웹 정보 통합 및 검색을 위한 XML기반 미디에이터 시스템의 개발

양정욱, 홍동완, 이덕형, 윤지희
한림대학교 컴퓨터공학과

Development of an XML-based Mediator System for Web Information Integration and Retrieval

Jeong-Uk Yang · Dong-Wan Hong · Duck-Hyung Lee · Jee-Hee Yoon
Dept. of Computer Engineering, Hallym Univ.

요 약

HMS(Hallym Mediator System)는 XML을 기본 데이터 모델로 하여 인터넷에 산재하여 있는 분산이질 정보에 대한 통합, 검색기능을 제공하는 미디에이터 시스템이다. 분산이질 정보의 공통 스키마 구조로서 XML DTD를 사용하며, 각종 정보에 대한 가상의 통합 뷰(view) 생성기능을 제공하여 웹 상의 통합된 가상정보 구조를 표현한다. 웹 상의 일반 사용자는 이와 같이 생성된 뷰 DTD를 근거로 분산이질 정보에 대한 구조적, 내용적 질의를 수행할 수 있다. HMS는 가상접근 기법(virtual approach) 기반의 정보검색 시스템으로서, 사용자 질의는 XML 형태의 소스 매핑정보를 이용하여 각 소스에 대한 부질의로 변환되며, 각 소스로부터의 검색결과는 시스템에 의하여 통합되어 XML 문서 형태로 돌아오게 된다. HMS에서는 DTD 구동형의 비쥬얼 사용자 인터페이스를 제공하여, 관리자와 일반 사용자에게 직관적이고 간편한 가상정보 구축 및 질의검색 환경을 제공한다.

1. 서론

괄목할 만한 인터넷의 성장은 기하 급수적으로 인터넷 사용 인구의 증가를 불러 왔다. 이에 따라 정보의 양도 증가하였으며, 그 정보의 형태도 동영상, 일반 텍스트, 이미지, 음성, RTF(Rich Text Format) 등으로 다양하며, 내용적으로도 같은 혹은 유사한 정보가 인터넷의 여러 곳에 중복, 산재되어 위치할 수 있다. 이 들 정보들은 대부분 유용한 정보로서 공유되어져야 하며, 이러한 공유에 의하여 새로운 데이터의 창출을 가져올 수 있다. 따라서 정보의 통합은 필수적인 요건으로 대두되고 있다.

최근 분산 이질 환경의 데이터를 통합하기 위한 방법으로서 미디에이터[Wiederhold 1992] 개념이 널리 활용되고 있으며, 미디에이터 시스템에서 데이터 통합을 위한 기본모델로 XML(eXtensible Makrup Language)[Bracy외 2인 1998]을 사용할 수 있다[Baru외 6인

1999]. XML은 HTML(Hyper Text Markup Language)과 같이 간단하고 이해하기 쉽다는 특성을 가지며 동시에 구조 정보를 효율적으로 표현할 수 있어, 차세대 인터넷상의 전자문서 표준으로 인정받고 있다.

본 논문에서는 분산이질 환경에서 웹 정보의 통합 및 검색이 가능한 HMS 미디에이터 시스템 개발에 대하여 논한다. HMS에서는 웹 상에 존재하는 분산이질 정보의 통합된 가상 정보구조를 표현하기 위하여 XML을 공통 데이터 모델로 이용하며, 공통 스키마 구조로서 XML의 DTD 구조를 사용한다.

시스템 기능은 크게 두 가지로서, 관리자에 의한 통합 뷰 생성 및 관리 기능과 통합 뷰를 기반으로 한 일반 사용자의 정보 검색 기능으로 나눌 수 있다. 관리자는 각종 응용 목적에 적합하도록 여러 개의 소스 DTD로부터 통합정보 구조를 표현하는 뷰 DTD를 선언적으로 정의할 수 있으며, 시스템은 뷰 DTD 정의에 근거하여 추론 과정에 의하여 정확한 최소의 의미를 표현하는 뷰 DTD를 자동 생성한다. 이와 같이 생성된 뷰 DTD는 관리자와 보완 과정을 거쳐, 최종적으로 XML 저장소에 저장되며, 시스템은 이 때 소스 정보와 뷰 DTD 간의 매핑 정보를 나타내는 소스 매핑 정보(Source Mapping Information)를 자동 생성 관리한다. 웹 상의 사용자는 이들 DTD를 웹 상의 정보 구조로 인식하여, 분산이질 정보에 대한 구조적, 내용적 질의를 수행할 수 있다. 사용자 질의는 XML 형태의 소스 매핑정보에 의하여 각 소스에 대한 부질의로 변환되며, 각 소스로부터의 검색결과는 시스템에 의하여 통합되어 XML 문서 형태로 돌아오게 된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대하여 간단히 설명하고, 시스템 구성 및 질의 언어, HMS에 의한 정보 통합 및 질의 평가방식 등에 대하여 개괄적으로 설명한다. 3장에서는 각 시스템 구성요소의 기능, 구현 방식에 대하여 설명하고, 예를 이용하여 관리자와 일반 사용자의 사용자 인터페이스 기능에 대하여 설명한다. 마지막으로 4장에서는 시스템의 실용성 및 차후 연구과제에 대하여 논한다.

2. HMS (Hallym Mediator System)

2.1 관련연구

기존의 이질적인 데이터베이스 시스템의 통합 방법[Florescu외 2인 1998]으로서, 연합 데이터베이스(Federated Database), 멀티 데이터베이스(Multi Database) 방식 등이 잘 알려져 있다. 웹 상의 정보 통합을 위하여는 이와 같은 이질적인 데이터의 통합이라는 문제 외에도 정보 소스의 수가 매우 많고, 계속 변화하며, 각 정보소스에 대한 메타정보가 부족하고, 각 정보 소스가 상당한 자치성을 갖고 있다는 등의 부가적인 특성이 고려되어야 한다.

