데이터베이스 통합을 위한 XML 개발 지원 시스템에 관한 연구*

조 종 덕** · 박 영 수***

A Study on XML Development Support System for DB Integration

Cho, Chong Duck · Park, Young Soo

〈Abstract〉

In this paper, we suggest an XML Development Support(XDS) System as a good developing tool to solve the problems that related with the database integration, increased to the developing terms and costs when the developing of XML programming base on the web server. XDS System is designed for the purpose of easily coding the XML program, of simply changing the programming structure and of completely maintaining the program environment.

According to the predefined policy, XDS System provides necessary informations such as tables, columns, or relational keys from Database for developer. The developer step by step selects or sets what he needs. And XDS System creates an XML Tree with the result of selected information. The XML Tree includes all information of making an XML Schema and some XML Stylesheet. XDS System creates an XML Schema for all clients and creates many XML Stylesheets for each client using the XML Tree.

Key Words : XML Tree, XML Schema, XML Stylesheet, XPath, XML Development Support System

Ⅰ. 서론

최근 인터넷과 정보통신의 비약적인 발전은 기존의 많은 생활 패턴과 패러다임을 매우 빠르고 급속히 변화시키고 있다. 인터넷의 등장으로 지역의 장벽이 무너지고, 무선인터넷의 등장으로 시간과 공간의 제약이 없어지면서, 일반 인터넷 사용자들의 요구 사항도 다양한 형태로 진화되고, 따라서 이에 대응하는 서비스를 준비하려면 보다 많은 개발 시간과 개발 비용이 증가하게 된다.

기업과 고객 간의 정보 전달은 대부분 컴퓨터나 노트북 또는 휴대폰 등 다양한 형태의 클라이언트 매체를 통해 서비스를 받기 때문에 웹 서버의 통합 환경과 통합 프로그램의 필요성이 더욱 강조되고 있다[1].

일반적으로 웹 서비스 프로그램의 처리 과정은 우선 사용자가 웹 서버에 접속하여 질의할 수 있는 형태의 문

* 본 논문은 2008년 서일대학 학술연구비 지원에 의해 연구됨.
** 서일대학 정보전자과 교수(제1저자 및 교신저자)
*** 광운대학교 정보과학교육원 강사

서를 제공하고, 웹 서버는 질의 받은 내용을 미리 정의한 프로그램의 매개변수로 받아들여 데이터베이스에 전달하며, 데이터베이스로부터 전달받은 결과를 사용자의 접속매체가 해독할 수 있는 형태의 문서로 변환하여 전달하게 된다.

따라서 사용자 위주의 웹 서비스를 제공하기 위해서는 사용자가 질의를 하거나 결과를 받을 수 있는 문서가 전적으로 사용자의 클라이언트 매체 환경에 따라 다르게 됨으로 웹 서버는 클라이언트의 정보를 통해 어떤 형태의 문서를 제공해야 할지를 동적으로 결정하여 처리해야 한다[2].

본 연구에서는 다양한 종류의 클라이언트 매체를 사용하는 사용자들의 매체 특성을 고려한 서비스를 제공하고, 이종의 데이터베이스를 통합하여 보다 다양한 정보를 제공하며, 개발자의 개발 시간과 비용을 절약하고, 개발된 프로그램의 유지보수를 쉽게 할 수 있도록 하기 위해 기존의 응용 프로그램 형태의 저작도구가 아닌 인터넷 상에서 웹 서비스를 위한 서버 프로그램을 생성할 수 있는 XML 기반의 개발 지원 시스템을 설계한다. 그리고 이 시스템을 통해 아웃소싱 방법으로 개발자가 웹 서비스에 필요한 서버 프로그램을 자동 생성할 수 있는 방법을 제시하고 설계하고자 한다.

본 연구에서 웹 프로그램을 생성하기 위한 XML 기반의 설계 단계에서 고려된 사항은 다음과 같다.

첫째, 인터넷을 사용하는 사용자는 접속 매체가 유무선에 관계없이 클라이언트 매체의 특성을 고려한 웹 서비스를 제공받기 위해 XML 스타일시트를 생성할 수 있도록 한다.

둘째, 개발자의 개발 환경은 인터넷이 가능한 곳이면 언제 어디서나 XML 기반의 개발 지원 시스템을 통해 새로운 웹 프로그램의 개발을 쉽게 할 수 있도록 하며, XML 트리의 편집기능을 통해 쉽게 구조변경 및 유지보수 할 수 있도록 한다[3].

셋째, 사용자 위주의 프로그램을 개발하기 위해서는 보다 많은 프로그램 모듈의 증가로 개발 비용과 시간이 증가하게 됨으로, 개발 시간과 비용을 줄이기 위해 XML 기반의 개발 지원 시스템을 통해 데이터 구조를 정의하는 XML 스키마, 사용자의 매체에 맞게 정보를 표현하기 위한 XML 스타일시트, 사용자의 요청을 이종의 데이터베이스에 질의할 수 있는 XML 문서를 XML 트리 기반으로 생성할 수 있도록 한다[4].

XML 기반의 개발 지원 시스템 구성은 XML 웹서버와 같은 형태로 구성되며, 개발자와 XML 웹서버 사이에서 개발자가 원하는 웹 프로그램을 생성을 지원하기 위해 다음과 같은 단계를 수행하게 된다.

첫째, 개발자와 XML 웹서버 간의 정보 전달을 통해 웹 프로그램 개발에 필요한 정보를 수집한다.

둘째, 수집된 정보들을 DOM 트리기반의 XML 트리를 통해 개발자가 설정한 계층 구조와 함께 저장된다.

셋째, XML 트리에 저장된 정보들을 이용하여 클라이언트가 사용하게 될 XML 스키마를 생성한다.

넷째, 클라이언트에 웹서비스를 제공하기 위한 XML 스타일시트를 생성한다.

끝으로, 클라이언트가 서비스를 요청하기 위한 XPath 질의를 포함한 XML 문서를 생성한다.

본 연구에서 제안한 XML 기반의 개발 지원 시스템은 W3C에서 표준(권고사항)으로 확정된 명세서를 기준으로 설계되었으며, 전문가를 위한 ASP(Application Service Provider) 형태로서 개발자를 위해 서비스되는 웹 기반의 CASE 지원 도구이다. 그러므로 개발자는 XML 기반의 개발 지원 시스템을 통해 웹서비스에 필요한 프로그램 생성하고, 아웃소싱 방법을 통해 결과물을 제공 받게 된다.

Ⅱ. 관련연구

2.1 XML 관련기술

W3C를 중심으로 그동안 제안된 수많은 XML 관련 기술들이 점차 권고사항인 표준안으로 확정되었다. 이러한 XML 관련 기술 중 본 연구와 관련된 사항들을 살펴보면 다음과 같다.

XML 스키마는 XML 스펙을 통한 데이터 타입 및 요소와 속성에 대한 표현을 지정하기 위해 정의된 XML 문서를 위한 스키마의 표준이다[5-6]. 그리고 XML 문서의 구조와 내용 그리고 의미 체계를 정의하는 내용을 포함하고 있다. 그러나 본 연구에서는 XML 스키마를 XML 스타일시트에 적용하기 용이하도록 문서의 논리적 구조를 재정의 하였다.

