이화여자대학교 과학기술대학원
2001학년도
석사학위 청구 논문

ebXML 메시지 표준에 기반 한 전자 거래 시스템 설계 및 구현

컴 퓨 터 학 과

홍 은 주

2002

ebXML 메시지 표준에 기반 한 전자 거래 시스템 설계 및 구현

이 論文을 碩士學位 論文으로 提出 함

2002 年 1月

梨花女子大學校 科學技術大學院

컴퓨터學科 洪 恩 珠

홍 은 주의 碩 士 學 位 論 文을 認准 함

指導敎授 용 환 승

審査委員 이 기 호 _____

박 승 수 _____

용 환 승 _____

梨 花 女 子 大 學 校 科 學 技 術 大 學 院

목 차

그 림 목 차

표 목 차

論 文 槪 要

기업간 거래는 주고 받는 메시지를 통해 이루어지는데 기업간 전자 거래는 정보 통신을 이용하는 것으로 이를 위해 확장성과 호환성을 만족하는 기술이 필요하게 되며 이를 만족하는 기술이 XML(extensible Markup Language)이다. 그러나 전자거래를 위한 단일한 표준이 형성되지 못하고 있으며 다른 분야 뿐 아니라 동일한 분야에서도 복수의 제안들이 등장하고 있어 표준화에 대한 필요성이 대두하게 되었다. 이러한 결과 추진된 단체가 ebXML(electronic business XML)이며 ebXML은 전세계를 하나의 전자거래 기반으로 통일하겠다는 목표를 가지고 있다.

이에 본 논문에서는 ebXML 메시지 표준을 기반으로 두 기업 사이에서 메시지를 전송하기 위한 전자 거래 시스템을 제안하였다. 전송되는 메시지는 MIME (Multipurpose Internet Mail Extension) 헤더로 포장되는 헤더 컨테이너와 적재 컨테이너로 구분되며 이들은 또 각각 MIME으로 포장되어 있다. 메시지를 송신하고자 하는 기업에서는 주문서에 해당하는 XML 문서를 관계 데이터베이스로부터 추출하여 SMTP(Simple Mail Transfer Protocol) 프로토콜을 이용해 주문을 처리하는 기업으로 송신한다. 주문 처리 기업에서 메시지를 수신하면 이를 각각 헤더에 해당하는 SOAP (Simple Object Access Protocol)봉투와 본문에 해당하는 적재 문서를 구분하여 다시 관계형 데이터베이스에 저장, 이를 관리할 수 있다.

이러한 ebXML 메시지 표준을 기반으로 한 전자 거래 시스템은 기업간에 미리 약속된 규약을 기반으로 메시지를 전송하기 때문에 표준화의 문제를 해결할 수 있으며 향후 전세계 전자 거래 시장에서 활발하게 이용될 것으로 전망된다.

I. 서 론

1.1 연구 배경 및 목적

XML은 전자상거래에서 데이터를 교환하는 표준으로 인식되고 있다. 그러나 XML은 표준을 정의하기 위한 기반일 뿐, XML 자체가 데이터의 구조나 의미를 정의하지는 않는다. 따라서 전자상거래를 위한 단일한 표준이 형성되지 못하고 있으며 다른 분야 뿐 아니라 동일한 분야에서도 복수의 제안들이 등장하고 있다. 이것은 XML 기반의 표준화 전쟁이 시작되었음을 의미하며 기업들은 N^2 변환문제(N개의 표준이 있을 경우 N(N-1) 가지의 변환이 필요)에 대응해야 한다[1].

이러한 표준화의 전쟁이 시작된 상황에서 ebXML이 설립되었다. ebXML은 UN/CEFACT(United Nations Center for Trade Facilitation and Electronic Business) 와 OASIS(Organization for the Advancement of Structured Information Standards)에 의해 설립되어 18개월의 작업 프로그램을 수행하는 국제적 시도로서 전세계적인 XML 구현을 위한 기술적 표준화의 기반을 제공하는 것을 목표로 한다. 이 목표는 다양한 환경에서 일관성이 있는 일정한 방법으로 전자거래 데이터를 교환하기 위해서, XML 기반의 개방형 기술 프레임워크를 제공하는 것으로, 단일한 전세계적 시장의 탄생을 예고하고 있다[1,2].

한편, XML이 WWW(World Wide Web)에서 비즈니스 데이터를 교환하는 표준으로 자리 잡음에 따라 기업들은 관계형 데이터베이스에 기반 한 데이터 소스 보다는 XML을 직접 다루기를 원하고 있다. 현재 혹은 가까운 미래에도 비즈니스 데이터는

관계형 데이터베이스 시스템에 저장되어 있을 것이므로 관계형 데이터베이스에 저장되어 있는 비즈니스 데이터를 표준화된 메시지 구조로 교환하는 메커니즘의 필요성이 대두하게 된다[3,4,5,6,7].

ebXML에는 비즈니스 프로세스, 기술 구조(Technical Architecture), 핵심 컴포넌트(Core Component), 등록기 및 저장소(Registry & Repository), TRP(Transport, Routing & Packaging) 등의 워킹 그룹이 있는데 TRP 프로젝트 팀은 기업간의 업무 문서 전달에 관련된 기술 규격을 담당하고 있다. TRP의 표준화 대상은 서비스 인터페이스, 메시징 정책, 행위, 메시지 형식 등에 관한 표준화를 주요 목표로 하고 있으며 메시징 서비스(Messaging Service)는 기존의 여러 통신 프로토콜을 사용하여 송신 서비스와 수신 서비스간에 메시지를 전송하는데 필요한 기능을 규정하고 있다[1,8,9,10].

TRP 팀은 2001년 5월에 ebXML 메시징 서비스 표준 버전 1.0을 발표하고 이를 승인 받았다. 현재 학계에서는 이와 관련한 연구가 활발히 진행되고 있는데 본 논문에서는 ebXML 메시징 서비스 표준 1.0을 기반으로 기업 A의 데이터를 XML로 변환해 주고 이를 기업 B로 전송해 주는 두 기업 사이의 메시지 전송에 대한 모델을 설계하고 이를 구축하였다[8].

1.2 연구 내용

본 논문에서는 두 기업 A와 B 사이의 ebXML 메시지 전송에 대해 살펴본다. 송신자 측에서는 보내고자 하는 적재(payload) 데이터를 XML 문서로 변환해주고 헤더(header)에 해당하는 데이터를 입력 받아 ebXML 규정에 근거한 SOAP 봉투

생성기를 설계 및 구현한다. ebXML 메시지 전송은 HTTP(HyperText Transfer Protocol), FTP(File Transfer Protocol), JMS(Java Message Service), SMTP 등의 프로토콜을 이용할 수 있다[11]. HTTP는 동기 및 비동기 전송을 지원하며 SMTP는 비동기 전송만 지원하는데, 본 논문에서는 SMTP를 이용하여 구현한다. 메시지를 받은 수신자 측에서는 헤더에 해당하는 SOAP 메시지와 적재(payload) XML 문서를 받게 되며 이들을 관계형 데이터베이스에 저장한다.

메시지를 송신하는 기업 A의 데이터를 XML 문서로 변환해 주는 부분은 데이터가 관계형 데이터베이스에 저장되어 있는 경우와 텍스트 파일에 저장되어 있는 경우로 나누었으며 관계형 데이터베이스에 저장되어 있는 데이터는 SQL Server 2000의 기능을 이용하여 쿼리를 통해 해당 데이터를 XML문서로 변환해 준다. 메시지를 수신하는 기업 B는 헤더에 해당하는 XML문서에 대해 MSXML 파서(Microsoft XML Parser)를 이용하여 메시지 송신과 관련한 데이터를 데이터베이스에 저장하며 적재 데이터는 SQL Server의 OpenXML 구문을 이용하여 파싱하고 데이터베이스에 저장한다.

본 논문의 구성은 다음과 같다. II장은 XML의 표준화와 관련한 ebXML의 동향 및 워킹 그룹을 살펴보고 III장에서는 ebXML TRP 그룹에서 제시한 ebXML 메시징 규격 1.0에 대해 기술하며 ebXML 메시지 헤더와 적재 문서 생성과 저장을 위한 기술에 대해 설계하고 IV장에서는 ebXML 메시지 생성 모듈, SMTP를 이용한 메시지 전송 모듈 및 수신한 ebXML 메시지 저장 모듈에 대해 구현한다. 마지막으로 V장에서는 본 연구의 결론과 함께 앞으로의 연구 방향을 제시한다.

II. 관련 기술 및 연구동향

본 장에서는 기업간 문서 교환에 대한 방법의 변화 및 '메시징'에 대해 알아 본다. 한편, XML의 기술적인 표준화를 진행하는 W3C에서는 XML 자체 뿐만 아니라 관련 핵심 기술들에 대하여 표준화를 진행하고 있는데, 전자 거래의 표준화와 관련한 ebXML 및 로제타넷(RosettaNet) 단체들의 동향 및 워킹 그룹을 살펴본다.

2.1 기업간(B2B) 문서 교환 메시징 개념

기업간에 거래를 하기 위해서는 문서가 필요하다. 1980년대 전자 상거래를 주도한 문서 교환의 핵심인 EDI(Electronic Data Interchange)는 거래 처리 시간 단축, 인력 절감 등의 효과를 가져왔다. 그러나 XML을 이용한 전자 문서 교환이 대두되면서 EDI보다 더 저렴하고 편리한 방법들을 제공하고 있다[12].

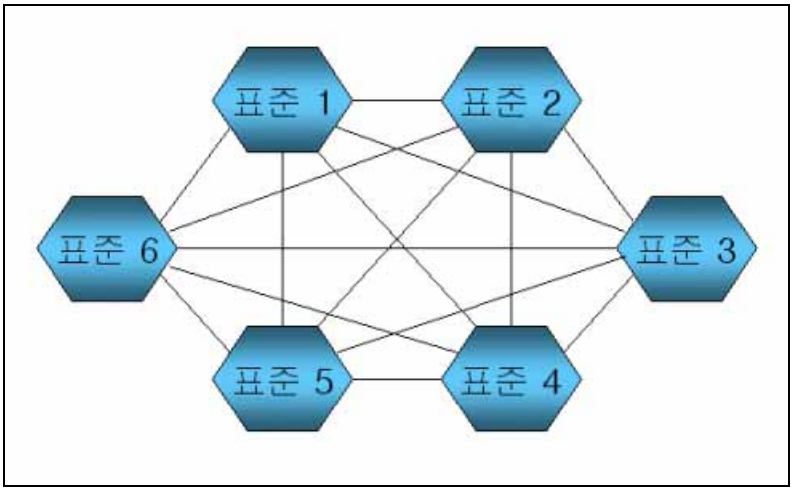
'메시징'은 문서를 메시지화 하는 것을 의미하는데 전자 거래에서도 주문서, 송장 등의 다양한 형식의 문서들도 어떤 규격화된 봉투에 담아서 보내야 한다. 이러한 규격화된 봉투에는 보낼 문서를 누구에게 보내는지, 어디로 보낼 것인지, 경유지는 어디인지를 명시해 주는데 보통의 메시지는 봉투(Envelope)와 헤더(Header), 본문(Body) 부분으로 나뉘어 있다. 봉투는 송신자와 수신자에 대한 정보가 담겨 있으며, 헤더에는 송신할 메시지가 어떤 내용을 담고 있는지에 대한 요약 내용 등에 대한 일반적인 정보를 담고 있고, 본문에서는 실제로 전송할 문서의 내용을 담고 있다[8,11].

ebXML TRP 워킹그룹에서 추진하고 있는 ebXML 메시징은 거래 당사자들 간에 비즈니스 메시지를 교환하기 위한 표준화된 방법을 제공하고 있는데 헤더(Header)와 본문(Body) 부분으로 구성된 SOAP 봉투(Envelope)와 MIME 봉투(Envelope) 그리고 프로토콜 봉투 (Envelope)으로 구성된다[8,11,13,15].

