지 않고 있다.

본 논문에서는 이러한 문제점을 해결하여 전역 요소에 대응하는 분산된 지역문서에 대한 동시 다중 매핑을 지원한다. 즉, 여러 노드나 요소가 동일한 의미이지만 구조가 다른 경우 동시에 여러 가지의 매핑을 중복해서 지원하여 지역문서의 구조가 불규칙적으로 반구조적인 문서에 대한 검색 결과를 정확히 얻을 수 있도록 하였다. 또한 DTD나 스키마가 제공되지 않을 경우 XML 문서 자체를 파싱하여 구조정보를 사용자에게 제공하므로 정확하게 태그된 어떠한 XML 문서도 검색할 수 있다. 부정확하게 태그된 문서는 문서구조 정보를 수집하기가 어렵기 때문에 본 시스템에서는 고려하지 않는다. 물론 DTD나 스키마와 같은 구조적인 정보가 사용자에 의해 주어진다면 질의를 얻기 위한 추가적인 작업이 필요하지 않다.

본 논문에서의 접근 방법은 메타데이타 인터페이스인 DDXMI(Distributed Documents XML Metadata Interface)에 저장된 매핑 정보를 기반으로 질의 처리를 모델화하고 시스템을 구현하는 것이다. DDXMI 파일은 반구조적 문서의 이름, 위치정보 및 경로정보, 반구조적 자료의 노드나 경로에 관한 의미정보를 포함한다. 시스템 모형은 사용자를 위해 메타데이타를 통합 할 수 있는 툴을 제공하고, 전역질의로부터 지역문서에 대한 질의를 생성하여 결과를 통합하기 위해 사용되는 DDXMI 파일을 생성하도록 구현되었다. 이 도구는 지역문서의 DTD 혹은 문서 자체를 파싱해서 각 요소에 대한 경로를 생성하며, 사용자가 클릭하면 트리를 펼치거나 접을 수 있는 동적인 경로트리를 제공한다. 지역요소에 대해서는 대응하는 전역요소 및 변환함수 이름과 연결되는 매핑 요소를 클릭하여 가시적인 색인번호를 할당하며, 변환함수는 본 시스템의 XML 질의 언어로 사용된 Quilt[1]에 내장되어 있거나 사용자가 정의한 함수를 사용할 수 있다. 시스템에서 색인번호를 내부적으로 수집해서 얻은 매핑 정보를 기반으로 DDXMI가 생성된다. 사용자 질의는 생성된 DDXMI에 따라 각각 상응하는 지역문서에 대한 실행 가능한 질의를 생성하여 Quilt에 의해 처리된다.

분산된 질의 및 질의 최적화와 같은 대부분의 복잡한 문제들이 Quilt 내에서 처리되므로 시스템은 상대적으로 간편하고, 간단한 GUI로 인해 사용하기 편하다. 본 논문에서 제시한 시스템은 사용자가 같은 스키마일지라도 서로 다른 가상의 통합정보 시스템을 구성할 수 있으며, 서로 다른 사용자가 그들 자신의 필요에 따라 서로 다른 가상의 정보 시스템을 가질 수 있는 유연성을 제공한다.

알고리즘의 설계, 구현 및 실험과 검증을 위해 본 논문에서는 [2]에서 제안한 OEM (Object Exchange Model) 데이타 모델을 변경하여 이구조적, 이질적인 형태를 동시에 포함할 수 있도록 세 개의 서로 다른 반구조적 레스토랑 안내 문서를 제작하고 이용하였다. 설계

한 알고리즘을 기반으로 시스템을 구현하였으며, 입력된 전역질의에 대하여 자동적으로 지역질의가 생성되도록 하였고, 지역질의를 실행한 결과도 함께 나타내었다. 프로토타입 시스템은 윈도우즈 환경에서 Java와 JavaCC 컴파일러를 이용하여 개발하였으며, XML 문서를 파싱하기 위해 SAX(Simple API for XML) 파서를 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 반구조적 문서의 질의에 대한 관련 연구를 조사하였으며, 3장에서는 지역질의 생성을 위한 추상적인 알고리즘을 매핑, 함수변환, 경로교체, 이질적인 노드에 대한 식별 알고리즘으로 제시하였고 4장에서는 3장에서 제시한 알고리즘을 구현한 전체 시스템의 개요와 각 주요 모듈에 대한 구체적인 구현 방법을 제시하였다. 5장에서는 구현 결과를 보이기 위해서 세 개의 반구조적 문서에 대한 전역스키마, 지역문서를 파싱하여 생성한 지역스키마, 전역질의를 이용하여 생성한 지역질의와 그 지역질의를 실행한 결과를 나타내었다.

2. 관련 연구

반구조적 데이타에 관한 연구는 데이타베이스의 구조가 불규칙하고, 구조를 알 수 없거나 때로 변경될 수 있는 구조를 가진 데이타베이스 관리 기술의 확장을 목표로 연구된 기술이다[3,4]. 여기서 반구조적 데이타란 엄격한 형태를 취하지 않을 뿐만 아니라, 불규칙적이고 부분적인 구조를 갖는 데이타 형태를 말한다[3-6]. 반구조적 데이타와 관련된 대부분의 연구는 반구조적 데이타에 대한 논리적 데이타 모델과 질의 언어에 중점을 두고 있으며, 시스템 구현은 미약한 실정이다[2,6,7]. 이와 관련된 언어 기반 접근 방법의 일부 시스템들은 데이타의 특징을 기술 할 수 있고, 스키마 변환을 위한 툴을 제공한다. 범용 데이타 관리 시스템인 LORE (Lightweight Object REpository)[8], 자료 변환 시스템인 YAT(Yet Another Tree-based system)[9], 이질적 데이타 통합을 위한 TSIMMIS(The Stanford-IBM Manager of Multiple Information Sources)[10,11]와 같이 반구조적인 데이타를 사용하는 프로토타입과 방법들이 소개되었으며, MSL(Mediator Specification Language)[12,13], Lorel(LORE Language)[8,14], UnQL (Unstructured Query Language)[15], YATL[9]과 같은 반구조적 데이타를 위한 질의 언어가 제안 되었다.

LORE는 데이타베이스 엔진 내부에 오토마타와 그래프 항행 기술을 사용하고 인덱싱 기법을 경로 표현식에 적용한 네비게이션 형태를 갖는 반구조적 범용 데이타베이스 시스템으로 OEM 데이타 구조를 데이타가이드에 저장하여 스키마 구조 정보를 얻고 있으나, 모든 경로에 대한 정규식을 지원하지는 못하고 있다. YAT는 이질적 자료 변환방법으로 일치 규칙에 의한 방법들을 제시하여 데이타의 특징들과 스키마간의 매핑을 가능하게 하지만 문서의 패턴을 정의하고 변수에 매핑시키는 규칙 적용이 복잡하다. TSIMMIS 시스템에서 데이타 통합을 위해 제시한 방법인 MIX Mediator System은 스키마 도메인 불일치, 스키마적 불일치, 구조적 불규칙성을 갖는 서로 다른 형식으로 작성된 데이타 모델을 정형화된 OEM 혹은 XML 모델로 변형하기 위해서 랩퍼를 사용하며, 상당히 좋은 성능을 가지고 있지만 논리적 데이타 합병과 물리적 데이타 합병 그래프를 사용하는 매우 복잡한 방법을 사용한다. 이러한 도구들은 분산된 문서를 지원하지 못하고 단지 특정 데이타베이스나 문서에 국한되어 있으며, 조건이나 함수 변환에 있어서 매핑 방법이 상당히 어렵다. 또한 반구조적 분산된 데이타나 문서상에서 하나의 노드나 경로에 대해 1:1, 1:N, N:1과 같은 다중 매핑을 자동으로 지원하지 않으며, 여러 노드나 경로에 대해서도 이러한 매핑을 동시에 허용하지 않는다.

반구조적 문서를 위한 질의 언어들이 다양하게 소개되고 있다. MIX Mediator System에서 사용된 MSL은 논리적 기반의 언어로, 상이한 데이타 로그를 객체 융합의 개념으로 반구조적 데이타에 확장 적용한 TSIMMIS 데이타 통합 프로젝트에서 사용된 질의 언어이다[12,13]. LORE 시스템에서 사용된 Lorel은 데이타 엔진 내부에 오토마타의 사용과 그래프 추적 기술을 요구하는 질의 표현 기반의 네비게이션 형태를 갖는다[8,14]. UnQL은 질의 언어의 구문과 의미를 기초로 구조적 반복이라는 대수식에 근거한 반구조적 데이타와 XML을 위한 질의 언어이다[15,16]. 여기서 구조적 반복이란 순환 그래프 상에서 명백한 의미를 갖는 것을 말한다. 이와 같은 반구조적 분산 데이타 처리를 위한 질의 언어들은 각 시스템에 종속적으로 이용되고 있어 범용성을 가지지 못한다. 이에 비해, 본 논문에서 사용한 Quilt[1] 질의 언어는 XML_QL, XQL, XPath, YATL, XSQL 등 XML 질의 언어와 SQL, OQL 등 데이타베이스 질의 언어의 장점을 적용하여 만든 범용성을 갖는 XML 질의 언어이므로 어떠한 XML 문서에 대해서도 적용이 가능하다.

본 논문의 예제로 사용된 문서에서 응용한 개념인 OEM은 TSIMMIS 데이타 통합 프로젝트를 위해 설계된 반구조적 데이타 모델이다[2,17,18]. OEM은 그래프 기반의 자기 묘사적인 객체 모델로서 데이타에 대한 규칙성이 없으며, 각 객체는 그 자신의 이름표를 가지므로 데이타 자체를 묘사할 수 있다. 이 모델에서의 데이타는 이름표가 붙은 방향성 그래프로 간주되며, 모든 스키마

적 정보는 동적으로 변하는 이름표에 포함되므로 고정된 스키마의 개념을 별도로 갖지 않는다. 이와 같은 특징 때문에 반구조적인 문서를 다루는 대부분의 응용에서 데이타 모델로 OEM을 사용하고 있다.