최근 웹 정보 통합을 위한 시스템 구성 방식으로서, 데이터 웨어하우징(Data Warehousing)방식과 가상 접근(Virtual Approach) 기법에 관한 연구[Ludascher외 2인 2000]가 활발히 이루어지고 있다. 데이터 웨어하우징 기법은 웹 소스로부터의 데이터를 시스템으로 가져와서 통합, 저장 한 후에, 모든 질의를 웨어하우스에 대하여 적용하는 시스템 구현 방식으로서, Lore(Lightweight Object REpositiory) 시스템[McHugh외 4인 1997] 등을 예로 들 수 있다. 가상 접근기법은 모든 정보가 분산 이질 형태로 웹 상에 그대로 존재한 상황에서, 가상 통합 뷰에 대한 질의가 실행시 각 소스에 대한 질의로 분해되어 평가되는 시스템 구현 방식으로 MIX(Mediation of Information using XML) 시스템[Baru외 6인

1999] 등을 예로 들 수 있다. 일반적으로 데이터 소스의 수가 많고 변화가 빈번하며, 소스에 대한 제어가 어려운 데이터 통합의 경우에는, 가상 접근 기법이 웨어하우징 기법보다 적합한 시스템 구현 형태로 알려져 있다. 그러나 가상접근 기법은 웨어하우징기법에 비하여 효율면에 문제가 있을 수 있으므로 질의 평가를 위한 최적화 작업[Florescu외 2인 1998]을 위한 연구가 수반되어야 하며, 실용화를 위한 다각적인 연구가 필요하다.

2.2 시스템 개요

HMS(Hallym Mediator System)는 가상 접근기법 기반의 웹 정보통합/검색 시스템으로서, 그림 1과 같이 사용자 인터페이스, 미디에이터, 래퍼를 기본 구조로 한다. 사용자 인터페이스는 통합 뷰 생성을 위한 관리자용 인터페이스와 정보검색을 위한 일반 사용자 인터페이스로 이루어져 있다. 미디에이터는 뷰 정의/질의 파싱모듈, DTD 추론 모듈, 질의 실행 모듈, 결과(XML문서와 DTD) 생성 모듈, 소스 매핑정보 처리 모듈 등으로 이루어져 있으며, 래퍼는 인터넷 상의 정보 매핑을 위하여 소스 정보별로 작성된다.

시스템의 기본 기능은 다음과 같이 통합 뷰 생성 및 관리 기능과 정보 검색 기능으로 나눌 수 있으며, HMS에서는 뷰 정의와 정보 검색을 위하여 질의어 HML(Hallym Mediator Language)을 제공하고 있다.

(1) 통합 뷰의 생성 및 관리

미디에이터 관리자는 웹 상의 각종 소스에 산재되어 있는 정보를 통합, 가공하여 응용 목적에 적합한 가상의 통합 뷰를 정의, 구축하여야 한다. HMS에서는 웹 상의 정보 소스에 대한 각종 정보(소스 내용, 속성 정보, 제약 조건, 소스 신뢰도, 질의처리 능력 등)를 메타 데이터 형태로 수집, 저장, 관리한다. 관리자는 이들 메타 데이터를 기반으로 사용자 인터페이스에서 제공되는 소스 DTD 브라우징 기능, 뷰 DTD 편집 기능 등을 이용하여 통합 뷰를 정의한다. 작성된 뷰 정의 질의는 HML View Definition Language 파서에 의해 파싱이 되고, 추론 계획이 수립된다. DTD 추론 모듈은 수립된 계획에 의하여 대수적인 방식에 의하여 소스 DTD 들로부터 뷰 정의 질의 의미에 정확한 최소의 뷰 DTD를 자동 생성한다. 통합 뷰를 생성하는 과정에서 웹 상의 각종 소스와 통합 뷰와의 매핑 정보가 자동 추출되며, 이 소스 매핑 정보는 소스 매핑 모듈에 저장 관리된다. 생성된 뷰 DTD와 소스 매핑 정보는 관리자에 의하여 다시 보완될 수 있으며, 이들은 최종적으로 XML 저장소에 저장된다.

(2) 정보 검색

사용자는 일반 사용자 인터페이스를 이용하여 응용 목적에 적합한 뷰를 구동시킨 후, 분산 이질정보에 대한 구조 정보 및 내용 정보에 대한 질의를 수행할 수 있다. 뷰 DTD를 근거로 작성된 사용자의 질의는 HML Query Parser로 보내진다. 입력된 HML 질의는 소스매핑 정보 참조에 의하여 소스 정보에 대한 HML 질의로 변환되고, 각종 정보 소스에 대한 메타 데이터를 이용하여 질의의 효율적 평가를 위한 실행 전략이 수립된다. 실행엔진은 실행 전략에 의하여 정보가 산재하여 있는 정보 소스로부터 래퍼(Wrapper)를 통하여 정보를 검색, 추출한다. 추출된 데이터는 결과 생성 모듈로 보내져 사용자 질의의 생성절에 적합한 결과 문서와 DTD가 생성된다. 이 결과문서와 DTD는 사용자에게 질의 결과로 전송된다.