XML 스타일시트는 XML 문서를 다른 문서로 변환하거나 표현하기 위한 목적으로 설계되었으며, XML이 갖는 장점인 플랫폼과 언어의 독립성을 그대로 계승한 일종의 XML 응용프로그램이다[5, 7]. 본 연구에서는 XML 스타일시트를 XML 스키마의 재구성과, XML 문서와 결합하여 클라이언트에 서비스하기 위해 사용되며, 결합 방법은 다음 3가지 유형 중 Unlinked Type을 적용하였다.

- Embedded Type : 스타일시트가 XML문서 내에 포함된 경우
- Referenced Type : 스타일시트 경로를 XML문서에 포함하는 경우
- Unlinked Type : 서버에 의해 동적으로 XML문서와 결합하는 경우

XML 질의는 웹상에서 문서나 데이터베이스의 접근을 통해 데이터를 추출할 목적으로 질의를 할 수 있도록 개발된 XML 질의의 표준이다[5, 8]. XML 질의는 크게 XQuery와 XPath Query로 분류되는데, 본 연구에서는 XQuery의 축약 형태인 XPath Query를 사용하여 간단한 경로와 매개변수를 통해 질의를 할 수 있도록 하였다.

DOM 트리는 XML의 구조와 정보의 접근을 용이하게 하고, 조작을 가능하게 하는 표준 인터페이스를 제공하며, XML 문서를 계층적인 노드의 형태로 나타내기 위한 목적으로 제정되었다[5, 9]. XML 문서의 요소는 계층 구조를 형성하고 있기 때문에, 문서의 모든 정보를 DOM 트리로 나타낼 수 있으며, 본 연구에서는 XML Tree를 생성하는 데 DOM 트리를 사용하였다.

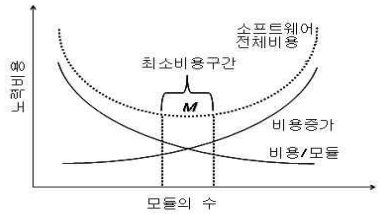
최근 대부분의 관계형 데이터베이스에서 XML을 지원하고 있으며, XML 문서를 저장하고 검색하거나, 계층적 구조를 갖는 질의에 응답할 수 있도록 설계되어, XML 기반의 질의어 및 프로그램이 가능한 포맷을 지원함으로서, XML 기반의 웹 프로그램 개발이 가능하도록 지원하고 있다[1, 10].

이와 같이 W3C를 중심으로 XML 관련 기술에 대해 제안된 많은 표준안들이 점차 권고사항으로 확정되고 있고, 따라서 본 연구에서도 이런 표준안을 기반으로 XML 개발 지원 시스템을 설계하였다.

2.2 XML CASE도구와 지원방식

본 논문에서 제안한 다양한 형태의 클라이언트 매체를 위한 웹 서비스를 제공하기 위해서는 기존에 비해 설계하고 구현해야 될 프로그램의 수가 증가하게 됨으로 <그림 1>(a)과 같은 개발 비용 증가를 초래하게 된다[11]. 따라서 본 연구에서는 <그림 1>(b)과 같이 여러 종류의 CASE 도구 중 XML을 위한 지원 도구(Support Tools)를 설계하여 웹상에서 프로그램을 작성을 도와주는 XML 기반의 개발 지원 시스템을 제안한다. 기존의 XML 지원 도구들과 본 연구에서 제안한 XML 기반의 개발 지원 시

스템이 갖는 가장 큰 차이점은 일반적인 로컬 컴퓨터상에서 응용 프로그램 형태로 지원되는 것이 아니라 웹 기반에서 ASP (Application Service Provider) 형태로 XML 프로그램 개발을 지원하는 것이다[12].



(a) 프로그램 개발 비용

(b) CASE 도구 종료

<그림 1> 프로그램 개발 비용과 CASE 도구 종류

본 연구에서 적용한 ASP는 전문가 유형의 ASP로, 정보시스템 인프라와 처리 기능을 기업 내부에서 소유하지 않고, 외부 전문 업체의 서비스를 통해 결과를 아웃소싱할 수 있도록 하였다. 따라서 이런 ASP 모델을 사용했을 경우 얻게 되는 장점은 다음과 같다.

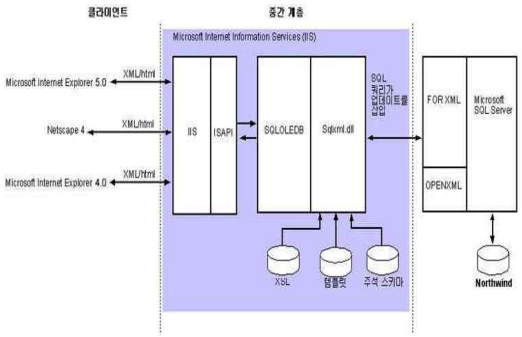
- 총괄적인 저렴한 소유 비용(Total Cost of Ownership, TCO)
- 새로운 응용프로그램에 대한 보다 빠른 접근과 신속한 갱신
- 일정한 서비스 수준의 보장
- 내부 정보시스템에 대한 관련 직원의 필요성 감소
- 하드웨어 및 소프트웨어, 유지보수 등의 관리 비용 절감
- 일정 비용 부담으로 응용프로그램의 비용 추정이 가능함

따라서 CASE 도구인 XML 개발 지원 시스템은 개발자와 웹 서버 사이에서 ASP 방식으로 인터넷 사용자를 위한 XML 서비스에 필요한 문서들을 생성할 수 있도록 지원하고, 이렇게 생성된 문서는 최종 사용자에 의해 XML 문서가 호출된다. XML 웹 서버는 서비스를 요청할 사용자의 클라이언트가 갖고 있는 운영체제와 브라우저를 확인하여 어떤 유형의 XML 스타일시트로 서비스해야 할지를 결정하게 된다. 그리고 XML 웹 서버는 XML 문서의 XPath 질의 내용을 관계형 데이터베이스에 질의하고, 질의 결과는 XML 스키마에 정의된 XML 계층 구조로 반환된다. 관계형 데이터베이스로부터 반환된 결과를 XML 웹 서버는 사용자 기기 종류에 맞는 XML 스타일시트와 동적으로 결합하여 최종 사용자에게 제공하게 된다.

Ⅲ. XML기반 개발 지원 시스템 설계

3.1 시스템 구조 및 수행 정책

본 연구에서 최종적으로 사용자에게 서비스하고자 하는 웹서버 구조는 <그림 2>와 같이 3계층으로 구성된다.

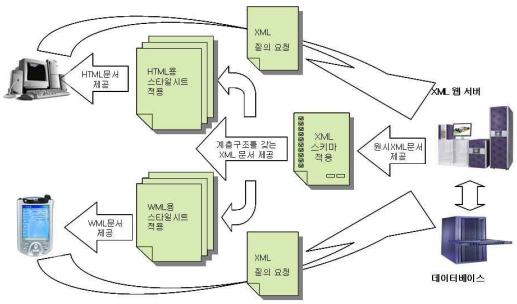


<그림 2> XML 웹서버 구조

본 연구에서 제안한 XML기반의 개발 지원 시스템 구조는 XML 서버와 동일한 구조를 갖지만, 개발자와 XML 서버 사이에서 웹 프로그램을 위한 저작 도구로서의 기

능을 수행하게 되며, 기존의 저작 도구들과의 가장 큰 차이점은 개발자의 로컬 컴퓨터에 내에서 응용 프로그램 형태로 수행되는 것이 아니라 웹서버 상에서 웹 프로그램 형태로 수행하게 된다는 것이다.