2.2 ebXML의 필요성

인터넷 및 웹을 사용한 비즈니스는 해결해야 할 많은 기술적 제약이 있는데 무엇보다도 전자적인 기업간 거래 처리를 위해서 컴퓨터 시스템간에 구조화된 정보를 교환할 수 있는 공통의 언어가 제공되어야 한다는 점이다. 일반적으로 웹에서 사용되는 HTML(HyperText Markup Language)은 단지 표현을 위한 정보만을 전달하기 때문에, 이러한 정보의 교환에는 부적절한다[1]. XML은 SGML(Standard Generalized Markup Language)의 복잡성을 단순화 한 것으로, SGML의 구조화, 확장성, 검증 등의 특성을 유지한다. XML은 SGML의 복잡성을 피하면서도, DTD(Document Type Definition) 나 XML 스키마를 사용해 문서 자신에 대해 기술할 수 있는 장점을 가진다[1,2]. 태그가 함축하고 있는 문법과 의미에 변화를 줄 수 있을 뿐 아니라, 다른 형태의 관계를 기술할 수 있어서, 문서간 연결 모델을 바꿀 수도 있다. 추가적으로 XML 문서에 대한 프리젠테이션 명세 언어가 제공되어 구조화 정보와 표현 정보가 실제 데이터로부터 독립적으로 유지될 수 있다. 이 언어는 개발자보다 쉽게 웹을 포함한 다양한 형태로 정보를 변환하는 것을 가능하게 한다. 또한 DOM(Document Object Model)은 프로그램이나 스크립트를 사용하여 이러한

정보를 동적으로 접근하고 조작할 수 있는 표준 API(Application Program Interface)를 제공한다. XML 표준이 제정된 이후 많은 응용 분야에서는 해당 분야 데이터에 관한 암묵적 지식을 표현하기 위해 노력을 해오고 있다. 특히 전자상거래 분야에서 정보의 표현과 교환 프로토콜 등을 표준화하기 위한 필요성이 대두되었지만 다른 분야 뿐만 아니라 동일한 분야에서도 복수의 제안들이 등장하게 되었다. 따라서 기업들은 N^2 변환문제(N개의 표준이 있을 경우 N(N-1) 가지의 변환이 필요)를 대응해야 하고 이를 해결하기 위한 노력으로 모든 기업이 동의하는 공통의 비즈니스 정보 표현 모델을 만들기 위해 설립된 것이 ebXML이다. 즉, ebXML은 복잡성과 변환 문제를 해결하기 위한 기반 프레임워크를 제공하는 것을 목적으로 삼고 있다[1].



[그림 2.1 N^2 변환문제]

2.2 ebXML의 워킹 그룹

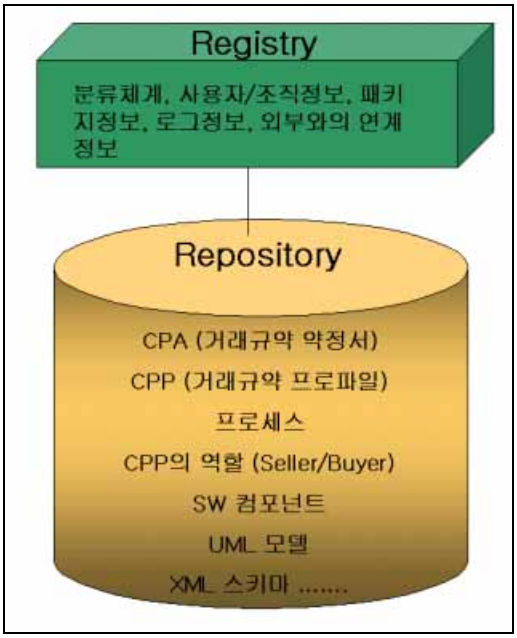
본 절에서는 ebXML의 워킹그룹에 대해 알아 본다. 워킹그룹은 비즈니스 프로세스, 기술적 구조, 핵심 컴포넌트, 레지스트리 및 리파지토리, TRP 등 다섯 가지의 워킹 그룹이 있으며 비즈니스 프로세스 그룹은 거래 절차에 대한 내용을 표준화된 방법으로 모델링하고 이를 자동으로 인식하여 거래가 일어나도록 하는 것을 목표로 하고 있다[1,8,9,10,11]. 기술적 구조 그룹에서는 ebXML 관련된 전반적인 기술 구조를 정의하고 있다. 핵심 컴포넌트 그룹에서는 메시지에서 발견되는 재사용 가능한 데이터 항목을 중립적인 개체로 정의하고 있다. 이 밖에 등록기 및 저장소 워킹 그룹과 TRP 워킹 그룹에 대해서는 좀더 자세히 알아 보겠다.

2.2.1 등록기 및 저장소 (Registry & Repository)

등록기 및 저장소는 ebXML 아키텍처의 핵심이 되는 부분으로 ebXML 기반 위에 모든 거래가 이루어지게 하기 위한 각종 정보들을 집결시켜 놓고 표준화된 방식으로 접근하도록 기본 모델을 제공하는 것을 목표로 하고 있다[1,10]. 즉, 기업간의 거래에 필요한 모든 정보들을 이 곳에 등록할 수 있는 기반 구조를 만들어 두고 거래를 원하는 사람이 필요할 때 찾아서 활용할 수 있도록 하는 것이다. 이를 위해 XML로 관리하고자 하는 객체들과 객체들에 대한 메타 정보 및 표준화된 접근을 위한 요청/응답 메시지를 XML로 표현해야 한다.

그리고 등록기 및 저장소를 통해 ebXML에서 최대 목표로 하고 있는 단일한 글로벌 전자 시장의 생성(Creating a Single Global Electronic Market)이 가능해 질 수 있는데 거래하고자 하는 기업에 대한 정보들이 표준화된 방법으로 표현되어 있는

이 곳을 통해 모든 정보들을 얻을 수 있고 이를 통해 자동으로 거래가 발생되도록 할 수 있다[12].



* CPA : Collaboration-Protocol Profile

* CPP : Collaboration-Protocol Agreement

[그림 2.2 ebXML Registry & Repository에 저장될 컨텐츠]

2.2.2 TRP (Transport, Routing & Packaging)

TRP 워킹 그룹은 상호간에 메시지를 플랫폼에 상관없이 안전하게 전송하는 것을 목표로 하고 있으며, 이를 위해 XML로 메시지의 헤더를 구성하여 상호 운용성을 증대 시킨다. TRP 프로젝트는 다음과 같은 세부적인 요구 사항을 추구한다[1,2,8,11].

① 업무 문서들을 포장하기 위한 봉투와 헤더의 규정

② 신뢰성 있는 메시지 전송과 에러 처리 능력 제공

③ 메시지 경로설정 확인

④ 보안에 관한 요구사항 충족

⑤ 감사를 위한 자료 제공

⑥ 수용 가능한 서비스 품질의 설정 및 충족

⑦ 플랫폼 독립적인 상호 연동성 지원

⑧ 재시작 및 복구의 지원

ebXML 메시지는 초기에는 고유의 메시지에 대한 포장 구조를 가지고 출발하였으나 결국 MS의 SOAP 형식으로 포장 구조를 표준으로 채택하였다. ebXML만의 고유 형식을 고집하지 않고 이미 시장에서 활용되고 있는 산업 표준을 채택함으로써 ebXML의 확산에 큰 기여를 할 수 있으리라는 판단 때문이다. ebXML 상의 모든 메시지 전송은 TRP에서 정의한 규정에 따라 메시지 헤더와 본문이 구성되어 전체 메시지가 구성된다[8,12].

2.3 로제타넷

로제타넷(RosettaNet)은 ebXML과 같이 전자상거래 프레임워크를 제시하기 위해 IT 중심의 산업계에서 발생하는 각종 거래들을 전자거래로 표준화하고자 하는 목표를 가지고 출발한 표준화 단체이다[12,15].

로제타넷에서 표준화 대상으로 다루고 있는 범위를 ebXML과 상호 비교해 보겠다. 로제타넷이 ebXML과 비교하여 가장 큰 다른 점은 활용하는 대상이 IT 중심의

산업계에 중점을 두고 거래 절차를 표준화한 점과 모든 거래 당사자들이 상호 공유하여 정보를 등록하기도 하고 검색하기도 하는 등록기와 저장소(Registry & Repository) 영역이 제외되어 있다는 점이다.

[표 2.1 ebXML과 로제타넷의 표준화 범위 상호 비교]

표준화 대상	ebXML	로제타넷
표준화 대상	전 산업 분야	IT 중심의 산업계
등록기와 저장소전	워킹그룹	없음
거래 절차	비즈니스 프로세스(BP) 워킹그룹	PIPs
공통 정보	핵심 컴포넌트(CC) 워킹그룹	Dictionary
메시지 교환	TRP 워킹그룹	RNIF
파트너 정보	트레이딩 파트너(TP) 워킹그룹	Partner Codes

로제타넷에서 진행하고 있는 표준화 대상과 그 역할은 다음과 같다[12,15].

① PIPs(Partner Interface Processes) : 거래 당사자간의 거래 절차에 대한 표준화를 목표로 하고 있으며, 이를 위하여 각 업무별로 상세한 절차와 필요한 문서들과 활용방법에 대한 자세한 내용을 담고 있다.

② Dictionary : PIPs에서 활용되는 각종 공통 정보들에 대하여 일관성 있는 상호간의 인식을 목표로 한다. 이를 위하여 공유해야 할 기본 정보들을 표준화해 둔 것이다.

③ RNIF(RosettaNet Implementation Framework) : 표준화된 방식으로 인터넷 상에서 전자거래에 관련된 비즈니스 문서들을 안전하게 상호 교환하는 것을 목표로 한

다. 이를 위하여 메시지의 구성과 보안을 고려한 전송 방식에 대한 자세한 내용을 담고 있다.

④ Product & Partner Codes : 거래할 물품과 거래 당사자에 대하여 표준화된 코드를 부여하는 것을 목표로 하고 있으며 이를 위하여 국제 표준 혹은 산업계에서 활용되고 있는 분류체계를 활용하고 있다. 거래 당사자에 대한 ID는 DUNS(Data Universal Numbering System)를 활용하고 물품에 대한 코드는 GTIN(Global Trade Item Number)와 UN/SPSC(United Nation's Standard Product and Service Code)를 활용하고 있다.

III. ebXML 메시지 표준을 기반으로 한 메시지 생성 및 저장 설계

본 장에서는 ebXML에서 지난 2001년 5월에 발표한 메시지 표준 버전 1.0에 대해 알아보고 이를 기반으로 한 전자 거래 설계에 대해 다루고 있다. 메시지를 전송하는 클라이언트에서 ebXML의 메시지를 생성하는 부분과 메시지를 전송 받은 수신 서버 측에서 메시지를 데이터베이스에 저장하는 부분의 설계에 대해 알아보도록 한다.