본 논문에서는 응용 사용자들에게 DDXMI를 통해 반구조적 데이타나 자료에 대한 질의와 검색을 가능하게 하는 쉬운 방법을 제시하며, 현재 제안되고 있는 여러 도구들이 하나의 노드나 경로에 대해 여러 유형의 다중 매핑을 동시에 허용하지 못하는 문제점을 해결하여, 정확한 검색 결과를 얻을 수 있도록 개발하였다. 반구조적 데이타나 문서의 검색을 위한 그래픽한 사용자 인터페이스(GUI)를 통해, 전역문서를 기반으로 입력으로 받은 여러 개의 각 지역문서에 대한 노드들 사이의 매핑, 변환, 경로교체 및 이질성을 해결한다. 매핑과 변환은 의미적 메타데이타 인터페이스인 DDXMI에 구현되었으며, 전체 시스템은 아주 간단한 구조를 가지기 때문에 응용 사용자들이 손쉽게 조작할 수 있고, 유용성과 의미적 데이타 통합을 위한 충분한 기능을 제공한다.

3. 질의 처리 알고리즘

전역질의 Q_{in}으로부터 N 개의 지역질의 Q_{out}을 생성하기 위해 세 개의 알고리즘 *M(GS, LS)*, *PS*, *NIP*를 제안한다. 여기서 *GS*는 전역스키마, *LS*는 지역스키마 L_i, ..., L_j 집합, *M*은 전역스키마와 지역스키마 간의 매핑, *PS*는 다음 단락에서 설명될 경로교체를 위한 알고리즘이며, *NIP*는 동일한 노드 이름이지만 이질적인 구조를 갖는 노드를 해결하기 위한 노드 식별 조건절을 생성하기 위한 알고리즘이다. 그림 1에 질의 생성을 위한 각 알고리즘과 전체 시스템과의 연관 관계를 도식화하여 나타내었으며, 각 알고리즘은 다음 단락에서 상세히 기술한다.

3.1 전역스키마와 지역스키마 간의 매핑(M(GS, LS))

*G*를 전역스키마 *GS*의 경로트리 *GT*의 노드 집합, *L*을 지역스키마 *LS*의 경로트리 *LT*의 노드 집합이라 가정하고, *PG*와 *PL*을 각각 *G*와 *L*의 멱집합이라 하자. 각 트리의 노드 o_i는 노드명 ol_i와 노드값 ov_i로 구성된 객체 (ol_i, ov_i)가 된다. 그림 2에서 노드 번호 **5**는 노드명 'location'과 노드값 '1900 King's Highway, Rolla, MO, 65401'을 갖는다. *GT*와 *LT*에서, 여러 노드가 동일한 노드명을 갖기 때문에, 필요할 경우에는 그림 2에 있는 'location$_1$', 'location$_2$'와 같이 노드명에 첨자를 추가하여 구분해서 사용한다.

정의 1. *PG*와 *PL*의 카티션곱(*Cartesian Product*) $CP=PG \times PL$은 (g, l) 원소들의 집합이다. 여기서 $g=(gn_1, gn_2, ..., gn_m)$이고 $gn_i \in G$이며, $l=(ln_1, ln_2, ..., ln_m)$이고 $ln_i \in L$이다.

정의 2. 전역스키마 *GS*와 지역스키마 *LS* 사이의 매핑 $M(GS, LS) \subset CP(GS, LS)$는 $M_{11}(GS, LS) \cup M_{1N}(GS, LS) \cup M_{N1}(GS, LS)$이며, 다음 조건을 만족해야 한다.

i) $M_{11}(GS, LS)$는 일대일 매핑 원소들의 집합으로서, $g \in PG$는 단일원소집합(singleton)이고, $l \in PL$에 대해 $m_{11}(g)=l$인 원소 $m_{11}=(g, l) \in M(GS, LS)$의 집합이다.

ii) $M_{N1}(GS, LS)$는 다대일 매핑 원소들의 집합으로서, $g=(gn_1, gn_2, ..., gn_m)$ (단 $m>1$)은 $g \in PG$이고, 단일원소집합 $l \in PL$에 대해 $m_{N1}(g)=l$인 원소 $m_{N1}=(g, l) \in M(GS, LS)$의 집합이다.

iii) $M_{1N}(GS, LS)$는 일대다 매핑 원소들의 집합으로서, 단일원소집합 $g \in PG$와 $l=(ln_1, ln_2, ..., ln_m)$이면서 $l \in PL$인 $m_{1N}=(g, l) \in M(GS, LS)$인 원소들의 집합이다.

iv) 이외의 다른 원소 $g \notin (domain(M_{11}) \cup domain(M_{N1}) \cup domain(M_{1N}))$에 대해서 $m_0(g)=\varnothing$ 이다.

만약 문맥상 이러한 의미가 명백하다면, 간단히 *M*과 *m*으로 표현한다.

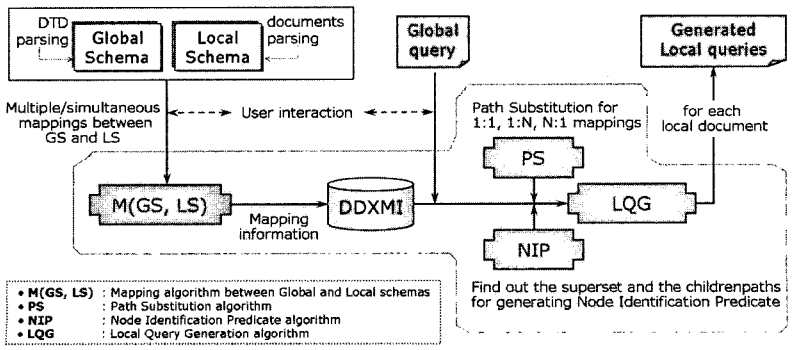


그림 1 질의 생성 알고리즘과 시스템 연관 관계

gn_i와 ln_i는 각각 트리 GT, LT의 내부 혹은 외부 노드일 수 있다. 'guide'를 전역문서, 'agency2'를 지역문서의 스키마 이름이라 하자. 예를 들어, 그림 2에서, M_{11}(guide, agency2)는 ((state, state_code), (zip, zip_code))로, M_{N1}(guide, agency2)는 (((street, city, zipcode), location), ((street, city, state, zip), location))으로, 그리고 M_{1N}(guide, agency2)는 (zipcode, (state_code, zip_code))로 표현된다.

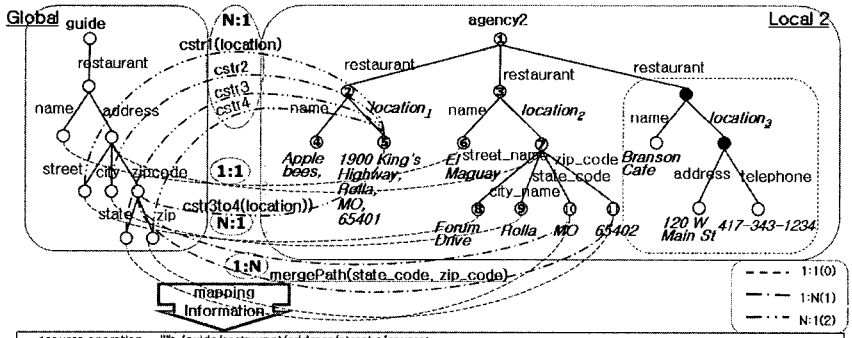
정의 3. $m \in M(GS, LS)$라 하자. $m=(l, g)$에 대한 변환 T_m은 $T_m: l \rightarrow g$로 정의되고 $T_m(l)=g$이다. 여기서 T_m은 벡터이고, T_m의 길이 $|T_m|$은 $|g|$이며, 객체로부터 적절한 값을 얻기 위해 l 객체의 값에 적용되는 함수 벡터를 의미한다.

예를 들어, m_{1N}=(zipcode, (state_code, zip_code))일 때 $mergePath_{m1N}$((state_code, 'MO'), (zip_code, '65402'))=(zipcode, 'MO 65402')이고, m_{N1}=((street, city, zipcode), location)일 때 $(cstr_1, cstr_2, cstr_3to_4)$ m_{N1}(location, '1900 King's Highway, Rolla, MO, 65401')=((street, '1900 King's Highway'), (city, 'Rolla'), (zipcode, 'MO, 65401'))이다. 여기서 '$mergePath$'는 문자열을 합병하는 함수이고, '$cstr_i$'는 노드의 값에서 콤마로 분류되는 i번째 단어를 추출하기 위한 함수이며, '$cstr_ito_k$'는 j번째부터 k번째까지의 일부 단어를 추출하는 함수이다. 때로 $(div(100)) \circ (cstr_i)$와 같이 문자함수에 나누기 연산자를 추가하는 복합함수가 사용되기도 한다. 이 복합함수는 전역문서의 화폐 단위는 달러인 반면 지역문서의 화폐단위는 센트인 경우, 전역문서내의 요소와 동일한 의미를 가지도록 지역문서의 화폐 단위를 바꾸기 위해 i번째 문자열에 나누기 연산자를 재 적용함을 의미한다.

3.2 지역질의를 위한 경로교체(PS)

XPath는 XML 문서를 노드의 트리로서 모델화하였다[19]. XPath의 중요한 표현 방식 중 하나는 위치경로이다. 위치경로는 문맥 노드에 관련된 노드 집합을 선택하는 방법이며, 위치경로의 수식 값을 구한 결과는 위치경로에 의해 선택된 노드를 포함하는 노드 집합이다. 위치경로는 노드의 집합을 여과하기 위해 사용되는 수식을 반복적으로 포함할 수 있다.