클라이언트의 요청에 의해 XML 웹서버가 서비스를 수행하게 되는 과정은 <그림 3>과 같다.



<그림 3> 인터넷 사용자를 위한 웹 서비스 경로

1. 사용자는 자신이 갖고 있는 클라이언트의 브라우저 또는 응용 프로그램을 통해 서버에 접속하게 된다.

2. 서버는 클라이언트가 서버 접속을 위해 제공되는 정보를 통해 어떤 서비스를 제공할지를 결정하고, 그 결과를 임시 보관한다.

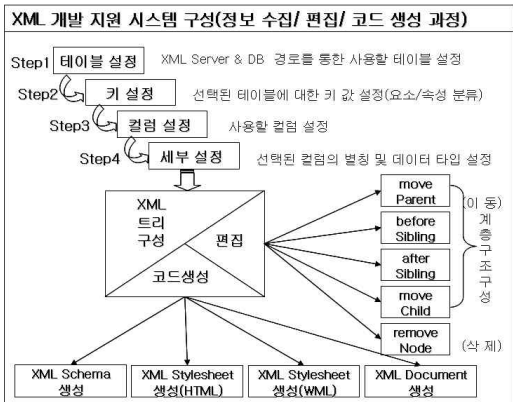
3. 서버는 템플릿의 XPath 질의를 포함하고 있는 XML 문서를 호출하게 된다.

4. XML문서는 사용자가 원하는 내용을 매개변수로 하여 관계형 데이터베이스(SQL Server)에 질의를 하게 된다.

5. 관계형 데이터베이스는 XPath 질의에 대한 결과를 FOR XML문에 의해 XML 태그를 포함하여 반환한다.

XML 개발 지원 시스템이 XML 문서 생성 과정은 <그림 4>와 같이 크게 4단계로 분류할 수 있다.

첫째, XML 트리를 위한 정보 수집 단계로 개발자가



<그림 4> XML 개발지원시스템의 정보수집 및 처리

사용할 데이터베이스, 테이블, 컬럼을 선택할 수 있도록 돕고, 개발자가 여러 개의 테이블을 선택할 경우 테이블 간의 조인 관계를 나타내는 키 관계를 설정하고 요소와 속성을 분류하는 단계이다. 이때 요소와 속성의 분류 기준은 테이블 내의 컬럼들 중 키 값을 갖는 경우에는 속성으로, 그렇지 않은 경우에는 요소로 분류하게 된다.

둘째, XML 트리의 편집 단계로, 앞서 설정된 키 값에 의해 분류된 요소들 간의 계층 구조를 형성하는 단계이다. 요소의 계층 구조 형성 기준은 XML 트리의 최상위 노드를 XMLRoot로 하여, 최종 웹서비스 문서에서 처음 출력될 항목을 루트 노드의 자식 노드로 하고, 루트 노드의 자식 노드를 클릭할 경우 나타내게 될 항목을 루트 노드의 손자 노드로 하는 방식으로 진행이 된다. 이때, 요소 노드의 검색을 위해 속성 노드들은 해당 요소 노드의 자식 노드로 설정하게 된다.

셋째, XML 트리의 구성 단계로, 앞서 수집된 정보와 편집 결과를 이용하여 XML 트리를 구성하는 단계이다. XML 트리는 DOM 트리를 기반으로 개발자 컴퓨터의 메모리상에서 구성되며, 그 결과를 저장할 수 있도록 하였다. 이렇게 구성된 XML 트리의 정보는 웹서비스에 필요한 XML 문서를 자동 생성하는데 사용된다.

끝으로, XML 트리를 기반으로 한 XML 문서 생성 단계이다. 이 단계에서는 XML 트리 내의 정보를 이용하여

XML 문서의 구조를 나타내기 위한 XML 스키마를 생성하고, 스키마를 통해 나온 결과를 최종 사용자가 브라우저를 통해 볼 수 있도록 XML 스타일시트를 각각 생성하게 된다. 그리고 최종 사용자가 위의 문서를 호출하는데 필요한 XPath 질의를 포함하는 XML 문서를 생성하게 된다.

XML 개발 지원 시스템이 정보 수집 단계를 수행하는데 필요한 정책은 <표 1>과 같다. 본 논문에서 제안한 XML 개발 지원 시스템이 인터넷을 통해 개발자와 XML 서버의 관계형 데이터베이스 간에 정보를 수집하기 위해서는 다음과 같은 일에 관한 정책이 필요하다.

첫째, XML 서버에 접속하여 데이터베이스, 테이블, 컬럼 등을 선택한다.

둘째, 선택된 컬럼들에 대한 관계형 데이터베이스내의 정보들을 수집한다.

셋째, 선택된 컬럼들에 대한 별칭 및 데이터 타입을 설정하고, 각각의 컬럼들에 대한 계층 구조를 형성한다. <표 1>에서는 이와 같은 일을 단계적으로(step by step) 개발자가 선택 또는 설정해야 할 내용들을 정의하고 있다.

개발자가 XML 개발 지원 시스템에 접속하여 미리 설정되어 있는 XML 서버의 가상 경로와 관계형 데이터베이스의 접속을 위한 가상 경로를 입력하게 되면, XML 개발 지원 시스템은 각 단계별로 개발자가 데이터베이스의 테이블 및 컬럼들을 선택할 수 있도록 하고, 각 컬럼들 중 키 값(주키/외래키)을 갖는 컬럼들을 속성으로 그 외의 컬럼들은 요소로 하여 각각의 순서와 계층 구조(부모-자식 관계) 및 각 컬럼들의 별칭을 개발자가 설정할 수 있도록 한다.

3.2 시스템의 XML Tree 설계

XML 개발 지원 시스템이 정보 수집을 위해서는 테이블 설정 단계, 키 설정 단계, 컬럼 설정 단계, 그리고 각 컬럼에 대한 별칭과 데이터 타입을 설정하는 4단계를 거치게 된다.

<표 1> XML 개발 지원 시스템의 정보 수집 정책

단 계	내 용	구분
1단계	개발 지원 시스템 접속 및 DB 접속 인증 단계	정보수집단계
2단계	RDBMS로부터 사용 가능한 테이블 선택 단계	
2.1	사용할 테이블 선택	
2.2	선택된 테이블간의 참조 키 설정	
3단계	선택된 테이블내의 컬럼 선택 단계	
4단계	각 컬럼들의 별칭 및 XML 데이터 타입 설정 단계	
5단계	요소의 세부 설정 단계(키 값으로 사용되지 않는 열)	편집단계
5.1	상위 레벨 요소 결정	
5.2	요소들 간의 순서 결정	
5.3	자식 요소들의 부모 노드 및 순서 결정	
6단계	속성의 세부 설정 단계(키 값으로 사용되는 열)	
6.1	속성의 부모 요소 및 속성들 간의 순서 결정	

이때 개발 지원 시스템이 테이블에 대한 정보를 수집하기 위해 생성되는 질의에 대한 관계 대수 표현식은 수식 1과 같다.

$$\Pi_{Table\_Name}\sigma_{Table\_Type='Base\ \ Table'}(Information\_Schema.Tables) \quad (1)$$

관계형 데이터베이스가 갖는 스키마 정보 중에서 개발자에 의해 선택된 테이블 내에 있는 컬럼 중 키 값을 갖는 컬럼에 대한 정보를 수집하게 위해 필요한 질의에 대한 관계 대수 표현식은 수식 2와 같다.

$$\Pi_{Table\_Name,\ Constraint\_Name,\ Column\_Name,\ Ordinal\_Position}\sigma_{Table\_Name='Table\ 1'\{\vee\ Table\_Name='Table\ N'\}}(Information\_Schema.Key\_Column\_Usage) \quad (2)$$

관계형 데이터베이스가 갖는 스키마 정보 중에서 개발자에 의해 선택된 테이블 내에 있는 컬럼들에 대한 정보를 수집하게 위해 필요한 질의에 대한 관계 대수 표현식은 수식 3과 같다.