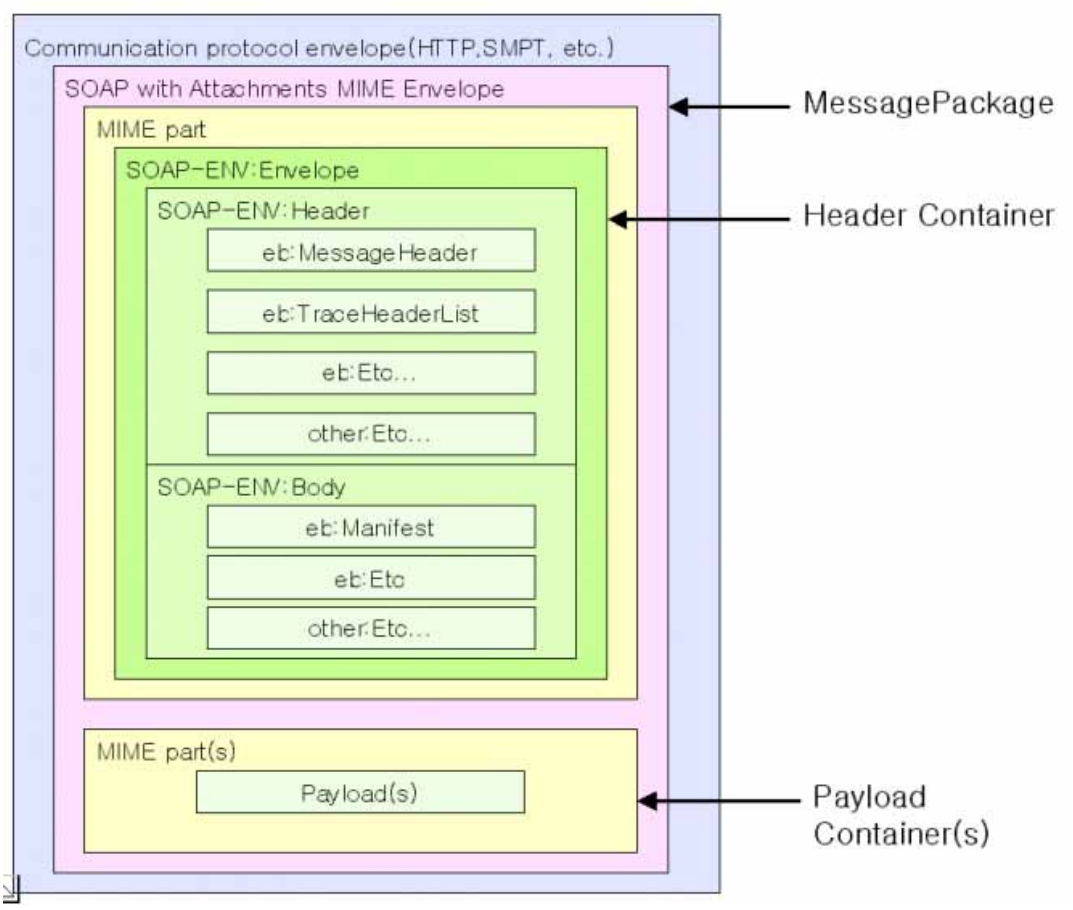
3.1 ebXML 메시지 서비스 표준

ebXML 메시지 서비스 표준은 전자 거래에서 사용되는 메시지를 교환하기 위한 통신 프로토콜에 초점을 맞추고 있으며 비즈니스 정보를 안전하게 전송하는데 목적이 있다[11]. 본 논문에서는 ebXML에서 제안한 메시지 표준을 기반으로 메시지를 생성하였는데 HTTP나 SMTP 프로토콜을 통해 전송될 ebXML 메시지를 어떻게 포장할 것인지에 대해 정의한 패키징 규격과 성공적인 ebXML 메시지를 생성할 수 있는 ebXML 메시지 서비스에 필요한 정보를 구성하고 구조화 하는데 대해 정의한 ebXML SOAP 확장에 대해 알아보겠다.

3.1.1 패키징 규격

메시지 패키지는 크게 헤더 컨테이너(Header Container)와 적재 컨테이너

(Payload Container)로 구분된다. 헤더 컨테이너는 SOAP 메시지로 표현되어 있으며 다시 SOAP Header 엘리먼트와 SOAP Body 엘리먼트로 구분된다. SOAP Header 엘리먼트는 ebXML 헤더 정보를 담고 있으며 SOAP Body 엘리먼트는 메시지의 적재 파트와 관련된 정보를 담고 있다[11].



[그림 3.1 ebXML 메시지 구조]

- 헤더 컨테이너 (Header Container) : 메시지 패키지의 루트 부분에 해당한다.

```
Content-ID: messagepackage123@example.com
Content-Type: text/xml; charset="UTF-8"

<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schema.xmlsoap.org/soap/envelope/>
  <SOAP-ENV:Header> ... </SOAP-ENV:Header>
  <SOAP-ENV:Body> ... </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[그림 3.2 헤더 컨테이너 예]

- 적재 컨테이너 (Payload Container) : 메시지 패키지 안에 있는 하나 이상의 적재 컨테이너이며 적재 내용이 없으면 존재하지 않을 수도 있다. 각각의 적재 컨테이너의 내용은 SOAP Body 부분의 Manifest 엘리먼트에 선언된 내용이어야 한다. ebXML에서는 적재되는 내용에 대해 아무런 규정이나 제한을 두고 있지 않다. 적재 내용은 단수한 텍스트 오브젝트가 될 수도 있으며 복잡한 계층을 가진 오브젝트도 될 수 있다.

```
Content-ID: <domainname.example.com>
Content-Type: text/xml

<Invoice>
  <Invoicedata>
  ...
  </Invoicedata>
</Invoice>
```

[그림 3.3 적재 컨테이너 예]

- ebXML 메시지의 MIME 헤더 : HTTP 프로토콜이나 SMTP 프로토콜 모두에게

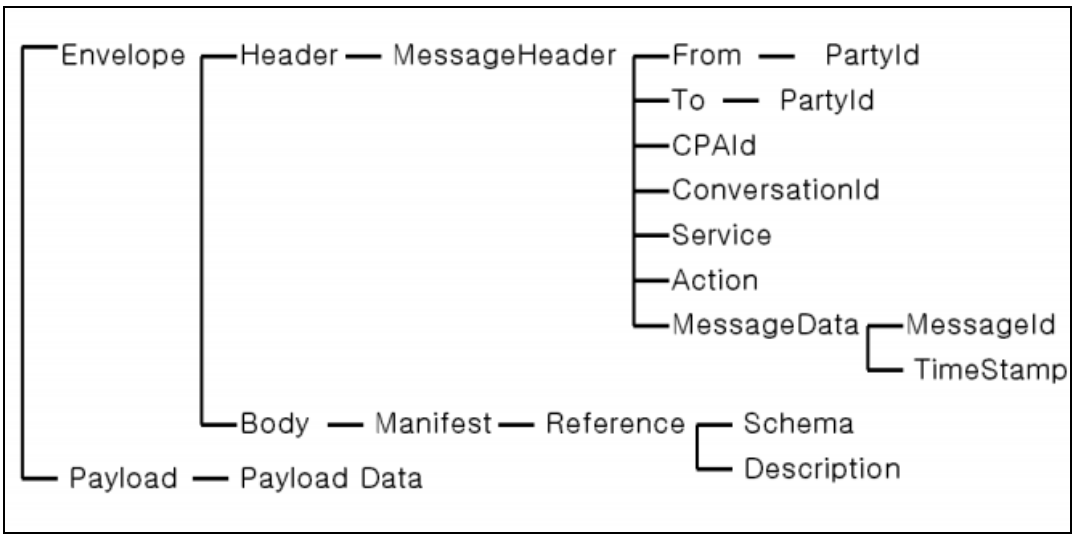
있어서 MIME은 공통적으로 적용된다. 즉, 전송 프로토콜에 따라 헤더의 내용은 달라질 수 있지만 MIME 헤더에 담기는 내용은 달라지지 않는다. MIME 헤더에서 Content-type으로 multipart/related를 사용하는데 이것은 다음에 나오게 될 파트가 여러 개이며 각각의 파트가 연관이 되어 있다는 것을 나타낸다. 또한 각각의 파트들은 boundary 값으로 구분되어 있다는 것을 지정할 수 있다. 또한 type으로 각 컨텐츠의 형식을 결정한다.