위치경로는 절대위치경로와 상대위치경로 두 종류가 있다. 상대위치경로는 '//' 로 분리되는 하나 혹은 그 이상의 순차적인 위치 단계로 구성되며, 이 단계는 좌측에서부터 우측으로 합병된다. 절대위치경로는 '/' 기호와 선택적으로 올 수 있는 상대위치경로로 구성된다. '/' 자체는 문맥 노드를 포함하고 있는 문서의 루트 노드를 선택한다. 만약 '/' 기호 다음에 상대위치경로가 온다면, 위치경로는 문맥 노드를 포함하는 문서의 루트 노드와 관련이 있는 상대위치경로에 의해 선택된 노드 집합을 선택한다.




```
<source operation = "">/guide/restaurant/address/street</source>
    <target>
        <targ type = "2" operation = "cstr1">/agency2/restaurant/location</targ>
        <targ type = "0" operation = "">/agency2/restaurant/location/street_name</targ>
    </target>
<source operation = "">/guide/restaurant/address/city</source>
    <target>
        <targ type = "2" operation = "cstr2">/agency2/restaurant/location</targ>
        <targ type = "0" operation = "">/agency2/restaurant/location/city_name</targ>
    </target>
<source operation = "">/guide/restaurant/address/zipcode</source>
    <target>
        <targ type = "2" operation = "cstr3to4">/agency2/restaurant/location</targ>
        <targ type = "1" operation = "0,1">/agency2/restaurant/location/state_code, /agency2/restaurant/location/zip_code</targ>
    </target>
<source operation = "">/guide/restaurant/address/zipcode/state</source>
    <target>
        <targ type = "2" operation = "cstr3">/agency2/restaurant/location</targ>
        <targ type = "0" operation = "">/agency2/restaurant/location/state_code</targ>
    </target>
<source operation = "">/guide/restaurant/address/zipcode/zip</source>
    <target>
        <targ type = "2" operation = "cstr4">/agency2/restaurant/location</targ>
        <targ type = "0" operation = "">/agency2/restaurant/location/zip_code</targ>
    </target>
```

그림 2 전역노드와 지역노드 간의 매핑

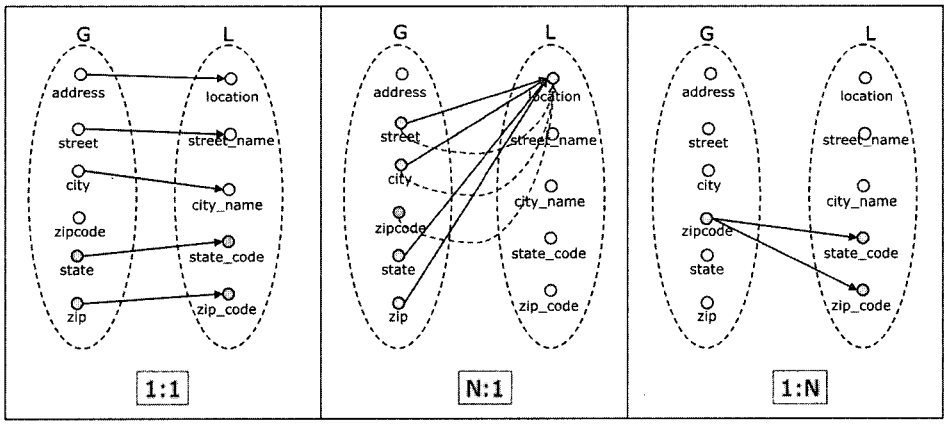


그림 3 그림 2에서의 노드를 집합으로 표현한 매핑

Quilt는 경로를 표현하기 위해 W3C에서 정의된 것과 같은 XPath 표현식을 사용하며, FOR, LET, WHERE와 RETURN 절로 구성되어 있다. FOR와 LET 절은 검색 대상이 되는 노드를 선택하여 변수에 결합하기 위한 기능을 가진다. FOR와 LET 절에서 조건을 주어 경로를 선택하기 위해서는 '[ ](predicate)' 안에 조건식을 작성한다. 따라서 FOR와 LET 절에서의 경로교체는 매핑 종류에 따라서 검색 노드의 선택 위치가 다르다. 1:N 매핑의 경우에는 하나의 전역경로가 N 개의 지역경로와 매핑이 되지만 FOR와 LET 절에서는 이를 N 개의 서로 다른 변수로 결합할 수 없기 때문에 매핑되는 지역 노드의 부모 노드를 매핑시키고 조건절에서 매핑되는 노드를 검색할 수 있도록 한다. WHERE 절은 조건에 맞는 요소나 속성을 선택하기 위한 문장으로, WHERE 절에서의 경로교체는 FOR와 LET 절과는 달리 XPath의 노드는 그 노드가 가지고 있는 값을 의미하기 때문에 1:N 혹은 N:1 매핑이 적용될 경우에는 매핑된 후에 값을 가질 수 있도록 변환함수를 적용하여 경로를 교체한다. RETURN 절은 조건에 의해 검색된 결과를 출력한다.

전역질의로부터 지역질의를 생성하기 위한 중요한 작업 중의 하나는 전역질의에서의 경로를 지역스키마의 경로트리에 적합한 경로로 교체하는 일이다. 경로교체 알고리즘은 아래와 같다 :

Path Substitution algorithm $PS(X_{in}) = X_{out}$;
Input : Tokenized XPath expression X_{in} in the global query
Output : the local XPath expression X_{out}
$X_{out} = (x_1, ..., x_i, ..., x_k)$ where k = the number of mappings for X_{in};
While (token is not null)
  Begin
    Read the next token of the input string
    Case *token* begin
      i) '/' : replaced by '/' and concatenate it to each element of X_{out}.
      ii) '//' : replaced by '//' and concatenate it to each element of X_{out}.
      iii) '.' : replaced by '.' and concatenate it to each element of X_{out}.
      iv) '..' : replaced by '..' and concatenate it to each element of X_{out}.
      v) a node label p :
        For each mapping element $m_i = (p, q)$,
          Case *i* for mapping type:
            (i) 1:1 : concatenate q to x_i;
            (ii) N:1 :
              if X_{in} belongs to the location path
                then concatenate q to x_i;
            (iii) 1:N :
              if X_{in} belongs to the location path
                then mapping with q's parent node;
              else make N duplicates of x_i and
                let x_i in X_{out} be replaced by N copies of x_i
                concatenate $q_j \in q$ to corresponding x_i
          End Case *i*
        End For
      End Case *token*
    End While
    Apply T_m over X_{out};

그림 2에서의 예를 들면, 1:N인 경우 m(address)=location이고 m(zipcode)=(state_code, zip_code)일 때 위치경로에서의 경로교체는 PS(address/zipcode)=(location/state_code, location/zip_code)이고, 조건경로에서의 경로교체는 PS(address/zipcode)=T_m(address/zipcode)=*mergePath*(location/state_code, location/zip_code)이

며, m(address)=null이고 m(street)=location일 때 위치 경로에서는 PS(address/street)=location이고, 조건경로에서는 PS(address/street)=T_m(address/street)=*cstr1*(location)이다.

3.3 지역문서에서 이질적인 노드에 대한 구분(NIP)

앞에서 언급한 바와 같이 구조적 자료와 반구조적 자료 처리 사이의 중요한 차이점은, 반구조적 자료는 동일한 이름을 가지면서 서로 다른 구조인 여러 개의 노드가 한 문서에 있거나, 서로 다른 이름을 가지면서 구조도 서로 다른 여러 개의 노드가 한 문서에 존재한다는 것이다. 만약 동일한 이름을 갖는 지역트리의 모든 노드가 전역트리의 한 노드와 매핑 된다면, 지역질의 생성은 단순해진다. 그러나 동일한 이름으로 서로 다른 구조를 갖는 지역노드가 두 개 이상의 전역노드와 매핑 되거나, 일부만 매핑 되고 나머지는 매핑 되지 않을 수 있다. 이와 같이 일부의 노드가 매핑 되지 않아 질의 대상으로 참여할 수 없는 노드를 구별하는 조건문이 지역질의 내에 필요하다.

그림 2에서의 예를 보면, 'guide' 문서의 'address'가 'agency2' 문서의 'location$_1$'과 'location$_2$' 노드와 매핑되고, 전역질의가 그림 4와 같을 경우,

```
<result>
FOR $addr IN document("guide.xml")//address
WHERE $addr/zipcode[CONTAINS(., "MO")]
RETURN $addr
</result>
```

그림 4 Query1 : 'guide' 스키마에 대한 전역질의

'location$_3$' 노드는 질의 대상이 아니므로 질의 생성기는 지역문서에서 질의 대상이 아닌 노드가 존재하는지의 여부를 점검해서, 만약 그러한 노드가 존재한다면 그러한 노드에 대한 경로 지정 술어식 '[ ]'을 사용하여 명백하게 검색 대상이 될 수 없도록 조건을 주어야 한다.

정의 4. l_i와 l_j를 지역경로 트리의 서로 다른 노드 색인 i와 j를 갖는 동일한 이름의 노드라 하자. 만약 l_i와 l_j가 $m^{-1}(l_i)=m^{-1}(l_j)$와 같이 동일한 전역노드와 매핑 된다면, l_i와 l_j는 동질성이라 하며, 집합 $homo(l_i)$ 혹은 $homo(l_j)$와 같이 표현한다. 그렇지 않으면 그들을 서로 상충한다고 하며, l_i와 상충하는 노드 집합을 $conflict(l_i)$라고 표현한다.

그림 2에서, m(address)={location$_1$, location$_2$}인 경우, 각각 $homo$(location$_1$)={location$_1$, location$_2$}, $conflict$(location$_1$)=location$_3$, 그리고 $conflict$(location$_2$)=location$_3$로 나타낸다.

그림 4의 'Query1' 질의에서, m(address)={location$_1$, location$_2$}이므로 술어식 CONTAINS (., "MO")는 'location$_1$'과 'location$_2$' 노드에 적용되고, 'location$_3$'은 질의 대상에서 제외된다. 동질 원소 'location$_1$'과 'location$_2$' 노드를 포함하기 위해서는 이러한 노드를 식별할 수 있는 어떤 특정 조건이 기술되어야 한다.