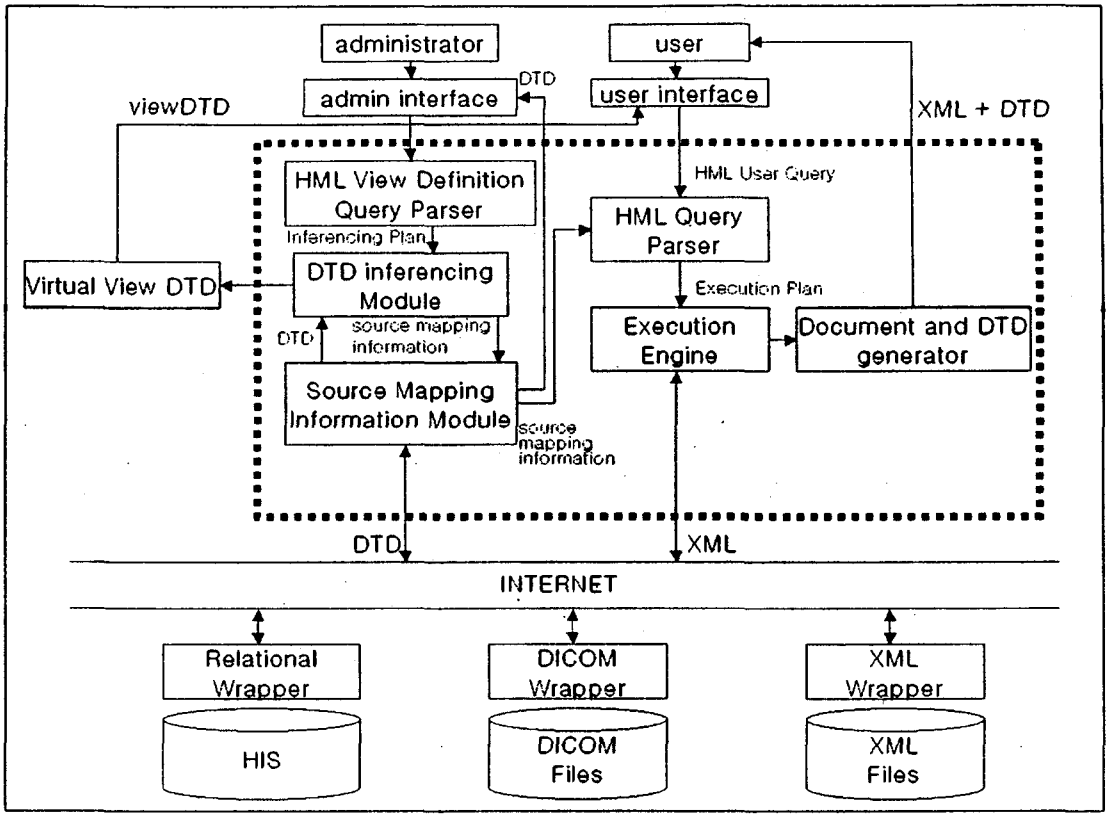

그림 1. HMS 구조도

2.3 HML(Hallym Mediator Language)

HMS에서는 뷰 정의와 정보 검색을 위한 미디에이터 언어 HML(Hallym Mediator Language)을 제공한다. HML은 XML-QL[Deutsch외 4인 1999], Yat, UnQL등의 XML 질의 언어와 동등한 기능을 가지는 고 수준 선언적 질의 언어로서, 정보 구조를 정의하는 생성절(construct clause)과 검색조건 기술을 위한 조건절(where clause)로 구성된다. HML의 뷰 정의 언어와 정보 검색을 위한 질의어는 두가지 모두 동일한 문법 구조를 갖는다. 그림 2에 HML의 BNF를 보인다.

HML은 웹 질의어로서, HMS 시스템에서 최적화 되어 동작을 할 수 있도록 개발, 구현되었으며, 다중 문서에 대한 질의가 가능하다. 다음 [예제 1]과 [예제 2]는 HML에 의한 정보 검색 예를 보인다. [예제 1]은 '==' 연산자를 이용한 값 질의 구성 예로서 엘리먼트 경로에 의한 구조질의를 보이고 있으며, [예제 2]는 생성절(construct clause)의 '{'와 '}'사이에 있는 엘리먼트에 대한 그룹핑(grouping) 실행 예를 보인다.

```
<Query> := 'construct' <construct_clause> 'where' <where_clause>
<construct_clause> = <elements>
<elements> := (<element> | ( <groupStart> <element> <groupEnd>)) ( <var> |
<blank> ) { <elements> } '</>'
<element> := '<' (<letters> | (<letters> ('!=' | '==') (<var> | <pcdata>) ) '>'
<where_clause> := <condition> { 'and' <condition> } {'opand' <opand_clause> }
<condition> := <cond_elements> 'in' <url>
<cond_elements> := ( <element> | <range_var> ) { <cond_elements> } ( <blank> |
<pcdata> | <elements> | <var> ) '</>'
<range_var> := <var> ':' <cond_elements> ':' <var>
<letter> := 'a' | 'b' | ··· | 'Z' | '0' | ··· | '9'
<letters> := ( 'a' | 'b' | ··· | 'Z' ) { <letters> }
<pcdata> := <letter> { <pcdata> }
<operator> := ( '!=' | '==' | '>' | '<' | '>=' | '<=' )
<var> := '$' <letters>
<groupStart> := '{'
<groupEnd> := '}'
<opand_clause> := <var> <operator> ( <var> | <pcdata> )
```

그림 2. HML의 BNF

[예제 1] '담당 의사 이름이 SIEMENS인 환자들을 출력하라.'

```
construct
    <patient_name> $pname </>
where
    <patients>
      <patient>
         <name> $pname </>
         <doctor> $docname </>
      </>
    </>
in "http://hyacinth.ce.hallym.ac.kr/data/patient.xml"
opand $docname == "SIEMENS"
```

[예제 2] '담당 의사별로 그룹화 하여 환자이름을 출력하라.'

```
construct
     <hospital>
        { <docname> } $doc </>
          <patientname> $pat </>
     </>
```

```
where
    <patients>
        <patient>
            <name> $pat </>
            <docname> $doc </>
        </>
    </>
in "http://hyacinth.ce.hallym.ac.kr/data/patient.xml"
```

3. 시스템 구현

본 절에서는 시스템을 구성하는 각 모듈의 기능과 구현 방식에 대하여 간단히 기술하고, 사용자 인터페이스를 통한 통합 뷰 정의, 뷰 생성 과정, 검색 질의 구성, 응답 결과 생성과정 등 시스템 사용의 예를 보인다.