```
Content-type: multipart/related; boundary="BoundarY"; type="text/xml";
        start=" <ebxmlheader111@example.com>
--BoundarY
Content-ID: <ebxmlheader111@example.com>
Content-Type: text/xml
.......
--BoundarY
Content-ID: <ebxmlpayload@example.com>
Content-Type: text/xml
.......
--BoundarY
```

[그림 3.4 ebXML 메시지 MIME 헤더의 예]

3.1.2 ebXML SOAP 확장

SOAP 봉투는 Header와 Body 부분으로 구성되어 있는 XML 문서이다. Header는 SOAP 봉투를 처리하기 위한 Service나 보내는 사람, 받는 사람등에 대한 정보를 담고 있으며, Body 부분에는 다음 파트에 오게 되는 적재(Payload)에 대한 정보를 담고 있다.



[그림 3.5 SOAP 메시지 엘리먼트 계층도]

- <MessageHeader> 엘리먼트 : 라우팅에 대한 정보와 메시지에 관련된 정보를 담고 있다.
  - <From>...</From>, <To>...</To> : DUNS 숫자나 이메일 주소 같은 구분자를 사용한다.
  - <CPAId>...</CPAId> : 두 기업간의 메시지 교환을 관리하는 파라미터를 구분하는 스트링으로 두 기업간에 동의된 내용이어야 한다.
  - <ConversationId>...</ConversationId> : 대화를 시작하는 기업이 이 id를 결정하며 from, to 사이에 유일한 값이어야 한다.
  - <Service>...</Service> : 개인이나 기업 같은 서비스 디자이너에 의해 설명되는 메시지 상의 행동을 확인한다.
  - <Action>...</Action> : 메시지를 프로세스하는 서비스 안에서의 프로세스를 확인한다.
  - <MessageData>...</MessageData> : 메시지를 유일하게 구분해준다. 하위

엘리먼트로는 유일한 구분자인 <MessageId>와 메시지가 생성된 시간을 표현하는 <TimeStamp>가 있다.

- <Manifest> 엘리먼트 : 하나 이상의 참조 엘리먼트로 구성되어 있으며 적재(Payload)에 들어있는 데이터에 대한 정보를 담고 있다.
  - <Reference>...</Reference> : 하위 엘리먼트 <Schema>와 <Description> 으로 이루어져 있다. <Schema> 엘리먼트는 <Reference>에서 확인된 인스턴스 문서를 정의하는 스키마 정보를 담고 있으며 <Description> 엘리먼트는 적재 오브젝트에 대해 명시하고 있다.

3.2 메시지 생성

ebXML 메시지를 포장하여 전송하기 위해서는 헤더 컨테이너에 포함되는 SOAP으로 이루어진 XML문서와 적재 컨테이너에 포함되는 XML 문서를 각각 생성해 주어야 한다. 본 논문에서 구현한 시스템은 기업 A 가 기업 B 에게 주문서를 전송하는 모델인데, 헤더 컨테이너 부분은 ebXML에서 제안한 스키마에 근거하여 SOAP 봉투(XML 문서)를 작성하며 적재 문서 부분에 대해서는 데이터베이스와 텍스트 파일로부터 XML 문서를 작성한다.

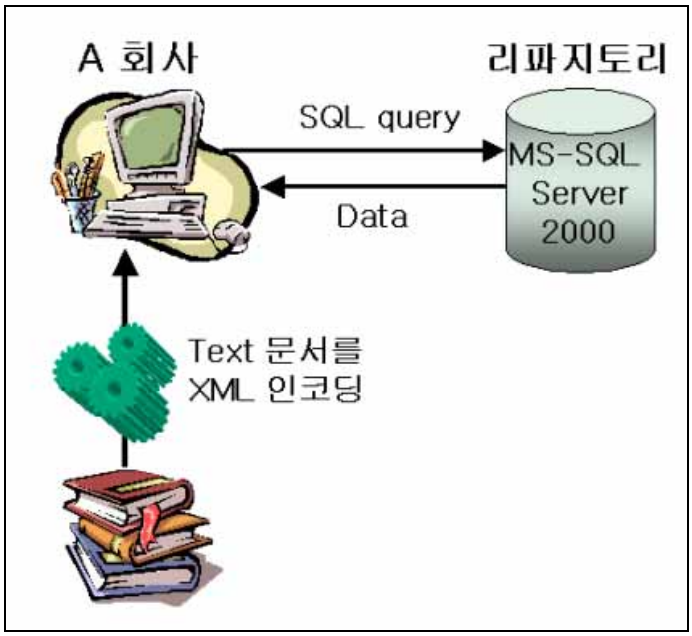
3.2.1 SOAP 봉투

SOAP 봉투는 ebXML 메시지 규격에 근거하여 작성하기 때문에 별도의 스키마를 작성할 필요가 없으며 XML문서의 텍스트 부분에 해당하는 데이터를 사용자로부터 입력 받거나 혹은 프로그램 내부에서 생성한 후 각 엘리먼트를 인코딩해 준다.

SOAP 봉투의 구조는 그림 3.4의 SOAP 구조도를 기반으로 작성된 XML 문서이다.

3.2.2 적재 문서

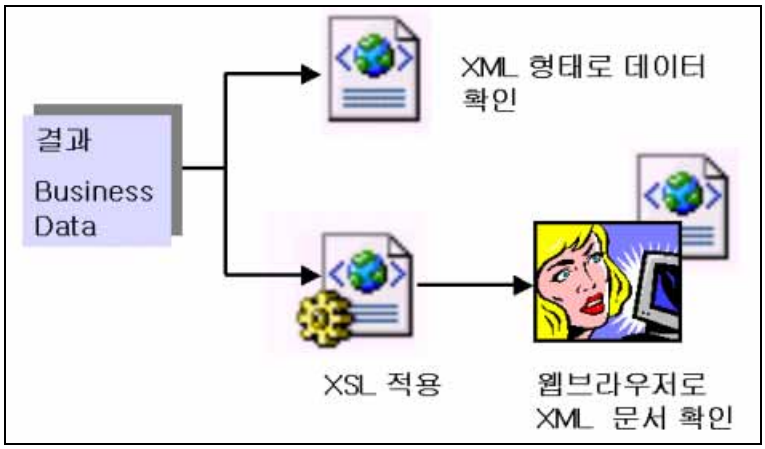
기업 A가 기업 B에 주문서를 전송할 경우 기업 A는 주문과 관련한 데이터를 XML로 인코딩 해주어야 한다. 본 논문에서는 데이터 소스를 SQL Server에서 가져오는 경우와 기업 A의 텍스트 문서를 가지고 각각 XML 인코딩 하는 경우를 고려하였다.



[그림 3.6 기업 A의 주문서 XML 인코딩]

주문하기 위해 필요한 데이터는 SQL Server의 XML 변환 기능을 이용하여 XML 문서로 변환한다. 이때, 기업 A가 기업 B의 데이터베이스 내용에 대해 질의하기 위해서는 기업 B의 데이터베이스 내용을 확인해야 하는데 리파지토리의 데이터베이스 내용을 XSL(eXtensible Stylesheet Language)을 통해서 웹브라우저를 통해

확인한다. 확인 과정은 그림 3.7과 같다.



[그림 3.7 리파지토리 데이터베이스 확인]

3.3 메시지 저장

기업 B는 전송 받은 메시지를 데이터베이스에 저장하여 주문서를 관리한다. 첫 번째, SOAP 봉투는 정해진 스키마에 근거하여 만들어진 테이블에 저장되며 두 번째, 적재 문서는 문서마다 스키마가 다르기 때문에 문서마다 테이블을 따로 생성하여 관리한다.

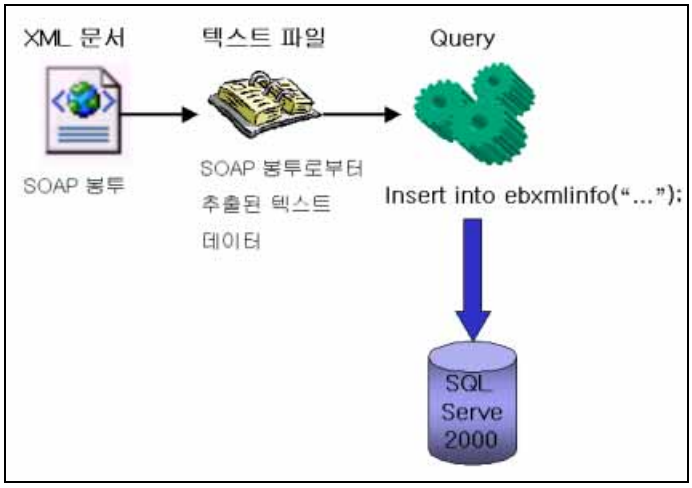
3.3.1. SOAP 봉투 저장

SOAP 봉투를 저장할 테이블은 다음과 같이 purchase_order라는 테이블 이름으로 설계하고 각 column에 SOAP 봉투의 텍스트 데이터를 저장한다.

[표 3.1 Purchase_order 테이블 설계]

Column Name	Data Type	Length
From	Varchar	100
To	Varchar	100
CPAId	Varchar	100
ConversationId	Varchar	100
Action	Varchar	100
MessageId	Varchar	100
Timestamp	Varchar	100
Description_1	Varchar	100
Description_2	varchar	100

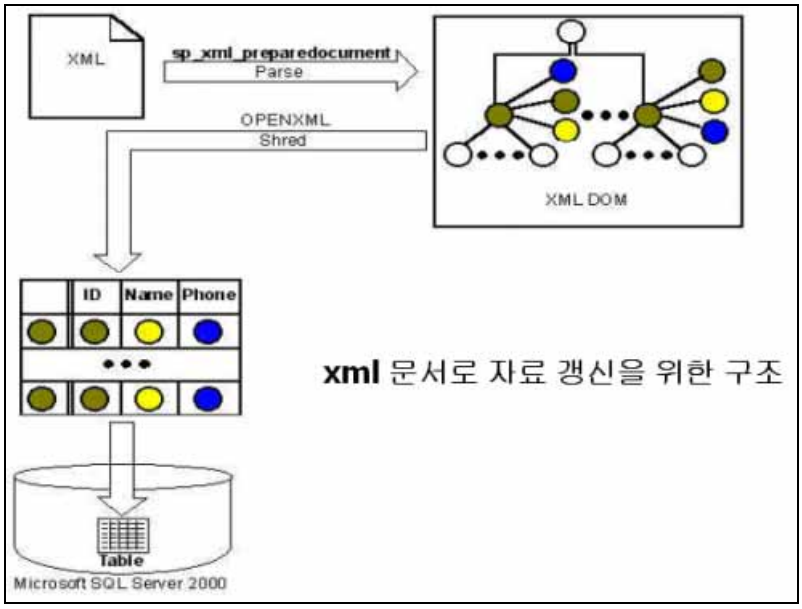
저장과 관련한 전체 과정은 다음 그림 3.8과 같다. SOAP 봉투로부터 마크업을 제외한 실제 데이트를 추출하여 이를 텍스트 파일에 저장하고 'insert into table_name' query를 통해 데이터베이스에 저장한다.



[그림 3.8 기업 A의 데이터베이스 확인과정]

3.3.2 적재 문서 저장

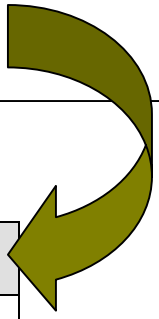
적재 문서는 SQL Server 2000에서 제공하는 sp_xml_preparedocument 시스템 저장 프로시저와 OpenXML 구문을 사용하여 데이터베이스에 저장한다[10,12].



[그림 3.9 OpenXML 구조도]

적재 문서의 예로 그림 3.7과 같은 XML문서를 데이터베이스에 저장하기 위해 이 문서를 기반으로 한 테이블을 설계 한 후 XML 문서의 애트리뷰트에 해당하는 데이터들을 테이블의 각 column에 저장한다. 다음 그림 3.10은 XML 문서를 관계 데이터베이스로 매핑해주는 화면이다.

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <ebxml_product id="3" product_name="Strawberry Drink" price="7" />
  <ebxml_product id="4" product_name="Cream Soda" price="9" />
  <ebxml_product id="5" product_name="Diet Soda" price="6" />
  <ebxml_product id="6" product_name="Cola" price="3" />
  <ebxml_product id="7" product_name="Orange Juice" price="4" />
……
</root>
```



ID	Product_Name	Price
3	Strawberry Drink	7
4	Cream Soda	9
5	Diet Soda	6
6	Cola	3
7	Orange Juice	4

[그림 3.10 XML의 RDB 데이터로의 매핑]

IV. 시스템 구현

본 장에서는 3장에서 언급한 ebXML 메시지 생성 및 저장 모델을 이용하여 ebXML 메시지 생성과 SMTP 프로토콜을 이용한 메시지 전송 및 수신 그리고 수신한 ebXML 메시지를 데이터베이스에 저장하는 시스템 등 각 네 가지 모듈을 구현한다. 구현된 시스템을 이용하여 실제로 회사가 주문 명령을 실행하고 명령을 처리하는 과정을 예로 든다.

4.1 시스템 구현 환경

본 논문에서 제안한 프로토타입 시스템의 구현 환경은 표 4.1과 같다.

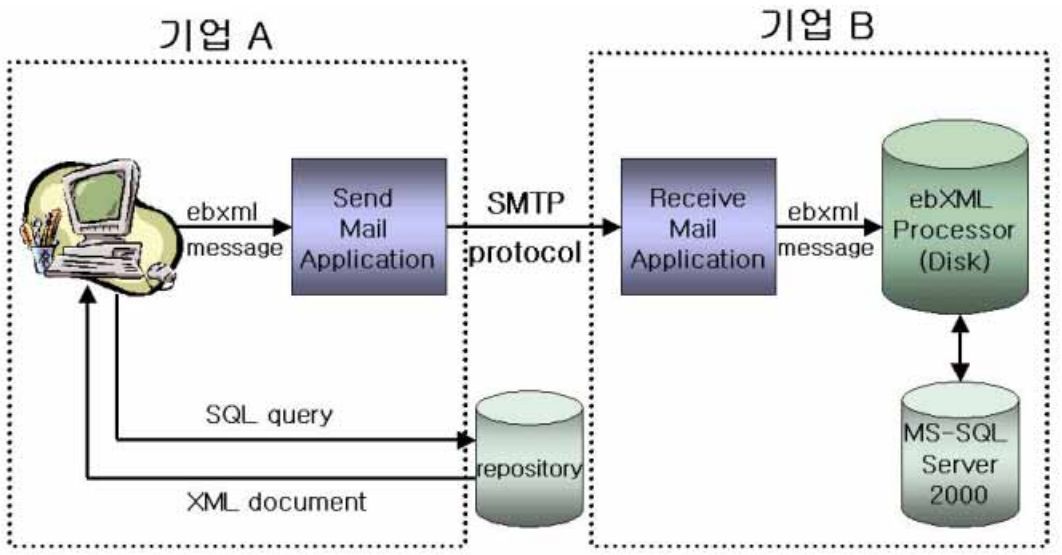
[표 4.1 시스템 구현 환경]

서버측 운영 체제	Windows 2000 Server
데이터 서버	Microsoft SQL Server 2000
웹 서버	Internet Information Server 5 (IIS 5)
웹 클라이언트	Microsoft Internet Explorer 5.0 이상
개발 도구 및 언어	Java Development Kit 1.2 Jakarta Tomcat 3.2.1, MSXML Parser 3.0 OpenXML (SQL Server 2000), JavaMail 1.2, XSL (Extensible Markup Language) Java Server Page

주문하고자 하는 기업 A가 주문을 처리하는 기업 B로 거래 문서를 전송하기 위한 ebXML 메시지 생성 과정을 톰캣(Tomcat) 환경에서 수행되는 JSP로 구현하였다. A 기업은 메시지를 생성하기 데이터베이스 정보를 가져와야 하는데 A는 제공된 테이블 정보를 바탕으로 쿼리문을 줄 수 있다. 이때 A는 제공된 SQL Server에 대해 IIS 가상 디렉토리를 설정하여 쿼리 결과를 HTTP를 통해 전송 받는다. 메시지 전송 및 수신은 SMTP 프로토콜을 이용한 JavaMail로 구현하였다. 기업 B측에 수신된 ebXML 메시지는 MSXML 파서와 SQL Server 2000의 OpenXML 구문을 사용하여 파싱 되도록 하였으며 SQL 쿼리를 통해 데이터베이스에 저장된다.