$ln_i \in l=m(x)$이고 $L_{hc}(ln_i)$를 LT에서 ln_i와 같은 이름을 가지면서 서로 다른 색인을 갖는 노드 집합이라면, $H=homo(ln_i)$이고 $C=conflict(ln_i)$라 할 때, $L_{hc}(ln_i)=H \cup C$ 가 성립된다.

정의 5. 처리중인 질의 내에서 현재 파싱되고 있는 노드 ln_i의 어떤 원소 $h_i \in L_{hc}(ln_i)$에 대해 $childpath(h_i)$는 노드 h_i에서 그 자신의 자식 경로로부터 단말노드까지의 경로 집합으로 정의하고, h_i의 수퍼 집합 $S(h_i)$는 $h_i \in L_{hc}(ln_i)$일 때, $S(h_i)= U_{i=1}^{k} childpath(h_i)$로 정의되며 집합 $L_{hc}(h_i)$ 내에 있는 모든 원소에 대한 모든 자식 경로들의 집합을 의미한다.

그러므로, 각 노드 h_i에 명백하게 적용 가능한 술어식을 쉽게 찾아낼 수 있다. 각 노드 $ln_i \in L_{hc}(ln_i)$에 대해, $p_i \in childpath(ln_i)$이고 $q_i \in (S(ln_i)-childpath(ln_i))$인 원소 p_i와 q_i에 대한 술어식 ((p_1 AND p_2 AND ... AND p_i) AND (NOT(q_1) AND NOT(q_2) ... AND NOT (q_j)))는 오직 노드 ln_j를 제한하기 위한 문장이 된다, 이 문장의 의미는 노드 식별 술어식에서 ln_i는 p_1에서부터 p_i까지의 자식 경로를 갖고, q_1에서 q_j까지의 자식 경로는 갖지 말아야 한다는 뜻이다.

'Query1' 질의 파싱 과정에서 'address' 노드에 도달하면, m(address)={location$_1$, location$_2$}이므로, L_{hc}(location$_1$)=$homo$(location$_1$)$\cup$$conflict$(location$_1$)={location$_1$, location$_2$}$\cup${location$_3$}이 된다. 또한 $childpath$(location$_2$)={street_name, city_name, state_code, zip_code}, $childpath$(location$_1$)=null 그리고 $childpath$(location$_3$)={address, telephone}이므로, S(location$_1$)={street_name, city_name, state_code, zip_code, address, telephone}이 된다. 그러므로 'location$_2$'를 위해서는 술어식 [/street_name AND /city_name AND /state_code AND /zip_code AND NOT(/telephone) AND NOT (/address)]가, 'location$_1$'을 위해서는 술어식 [NOT (/street_name) AND NOT(/city_name) AND NOT (/state_code) AND NOT(/zip_code) AND NOT (/telephone) AND NOT(/address)]가 필요하다.

파싱 중에 상충되는 노드가 발견되면 $homo(q)$ 집합에 있는 노드의 개수만큼 FOR나 LET 절을 생성하고 전역질의의 WHERE 절에 있는 조건절도 각 상충되는 노드에 맞게 생성해야 한다. 지역문서에서 이질적인 노드에 대한 구분을 위한 알고리즘은 아래와 같다 :

Node **I**dentification **P**redicate generation algorithm *NIP(q)* for node *q*
Find out the superset *S(q)*;
For each $q_i \in homo(q)$
  Copy previous FOR/LET clause and concatenate the following statement;
  For each path $p \in childpath(q)$
    Concatenate *'AND /p'*;
  For each path $r \in S(q)-childpath(q)$
    Concatenate *'AND NOT(/r)'*;
EndFor
Return *n* FOR/LET statement where $n = |homo(q)|$

3.4 지역질의 생성 알고리즘(LQG)

사용자에 의해서 입력되어 저장된 매핑정보 및 변환함수 정보, 경로교체 알고리즘 *PS*와 노드 식별을 위한 조건절 생성 알고리즘 *NIP*를 이용하여 다음과 같이 각 지역문서에 대한 지역질의를 생성하는 알고리즘을 제시한다 :

Local **Q**uery **G**eneration algorithm $QG(Q_{in}) = Q_{out}$;
Input : Tokenized global query Q_{in};
Output : the set of *n* local query Q_{out}, where *n* = the number of distributed local documents
for each $i = 1$ to n, q_i = null;
While (token is not null)
  Read the next token of the input global query Q_{in};
  Concatenate token to each q_i;
  if token is node label for *gn*
    then call *PS(gn)*, get a corresponding local node *ln*;
    If $(homo(ln)-ln) \neq \varnothing$ then
      For each element $r_j \in homo(ln)$
        q_{ij} = call $NIP(r_j)$
      q_i = union of q_{ij};
End While
Q_{out} = the set of q_i;

4. 알고리즘의 구현

4.1 시스템 구조

DDXMI를 기반으로 하는 반구조적 분산 문서를 위한 정보검색 시스템의 전체적인 구조는 그림 5와 같다. XML 문서가 올바른 태그 형식을 가진다면, DTD로 데이타 구조를 표현하지 않더라도 모든 문서는 잘 정의되었다고 가정한다. 즉, 어떠한 XML 문서라도 DTD 존재 유무에 관계없이 처리할 수 있다는 것을 의미한다. 전역질의어는 질의어 생성기(Query Generator)를 통해 DDXMI에 저장된 정보를 이용하여 반구조적 분산 문서를 위한 각각의 지역문서 형식에 적합한 지역질의로 자동 변환된다. DDXMI는 저자, 날짜, 주석과 같은 사용자 식별 정보 및 각 지역 파일에 적용되는 경로정보와 각 지역문서에 적용될 함수 정보를 가지고 있다.

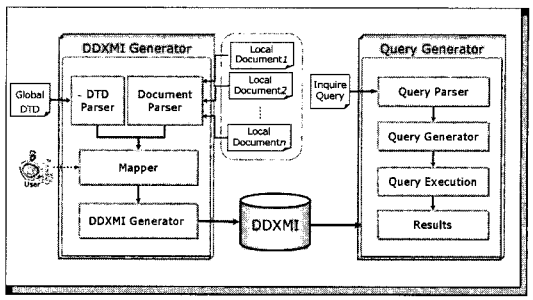


그림 5 분산된 반구조적 문서검색을 위한 시스템 구조

본 시스템은 첫째, 구조적, 의미적으로 상이한 지역문서를 처리하기 위한 DDXMI 생성기(DDXMI Generator)와 둘째, 사용자가 생성한 전역질의어를 DDXMI에 저장된 메타 정보를 이용하여 지역질의어로 생성해주는 질의어 생성기(Query Generator)의 두 부분으로 구성한다.

전역질의어 내에 존재하는 경로정보는 질의어 생성기에 의해 파싱(Query Parser)된 후, 각 요소에 해당되는 매핑 정보가 DDXMI 내에 있다면, 각각의 지역문서 형식에 해당되는 경로정보로 교체 된다. 만일 DDXMI 내에 해당하는 매핑 정보가 없다면, 질의가 생성되지 않으며, 이러한 질의어는 지역문서에 적용되지 않음을 의미한다. 질의 실행기(Query Execution)는 생성된 질의어를 실행하여 각 문서로부터 원하는 모든 답을 얻는다.

4.2 DDXMI 생성기와 질의 실행기 개요

각 문서에 대한 매핑 정보는 각각의 XML 문서 자체를 XML 파서를 이용하거나 혹은 문서의 DTD 파일을 DTD 파서를 이용하여 파싱하므로써 생성한다. 전역 DTD의 각 노드는 전역 DTD 트리의 각 노드에 색인번호를 부여하고, 이 색인번호를 지역경로 트리의 동일한 의미를 가진 노드에 할당함으로써 지역문서의 노드 집합과 매핑한다. 'source'와 동일한 색인번호를 갖는 'target'의 모든 노드를 모아 'source'와 'target' 경로의 집합들로 자동 생성하고, DDXMI는 이러한 경로 집합을 모아서 작성된다. 전역 DTD의 한 요소가 지역문서의 한 요소와 매핑되는 단순한 경우의 노드는 전역 DTD와 동일한 의미를 갖는다.

전역문서와 지역문서의 노드 사이의 매핑을 쉽게 처리하기 위해서 그림 6과 같은 GUI를 제공한다. 그림 6의 좌측 상위 프레임에 'guide.idx'를 루트 이름으로 갖는 동적인 경로 트리를 'guide.DTD'로부터 생성하고, 좌측 하위 프레임의 트리는 지역문서 자체 혹은 문서의 DTD를 파싱하여 생성한다.