3.1 시스템 구성요소

(1) HML Query Parser

HMS의 사용자는 직관적인 비쥬얼 사용자 인터페이스를 통하여 질의를 생성한다. 따라서 사용자는 HMS에서 사용되는 HML 문법을 알지 못해도 질의 구성이 가능하다. HML 질의 파서는 사용자가 인터페이스를 통해 생성한 질의 정보와 소스 매핑정보 모듈에서 받은 메타 정보를 이용하여 완전한 HML 질의를 구성한다. 즉 매핑정보 모듈에서 넘겨받는 정보는 사용자 인터페이스를 통해서 입력받은 질의를 소스 문서에의 서브 질의로 매핑 시키는데 사용되며, 이는 사용자에게 정보의 위치를 알지 못해도 질의를 가능하게 하는 위치 투명성을 제공한다. 완성된 HML 서브 질의는 파싱되어 각각의 문서별로 실행 전략이 수립되어 실행 엔진에 전달된다.

(2) Execution Engine

실행 전략에 따라 실행 엔진은 분할된 서브 질의에 대한 결과를 각 래퍼에 요청한다. 래퍼는 서브 질의에 해당하는 정보를 검색하여 이를 XML 문서 형태로 실행엔진에 전송하며, 실행 엔진은 이 들 결과를 취합하여 그 내용을 내부적으로 테이블형태로 저장한다.

(3) Document and DTD Generator

실행 엔진에 의하여 검색된 결과는 결과 생성 모듈에 의하여 사용자 질의의 생성절에 명시한 엘리먼트 구조에 맞게 XML 문서로 변환된다. 이때 결과 XML 문서와 함께 DTD 정보가 자동 생성된다. 따라서 이를 이용하여 생성된 결과 XML 문서의 유효성 검사 및 결과 XML 문서에 대한 재 질의가 가능해진다.

(4) DTD Inferene Module

DTD 추론 모듈은 정확한(tight) 최소의 뷰 DTD를 생성한다. 임의의 소스 DTD $d_1 \sim d_n$과 이에 대한 뷰 정의가 주어진 경우, 소스 DTD $d_1 \sim d_n$로부터 뷰 정의를 만

족하는 뷰 DTD는 여러 개가 존재할 수 있다. DTD $d_1 \sim d_n$과 뷰 정의를 만족하는 뷰 DTD v_1, v_2에 대하여 $v_1 \subset v_2$ 의 관계가 성립할 경우, v_1이 v_2보다 더 정확하다고 말한다[Papakonstantinou & Velikhov 1999]. 즉 DTD $d_1 \sim d_n$과 뷰정의를 만족하는 뷰 DTD $v_a\ (\forall a \in N)$에 대하여 $v_1 \subset v_a\ (\forall a \in N)$인 v_1을 정확한 뷰 DTD라고 하며, v_1은 뷰 정의와 DTD $d_1 \sim d_n$을 만족하는 최소의 뷰 DTD를 의미한다.

본 논문의 DTD 추론 모듈에서 사용하고 있는 알고리즘은 San Diego 대학의 MIX 시스템에 사용된 알고리즘[Papakonstantinou & Velikhov 1999]에 기초한다. MIX 시스템에서는 정확한 뷰 DTD 구조를 얻기 위하여 특수화(specialization) 개념을 도입한 엘리먼트 구조를 제안하고, 모든 엘리먼트와 그것의 자식 엘리먼트를 입력으로 하는 refinement 함수를 실행하여 결과를 얻으며, refinement 함수에서는 대수적인 표현으로 엘리먼트를 풀어서 정확한(tight) 뷰 DTD를 생성한다.

HMS는 표준 DTD를 사용하는 시스템으로서, DTD 추론 모듈에서는 MIX의 DTD 추론 알고리즘의 정확도(tightness)를 개선하기 위하여 refinement 함수 정의를 부분적으로 수정하였다. 그림 3은 본 시스템에서 사용한 refinement 함수 정의를 나타내며, 이 중 밑줄 친 부분이 수정된 부분이다. 다음 [예제 3]에 대하여 뷰 DTD 추론 방식을 설명하면 다음과 같다. [예제 3]의 DTD는 가족 구성원을 나타내며, <myname>에는 자신의 이름이, <childname>에는 자식의 이름이 들어가며, 자식은 여러 명 있을 수 있다. 뷰 DTD 정의문은 '적어도 2명 이상의 자식을 갖는 family를 추출하라'는 HML 질의로 표현되며, $result는 영역 변수로서 '$result:'에서 ':$result'의 사이에 있는 엘리먼트들을 $result변수에 바인딩을 시킨다.

[예제 3]

```
DTD : <!ELEMENT family (myname, childname*)>
      <!ELEMENT myname (#PCDATA)>
      <!ELEMENT childname (#PCDATA)>

viewDTD = construct $result
          where
          $result: <family>
                     <childname> $ch1 </>
                     <childname> $ch2 </>
                   </> :$result
          opand
               $ch1 != $ch2
```

위의 [예제 3]과 같은 소스 DTD와 뷰 DTD 정의가 주어진 경우, 표준 DTD만을 사용하여 결과를 표현하면 MIX에서는 다음과 같은 뷰 DTD를 생성하게 된다.

<!ELEMENT family (myname, childname*, childname, childname*, childname, childname*)>

<!ELEMENT myname (#PCDATA)>
<!ELEMENT childname (#PCDATA)>

그러나, HMS는 다음과 같이 보다 정확한 뷰 DTD를 얻을 수 있다.