4.2 시스템 전체 구조

시스템 전체 구조는 크게 4개의 모듈로 나누어지며 각 부분의 해당 알고리즘을 기술하도록 한다. 첫번째 모듈은 ebXML 메시지를 생성하는 모듈이고 두 번째는 생성된 메시지를 다른 기업으로 송신해주는 통신 프로토콜에 관한 모듈이며 세 번째 역시 통신 프로토콜인데 전달 받은 메시지를 시스템에 저장해 주는 역할을 한다. 네 번째 모듈은 시스템에 저장된 메시지를 데이터베이스에 저장한다.



[그림 4.1 전체 시스템 구조도]

4.3 ebXML 메시지 표준을 기반으로 한 전자 거래 시스템

ebXML에 담겨 있는 기본적인 개념은 서비스를 요청하고 결과에 대한 응답 정보를 받는 기본적인 거래들이 모두 XML 형태의 표준화된 규격을 가진 문서에 의해 이루어 진다는 것이다[11]. 본 논문에서는 서비스를 요청하고 요청 받은 데이터를 처리하는 과정에 대해 구현한다.

4.3.1 ebXML 메시지 생성 모듈

ebXML 메시지를 생성하기 위해 사용자는 SOAP 봉투와 적재 문서를 만들기 위한 정보를 입력해 주어야 한다. SOAP 봉투를 만들기 위해서는 먼저 메시지를 어디로부터 어디로 전달하는지에 대한 정보를 명시해 주어야 하며 명령의 형태 및 적재 문서의 수, 그리고 적재 문서에 대한 설명을 기재한다. 적재 문서에 대해서는 본

논문에서는 주문서를 기준으로 고려하였으며 기업 A의 회사가 기업 B의 회사로 주문서를 전달하는 과정을 그리고 있다. SOAP 봉투 문서를 이루기 위한 데이터는 사용자로부터 입력된 데이터 및 프로그램 내부에서 생성된 데이터들로 ebXML 메시지 표준을 기반으로 하여 XML 문서로 엘리먼트를 인코딩 해준다.

① SOAP 봉투 생성

먼저 SOAP 봉투 문서를 작성하기 위해 사용자로부터 입력 받아야 할 데이터는 From, To, Action의 형태, 적재 문서의 개수, 적재 문서에 대한 URI(Uniform Resource Identifier)와 설명 등이다. From과 To는 각각 메시지를 발생시키는 회사와 메시지를 수신하는 회사를 명시해 주는 것으로 기업의 고유한 id인 DUNS 숫자나 URI 혹은 이메일 주소로 기입할 수 있다. 또한 SOAP 봉투와 함께 적재하고자 하는 문서의 수를 기입하고 이 적재 문서 수를 바탕으로 그 수 만큼 적재 문서에 대한 레퍼런스를 입력 받는다.

이 밖에 CPAId, ConversationId, Timestamp, Payload id 등은 프로그램 내부에서 생성해 주는데 CPAId는 From 정보와 To 정보를 연결한 스트링 형태로 만들어 주며 ConversationId는 주문 명령이 발생한 시간 데이터와 임의로 주어진 다섯 자리 숫자 데이터를 스트링으로 연결하여 사용한다. Timestamp는 ConversationId와 마찬가지로 주문 명령이 발생한 시간을 기록한다. 이렇게 생성된 데이터들은 ebXML 메시지 표준을 바탕으로 인코딩 한다.

```
import   java.io.*;                    // XML 문서를 파일에 기록하기 위한 클래스
import   java.util.Random;          // id 생성을 위한 클래스
<jsp:useBean id="clock" class="java.util.Date" />   // 시간 데이터 호출을 위한
……                                                          //   Bean 사용
int year      = clock.getYear();
int month    = clock.getMonth();
int date      = clock.getDate();
int hours     = clock.getHours();
int minutes = clock.getMinutes();
……
try                                 // 입력 및 생성된 데이터로 SOAP 봉투 문서 작성
{ BufferedWriter bw = new BufferedWriter( new FileWriter("ebxml.xml"));
……
bw.write("<eb:From>");
bw.write("<eb:기업Id>"+ from+ "</eb:기업Id>");   bw.write("</eb:From>");
bw.write("<eb:To>");
bw.write("<eb:기업Id>"+ to+ "</eb:기업Id>");   bw.write("</eb:To>");
bw.write("<eb:CPAId>"+ from+ to+ "</eb:CPAId>");
bw.write("<eb:Manifest>");
……                  // 입력 받은 적재 문서 수 만큼 Reference 엘리먼트 생성
          for ( int i = 0 ; i < num; i++ ) {
                    int paynum = i+ 1;
                    String payid = "pay"+ paynum;
                    bw.write("<eb:Reference eb:id='"+ payid+ "'>");
                    bw.write("<eb:Schema eb:location='"+ temp1[i]+ "'/>");
                    bw.write("<eb:Description>"+ temp2[i]+ "</eb:Description>");
                    bw.write("</eb:Reference>");
          }
……
```

[그림 4.2 ebXML SOAP 봉투 생성 코드]

② 적재 문서 생성

본 논문에서는 적재 문서의 데이터 소스에 대해 두 가지를 고려하였다. 첫번째는 리파지토리의 SQL Server에 접속하여 해당 테이블을 질의하여 필요로 하는 컬럼 정보에 대해 XML 문서화 하는 것이고 두 번째는 기업 A 자체적으로 데이터베이스화 되어 있지 않은 플랫 파일로부터 데이터를 라인별로 읽어와 XML 문서화 하는 두 가지 방법이다.

- SQL Server 2000 데이터베이스의 XML 문서화

먼저 기업 B의 SQL Server에 접속하기 위해, 기업 A는 SQL Server 용 가상 디레터리를 만들어야 한다. SQL Server용 가상 디렉터리는 일반적 IIS 가상 디렉터리와 목적과 기능이 다른 것으로 MS-SQL Server 프로그램 그룹의 'IIS의 SQL XML 지원 구성'을 통해 설정한다[16,17]. 본 논문에서는 설정과 관련하여 템플릿에 대한 가상 이름 설정에 대해 언급하겠다. 템플릿 가상 이름은 본 논문의 시스템 수행 시 생성되는 템플릿 파일들의 위치에 대해 물리적 경로를 설정해 준다.

설정을 마치면 기업 A는 리파지토리의 SQL Server 데이터에 접근할 수 있게 된다. 이때 기업 A는 리파지토리가 공개하는 데이터베이스의 테이블을 XSL을 통해 웹 브라우저에서 확인할 수 있으며 확인된 테이블을 통해 기업 A의 사용자는 자사에서 필요로 하는 정보에 대해 질의를 입력한다. 이때 질의는 템플릿 파일에 저장되며 결과적으로 질의를 담고 있는 템플릿 파일을 통해 리파지토리의 데이터베이스로부터 기업 A가 원하는 데이터를 XML 문서화 할 수 있다. 다음 그림은 기업 A의 질의를 받아 그에 대한 템플릿 파일을 생성하고 질의 결과를 XML 문서화 하는 작업을 설명하고 있다.

```
import java.io.*;

String select = request.getParameter("select");     // 입력 받은 Query
String tfrom = request.getParameter("tfrom");
String where = request.getParameter("where");
……
try {   BufferedWriter qr = new BufferedWriter( new FileWriter("auto.xml"));

          qr.write("<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>");
          qr.write("<sql:query>");
          qr.write(" SELECT "+ select);
          qr.write(" FROM "+ tfrom);
          if ( where.equals("null") ) qr.newLine();     // WHERE 조건이 없을 경우
          else qr.write(" WHERE "+ where);
          qr.write(" FOR XML AUTO");
          qr.write("</sql:query>");
          qr.write("</ROOT>");
          qr.close();
}
……
```

[그림 4.3 템플릿 파일 생성 코드]

```
<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>
<sql:query>       SELECT id, product_name, price
                  FROM ebxml_product
                  WHERE recyclable_pack = 1
                  FOR XML AUTO
</sql:query> </ROOT>
```

[그림 4.4 생성된 템플릿 파일]

● 플랫 파일로부터 XML 문서화

플랫 파일은 기업 A의 데이터베이스화 되지 않은 텍스트 문서들을 의미한다. 텍스트 파일에 대한 XML 문서화는 텍스트 파일로부터 라인별로 데이터를 읽어 들여 각 라인에 대해 태깅을 해주는 방법으로 구현한다.

```
import java.io.*;
import java.lang.*;

String doc = request.getParameter("doc");     // 입력받은 플랫 파일의 경로
String tag = request.getParameter("tag");      // 엘리먼트 이름
try {

PrintStream ods = new PrintStream(new FileOutputStream("fromtext.xml"));
FileReader file = new FileReader (doc);
BufferedReader buff = new BufferedReader (file);
ods.println("<invoice>");
boolean eof = false;
        while (!eof) {                          // 파일의 라인 끝이 아닐 동안
        String line = buff.readLine();          // 각 라인별로 읽어와서
        if ( line == null ) eof = true;
        else   {
                System.out.println(line);       // XML 태깅
                ods.println("<"+ tag+ ">"+ line+ "</"+ tag+ ">"); }
        }
        ods.println("</invoice>");
        buff.close();
……
```

[그림 4.5 플랫 파일의 XML 문서화]