우측 상위 프레임은 트리의 각 노드에 대한 색인 번호와 전체 경로를 포함한다. 사용자가 전역노드와 매핑

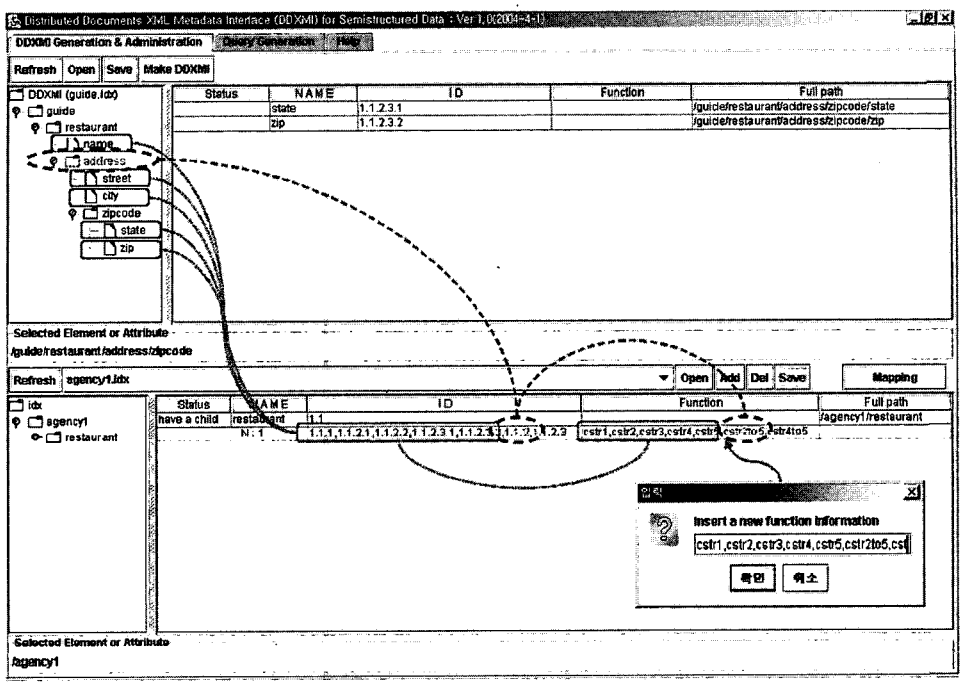


그림 6 전역문서와 지역문서 경로를 매핑하기 위한 GUI 인터페이스

될 지역노드를 선택 한 후 'Mapping' 버튼을 누르면, 연관된 색인 번호가 'ID' 필드에 복사되며, 함수 벡터를 팝업 윈도우에 입력하면, 입력된 함수들은 'Function' 필드에 복사된다. 함수 벡터는 1:1 매핑의 경우에는 노드 값의 이질성을 해결하기 위한 함수를 입력하며, 1:N 매핑의 경우에는 지역경로의 합병 순서를, N:1 매핑의 경우는 지역 노드의 값에서 일부 단어를 추출하기 위한 함수를 입력한다. 모든 연관된 색인 번호를 설정한 후 'Make DDXMI' 버튼을 누르면 DDXMI 파일이 생성된다. 이러한 매핑 과정은 모든 지역문서 파일에 대해 필요하며, 사용자는 수정 및 새로운 매핑 작업을 필요할 때마다 수행할 수 있다.

질의어 생성 및 실행은 그림 7의 화면에서 이루어진다. 생성된 DDXMI를 기반으로 사용자가 전역질의어를 좌측 상단의 'Original query' 프레임에 입력한 후 'Make local queries' 버튼을 누르면 우측 상단에 지역 질의어가 생성되고, 'Execution' 버튼에 의해 모든 지역 문서 혹은 선택된 개별 지역문서를 실행하여 하단의 결과 창에 출력한다.

4.3 매핑과 경로교체에 대한 구현

DDXMI 문서를 위한 DTD는 그림 8과 같다. 전역문서 DTD에 존재하는 요소를 'source' 요소라 하고, 지역문서의 요소는 'target' 요소라 한다. 질의 생성기가 전역질의에서 'source' 요소에 해당하는 요소 이름을 찾아내고 이에 대응하는 지역문서의 'target' 요소가 존재한다면, 질의 경로는 지역질의를 얻기 위한 'target' 요소의 경로로 바뀌게 된다. 'targ'에 기술된 'type' 속성은 매핑 종류를 표현하는 것으로, 0, 1, 2의 값을 가지며 각각 1:1, 1:N 그리고 N:1 매핑을 의미한다. 'operation' 속성의 값은 매핑 형태에 따라 각각 의미정보 함수, 지역경로의 합병 순서, 변환된 값을 얻기 위해 지역 값에 적용되어 'target' 노드로 전달되는 함수를 의미한다. 앞에서 설명된 그림 2의 아래 부분에 이러한 개념을 적용해서 생성된 전역문서인 'guide'와 지역문서인 'agency'를 위한 매핑 정보의 일부가 XML 형식으로 표현되어 있다.

'source' 요소와 'target' 요소의 구성은 '/' 기호부터 시작하여 루트 노드부터 매핑되는 단말 노드까지의 절대적인 경로로 구성된다. 그림 2에서 'street' 노드에 대한 'source' 요소는 '/guide/restaurant/address/street'이고, 그에 해당되는 'target' 요소는 '/agency2/restaurant/location'이다. 따라서 'street' 노드에 대한 매핑 노드는 'street'의 부모 노드인 'address'가 매핑되어 있을 경우에는 'location'이고, 그렇지 않을 경우에는 루트부터 현재 노드까지의 경로에서 가장 가까운 조상에 매핑된 노드와의 차이를 구하여 매핑시킨다. 즉, m(street)=location에 대한 구현은 'street'의 'source' 경로를 찾아서 이에 해당되는 'target' 요소의 경로에서 DDXMI에

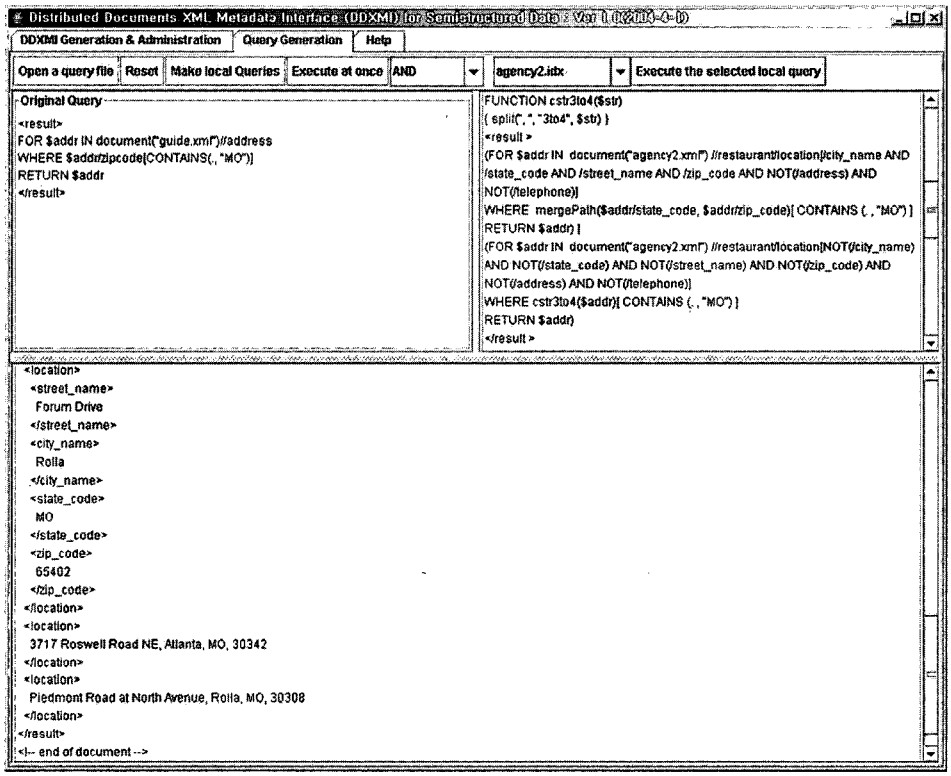


그림 7 질의 생성 및 실행 화면

```
<!ELEMENT DDXMI (DDXMI.header, DDXMI.isequivalent, documentspec)>
<!ELEMENT DDXMI.header (documentation,version,date,authorization)>
<!ELEMENT documentation (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT authorization (#PCDATA)>
<!ELEMENT DDXMI.isequivalent (source,target*)*>
<!ELEMENT source (#PCDATA)>
<!ELEMENT target (targ*)>
<!ELEMENT targ (#PCDATA)>
<!ATTLIST source operation CDATA #IMPLIED>
<!ATTLIST targ
    type CDATA #REQUIRED
    operation CDATA #IMPLIED
>
```

그림 8 DDXMI DTD

있는 'street'의 가장 가까운 조상의 'source'를 찾아 그에 해당되는 'target' 요소의 경로의 차이를 구하여 얻을 수 있으며, *m*(street)=*strdiff*('/agency2/restaurant/location'-'/agency2/restaurant')='location'이 된다.

'street'에 적용되는 매핑의 종류는 'type' 속성의 값이 2이고 'operation' 속성 값이 '*cstr1*'이기 때문에 N:1로 매핑이 되며, 변환함수로 '*cstr1*'을 가지고 있다는 의미이다. 이때 변환함수를 적용해야 하는지에 관한 결정은 현재 파싱되고 있는 노드가 위치경로인지 혹은 조건경로인지를 결정하여 적용한다. 질의 파싱 중에 'street' 노드가 위치경로이면 'location'으로 교체하고, 조건경로에 속하면 '*cstr1*(location)'으로 변환해야 한다. 즉 'location' 노드의 값에서 첫 번째 문자열을 분리하는 함수를 적용하면 'street'의 값을 얻을 수 있다는 의미이다.

1:N의 경우에 대한 매핑 정보는 'target' 요소 안에 N개의 경로 이름을 콤마로 구분하여 모든 경로이름을 입력한다. 그림 2에서 'zipcode' 노드에 대한 'target' 요소로 '/agency2/restaurant/location/state_code', '/agency2/restaurant/location/zip_code' 두개가 입력되어 있고, 'operation' 속성값에 '0,1'이 입력되어 있는데 이것은 첫 번째와 두 번째 문자열 즉, 'state_code'와 'zip_code' 노드의 값을 합병할 때의 합병 순서를 의미하는 문자이다. 이 순서 정보는 DDXMI 생성을 위한 1:N 매핑시 매핑 순서 자체가 문자열을 합병하는 순서가 되지는 않기 때문에 합병 순서를 입력해야 한다. N:1 경우와 마찬가지로 'zipcode'가 위치경로에 속하면 (state_code, zip_code)의 부모 노드에 매핑시키고, 조건경로에 속하면 'zipcode' 노드를 '*mergePath*(state_code, zip_code)'로 변환한다. 위치경로와 조건경로에 대한 구현은 Quilt 파서에서 속성 문법을 이용하여 구현하였다.