<!ELEMENT family (myname, childname*, childname, childname)>
<!ELEMENT myname (#PCDATA)>
<!ELEMENT childname (#PCDATA)>

```
Definition refinement(r, n)
        if r = n, then return n
        if r = n' then where n' is a name and n' ≠ n then return fail
        if r = r'? then return refinement(r', n) || fail
        if r = g* then return g* ⊗ refinement(g, n)
        if r = r1, r' then return (refinement(r1, n) ⊗ r') || (r1 ⊗ refinement(r', n))
        if r = r1|r' then return refinement(r1, n) || refinement(r', n)

Definition ⊗, ||
r1 ⊗ r2 = [ fail, if r1 = fail or r2 = fail
           [ r1, r2, otherwise
r1 || r2  = [ fail, if r1 = fail and r2 = fail
           [ r1, if r1 ≠ fail and r2 = fail
           [ r2, if r1 = fail and r2 ≠ fail
           [ r1|r2, otherwise
```

그림 3. type refinement 알고리즘

(5) Source Mapping Information Module

HMS에서는 사용자에게 질의 데이터에 대한 위치 투명성을 제공하기 위하여 소스 매핑 정보를 사용한다. 소스 매핑 정보는 그림 4와 같은 구조로 되어 있고, HMS의 기본 데이터 모델과의 일관성을 위하여 XML형태로 저장된다. '가상뷰이름'과 '가상엘리먼트이름'은 각각 가상 뷰에서 표현되어지는 뷰 DTD의 파일이름과 엘리먼트이름으로 관리자가 뷰 정의 질의로 정의한 가상이름이다. '실재파일이름'은 원본 소스 DTD의 파일을 나타내며, '실재엘리먼트이름'은 '실재파일이름'에 존재하는 엘리먼트 이름을 말한다. '실재엘리먼트경로'는 '실재엘리먼트이름'이 '실재파일이름'에서 존재하는 경로의 정규 경로 표현이다.

가상뷰이름	가상엘리먼트이름	실재파일이름	실재엘리먼트이름	실재엘리먼트경로	주석

그림 4. 소스 매핑 정보 구조

3.2 사용자 인터페이스

본 시스템에서는 관리자를 위한 뷰 정의 인터페이스와 일반 사용자를 위한 정보 검색 인터페이스를 기본으로 제공한다. 그림 5의 예제 DTD를 이용하여 관리자 인터페이스의 뷰 정의 과정을 설명한다. 그림 5는 DICOM 파일[Revet 1997]에서 추출된 환자의 영상정보에 대한 DTD와 HIS(Hospital Information System)의 기초 환자 정보에 대한 DTD의 일부를 보

이고 있다.

그림 6은 통합 뷰 생성을 위한 관리자 인터페이스의 사용 예를 보인다. 왼쪽 상단의 화면은 원본 소스 DTD를 읽어 들이는 화면으로 임의의 개수의 DTD를 읽어 들일 수 있으며, 구조를 나타내는 디렉토리 형태의 트리구조로 보여지게 된다. 왼쪽 하단은 뷰 DTD생성 질의를 입력하는 부분이다. 관리자는 원본 소스 DTD를 보면서 원하는 선택된 엘리먼트를 왼쪽 하단의 'Query Window'부분에 추가하여 질의를 완성한다. 완성된 질의는 실행 버튼에 의해서 뷰 DTD로 변환되어 오른쪽 화면에 결과가 출력된다. 이 예에서는 그림 5에 보인 2개의 DTD를 이용하여 환자 정보와 담당의사에 대한 정보를 통합한 뷰 DTD를 생성하는 과정을 나타내고 있다. 그림 7은 그림 6의 관리자 인터페이스를 통해 생성된 HML의 뷰 정의 질의문과 결과로 생성된 뷰 DTD를 보인다.

dicom.dtd : DICOM 공통 속성	patient.dtd : 환자 정보
<!ELEMENT dcms (dcm)*> <!ELEMENT dcm (name, providers)*> <!ELEMENT name (#PCDATA)> <!ELEMENT providers (provider*)> <!ELEMENT provider (group, groupname)*> <!ELEMENT group (service*)> <!ELEMENT groupname (#PCDATA)> <!ELEMENT service (id, element_name, value)> <!ELEMENT id (#PCDATA)> <!ELEMENT element_name (#PCDATA)> <!ELEMENT value (#PCDATA)>	<!ELEMENT patients (patient)*> <!ELEMENT patient (chnum, name, jumin, icode, iname, doctor)> <!ELEMENT chnum (#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT jumin (#PCDATA)> <!ELEMENT icode (#PCDATA)> <!ELEMENT iname (#PCDATA)> <!ELEMENT doctor (#PCDATA)>