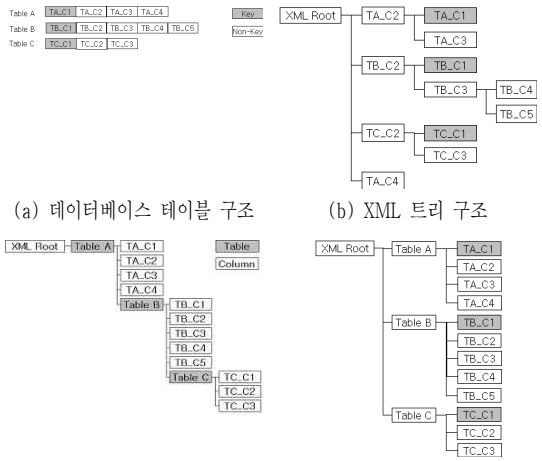
$$\Pi_{Table\_Name,\ Ordinal\_Position,\ Column\_Name}\sigma_{((Table\_Name='Table\ 1')\vee(Ordinal\_Position='1,...,n'))\{...\wedge((Table\_Name='Table\ N')\vee(Ordinal\_Position='1,...,n'))\}}(Information\_Schema.Columns) \quad (3)$$

관계형 데이터베이스가 갖는 스키마 정보 중에서 개발자가 선택한 모든 컬럼들에 대한 정보를 추출하기 위해 필요한 질의에 대한 관계 대수 표현식은 수식 4와 같다.

$$\Pi_{Table\_Name,Ordinal\_Position,Column\_Name,Data\_Type,Is\_Nullable} \quad (4)$$
$$\sigma_{((Table\_Name='Table\ 1')\vee(Ordinal\_Position='1,...,i'))}$$
$$\{...\wedge((Table\_Name='Table\ N')\vee(Ordinal\_Position='1,...,i'))\}$$
$$(Information\_Schema.Columns)$$
$$\Pi_{Table\_Name,Ordinal\_Position,Column\_Name,Data\_Type,Is\_Nullable}$$
$$\sigma_{((Table\_Name='Table\ 1')\vee(Ordinal\_Position='(n-i,...,n')}$$
$$\{...\wedge((Table\_Name='Table\ N')\vee(Ordinal\_Position='n-i,...,n'))\}$$
$$(Information\_Schema.Columns)$$

3.3 시스템의 XML 스키마 설계

XML 스키마는 XML 문서의 계층 구조를 나타내며, 본 논문에서는 이미 생성된 XML 트리의 정보를 이용하여 XML 스키마를 생성하게 된다. 그리고 XML 스키마를 통해 생성되는 XML 문서가 갖게 될 계층 구조는 <그림 5>(a)와 같은 데이터베이스 구조에서 XML 트리 구조를 갖는 (b)유형과 관계형 데이터베이스 스키마 구조를 갖는 (c)유형 2가지가 가능하다. XML 스키마는 XML 문서의 구조를 나타내기 때문에, 개발자가 설계한 <그림 5>(b)와 같은 계층 구조를 갖는 XML 스키마를 설계하는 것이 가장 이상적일 것이다. 그러나 XML 스키마를 구성하기 위해 사용된 테이블이 <그림 5>(a)와 같이 하나 이상일 경우, 각각의 테이블에 대한 참조키를 통해 테이블 간의 관계를 설정하게 되지만, 이질적인 데이터베이스 테이블이 통합되지 않는 한, 하나의 부모 노드에 서로 다른 테이블내의 컬럼들이 혼합되어 자식 노드로 존재하기는 어렵다. 또한, 사용된 컬럼들의 변경 없이 단지 계층 구조만 변경되었을 지라도 (b)유형은 매번 XML 스키마를 새로 작성해야 하는 문제점을 갖게 된다. (c)의 유형은 XML 질의에 의해 SQL 서버에 의해 동적으로 자동 생성되는 스키마 유형으로 따로 스키마를 설계할 필요가 없지만, <표 2>와 같이 질의에 따라 항상 같은 결과를 기대할 수 없다는 단점을 갖게 된다.



(a) 데이터베이스 테이블 구조 (b) XML 트리 구조

(c)RDBMS의 스키마 구조 (d)본 연구에서의 스키마 구조

<그림 5> XML 스키마가 적용된 XML 문서의 유형

<표 2> RDBMS 지원 XML 스키마의 구조

SELECT *ProductName, CategoryName* FROM Products, Categories FOR XML Auto, Elements	질의	SELECT *CategoryName, ProductName* FROM Products, Categories FOR XML Auto, Elements
<?xml version='1.0' ?> <Root> <Products> <ProductName> Alice Mutton </ProductName> <Categories> <CategoryName> Beverages </CategoryName> </Categories> </Products> ..... </Root>	결과	<?xml version='1.0' ?> <Root> <Categories> <CategoryName> Beverages </CategoryName> <Products> <ProductName> Alice Mutton </ProductName> </Products> </Categories> ..... </Root>

<표 2>의 결과에 대한 XPath 경로는 각각 다음과 같다.

좌측	ProductName	/Root [Products/@ProductName]
	CategoryName	/Root [Products/Categories/@CategoryName]

우측	ProductName	/Root [Categories/Products/@ProductName]
	CategoryName	/Root [Categories/@CategoryName]

위와 같은 결과는 <그림 5>(c)를 XML 스키마로 사용했을 때 발생할 수 있는 경우를 나타낸 것이며, 여기에는 크게 2가지의 문제점을 갖고 있다.

첫째, <표 2>의 SELECT문에서 테이블 내의 컬럼에 대한 순서만 다르게 하더라도 해당 데이터를 찾기 위한 XPath 경로가 달라진다. 이것은 SQL이 갖고 있는 문제라고 하기 보다는 XML의 엄격한 규칙에 의해서 비롯된 것이며, 이 문서를 사용하여 질의를 할 경우 유동적인 경로로 인해 프로그램 오류를 범할 가능성이 높아진다.

둘째, 서로 다른 테이블에서 서로 다른 컬럼들을 찾기 위한 질의를 할 경우 XPath 경로가 길어진 다는 것이다. 이것은 기존의 SQL 질의에서는 전혀 문제가 되지 않았지만, XML에서는 조인되는 테이블의 수만큼 찾고자 하는 컬럼에 대한 깊이 역시 깊어지는 문제점을 갖게 된다.

따라서 본 논문에서는 <그림 5>(d)와 같은 형식의 XML 스키마를 기준으로 설계하고자 한다. (d)와 같은 형식의 XML 스키마를 적용하였을 경우, XML 스키마가 XML의 구조를 나타내지 못하는 단점을 갖게 되지만, 앞서 언급한 (b)와 (c)의 경우 발생할 수 있는 단점들을 극복하는 장점을 갖게 된다.