4.3.2 SMTP 프로토콜을 이용한 메시지 전송 모듈

메시지 전송 및 수신은 SMTP 프로토콜을 이용한 메일 어플리케이션 구현을 통해 실행한다. 메일 서버는 POP3(Post Office Protocol)를 지원하는 이화여대 mm 서버를 이용한다. 본 논문의 메일 송신 어플리케이션은 일반 메일 송신 어플리케이션과 크게 다르지 않다. 단 ebXML에서 지정한 표준으로 MIME을 사용해야 하는데 이는 다음과 같다. 메일 어플리케이션 구동 시 첨부 되는 파일을 지정하는 부분에서 SOAP 봉투와 적재 문서를 선택해 주는데 이때 SOAP 봉투와 적재 문서는 각각 같은 MIME을 사용한다. 그러면 수신자는 MIME 헤더에서 지정한 대로 start로 지정된 첫번째 첨부가 SOAP 봉투라는 사실을 확인할 수 있다. 그리고 SOAP 봉투와 적재 문서를 합한 전체 메시지는 MIME 헤더가 포장하며 다음 그림의 4.6에서 첫번째 파트에 해당한다. 또한 그림 4.6의 두 번째 파트는 SOAP 봉투와 적재 문서 각각에 대한 MIME을 지정하는 그림이다.

```
send("MAIL FROM: "+ mailfrom.getText());     // 이메일 송신자
send("RCPT TO: "+ mailto.getText());         // 이메일 수신자
send("MIME-Version: 1.0");                   // MIME 버전
send("SOAPAction: 'ebXML'");                 // SOAPAction

if (attachFiles.size() != 0) {               // ebXML 메시지 파일이 있을 때
…
// 전체를 포장하는 MIME 헤더, start를 SOAP 봉투로 지정
  send("Content-Type:  multipart/related;  boundary="+ '₩"'+ boundary+ '₩"'+
        "type='text/xml';");
  send("start='" +  file.getName());
  sent("--" +  boundary);
……
```

```
// SOAP 봉투와 적재 문서의 MIME 지정하는 부분
public void sendFile(File file) throws IOException {
   send(“Content-ID : “‘ +  file.getName() + “‘”");
   send("Content-Type: text/xml; charset=”UTF-8”");
…}
```

[그림 4.6 메시지 송신 어플리케이션 코드]

4.3.3 SMTP 프로토콜을 이용한 메시지 수신 모듈

메일 수신은 JavaMail API를 이용한다. 크게 Inbox의 폴더를 얻어오는 단계, 폴더를 여는 단계, 메시지를 폴더로부터 얻어오는 세 단계로 나누겠다.

```
Folder folder = store.getFolder(“INBOX”); // INBOX 폴더 가져옴
Folder.open(Folder.READ_ONLY);            // 폴더 열기
Message message[] = folder.getMessages();   //메시지를 폴더로
                                              부터 가져옴
```

[그림 4.7 메시지 수신 어플리케이션 코드]

얻어온 메시지 SOAP 봉투와 적재 문서는 시스템 하드의 특정 디렉터리에 저장시켜서 메시지 저장 모듈에서 파일을 읽어올 수 있도록 한다.

4.3.4 ebXML 메시지 저장 모듈

XML 스키마에서 다시 관계형 데이터베이스 모델링하는 과정을 자동화 시키고자 많은 연구들이 진행되고 있지만 아직 실용화 단계에 와 있지 못하다. XML은 계층적 구조를 가지고 있어 이를 2차원적 테이블의 집합인 관계형 정보로 표현하기에

는 한계를 지니고 있기 때문이다. 하지만 현재 또는 가까운 미래에도 관계형 데이터베이스를 사용하는 고객들이 많을 것이므로 XML을 관계형 데이터베이스에 저장하는 일이 필요하기 때문에 이번 절에서는 시스템에 저장되어 있는 수신된 SOAP 봉투와 적재 문서를 데이터베이스에 저장하는 방법을 구현한다. 먼저 SOAP 봉투의 경우에는 정해진 스키마가 있지만 적재 문서의 경우에는 문서에 따라 스키마가 다르기 때문에 이 부분의 자동화는 현재 각계의 연구 상황도 그러하고 아직 접근하지 못한 부분이다. 따라서 본 논문에서는 적재 문서를 한가지 스키마로 고정하고 고려하였다. SOAP 봉투와 적재 문서를 관계 데이터베이스에 저장하는 방법을 각각 달리하였는데 SOAP 봉투는 'MSXML 파서'를 사용하였으며 적재 문서는 SQL Server 2000의 OpenXML 구문을 사용하여 각각 데이터베이스에 저장하였다.

① SOAP 봉투의 데이터베이스 저장

MSXML 4.0 파서에서 지원하는 com.ms.xml.Document 클래스의 getText() 메소드를 SOAP 봉투의 XML 문서에서 텍스트 데이터 부분만 추출할 수 있다. SOAP 봉투인 auto.xml문서는 현재 시스템에 저장되어 있다. 이 문서에서 markup이 아닌 텍스트를 추려내어 데이터베이스에 저장하는 것이 목적이다[19].

[표 4.2 MSXML 파서 클래스 및 메소드]

MS 파서 클래스	동작
com.ms.xml.Element	XML 요소 처리
com.ms.xml.ElementFactory	파서에 내부
com.ms.xml.Attribute	요소의 애트리뷰트를 저장
com.ms.xml.Document	*XML 문서를 로드하고 처리*
com.ms.xml.ParseException	파싱 예외에러를 처리

메소드	동작
getText()	전체 문서의 markup이 아닌 텍스트 표현을 반환
load(URL)	주어진 URL에서 문서를 로드함

```
import com.ms.xml.ParseException   // MSXML 클래스 임포트
import com.ms.xml.Document         // MSXML 클래스 임포트
import java.net.*;                 // 파일 URL
import java.io.*;                  // SQL 파일 작성하기 위한 클래스
……
    filename = "file:////c://purchase_order//auto.xml";
    URL url = null;
        …
        try {
        d.load(url);
    }
        try {   …if ( d != null ) {
                        fw.write(d.getText());    // 읽어들인 파일로부터 태그
                        fw.newLine();             // 이외의 정보 읽어서 텍스트
                }                                 // 파일에 기록
```

[그림 4.8 XML 문서인 SOAP 봉투로부터 텍스트 만을 추출]

SOAP 봉투 문서에서 태그를 제외한 데이터가 텍스트 파일에 저장된다.

```
petitsfrance@hanmail.net
sushibar@ewha.ac.kr
petitsfrance@hanmail.netsushibar@ewha.ac.kr
2001129-1754-29144
NewOrder
2001129-1754-29144
2001-11-29T17:54Z
This payload is purchase order of our company.
This payload is the list of the people that ordered the items.
```

[그림 4.9 SOAP 봉투로부터 추출된 텍스트 데이터]

이 데이터를 데이터베이스의 테이블에 저장하기 위해 SQL을 만들어주고 이를 실행해 준다.

purchase_order 테이블에 위의 텍스트 데이터를 저장하기 위한 SQL 생성한 텟스트 파일로부터 라인별로 데이터를 읽어 들여 SQL 문 “insert purchase order values (...) “을 만들어 데이터베이스에 데이터를 저장할 수 있도록 한다.

```
import java.io.*
……
try {
PrintStream ods = new PrintStream(new FileOutputStream("soap_save.sql"));
FileReader file = new FileReader (doc);
BufferedReader buff = new BufferedReader (file);
ods.println("insert into purchase_order values (");
        boolean eof = false;
        while (!eof) {
                String line = buff.readLine();    //텍스트 데이터를 라인별로
                if ( line == null )  eof = true;                    //읽어옴
                else  {
                        ods.println("'" +  line+  "'");
                        if ( line != null ) ods.println(",");
                }
        }
        ods.println(")");
        buff.close();
} …
```

[그림 4.10 SQL 문 생성을 위한 코드]

insert purchase_order values (*petitsfrance@hanmail.net, sushibar@ewha.ac.kr,*
petitsfrance@hanmail.netsushibar@ewha.ac.kr, 2001129-1754-29144
NewOrder, 2001129-1754-29144, 2001-11-29T17:54Z,
This payload is purchase order of our company,
This payload is the list of the people that ordered the items.)

[그림 4.11 생성된 SQL 문]

② 적재 문서의 데이터베이스 저장

적재 문서의 데이터베이스 저장은 SQL Server 2000의 OpenXML 구문을 이용하는 방법으로 구현한다. 전송 받은 XML 문서를 바탕으로 table을 만들기 위해 테이블 이름과 애트리뷰트의 개수, 애트리뷰트 이름 및 타입 그리고 저장하고자 하는 XML 문서의 라인 수를 입력해준다. 이러한 입력된 정보를 바탕으로 프로그램에서는 XML 문서를 데이터베이스에 저장하기 위한 질의를 담고 있는 sql 파일과 템플릿 파일을 생성해 준다. 먼저 sql 파일은 입력된 테이블 이름으로 테이블을 생성하고 입력된 애트리뷰트 수와 XML 라인 수만큼 가상 데이터를 입력해 놓는다. 다음 그림은 프로그램을 통해 생성된 sql 문인데 XML 문서를 데이터베이스에 저장하는 과정을 설명하겠다.

```
.....
01. CREATE PROC sp_update_ebxml_product @empdata ntext
02. AS
03. DECLARE @hDoc int
04. EXEC sp_xml_preparedocument @hDoc OUTPUT, @empdata
05. UPDATE ebxml_product
06. SET
07. ebxml_product.id   = XMLebxml_product.id ,
08. ebxml_product.product_name = XMLebxml_product.product_name,
09. ebxml_product.price = XMLebxml_product.price
10. FROM OPENXML(@hDoc, '/ root/ebxml_product')
11. WITH ebxml_product XMLebxml_product
12. WHERE ebxml_product.id   =   XMLebxml_product.id
13. EXEC sp_xml_removedocument @hDoc
14. SELECT *
15. FROM ebxml_product
16. FOR XML AUTO 17. GO
```

[그림 4.12 XML 문서를 DB에 저장하기 위한 저장 프로시저 생성]

첫번째 라인의 저장 프로시저 sp_update_tablename은 파라미터로 받는 데이터를 데이터베이스에 업데이트 하는 프로시저이다. 후에 데이터베이스에 저장될 XML 문서를 입력 받고 템플릿 파일이 호출되면 그 때 업데이트를 수행한다. 네번째 라인은 이미 @empdata라는 이름으로 입력된 XML문서를 @hDoc 파라미터로 넘기고 이를 프로시저 sp_prepare_document를 이용해 XML 문서를 파싱하는 작업을 거친다. 이때 내부에서 DOM이 생성되면서 관계 데이터베이스에 저장될 수 있게 된다. 다음 그림은 html 폼으로부터 입력 받은 XML문서를 데이터베이스에 업데이트 해주는 템플릿 파일이다.

```
<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>
<SQL:header>
<SQL:param name='empdata'><ebxml_product/></SQL:param>
</SQL:header>
<SQL:query>
EXEC sp_update_ebxml_product @empdata
</SQL:query>
</ROOT>
```