그림 5. 예제 DTD

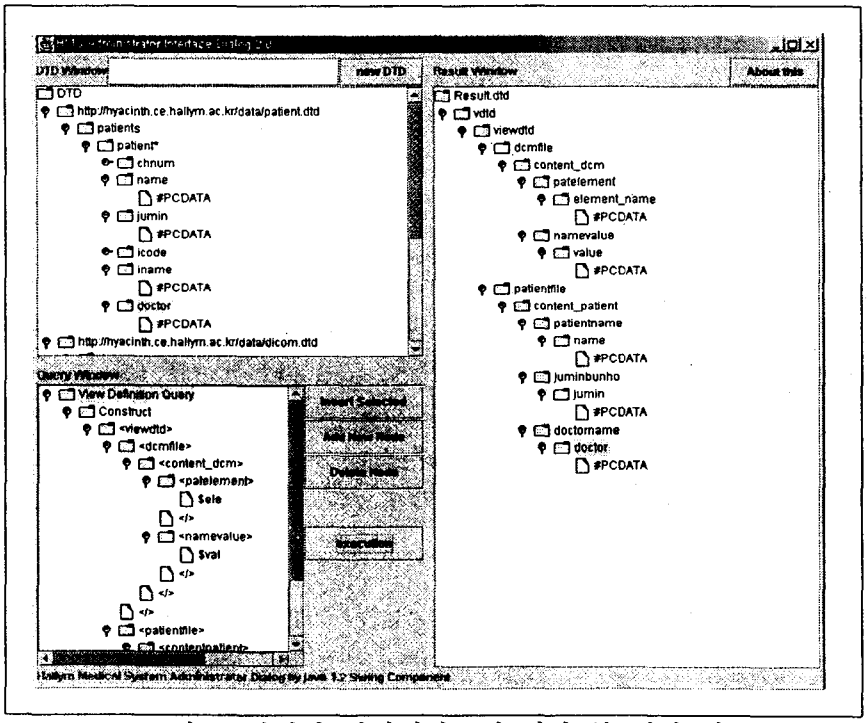

그림 6. 관리자 인터페이스에 의한 뷰 생성 예

본 시스템에서는 관리자의 뷰 정의 질의에 의한 뷰 DTD 생성 시, 뷰 DTD와 원본 소스 DTD와의 매핑 정보를 자동 추출한다. 소스 매핑정보는 그림 8과 같은 XML 형태로 저장되며, 그림 9는 이와 같이 추출, 생성되는 소스 매핑 정보의 DTD 구조를 보인다.

```
construct <viewdtd>
            <dcmfile>
              <content_dcm>
                <patelement> $ele </>
                <namevalue> $val </>
              </>
            </>
            <patientfile>
              <content_patient>
                <patientname> $nm </>
                <juminbunho> $jm </>
                <doctorname> $doc </>
              </>
            </>
          </>
where
          <dcms>
            <dcm>
              <providers>
                <provider>
                  <group>
                    <service>
                      <element_name> $ele </>
                      <value> $val </>
                    </>
                  </>
                </>
              </>
            </>
          </>
          in "http://hyacinth.ce.hallym.ac.kr/data/dicom.dtd" .
and
          <patients>
            <patient>
              <name> $nm </>
              <jumin> $jm </>
              <doctor> $doc </>
            </>
          </>
          in "http://hyacinth.ce.hallym.ac.kr/data/patient.dtd"
```

```
<!ELEMENT vdtd (viewdtd)>
<!ELEMENT viewdtd (dcmfile, patientfile)>
<!ELEMENT dcmfile (content_dcm)>
<!ELEMENT patientfile (content_patient)>
<!ELEMENT content_dcm (patelement, namevalue)>
<!ELEMENT content_patient (patientname,
juminbunho, doctorname)>
<!ELEMENT patelement (element_name)>
<!ELEMENT namevalue (value)>
<!ELEMENT patientname (name)>
<!ELEMENT juminbunho (jumin)>
<!ELEMENT doctorname (dotcor)>
<!ELEMENT element_name (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT jumin (#PCDATA)>
<!ELEMENT doctor (#PCDATA)>
```