본 논문에서는 일반 사용자가 사용할 XML 스키마를 생성을 목적으로 하고 있기 때문에, 클라이언트에 따른 XML 문서의 구조는 다를 수밖에 없고, 따라서 XML 문서와 XML 스키마의 구조가 항상 같아야 하는 (b)의 경우는 적합하지 않다. 또한 앞서 언급한 (c)의 경우도 유동적인 경로 및 경로의 길이가 길어지는 문제점을 갖고 있기 때문에 본 논문에서는 (d)와 같은 유형의 XML 스키마를 제안하였다. <표 3>은 본 논문에서 제안한 XML 스키마 구조를 이용한 질의와 결과이다.

<표 3> 본 논문에서 제안한 XML 스키마의 구조

SELECT *ProductName, CategoryName* FROM Products, Categories FOR XML Auto, Elements	질의	SELECT *CategoryName, ProductName* FROM Products, Categories FOR XML Auto, Elements

<?xml version='1.0' ?> <XMLRoot> <Products> <ProductName> Alice Mutton </ProductName> </Products> <Categories> <CategoryName> Beverages </CategoryName> </Categories> ..... </XMLRoot>	결과	<?xml version='1.0' ?> <XMLRoot> <Categories> <CategoryName> Beverages </CategoryName> </Categories> <Products> <ProductName> Alice Mutton </ProductName> </Products> ..... </XMLRoot>

본 논문에서 제안한 <그림 5>(d) 유형을 XML 스키마로 적용한 결과에 대한 XPath 경로는 아래와 같이 좌우 모두 동일한 경로를 갖고 있음을 알 수 있다.

좌측	ProductName	/XMLRoot [Products/@ProductName]
	CategoryName	/XMLRoot [Categories/@CategoryName]

우측	ProductName	/XMLRoot [Products/@ProductName]
	CategoryName	/XMLRoot [Categories/@CategoryName]

이때 테이블과 키 값을 갖는 컬럼들 간에 전제되어야 하는 조건은 다음과 같다.

1. 모든 테이블에는 주키(PK)를 갖는 컬럼이 반듯이 1개 존재한다.
2. 조인되는 모든 테이블(T)들은 조인을 위한 최소한의 외래키(FK)를 갖는다.
3. 각 테이블(T)에서 키(PK/FK)값을 갖는 모든 컬럼들은 속성 (A : attribute)노드로 사용된다. (A = PK + FK)
4, 각 테이블에서 키 값을 갖지 않는 모든 컬럼들은 요소(E: element)노드로 사용된다.

위의 조건하에서 조인되는 테이블의 수와 테이블에 속한 키 값을 갖는 컬럼들과의 관계를 살펴보면 다음과 같은 수식을 얻을 수 있다.

Ti : Table i (i=1,... , N) TN : 조인되는 테이블의 수

Ni : Ti의 총 Node 수　　Ei : Ti의 총 Element 수

PKi : Ti의 Primary Key 수

FKi : Ti의 Foreign Key 수

Ai : Ti의 총 Attribute 수 (= PKi + FKi)

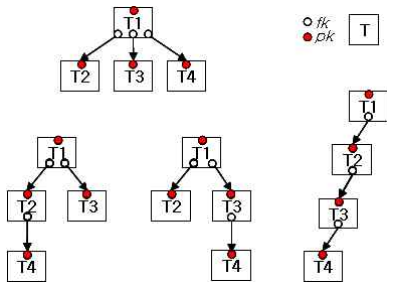
$$\sum_{i=1}^{n} N_i = \sum_{i=1}^{n} A_i + \sum_{i=1}^{n} E_i \quad (5)$$

$$\sum_{i=1}^{n} A_i = 2T_N - 1 = \sum_{i=1}^{n} PK\hat{i} + \sum_{i=1}^{n} FK\hat{i} \quad (6)$$

$$T_N = (\sum_{i=1}^{n} Ai + 1)/2 = \sum_{i=1}^{n} PK\hat{i} = \sum_{i=1}^{n} FK\hat{i} + 1 \quad (7)$$

$$\sum_{i=1}^{n} E\hat{i} = \sum_{i=1}^{n} \ni - 2T_N + 1 \quad (8)$$

위의 식들을 이용하여 조인된 테이블의 수가 4이고, 이때 사용되는 노드들의 수가 20개 일 때, 키 값을 갖는 속성 노드(A)의 개수는 수식 6에 의해 7개가 된다. 그리고 수식8에 의해 요소 노드(E)의 개수는 13이 된다. 그리고 수식 7에 의해 주키는 4개, 외래키는 3개가 된다. <그림 6>은 조인되는 테이블의 개수와 테이블 내에 존재하는 키의 개수와의 관계를 보여주고 있다.

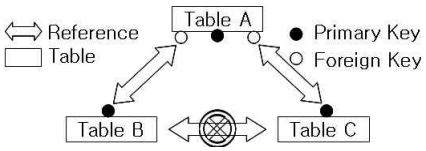


<그림 6> 조인되는 테이블과 키와의 관계

따라서 조인되는 테이블의 개수(TN)와 테이블 간의 조인관계 개수(RT)의 관계는 다음과 같은 수식이 성립하게 된다.

$$T_N = R_T + 1 \quad (9)$$

기존의 XML생성 도구에서는 평면 방식과 계층 방식 모두 스키마로서의 역할을 수행하는 데는 큰 문제가 없지만, 테이블 참조에는 약간의 문제가 존재하게 된다. <그림 7>과 같이, 테이블 A, B, C가 있고, 각각의 주키 pk_a, pk_b, pk_c를 갖고 있으며, 테이블 A에 테이블 B, C의 외래키가 fk_b, fk_c 라고 하면, 테이블 A와 B, 테이블 A와 C간의 컬럼 참조에는 문제가 없지만, 테이블 B와 C간의 참조는 별도의 처리 과정을 두지 않는 한 이루어 지지 않게 된다.

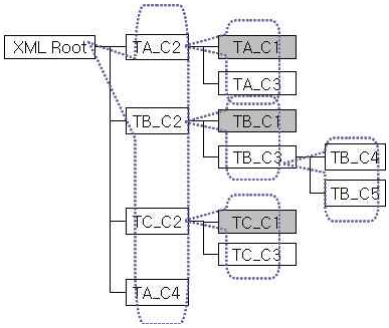


<그림 7> 기존 스키마의 테이블 키 참조 관계

본 연구에서는 XML 트리를 기반으로 XML 스키마, 스타일시트, 그리고 XML 문서를 함께 생성하기 때문에 참조 관계에서 발생될 수 있는 문제들을 Table A의 주키와 외래키를 이용하여 해결하였다. Table A에서의 주키는 자신에게 주키이며, 다른 테이블에 대해서는 외래키임으로, Table B와 Table C 테이블의 부모인 Table A의 주키와 외래키를 이용하여 <그림 7>의 Table B와 Table C 간의 참조를 가능하게 하였다.

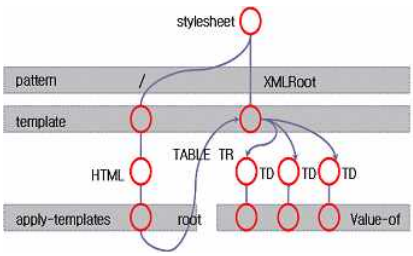
3.4 시스템의 XML 스타일시트 설계

XML 개발 지원 시스템이 XML 스타일시트를 생성하기 위해서는 XML 트리의 계층 구조와 테이블 컬럼의 별칭, 그리고 키 값을 갖는 속성 정보를 활용하며, <그림 8>과 같이 최상위 노드인 XMLRoot를 시작으로 자식 노드가 모두 키 값으로만 구성된 경우를 제외한 부모 노드가 존재하는 수만큼의 XML 스타일시트 문서를 생성하게 된다[4, 13].