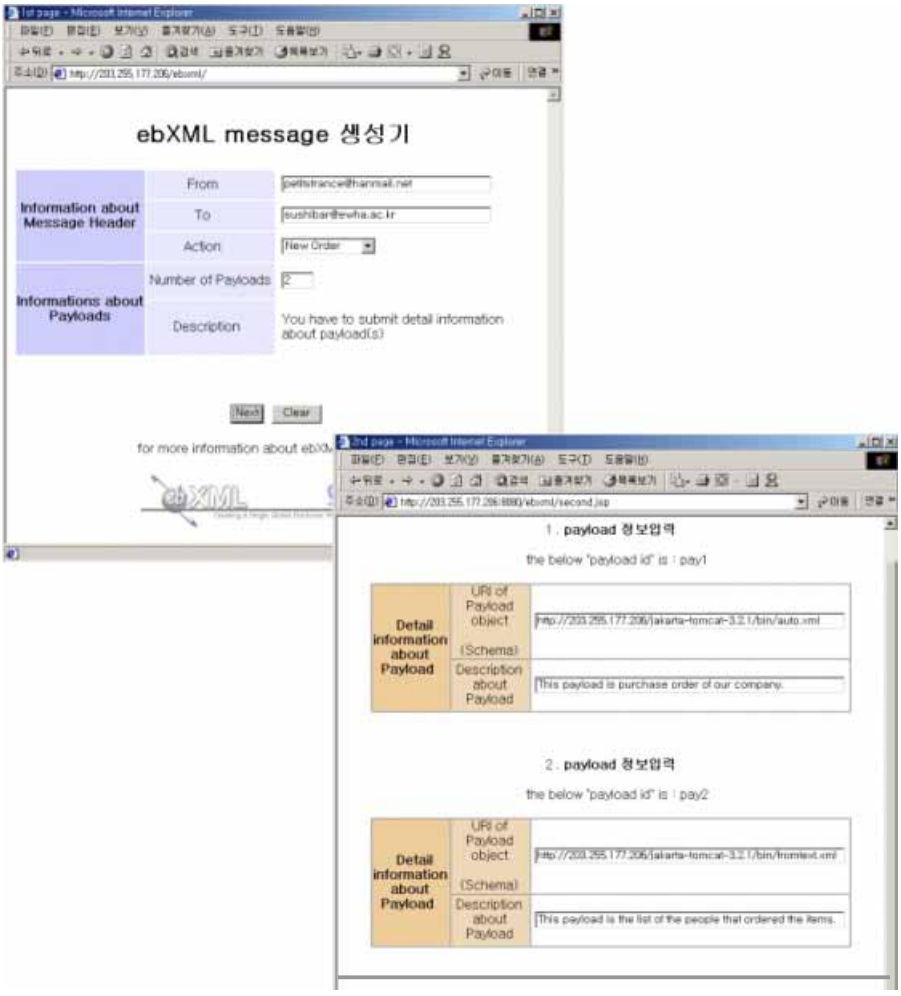
[그림 4.13 XML 문서를 DB에 업데이트 하기 위한 템플릿 파일]

4.4 구현된 시스템을 통한 예제

본 절에서는 기업 A가 주문 메시지를 생성하고 이를 SMTP프로토콜을 이용한 메일 송신 어플리케이션을 통해 전송, 이를 기업 B가 수신하고 데이터베이스에 저장하는 과정을 보이겠다.

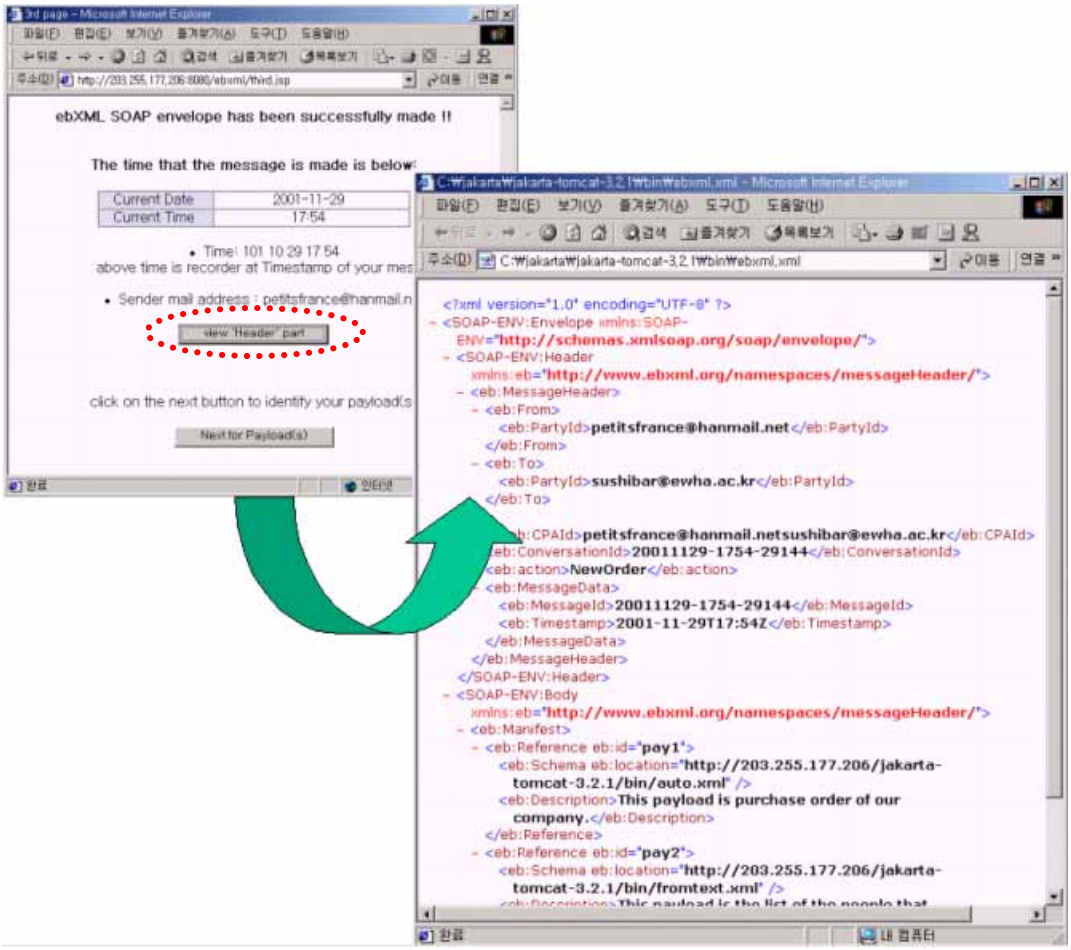
4.4.1. 기업 A의 메시지 생성 및 송신

다음 그림들은 메시지 생성을 위한 초기 화면부터 SOAP 봉투 생성까지를 보여주고 있다. 초기 화면으로 From과 To, Action 그리고 적재 문서의 수를 입력한다. 다음 화면에서는 적재 문서의 수를 입력한 만큼 적재 문서에 대한 정보를 입력하는 폼이 뜬다. 적재 문서에 대한 정보를 기록하면 주문을 시작한 시간을 기준으로 하여 SOAP 봉투가 생성된다.



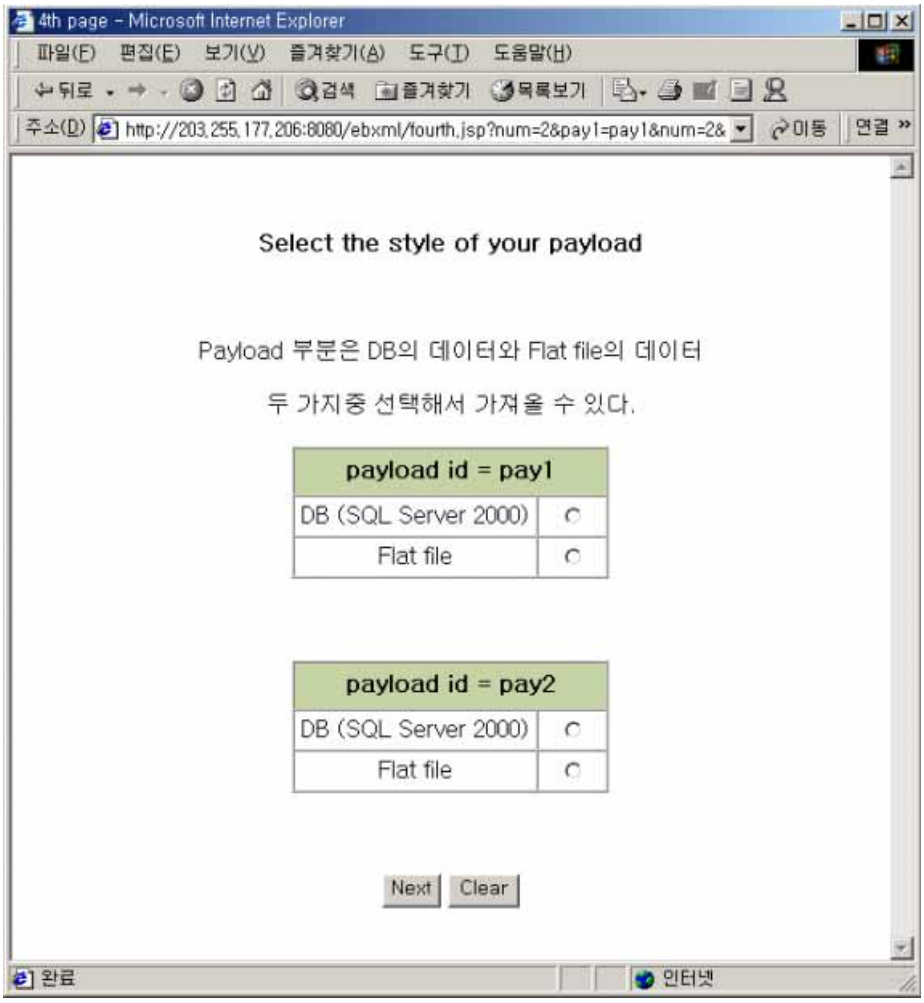
[그림 4.14 메시지를 생성하기 위한 화면]

다음은 SOAP 봉투가 성공적으로 생성되었음을 알려주고 view ‘Header’ part 버튼을 통해 생성된 XML 문서를 확인할 수 있다. 이때 나타나는 시간은 SOAP 봉투 문서의 Timestamp에 기록된다.



[그림 4.15 SOAP 봉투 생성을 확인]

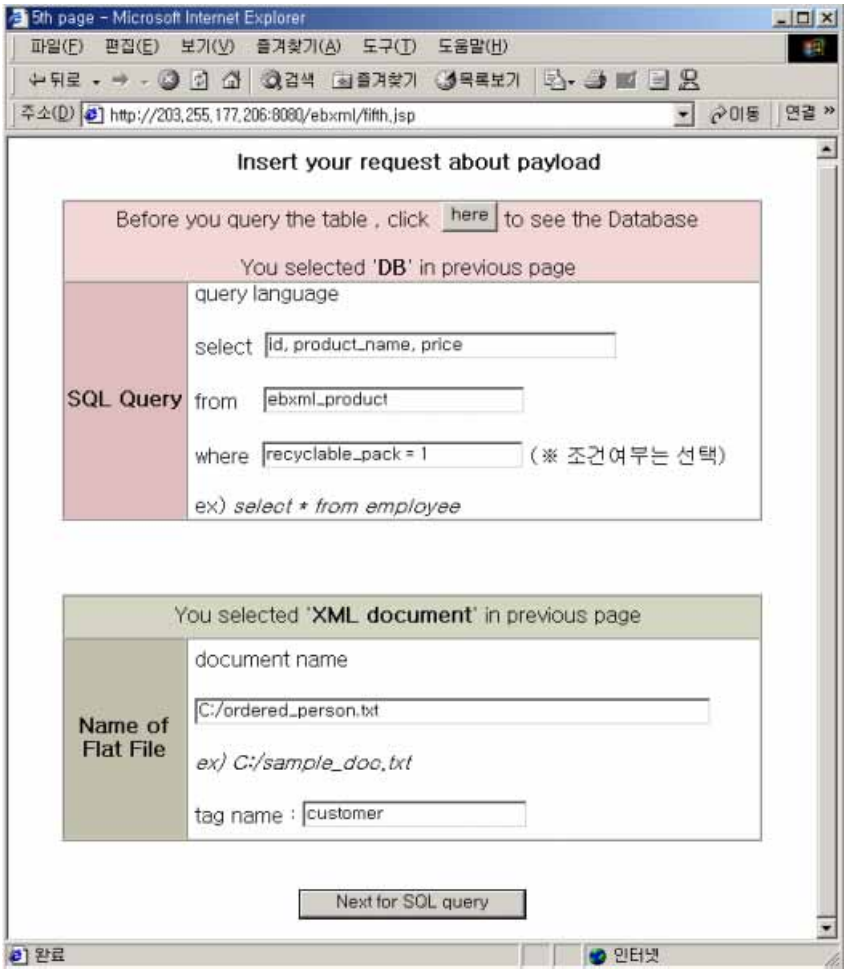
다음은 적재 문서를 생성하기 위해 데이터를 SQL Server에서 가져올 것인지 플랫 파일로부터 가져올 것인지를 선택한다. 예로 첫번 째 적재의 데이터 소스는 SQL Server로부터 가져오는 것으로 하고 두 번째 적재의 데이터 소스는 플랫 파일로부터 가져오는 것으로 하겠다.