그림 7. 생성된 HML 뷰 정의문과 뷰 DTD

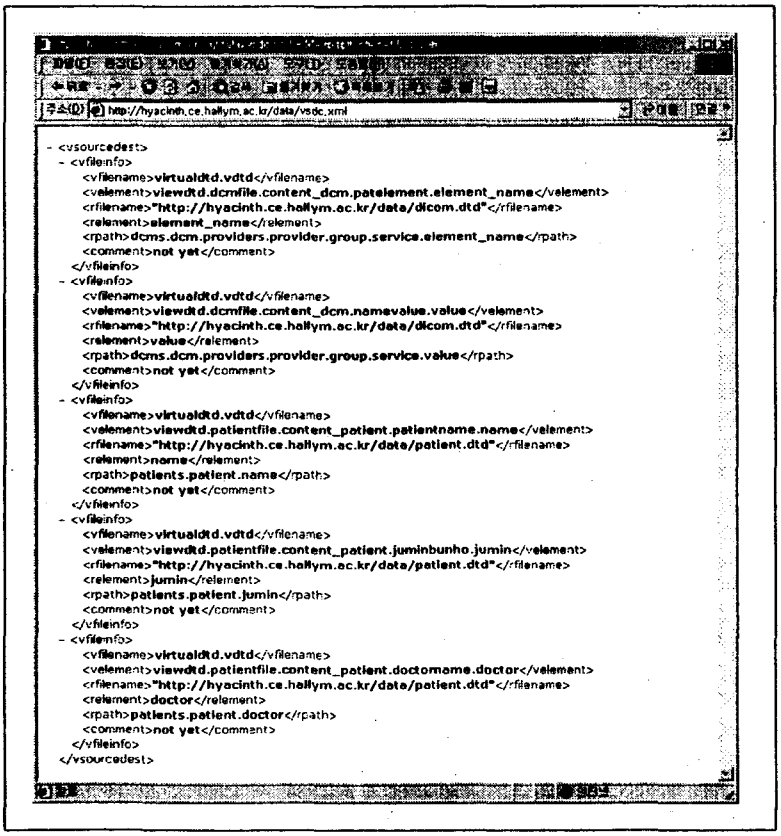

그림 8. 소스 매핑정보의 예

```
<!ELEMENT vsourcedest (vfileinfo)*>
<!ELEMENT vfileinfo (vfilename, velement, rfilename, relement, rpath, comment)>
<!ELEMENT vfilename (#PCDATA)>
<!ELEMENT velement (#PCDATA)>
<!ELEMENT rfilename (#PCDATA)>
<!ELEMENT relement (#PCDATA)>
<!ELEMENT rpath (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

그림 9. 소스 매핑정보의 DTD 구조

그림 10은 일반 사용자 인터페이스에 의한 질의 검색 과정을 나타낸다. 왼쪽 상단 화면은 소스 DTD를 읽어 들이는 화면으로 앞에서 생성된 뷰 DTD를 읽어 들일 수 있다. 우측 화면은 좌측의 뷰 DTD에 근거하여 질의를 작성하는 부분이다. 사용자가 원하는 엘리먼트는 'and window'와 'or window'를 통해서 변수로 선언된다. 'grouping window'에서는 앞에서 사용된 엘리먼트 중에서 그룹핑하려는 엘리먼트를 넣는 부분이다. 우측 화면 하단에 있는 'construct window'는 앞에서 구성한 조건 질의의 결과 형태를 작성하는 부분이다. 실행 버튼을 클릭을 하면 좌측 하단에 정의된 질의는 HML형태의 질의로 자동 변환된다.

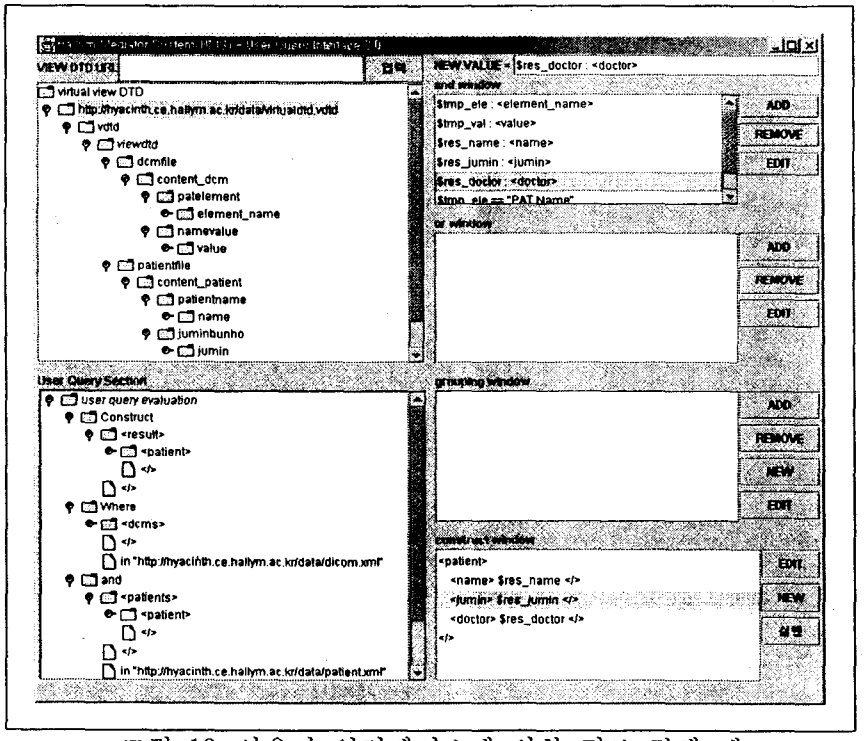

그림 10. 사용자 인터페이스에 의한 정보 검색 예

즉, 이와 같이 생성된 사용자 질의는 소스 매핑 정보를 참조하여 소스 정보에 대한 HML 질의로 자동 변환되며, 그 예를 그림 11에 보인다.

그림 11은 '환자 이름과 환자의 주민 번호, 담당의사 이름을 출력하라'는 질의를 나타내고 있다. 다음의 그림 12는 그림 5의 예제 DTD를 따르는 XML 문서의 예를 보인다. 즉, 그림

11의 질의를 실행하면 그 결과로 그림 13의 XML 문서가 출력되며, 동시에 결과 문서의 DTD가 그림 14와 같이 자동 생성된다.