<그림 8> XML 트리와 스타일시트의 계층 구조

이렇게 생성된 XML 스타일시트는 XML문서와 결합하여 <그림 9>와 같은 매핑 과정을 통해 최종 문서 생성을 위한 포맷팅과 변환을 수행한다.



<그림 9> XML 스타일시트의 템플릿 매핑 과정

Ⅳ. XML기반 개발 지원 시스템 구현

4.1 시스템의 XML 트리 구현

XML 개발 지원 시스템은 개발자가 입력한 XML 서버 및 데이터베이스의 가상 경로를 통해 개발자가 사용할 테이블을 선택할 수 있도록 하고, 선택한 테이블을 저장하는 과정이다. 우선 개발자가 입력한 경로를 이용하여 관계형 데이터베이스의 스키마에 저장된 테이블의 정보를 수집하기 위해 수식 3에서 제시한 관계 대수 표현식에 따라 XML 개발 지원 시스템은 관계형 데이터베이스에 질의할 XML질의를 생성하게 된다.

개발 지원 시스템에 의해 제공된 테이블들 중에서 개발자는 사용할 테이블을 선택하게 되고, 개발 지원 시스템은 선택된 테이블을 검색하여 저장하게 된다. 이를 위한 의사코드는 다음과 같다.

```
IF checkList. length != null THEN // 선택된 테이블
 FOR i = 0 to checkList. length DO  // 테이블검색
  IF checkList[i]. checked == true THEN
  st = st + checkList[i]. value + ';//      테이블저장
     tc = tc + 1
 ENDIF
 ENDFOR
ENDIF
```

그리고 Constraint_Name의 값을 통해 해당 컬럼이 테이블 내에서 주키(PK) 또는 외래키(FK)로 사용되는지에 대한 정보와, 테이블간의 키 관계에 대한 정보도 얻을 수 있다. Constraint_Name의 값은 2가지 형태로 출력되는데, 테이블의 컬럼이 주키로 사용될 경우에는 "PK_테이블명" 형태로 출력되고, 테이블의 컬럼이 외래키로 사용될 경우에는 "FK_외래키를 갖는 테이블명"과 "PK_주키를 갖는 테이블명" 형태로 출력된다.

이렇게 생성된 질의를 통해 얻은 XML 문서를 XML 스타일시트를 통해 개발자가 각 컬럼들에 대한 별칭을 설정하고, XML 데이터 타입을 선택할 수 있도록 준비한다.

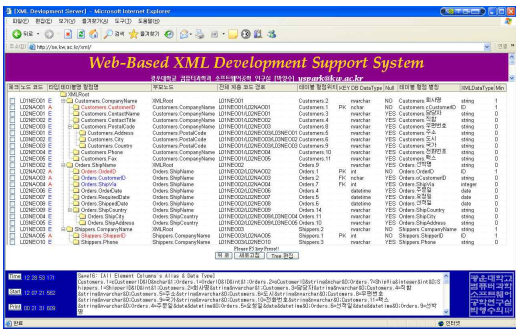
XML 개발 지원 시스템에 의해 XML 트리 생성을 위한 정보 수집 과정이 끝나게 되면, 수집된 정보를 이용하여 XML 트리를 구성하게 된다.

```
FOR i=0 to Constraint_Name. length-1 DO
   key_seperator = Constraint_Name. split("_")
   IF key_seperator[0]=="PK" THEN        // 주키인 경우
     pk +=key_seperator[1] +";" // 주키테이블 저장
   ELSE
     fk = fk +key_seperator[1] +";"// 외래키 테이블 저장
     rk = rk + key_seperator[2] +";"// 외래키 관계 테이블
   ENDIF
ENDFOR
```

XML 트리 구성의 첫 단계는 최상위 노드인 루트 노드(XMLRoot)를 선언하고 생성하는 것이다. 두 번째 단계는 수집된 정보를 이용하여 각 컬럼들마다 각자 고유의 노드 코드를 부여하고, 부여된 노드 코드에 XML 개발 지원 시스템의 정보 수집 과정을 통해 수집된 각 컬럼별 해당 정보를 추가하며, 이렇게 생성된 노드 코드들을 루트 노드 또는 부모 노드에 삽입하게 된다. XML 트리의 구성을 위한 노드 코드의 삽입을 위한 의사코드는 다음과 같다.

```
// 최상위 루트 노드 선언 및 생성
XMLRoot = new TreeListControl('XMLRoot')
// 노드 코드 선언
NodeCode = new TreeListControlNode(true)
NodeCode. setText(수집된 각 노드들의 정보 추가)
// 루트 노드(부모 노드)에 노드 코드 삽입
XMLRoot. add(NodeCode)
```

이때 사용되는 노드 코드들에 대한 관리는 생성될 때 부여받게 되는 각 노드들의 고유한 ID를 이용하여 관리하게 된다. 최종적으로 생성된 XML 트리는 <그림 10>과 같다.



<그림 10> XML 트리 출력 화면

4.2 시스템의 XML 스키마 구현

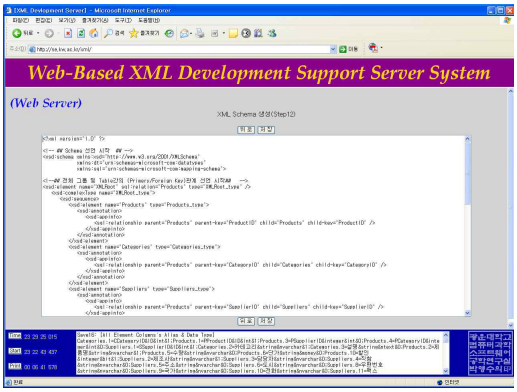
본 연구에서는 W3C에서 2001년 권고된 XML 스키마(XSD)를 기준으로 구현되며 스키마 코드의 생성 순서는 XML선언, 스키마 시작 선언, 테이블간의 조인관계 설정, 사용된 테이블간의 요소와 속성 정의 그리고 스키마 종료선언 순으로 진행된다. 우선 XML 트리가 갖고 있는 외래키를 갖는 테이블의 키 정보를 이용하여 테이블 간의 부모-자식 관계를 추출하게 된다. 또한, 주키를 갖는 테이블의 키 정보를 이용하여 해당 테이블에 대한 컬럼명을 얻게 된다.