4.4 이질적인 노드를 구분하기 위한 조건절의 생성

이 절에서는 3.3절에서 설명한 이질적인 노드를 구분하여 질의 대상에서 제외시키기 위한 조건절을 생성하는 방법에 대하여 설명한다. 어떤 전역노드 ln_i가 파싱될 때, 이 노드가 매핑되는 지역노드의 정보는 DDXMI의 'target' 요소에 포함되어 있다. 그림 2에서 전역노드 'restaurant'이 'agency2'의 'location$_1$', 'location$_2$'와 매핑되지만 'location$_3$'와는 매핑이 되지 않기 때문에 세 번째 'location$_3$' 노드는 검색 대상이 되지 않는다. 즉 'local2' 문서만을 놓고 사용자가 검색할 경우에 'location$_1$', 'location$_2$' 노드는 검색을 하고 'location$_3$' 노드는 검색을 하지 않도록 조건을 주는 문장을 자동으로 생성해주어야 한다. (location$_1$, location$_2$)와 location$_3$ 노드를 각자 구분하기 위해서는 앞에서 정의한 *childpath*를 이용하여 구현한다. 즉 각 노드에 대해 현재 노드부터 단말 노드까지의 경로를 모두 모아서 현재 노드를 구분하도록 조건문을 작성한다. 단, 이때 *childpath*의 논리합이 현재의 노드를 구분할 수 없을 경우에는 문서자체 혹은 매핑이 가지고 있는 고유의 문제이기 때문에 이를 해결할 수 있는 방법은 없다.

어떤 노드가 *homo*(ln_i)에 속하거나 혹은 속하지 않는 것에 관한 정보는 DDXMI에 있으며, 각 노드에 대한 *childpath* 정보는 문서에 대한 구조 정보를 생성하거나, DDXMI에 있는 'target' 요소를 노드마다 분리하여 생성할 수 있다. 본 논문에서는 편리하게 문서의 구조정보를 파싱시에 생성하여 사용하였다. 이러한 *childpath* 정보를 이용하여 현재 파싱되고 있는 노드 ln_i에 대한 수퍼셋 $S(ln_i)$를 구하고 *childpath*(ln_i)에 대해서는 'AND /pathname' 조건을, $S(ln_i)$-*childpath*(ln_i)에 있는 경로에 대해서는 'AND NOT(/pathname)'을 삽입한다.

그림 9의 'Query2'는 그림 4의 전역질의 'Query1'에 대한 지역질의 중 하나로서 'location' 노드를 구분하는 절을 포함한다. *homo*(location)에 대한 노드에 대해서는 *childpath*를 구하여 'AND' 연산자를 사용하였으며, *conflict*(location)에 속한 노드의 *childpath*는 'AND NOT'을 사용하여 각각의 노드가 구분될 수 있게 하였다.

5. 질의 실행 예

5.1 DDXMI 생성

실행의 결과를 얻기 위해 DTD로 정의된 전역문서인 '레스토랑 안내(guide)' 문서와 세 개의 지역문서인 '레스토랑 대행사(agency)' 문서를 생성한다고 가정한다. 전역문서의 DTD와 색인은 그림 10과 같다.

그림 11은 전역문서와 세 개의 색인된 지역문서의 경로 트리를 나타낸다. 전역문서의 각 색인 번호가 지역문서에서 동일한 의미를 갖는 요소에 주어지고, 각 색인 순서는 서로 다르다. 지역문서 내 경로트리의 일부 노드들은 전역 색인이 이러한 경로들을 포함하지 않을 경우 색인 번호를 갖지 않는다. 전역스키마의 'address' 노드는 'Local1'의 경로에서 'address' 노드를, 'Local2'의 경로에서 'location' 노드를 동일한 색인으로 갖지만, 'Local3'의 경로는 그렇지 못하다. 만약 어떤 질의가 'address' 요소를 포함한다면, 'Local3' 경로는 'address'에 상응하는 요소를 갖고 있지 않으므로 'Local3' 문서에 대한 질의는 생성되지 않는다. 그림 11과 같은 색인 할당을 기반으로 동일한 번호를 갖는 경로들을 같은 경로 집합으로 모아 DDXMI 파일을 자동 생성한다. 이 DDXMI 파일은 사용자가 쉽게 변경이나 수정할 수 있으며, 색인 번호를 재할당시 쉽게 재생성할 수 있다. 지역문서에서의 색인 파일은 2장에서 언급한 매핑 형태를 지칭하는 '**0**(1:1)', '**1**(1:N)', '**2**(N:1)'의 값을 갖는 'type' 필드와 전역문서와의 이질성을 해결하기 위한 함수 값을 갖는 'operation' 필드가 필요하다.

```
import mergePath as UDF_merge;
import split as USER_split;
FUNCTION cstr3to4($str)
{ split(", ", "3to4", $str) }
<result >
 (FOR $addr IN document("agency2.xml") //restaurant/location[/street_name AND /city_name
     AND /state_code AND /zip_code AND NOT(/telephone) AND NOT(/address)]
 WHERE mergePath($addr/state_code, $addr/zip_code)[ CONTAINS (. , "MO") ]
 RETURN $addr) UNION
 (FOR $addr IN document("agency2.xml") //restaurant/location[NOT(/street_name)] AND
     NOT(/city_name) AND NOT(/state_code) AND NOT(/zip_code) AND NOT(/telephone)
     AND NOT(/address)]
 WHERE cstr3to4($addr)[ CONTAINS (. , "MO") ]
 RETURN $addr)
</result >
```

그림 9 Query2 : 전역질의 Query1으로 부터 생성된 지역질의

<!ELEMENT guide (restaurant+)>	0	guide.xml
<!ELEMENT restaurant (name, address*)*>	1	/guide
<!ELEMENT address (street, city, zipcode)*>	1.1	/guide/restaurant
<!ELEMENT zipcode (state, zip)*>	1.1.1	/guide/restaurant/name
<!ELEMENT name EMPTY>	1.1.2	/guide/restaurant/address
<!ELEMENT street EMPTY>	1.1.2.1	/guide/restaurant/address/street
<!ELEMENT city EMPTY>	1.1.2.2	/guide/restaurant/address/city
<!ELEMENT state EMPTY>	1.1.2.3	/guide/restaurant/address/zipcode
<!ELEMENT zip EMPTY>	1.1.2.3.1	/guide/restaurant/address/zipcode/state
	1.1.2.3.2	/guide/restaurant/address/zipcode/zip

그림 10 전역문서의 DTD 및 색인된 경로 스키마

Index	Global Path	Local1 Path	Local2 Path	Local3 Path
		agency1.xml	agency2.xml	agency3.xml
0	guide.xml	0 /agency1	0 /agency2	0 /agency3
1	/guide	0 /agency1/restaurant	0 /agency2/restaurant	0 /agency3/restaurant
1.1	/guide/restaurant	0 /agency1/restaurant/name	0 /agency2/restaurant	0 /agency3/restaurant/name
1.1.1	/guide/restaurant/name	2 /agency1/restaurant		2 /agency3/restaurant
1.1.2	/guide/restaurant/address	0 /agency1/restaurant/address	0 /agency2/restaurant/location	1 /agency3/restaurant/streetname, /agency3/restaurant/cityname, /agency3/restaurant/statecode, /agency3/restaurant/zipcode
		2 /agency1/restaurant		2 /agency3/restaurant
1.1.2.1	/guide/restaurant/address/street	0 /agency1/restaurant/address/street	0 /agency2/restaurant/location/street_name	0 /agency3/restaurant/streetname
		2 /agency1/restaurant	2 /agency2/restaurant/location	2 /agency3/restaurant
1.1.2.2	/guide/restaurant/address/city	0 /agency1/restaurant/address/city	0 /agency2/restaurant/location/city_name	0 /agency3/restaurant/cityname
		2 /agency1/restaurant	2 /agency2/restaurant/location	2 /agency3/restaurant
1.1.2.3	/guide/restaurant/address/zipcode	0 /agency1/restaurant/address/zipcode	1 /agency2/restaurant/location/state_code, /agency2/restaurant/location/zip_code	1 /agency3/restaurant/statecode, /agency3/restaurant/zipcode
		2 /agency1/restaurant	2 /agency2/restaurant/location	2 /agency3/restaurant
1.1.2.3.1	/guide/restaurant/address/zipcode/state	0 /agency1/restaurant/address/zipcode/state	0 /agency2/restaurant/location/state_code	0 /agency3/restaurant/statecode
		2 /agency1/restaurant	2 /agency2/restaurant/location	2 /agency3/restaurant
1.1.2.3.2	/guide/restaurant/address/zipcode/zip	0 /agency1/restaurant/address/zipcode/zip	0 /agency2/restaurant/location/zip_code	0 /agency3/restaurant/zipcode
		2 /agency1/restaurant	2 /agency2/restaurant/location	2 /agency3/restaurant

그림 11 전역스키마와 세 개의 지역스키마 Local1, Local2, Local3 사이의 색인 할당

5.2 질의 생성 및 실행 예

본 논문에서 구현을 위해 사용한 XML 질의 언어인 Quilt 질의를 실행하기 위한 질의 엔진으로는 University of Pennsylvania에서 개발한 Kweelt[20]를 사용하였다. 사용자가 그림 7의 질의 실행 화면에서 질의 이름을 입력하고 질의생성 버튼을 누르면 지역질의가 문서의 개수만큼 생성된다. 실험의 결과를 얻기 위해 발생 가능한 모든 경우의 이구조적이고 이질적인 반구조 분산 XML 문서를 OEM 모델로 제작하였으며, 제작된 세 개의 문서를 파싱하여 그림 12의 local1, local2, local3와 같은 지역스키마를 생성하였다. 전역스키마와 지역스키마 간의 동시 다중 매핑이 그림 12에, 매핑에 대한 DDXMI의 일부가 그림 13에 나타나 있다. 그림 14는 전역질의에 대해 생성된 세 개의 지역질의를, 그림 15는 생성된 지역질의를 실행한 결과를 보여준다.

6. 결 론

본 논문에서는 분산된 문서에 대한 구조적 정보가 제공되지 않고, 자료 구조에 대한 엄격한 형식이 없는 반구조적 문서 환경에서의 의미적, 구조적인 불일치 문제를 해결하기 위한 질의 처리 시스템을 매핑, 경로교체, 노드 식별 처리 개념을 기반으로 모델화하고 구현하였다.