```
construct <result>
                    <patient>
                              <name> $res_name </>
                              <jumin> $res_jumin </>
                              <doctor> $res_doctor </>
                    </>
          </>
where
          <dcms>
                    <dcm>
                              <providers>
                                        <provider>
                                                  <group>
                                                            <service>
                                                                      <element_name> $tmp_ele </>
                                                                      <value> $tmp_val </>
                                                            </>
                                                  </>
                                        </>
                              </>
                    </>
          </>
          in "http://hyacinth.ce.hallym.ac.kr/data/dicom.xml"
and
          <patients>
                    <patient>
                              <name> $res_name </>
                              <jumin> $res_jumin </>
                              <doctor> $res_doctor </>
                    </>
          </>
          in "http://hyacinth.ce.hallym.ac.kr/data/patient.xml"
opand
          $tmp_ele == "PAT Name"
opand
          $tmp_val == $res_name
```

그림 11. 생성된 HML 질의문

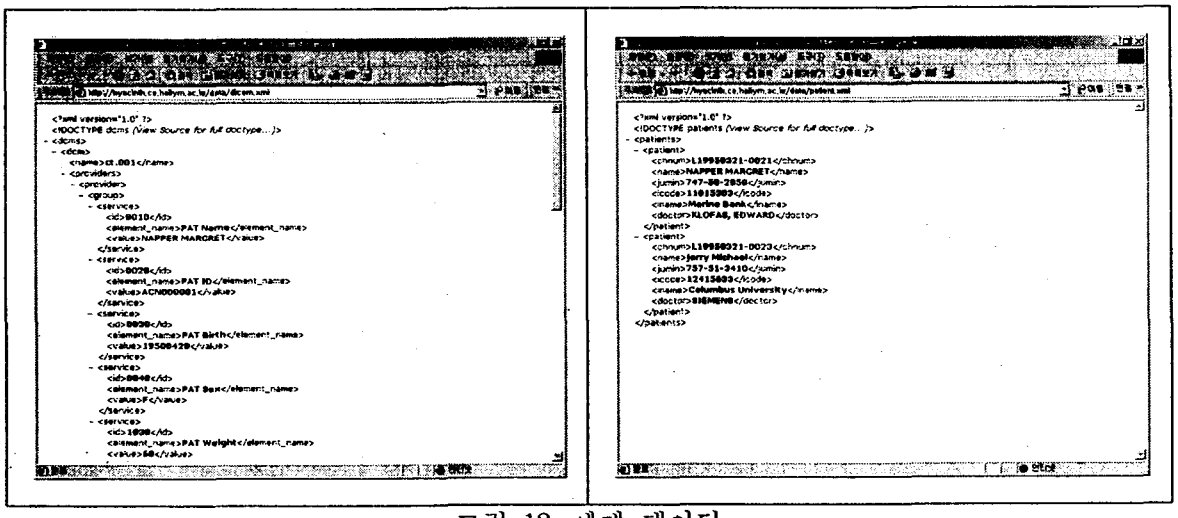

그림 12. 예제 데이터

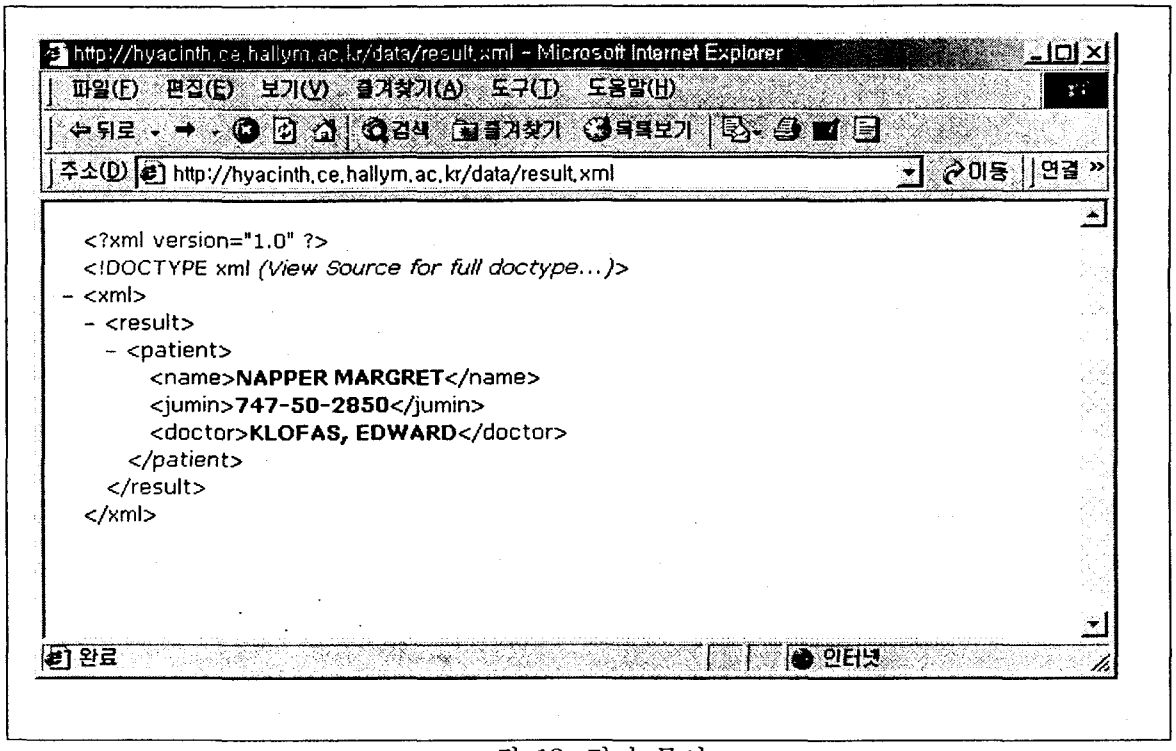

그림 13. 결과 문서

```
<!ELEMENT xml (result)* >
<!ELEMENT result (patient)>
<!ELEMENT patient (jumin, doctor, name)>
<!ELEMENT jumin (#PCDATA)>
<!ELEMENT doctor (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

그림 14. 결과 문서 DTD

4. 결론

본 시스템은 윈도우즈 NT기반에서 JDK 1.2를 이용하여 구현하였으며, XML 파서로는 IBM Parser for java를 이용하였다. 본 시스템은 웹 정보에 대한 직관적이고 쉬운 질의 환경을 제공하는 것을 그 목적으로 두고 있으며, 이를 위해 XML DTD를 기반으로 한 가상 뷰 기능을 제공하고 있다. 본 시스템은 현재 프로토타입이 가동 중인 상황으로 그 실용성 및 성능평가를 위하여, 현재 HMS를 기반으로 하는 병원정보 통합/검색 시스템[홍동완 & 윤지희 2000]을 구현 중에 있다.

참고문헌

[1] Florescu, D, Levy, A., Mendelzon, A., "Database Techniques for the World-Wide Web: A Survey," SIGMOD Record, Vol.27, No.3, 1998, pp. 59-74.

[2] Revet, B., "DICOM CookBook for Implementations in Modalities," Philipse Draft

Version 1.1, 1997, pp. 28-30.
[3] Papakonstantinou, Y., Velikhov, P., "Enhancing Semistructured Data Mediators with Document Type Definitions," ICDE, 1999, pp. 136-145.
[4] Bracy, T., Paoli, J., Sperberg-McQueen, C. M., "Extensible Markup Language (XML) 1.0," http://www.w3.org/TR/1998/REC-xml-19980210, w3c, February ,1998.
[5] McHugh, J., Abiteboul, S., Goldman, R., Quass, D., Widom. J., "Lore: A Database Management System for Semistructured Data," SIGMOD Record, Vol.26, No.3, 1997, pp. 54-66.
[6] Wiederhold, G., "Mediators in the Architecture of Future Information Systems," IEEE Computer Vol.25, No.3, 1992, pp. 38-49.
[7] Ludascher, B., Papakonstantinou, Y., Velikhov, P., "Navigation-Driven Evaluation of Virtual Mediated Views," EDBT, 2000, pp. 150-165.
[8] Baru, C. K., Gupta, A., Ludascher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P., Chu, V., "XML-Based Information Mediation with MIX," SIGMOD Conference, 1999, pp. 597-599.
[9] Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D., "XML-QL - A query language for XML," In Proc. Of the Int. World Wide Web Conf., Canada, 1999
[10] 홍동완, 윤지희, "XML문서를 이용한 환자 정보 교환 시스템(HIES)의 설계 및 구현," 한국 정보과학회 학술 발표 논문집, Vol 27, No 2, 2000, pp. 234-236.