끝으로, 조인되는 모든 테이블의 이름을 Element Name으로 구성하기 위해 외래 키 정보가 없는 테이블을 추출하게 된다. 다음은 XML 트리 정보에서 필요한 항목 필드 추출 의사코드이다.

```
// XML Tree 정보에서 필요한 항목 추출======
tbl=pk_tbl              // 선택된 테이블 정보
xt_info=tree_info // XML Tree 정보
e_tbl                   // 요소인 컬럼 정보 저장 변수
a_tbl                   // 속성인 컬럼 정보 저장 변수
tbl_sp=tbl. split(";")    //xt_sp. length와  tbl_sp. length는 같다
xt_sp=xt_info. split("/")
FOR i=0 to xt_sp. length-1 DO
    xt_tbl_sp=xt_sp. split(";")
    IF tbl_sp[i] == xt_tbl_sp[k] THEN    // k:테이블명 필드
       IF xt_tbl_sp[l]=='E' THEN         // l :타입 필드
          e_tbl = e_tbl + xt_tbl_sp[m]+", "+  //m:  별칭 필드
                + xt_tbl_sp[n]+", "+        // n : 컬럼명 필드
                + xt_tbl_sp[o]+";"+ // o:XML Data Type필드
       ELSE
          a_tbl= a_tbl + xt_tbl_sp[m]+", "+  //m:  별칭 필드
                + xt_tbl_sp[n]+", "+        // n : 컬럼명 필드
                + xt_tbl_sp[o]+";"+ //o: XML Data Type필드
       ENDIF
    ENDIF
ENDFOR
```

이와 같이 XML 트리 정보에서 추출된 "테이블명", "컬럼명", "별칭", "XML Data Type"은 조인된 테이블 수만큼 반복하여 매핑되며 코드를 생성하게 된다.

<그림 11>은 개발 지원 시스템이 XML 트리의 정보를 이용해 XML 스키마를 생성한 화면이다.



<그림 11> XML 스키마 생성 화면

4.3 시스템의 XML 스타일시트 구현

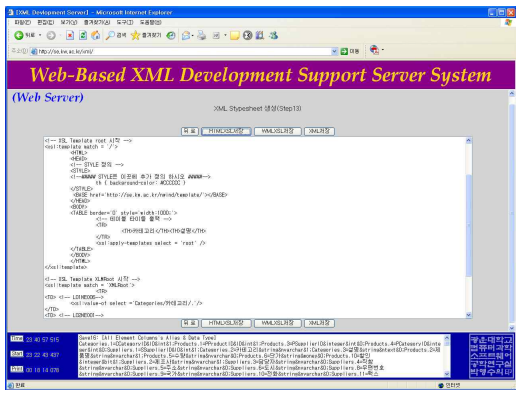
본 연구에서는 W3C에서 1999년 권고된 XML 스타일시트를 기준으로 XML선언, 스타일시트 시작선언, XML Template "root" 와 "XMLRoot"을 시작하고, 끝으로 스타일시트의 종료 순으로 스타일시트 생성이 구현된다.

이중 XML Root부분은 XML 트리가 갖고 있는 정보를 기반으로 "value-of" 부분에 대한 코드 생성 알고리즘을 적용하여 스타일시트 코드를 생성하게 된다. 위의 프로그램은 XML 스타일시트 코드 생성을 위한 자식 노드의 별칭 항목과 손자 노드 유무에 관한 정보 추출 프로그램으로 손자 노드가 없을 경우에는 링크 없이 value-of의 select 값으로 별칭 항목만 사용하면 되지만, 손자 노드가 존재하고, 손자 노드의 요소의 존재를 나타내는 ansc_elem_flg가 참이면, 해당 문서를 찾기 위한 매개변수로 achild_node의 별칭을 사용하게 된다.

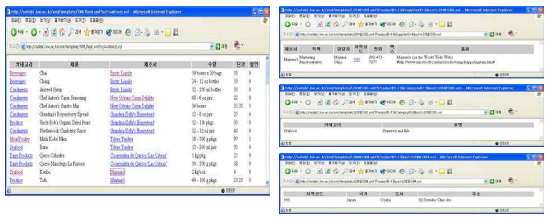
<그림 12>는 XML 개발 지원 시스템에 의해 자동 생성된 스타일시트 생성 화면이고, <그림 13>은 이렇게 생성된 결과를 실행시킨 화면이다.

<그림 14>는 XML문서에 무선(WML)용 스타일시트가 결합되어 실행된 화면이다.

```
// XML Tree의 자식-손자 노드 정보에서 필요한 항목 추출==
my_node=node    // 자신의 노드 코드 정보
xt_info=tree_info // XML Tree 정보
child_node=''       //요소인 자식 노드 정보 저장 변수
achild_node=''      //속성인 자식 노드 정보 저장 변수
temp=''
nodes=''
xt_sp=xt_info. split("/")
FOR i=0 to xt_sp. length-1 DO
  xt_field_sp=xt_sp. split(";")
  ansc_elem_flg=false
  temp=xt_field_sp[p]. split(my_node) // p:전체계층코드경로 필드
  IF temp[1]!="" THEN
    nodes=(temp[1]. length)%8
    CASE nodes IS
      WHEN 1 =>
        IF xt-field_sp[a]=='E' THEN // 손자 노드 없음
          child_node+=xt-field_sp[n]+", " // 자식 노드
        ENDIF                          //n: 별칭 필드
        EXIT
      WHEN 2 =>
        IF xt-field_sp[a]=='E' THEN // 손자 노드 있음
           child_node=child_node + xt-field_sp[n]+", "  //
           ansc_elem_flg=true  //n: 별칭 필드
        ELSEIF child_node!="" OR nsc_elem_flg!=false THEN
           achild_node+=xt-field_sp[n]+", " //
        ENDIF
        STOP
    ENDCASE
  ENDIF
ENDFOR
```



<그림 12> XML 스타일시트 생성 화면



<그림 13> HTML을 위한 XML 문서의 실행 화면

<그림 14> WML을 위한 XML 문서의 실행 화면

V. 시스템의 기능비교 및 결과분석

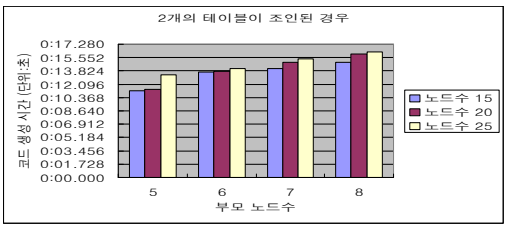
5.1 시스템의 기능비교

XML 개발지원시스템과 웹서버의 환경은 운영체제와 RAM이 동일 조건인 MS Windows Server 2003과 1GB 이고, CPU는 각각 Pantium4 3.2Ghz와 2.6Ghz이며, DB 서버는 웹서버에 MS SQL Server 2000을 설치하여, Altova사 XML Spy 2006의 문서 생성에 대한 측면을 비교하였으며, 기능적 측면의 비교는 <표 4>와 같다.

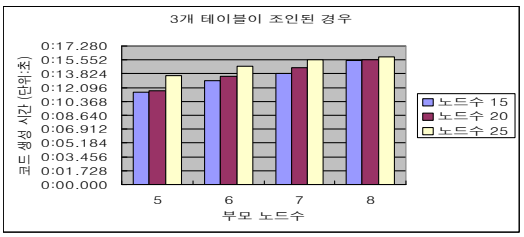
<표 4> 본 기존개발도구와 기능적 비교

평가 항목	기존 XML 개발 도구 (XML SPY 2006)	본 연구 (XML개발 지원 시스템)
생성 기반	응용프로그램 기반	웹 기반
설치장소	로컬 컴퓨터	개발 지원 시스템(서버)
개발 방법	사용자 인터페이스를 이용한 응용프로그램 개발 방법	개발 지원 시스템을 이용한 단계적 개발 방법 (아웃소싱 기법 적용)
자동 생성	스키마	스키마, 스타일시트, XML문서
유지보수 재사용성	가능 XML 편집 기능 이용	가능 XML Tree 편집을 통한 재구성

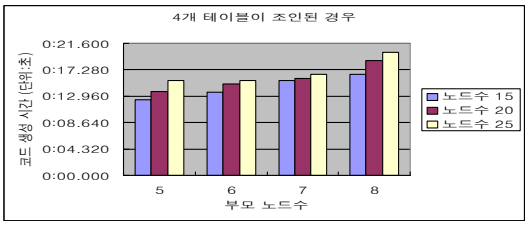
XML 문서를 웹 서비스 하기 위해 필요한 이들 문서를 총 사용된 노드수가 동일한 경우, 부모노드의 수 증가에 따른 코드 생성 시간을 조인된 테이블의 수의 증가에 따라 <그림 15>부터 <그림 17>에 걸쳐 비교하였다.