반구조적인 문서로부터 사용자가 원하는 형태의 검색 결과를 얻기 위한 시스템으로 LORE[8], YAT[9], TSIMMIS[10,11] 등이 제안되고 있으나, 매핑 방법과 변환이 어렵고, 전역요소에 대한 지역요소의 매핑시 1:1이거나, 1:N이거나, 혹은 N:1인 경우의 선택적 매핑만을 지원한다. 따라서 동일한 의미이지만 구조가 다른 여러

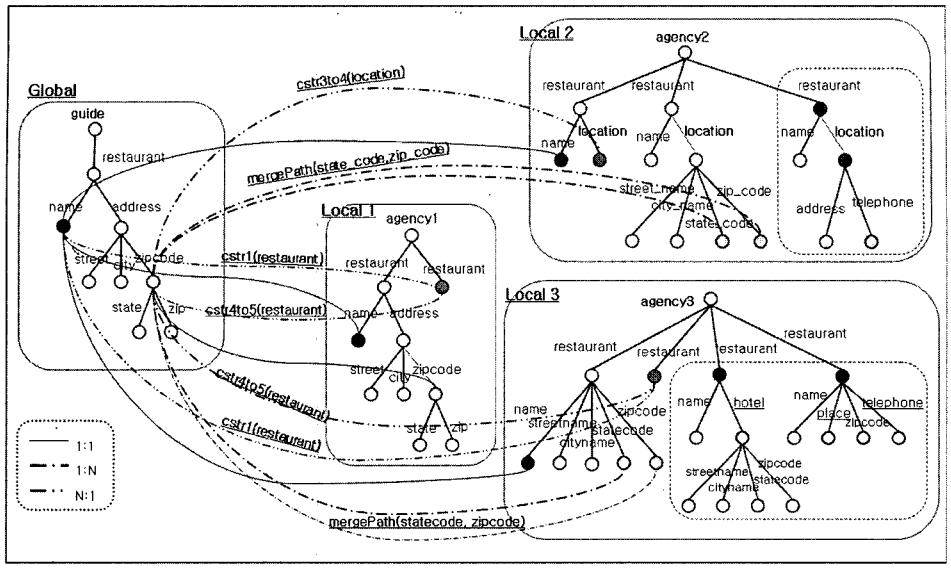


그림 12 전역스키마와 세 개의 지역문서 사이의 매핑

```
<source operation = "">/guide/restaurant/name</source>
    <target>
        <targ type = "0" operation = "">/agency1/restaurant/name</targ>
        <targ type = "2" operation = "cstr1">/agency1/restaurant</targ>
    </target>
    <target>
        <targ type = "0" operation = "">/agency2/restaurant/name</targ>
    </target>
    <target>
        <targ type = "0" operation = "">/agency3/restaurant/name</targ>
        <targ type = "2" operation = "cstr1">/agency3/restaurant</targ>
    </target>
<source operation = "">/guide/restaurant/address/zipcode</source>
    <target>
        <targ type = "2" operation = " cstr3to4 ">/agency1/restaurant</targ>
        <targ type = "0" operation = "">/agency1/restaurant/address/zipcode</targ>
    </target>
    <target>
        <targ type = "2" operation = "cstr3to4">/agency2/restaurant/location</targ>
        <targ type = "1" operation = "0,1">/agency2/restaurant/location/state_code,
                                    /agency2/restaurant/location/zip_code</targ>
    </target>
    <target>
        <targ type = "2" operation = "cstr4to5">/agency3/restaurant</targ>
        <targ type = "1" operation = "0,1">/agency3/restaurant/statecode,
                                    /agency3/restaurant/zipcode</targ>
    </target>
```

그림 13 그림 12의 전역스키마와 세 개의 지역스키마에 대한 DDXMI

요소가 지역문서 상에서 동시에 발생하는 경우에 대해 동시 다중 매핑을 지원하지 못하기 때문에 매핑되지 못한 요소는 검색 대상에서 제외되는 문제점이 있다. 또한 반구조적 문서를 위한 여러 질의 언어로 MSL[12,13], Lorel[8,14], UnQL[15,16], YATL[9] 등이 소개되고 있으나, 대부분 시스템에 종속적이어서 범용성을 갖지 못한다.

본 논문에서는 이러한 문제점을 해결하여 이질적이고 이구조적인 여러 노드나 요소를 동시에 중복해서 매핑할 수 있는 동시 다중 매핑 방식을 지원한다. 불규칙적으로 반구조적인 어떠한 문서에 대해서도 정확한 검색 결과를 얻을 수 있도록 알고리즘을 설계하고 구현 방법을 제시하였으며, 구현된 시스템을 실험하여 결과를 얻었다. XML 질의 언어로는 W3C에서 채택되어 모든 XML 문서에 적용 가능한 Quilt[1]를 사용하였다.

문서에 대한 경로 트리는 DTD 파서가 DTD를 파싱하거나 XML 파서가 XML 문서를 파싱하여 자동으로 생성한다. 통합검색 사용자의 편리성을 도모하기 위해서

Global Query

```
<result>
FOR $addr IN document("guide.xml")//address
WHERE $addr/zipcode[CONTAINS(., "MO")]
RETURN $addr
</result>
```

Generated Query — Local 1

```
import split as USER_split;
FUNCTION cstr2to5($str)
{ split(", ", "2to5", $str) }
FUNCTION cstr4to5($str)
{ split(", ", "4to5", $str) }
<result >
{FOR $addr IN document("agency1.xml") //address
WHERE $addr/zipcode[ CONTAINS (. , "MO") ]
RETURN $addr) }
{FOR $addr IN document("agency1.xml") //restaurant[ NOT(/address) AND NOT(/name)]
WHERE cstr4to5($addr)[ CONTAINS (. , "MO") ]
RETURN $addr)
</result >
```

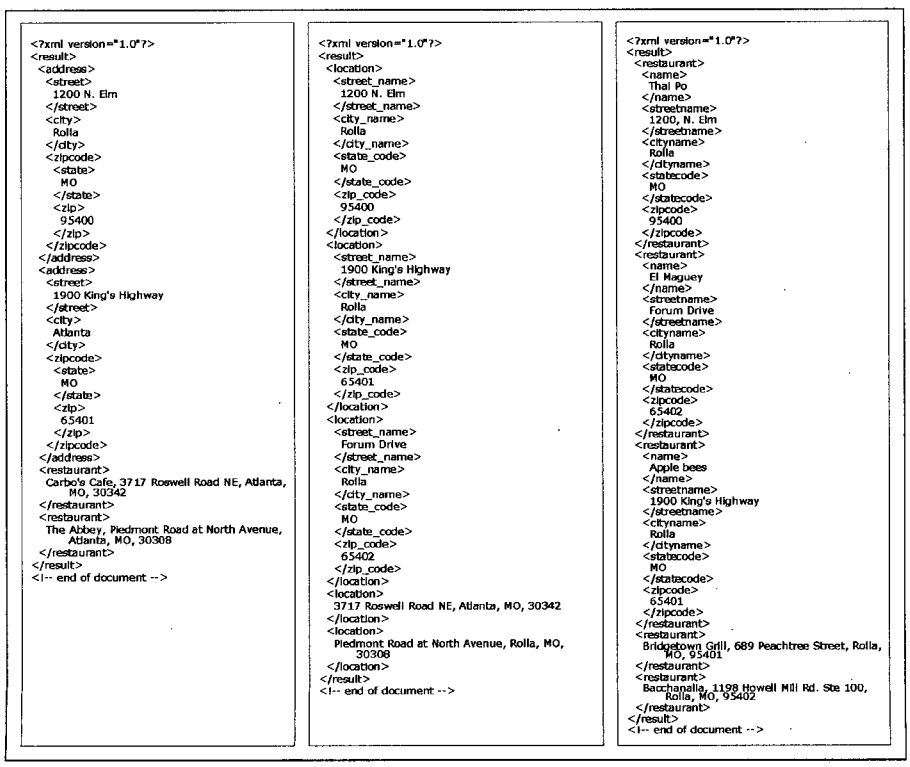
Local 2

```
import mergePath as UDF_merge;
import split as USER_split;
FUNCTION cstr3to4($str)
{ split(", ", "3to4", $str) }
<result >
{FOR $addr IN document("agency2.xml") //restaurant/location[/city_name AND /state_code AND /street_name AND
      /zip_code AND NOT(/address) AND NOT(/telephone)]
WHERE mergePath($addr/state_code, $addr/zip_code)[ CONTAINS (. , "MO") ]
RETURN $addr) }
{FOR $addr IN document("agency2.xml") //restaurant/location[NOT(/city_name) AND NOT(/state_code) AND
      NOT(/street_name) AND NOT(/zip_code) AND NOT(/address) AND NOT(/telephone)]
WHERE cstr3to4($addr)[ CONTAINS (. , "MO") ]
RETURN $addr)
</result >
```

Local 3

```
import mergePath as UDF_merge;
import split as USER_split;
FUNCTION cstr2to5($str)
{ split(", ", "2to5", $str) }
FUNCTION cstr4to5($str)
{ split(", ", "4to5", $str) }
<result >
{FOR $addr IN document("agency3.xml") //restaurant[/cityname AND /name AND /statecode AND /streetname AND
      /zipcode AND NOT(/hotel) AND NOT(/hotel/cityname) AND NOT(/hotel/statecode) AND NOT(/hotel/streetname) AND
      NOT(/hotel/zipcode) AND NOT(/place) AND NOT(/telephone)]
WHERE mergePath($addr/statecode, $addr/zipcode)[ CONTAINS (. , "MO") ]
RETURN $addr) }
{FOR $addr IN document("agency3.xml") //restaurant[NOT(/cityname) AND NOT(/name) AND NOT(/statecode) AND
      NOT(/streetname) AND NOT(/zipcode) AND NOT(/hotel) AND NOT(/hotel/cityname) AND NOT(/hotel/statecode) AND
      NOT(/hotel/streetname) AND NOT(/hotel/zipcode) AND NOT(/place) AND NOT(/telephone)]
WHERE cstr4to5($addr)[ CONTAINS (. , "MO") ]
RETURN $addr)
</result >
```

그림 14 전역질의에 대한 세 개의 지역문서의 질의 생성 결과



```
<?xml version="1.0"?>
<result>
 <address>
  <street>
   1200 N. Elm
  </street>
  <city>
   Rolla
  </city>
  <zipcode>
   <state>
    MO
   </state>
   <zip>
    95400
   </zip>
  </zipcode>
 </address>
 <address>
  <street>
   1900 King's Highway
  </street>
  <city>
   Atlanta
  </city>
  <zipcode>
   <state>
    MO
   </state>
   <zip>
    65401
   </zip>
  </zipcode>
 </address>
 <restaurant>
  Carbo's Cafe, 3717 Roswell Road NE, Atlanta,
      MO, 30342
 </restaurant>
 <restaurant>
  The Abbey, Piedmont Road at North Avenue,
      Atlanta, MO, 30308
 </restaurant>
</result>
<!-- end of document -->
```

```
<?xml version="1.0"?>
<result>
 <location>
  <street_name>
   1200 N. Elm
  </street_name>
  <city_name>
   Rolla
  </city_name>
  <state_code>
   MO
  </state_code>
  <zip_code>
   95400
  </zip_code>
 </location>
 <location>
  <street_name>
   1900 King's Highway
  </street_name>
  <city_name>
   Rolla
  </city_name>
  <state_code>
   MO
  </state_code>
  <zip_code>
   65401
  </zip_code>
 </location>
 <location>
  <street_name>
   Forum Drive
  </street_name>
  <city_name>
   Rolla
  </city_name>
  <state_code>
   MO
  </state_code>
  <zip_code>
   65402
  </zip_code>
 </location>
 <location>
  3717 Roswell Road NE, Atlanta, MO, 30342
 </location>
 <location>
  Piedmont Road at North Avenue, Rolla, MO,
      30308
 </location>
</result>
<!-- end of document -->
```

```
<?xml version="1.0"?>
<result>
 <restaurant>
  <name>
   Thai Po
  </name>
  <streetname>
   1200, N. Elm
  </streetname>
  <cityname>
   Rolla
  </cityname>
  <statecode>
   MO
  </statecode>
  <zipcode>
   95400
  </zipcode>
 </restaurant>
 <restaurant>
  <name>
   El Maguey
  </name>
  <streetname>
   Forum Drive
  </streetname>
  <cityname>
   Rolla
  </cityname>
  <statecode>
   MO
  </statecode>
  <zipcode>
   65402
  </zipcode>
 </restaurant>
 <restaurant>
  <name>
   Apple bees
  </name>
  <streetname>
   1900 King's Highway
  </streetname>
  <cityname>
   Rolla
  </cityname>
  <statecode>
   MO
  </statecode>
  <zipcode>
   65401
  </zipcode>
 </restaurant>
 <restaurant>
  Bridgetown Grill, 689 Peachtree Street, Rolla,
      MO, 95401
 </restaurant>
 <restaurant>
  Bacchanalia, 1198 Howell Mill Rd. Ste 100,
      Rolla, MO, 95402
 </restaurant>
</result>
<!-- end of document -->
```

그림 15 세 개의 레스토랑 대행사 문서에 대한 질의 실행 결과

트리 내의 노드를 클릭하여 동일한 의미를 갖는 노드를 찾고, 표현상의 이질성을 해결하기 위해 필요한 함수 이름을 부여할 수 있도록 하였다. DDXMI 파일을 생성하는 과정은 전역경로에 대해 동일한 색인번호를 갖는 지역경로를 하나의 셋으로 모아 DDXMI 파일을 생성한다. 매핑정보를 바탕으로 질의는 질의생성기에 의해 DDXMI에 존재하는 대응 경로를 찾아 입력된 의미정보 함수 및 변환함수를 적용하여 지역문서에 대한 지역질의를 생성한다. 이렇게 생성된 지역질의는 질의 실행 엔진에 의해 지역문서로부터의 결과가 처리되어 통합된 후 하나의 결과로 사용자에게 출력된다.

본 시스템은 XML 문서를 파싱하기 위해 SAX 파서를 사용하였으며, Java와 JavaCC 컴파일러를 이용하여 윈도우즈 환경에서 구현되었다. XML로 표현 가능한 어떠한 반구조적 문서라도 변경이나 재구성 없이 본 시스템에 사용될 수 있다. 이 시스템을 이용하려는 사용자를 위한 한 가지 요구 조건은 경로정보와 관련된 문서의 내용을 잘 파악해야 한다.

현재의 매핑 개념은 전역경로와 지역경로간의 동치 매핑을 허용하고 있으나, 향후 연구과제로 지역문서의 완전한 사용을 위해 지역경로 대 지역경로, 문서 대 문서, 문서 대 경로와 같은 좀더 복잡한 종류의 관계를 허용하고 표현할 수 있는 강력한 매핑 개념을 확장하는 연구가 필요하다. 또한 현재의 모형은 가족 트리를 위한 문서나 부가적인 요소를 포함하고 있는 요소와, 순환 요소를 포함하는 문서의 DTD에서 발생하는 와일드카드를 포함하는 경로를 지원하지 못하기 때문에 이를 위한 연구가 필요하다.

참 고 문 헌

[1] Don Chamberlin, Jonathan Robie, Daniela Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. Proceedings of WebDB 2000 Conference, in Lecture Notes in Computer Science, Springer-Verlag, 2000.

[2] Antonio Badia, Sanjay Kumer Madria. Handling Partial Matches in Semistructured Data with Cooperative Query Answering Techniques, Confederated International Conferences DOA, CoopIS and ODBASE, Pages:449-467, 2002.

[3] Dan Suciu. Distributed Query Evaluation on Semistructured Data, ACM Transactions on Database Systems, Vol. 27, No. 1, Pages:1-62, March 2002.

[4] Dan Suciu. Semistructured Data and XML. Information organization and databases, 2000.

[5] Peter Buneman. Tutorial: Semistructured data. In Proceedings of PODs, 1997.

[6] Serge Abiteboul. Querying semistructured data. In Proceedings of ICDT, 1997.

[7] Jason McHugh, Jennifer Widom, Serge Abiteboul, Qinghan Luo, Anand Rajaraman. Indexing Semistructured Data, Technical Report, Stanford University, 1998.

[8] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, Jennifer Widom. Lore: A database management systems for semistructured data. SIGMOD Record, 26, 1997.

[9] Sophie Cluet, Claude Delobel, Jérôme Siméon, Katarzyna Smaga. Your Mediators Need Data Conversion! In Proceedings ACM-SIGMOD International Conference on Management of Data, pages:177-188, 1998.

[10] Yannis Papakonstantinou. Query Processing in Heterogeneous Information Sources, Technical Report, Stanford University Thesis, 1996.

[11] Yannis Papakonstantinou, Hector Garcia-Molina, Jeniffer Widom. Object exchange across heterogeneous information sources. In Proceedings of the 11th ICDE, 1995.

[12] Chaitanya Baru, Amarnath Gupta, Bertram Ludäscher, Richard Marciano, Yannis Papakonstantinou, Pavel Velikhov, Vincent Chu. XML-Based Information Mediation with MIX. Exhibition program, ACM Conf. on Management of Data, SIGMOD'99, Philadelphia, 1999.

[13] Yannis Papakonstantinou, Hector Garcia-Molina, Jeffrey Ullman. MedMaker: A Mediation System Based on Declarative Specifications. Data Engineering(ICDE), 1996.

[14] Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey Ullman, Jennifer Widom. Querying Semistructured Heterogeneous Information, Proceedings of the Fourth International Conference on Deductive and Object-Oriented Databases, pages:319-344, December 04-07, 1995.

[15] Peter Buneman, Mary Fernandez, Dan Suiciu. UnQL: A Query Language and Algebra for Semistructured Data Based on Structural Recursion, VLDB Journal manuscript, 2000.

[16] Peter Buneman, Susan Davidson, Gerd Hillebrand, Dan Suciu. A Query Language and Optimization Techniques for Unstructured Data. In Proceedings of ACM-SIGMOD International Conference on Management of Data, pages:505-516, 1996.

[17] Svetlozar Nestorov, Serge Abiteboul, Rajeev Motwani. Inferring Structure in Semistructured Data. In Proceedings of the Workshop on Management of Semistructured Data, 1997.

[18] Yannis Papakonstantinou, Serge Abiteboul, Hector Garcia-Molina. Object Fusion in Mediator Systems, In Proceedings of Very Large Data Bases, pages:413-424, September 1996.

[19] XPath(XML Path Language), http://www.w3.org/TR/xpath

[20] Arnaud Sahuguet. Kweelt: More than just "yet another framework to query XML!," Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, 2001.

[21] Young-Kwang Nam, Joseph Goguen, Guilian Wang. A Metadata Integration Assistant Generator for Heterogeneous Distributed Databases, by, in Proceedings, International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems, Springer, Lecture Notes in Computer Science, Volume 2519, pages:1332-1344, 2002, from a conference held in Irvine CA, 29-31, October 2002.

최 귀 자

1991년 서울산업대학교 졸업(학사). 2005년 연세대학교 대학원 전산학과(석사) 2005년~현재 연세대학교 대학원 전산학과 박사과정. 관심분야는 XML, 메타데이타, 프로그래밍언어, 소프트웨어공학, 정보검색

남 영 광

1978년 연세대학교 수학과 졸업(학사) 1985년 한국과학기술원 전산학과 졸업(석사). 1992년 Northwestern University 전산학과 졸업(박사). 1993년~1994년 시스템공학연구소 선임연구원. 1995년~현재 연세대학교 전산학과 교수. 관심분야는 프로그래밍언어, 소프트웨어공학, 정보검색, XML