<그림 15> 조인 테이블이 2개일 때의 변화



<그림 16> 조인 테이블이 3개일 때의 변화



<그림 17> 조인 테이블이 4개일 때의 변화

따라서 부모 노드 수에 관계없이 공통으로 사용 가능한 1개의 XML 스키마를 제외하면, 부모 노드 수가 8인 경우는 결과적으로 부모 노드 수의 3배인 24개의 문서를 생성하게 됨으로 대략 2~4초가량의 시간이 더 소요되지만, 화면이 적은 PDA나 휴대폰을 위한 프로그램에 적합한 형태이고, 반대로 부모 노드 수가 5인 경우는 부모 노드 수의 3배인 15개의 문서를 생성하게 되며, 이 경우는 화면이 크고 한 화면에 많은 정보를 출력할 수 있는 데

스크톱이나 노트북 사용자에게 적합한 형태이다. 최근 개발 경향은 클라이언트 기기 특성을 고려하여 개발하게 됨으로, 유선과 무선의 부모 노드 수를 각각 5와 8로 다르게 3배할 경우 부모 노드 수의 2배인 10개와 16개의 문서가 3배된다. 그러나 하나의 형태를 생성하였다면, 나머지 형태는 XML 트리의 편집 기능을 이용한 계층 구조 변경을 통하여 부모 노드 수를 증감할 수 있으므로, 코드 생성을 위한 준비 시간을 단축할 수 있도록 하였다.

5.2 시스템의 결과분석

기존의 상용 XML 저작도구와 본 연구에서 제안한 XML 개발지원 시스템의 성능비교를 통해 다음과 같은 결과를 얻게 되었다.

첫째, 데이터베이스로부터 XML 트리를 생성하는 부분에서는 기존의 방법이 본 연구에서 제안한 시스템에 비해 생성 속도가 빠르다. 그 원인은 로컬이 아닌 인터넷상에서 수행하기 때문에 전송지연이 발생하고, 스키마 생성 정보뿐만 아니라 테이블간의 키관계 및 데이터 타입 등 보다 더 많은 정보를 수집하게 됨으로 상대적으로 생성속도가 더 발생하게 된다.

그러나 본 연구에서 제안한 XML 트리를 생성한 후에는 기존의 저작도구보다 더 많은 정보를 갖고 있기 때문에 XML 스키마 및 스타일시트를 생성하거나 변경이 빠르게 이루어진다. 또한 트리의 계층구조를 변경하거나 재구성하는 것이 가능하기 때문에 클라이언트 매체에 대응되는 각기 다른 스타일시트들을 쉽고 빠르게 작성할 수 있는 장점을 갖게 된다.

끝으로 기존의 저작도구들은 응용프로그램 형태로 로컬 컴퓨터상에서 수행되지만, 본 연구에서 제안한 시스템은 인터넷상에서 ASP(Application Service Provider)방식으로 개발이 가능함으로 개발의 편의성과 비용절감을 갖게 된다.

Ⅵ. 결론

기존의 XML 저작 도구들 대부분은 응용프로그램 형태로 사용자 인터페이스에 기초를 둔 XML 편집 기능을 위주로 개발되었기 때문에, 해당 응용 프로그램이 없는 곳에서는 직접 프로그램을 작성해야하는 단점을 갖게 되지만, 본 연구에서 제안한 XML 개발 지원 시스템은 ASP 방식의 웹 서버 형태로 구성되어 있기 때문에, 인터넷상에서 언제 어디서나 프로그램을 생성하거나 수정할 수 있고, 그 결과를 아웃소싱 방법으로 제공받을 수 있는 장점을 갖는다.

또한 각기 다른 유형의 클라이언트 매체들에 공통으로 사용될 XML 스키마와 각기 다른 클라이언트 매체의 특성에 맞는 XML 스타일시트를 자동 생성할 수 있도록 하였으며, 프로그램의 구조 변경이나 유지보수가 필요할 경우 이미 구축된 XML 트리의 재구성을 통하여 쉽고 빠르게 프로그램을 자동 생성 할 수 있는 방법을 제시하였다.

참고문헌

[1] David S. Linthicum, “Next generation Integration: Form Simple information to Web Services,” Addision Wesley Professional, 2004.

[2] Deitel et al, “XML How to Program,” Prentice Hall, 2000.

[3] Nathan Ridley, “Jscript Tree List Control,” http://www.codeproject.com/jscript, Dec 2002.

[4] J. Craig Cleaveland, “Program Generators with XML and Java,” PH PTR, 2001.

[5] W3C online, http://www.w3.org/(XML/DOM/XML Schema/XM LQuery/XPath)

[6] Jon Duckett at al, “Professional XML Schemas,” Wrox Press, 2001.

[7] Neil Bradley, “The XSL Companion,” Addition Wesley, 2000.

[8] J. Shanmugasundaran, E. Shekita et al, "A General Technique for Querying XML Documents using A Relational Database System, " SIGMOD Record 30(3), Sep. 2001, pp. 20-26.

[9] MSXML, http://msdn.microsoft.com/library/en-us/xmlsdk/html/xmmsc XML.asp

[10] Kevin Williams et al, “Professional XML Databases,” Wrox Press, 2001.

[11] Roger S. Pressman, “Software Engineering: A Practitioner’s Approach,” sixth edition, McGraw-Hill, 2003.

[12] A. Umar, “The Emerging Role of the Web for Enterprise Applications and ASPs,” IEEE Proceedings, VOL. 92 NO. 9, Sep. 2004, pp. 1420-1438.

[13] I. Tararinov and S. D. Viglas, “Sorting and Querying Ordered XML using a Relational Database System,” In Proc. Int’l conf. on Managing of Data, ACM SIGMOD, 2002.

■ 저자소개 ■

조 종 덕
Cho, Chong Duck

1992년 3월~현재 서일대학 정보전자과 교수
2001년 8월 광운대학교 전자공학과(공학박사)
1982년 2월 경희대학교 전자공학과(공학석사)
1979년 2월 광운대학교 전자공학과(공학사)

관심분야 : 정보보안, 컴퓨터 네트워크, 반도체 계면현상
E-mail : jjd@seoil.ac.kr

박 영 수
Park, Young Soo

2006년 3월~현재 광운대학교 정보과학교육원 강사
2006년 2월 광운대 컴퓨터과학과(공학박사)
1996년 8월 광운대학교 전산학과(이학석사)
1993년 2월 방송통신대학교 전산학과(이학사)

관심분야 : 소프트웨어엔지니어링, XML 웹서비스, 분산처리, 무선인터넷, 모바일컴퓨팅
E-mail : yspark@kw.ac.kr

논문접수일 : 2010년 3월 15일
수 정 일 : 2010년 4월 9일
게재확정일 : 2010년 6월 6일