[그림 4.16 데이터 소스 선택]

각각을 선택하면 다음 페이지에서 질의를 할 수 있는 폼이 나타난다. 먼저 기업 B에서 제공하는 데이터베이스의 테이블을 확인하겠는데 'here' 단추를 클릭하면 템플릿 파일에 담겨 있는 질의어 Select * from ebxml_product 를 XSL 파일을 적용시킨 결과를 웹 브라우저 상에 보여준다. 질의는 그림에서 보이는 바와 같이 "SELECT id, product_name, price FROM ebxml_product WHERE recyclable_pack = 1"과 같이 주겠다. 다음에 나오는 그림 4.17은 그림 4.16에서 SQL Server를 선택했을 경우에

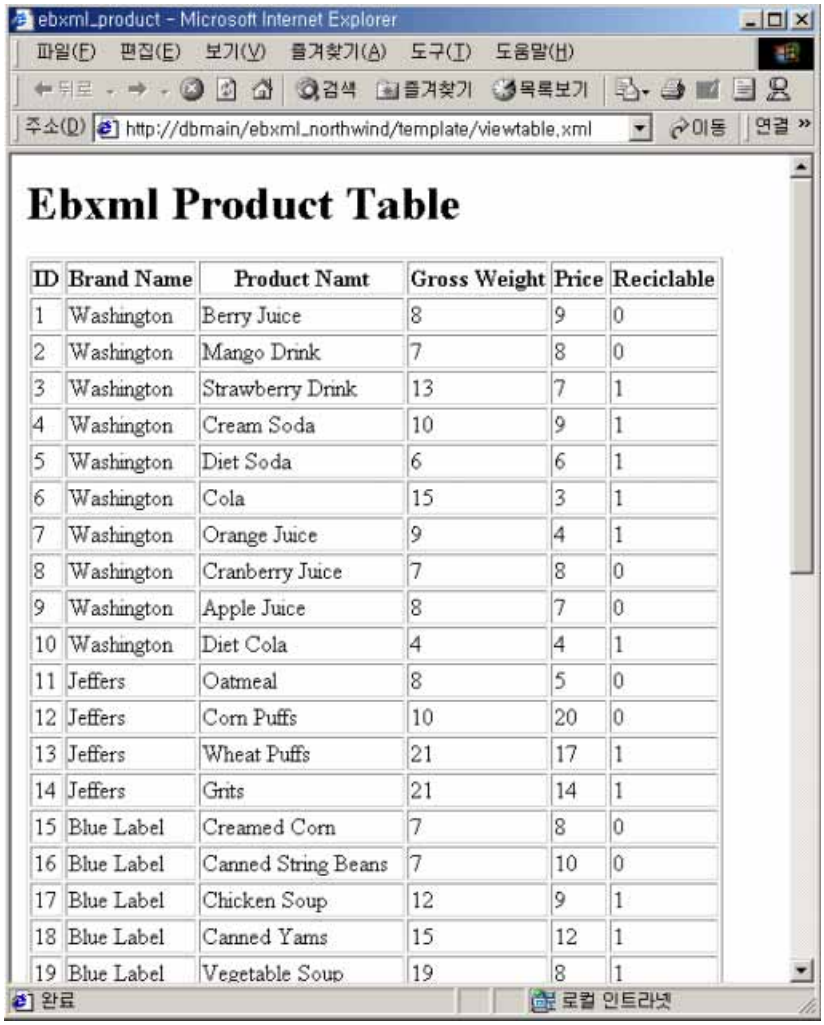
질의하는 폼이며 그림 4.18은 전체 테이블을 나타내고 있다.



[그림 4.17 데이터베이스를 질의하기 위한 폼]

한편 데이터소스를 플랫파일로 선택한 두 번째 적재에 대해서는 사용자가 직접 텍스트파일의 경로를 지정해 주고 라인별로 읽히게 될 텍스트 파일의 태그 이름을 지정해 주게 되어 있다. 예에서는 C:/ordered_person.txt 라는 파일을 읽어오겠다. 이는 기업 A의 기업 B에 대한 주문자 명단으로 라인별로 읽은 주문자에 대해 XML 엘리먼트 이름은 customer라고 지정해 준다. 그림 4.19와 그림 4.20은 생성된

XML 적재 문서이다.



Ebxml Product Table

ID	Brand Name	Product Namt	Gross Weight	Price	Reciclable
1	Washington	Berry Juice	8	9	0
2	Washington	Mango Drink	7	8	0
3	Washington	Strawberry Drink	13	7	1
4	Washington	Cream Soda	10	9	1
5	Washington	Diet Soda	6	6	1
6	Washington	Cola	15	3	1
7	Washington	Orange Juice	9	4	1
8	Washington	Cranberry Juice	7	8	0
9	Washington	Apple Juice	8	7	0
10	Washington	Diet Cola	4	4	1
11	Jeffers	Oatmeal	8	5	0
12	Jeffers	Corn Puffs	10	20	0
13	Jeffers	Wheat Puffs	21	17	1
14	Jeffers	Grits	21	14	1
15	Blue Label	Creamed Corn	7	8	0
16	Blue Label	Canned String Beans	7	10	0
17	Blue Label	Chicken Soup	12	9	1
18	Blue Label	Canned Yams	15	12	1
19	Blue Label	Vegetable Soup	19	8	1

[그림 4.18 XSL을 통해 보여준 기업 A의 테이블]

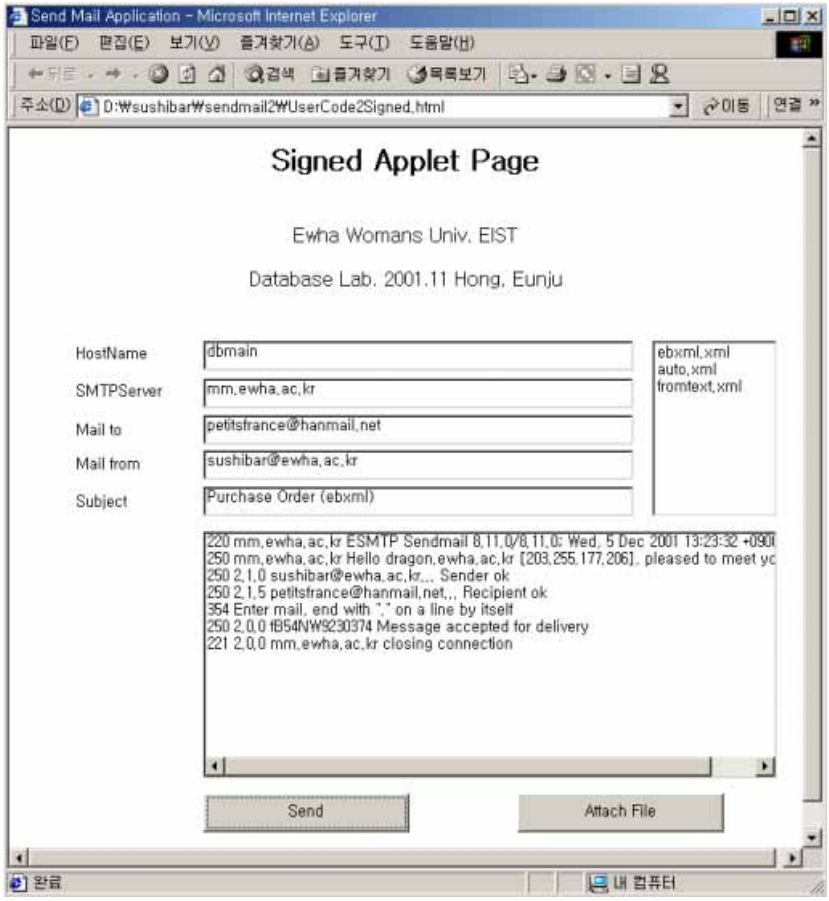
```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<ebxml_product id="3" product_name="Strawberry Drink      " price="7"/>
<ebxml_product id="4" product_name="Cream Soda            " price="9"/>
<ebxml_product id="5" product_name="Diet Soda             " price="6"/>
<ebxml_product id="6" product_name="Cola                  " price="3"/>
<ebxml_product id="7" product_name="Orange Juice          " price="4"/>
<ebxml_product id="10" product_name="Diet Cola            " price="4"/>
<ebxml_product id="13" product_name="Wheat Puffs          " price="17"/>
<ebxml_product id="14" product_name="Grits                " price="14"/>
<ebxml_product id="32" product_name="String Cheese        " price="1"/>
<ebxml_product id="35" product_name="Chicken Hot Dogs     " price="8"/>
……
</ROOT>
```

[그림 4.19 SQL 질의 결과 생성된 XML 적재 문서]

```
<invoice>
<customer>Sheri Nowmer</customer>
<customer>Derrick Whelply</customer>
<customer>Jeanne Derry</customer>
<customer>Michael Spence</customer>
<customer>Maya Gutierrez</customer>
<customer>Robert Damstra</customer>
<customer>Rebecca Kanagaki</customer>
<customer>Kim Brunner</customer>
<customer>Brenda Blumberg</customer>
……
</invoice>
```

[그림 4.20 텍스트 파일로부터 생성된 XML 적재 문서]

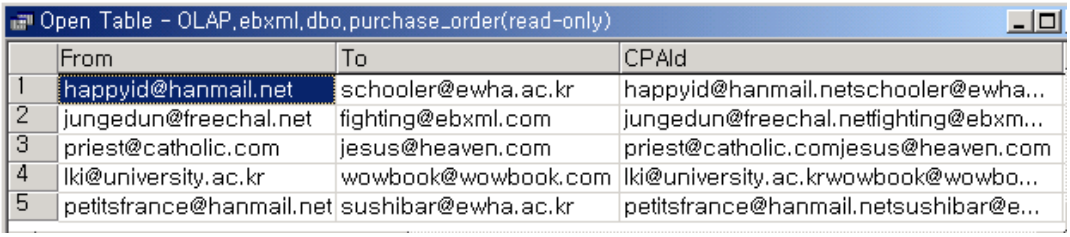
다음은 메일 전송 어플리케이션이다. 첨부되는 ebXML 메시지 즉, XML 문서는 프로그램 내부에서 각각 같은 MIME으로 포장되도록 지정해 두었기 때문에 메시지 수신자와의 약속을 위해 첫번째 첨부파일은 SOAP 봉투를 그리고 두 번째 이후는 적재 문서에 해당하는 파일을 첨부한다. 그리고 이 메시지 전체에 대해서는 MIME 헤더로 포장이 된다. 여기에서 사용자가 입력할 사항은 SMTP Server와, 받는 사람, 보내는 사람, XML 메시지이다. SMTP Server는 POP3를 지원하는 메일 서버로 'mm.ewha.ac.kr' 을 사용하였다.



[그림 4.21 Send Mail 어플리케이션을 이용해 메시지 전송]

4.4.2 기업 B의 메시지 수신 및 저장

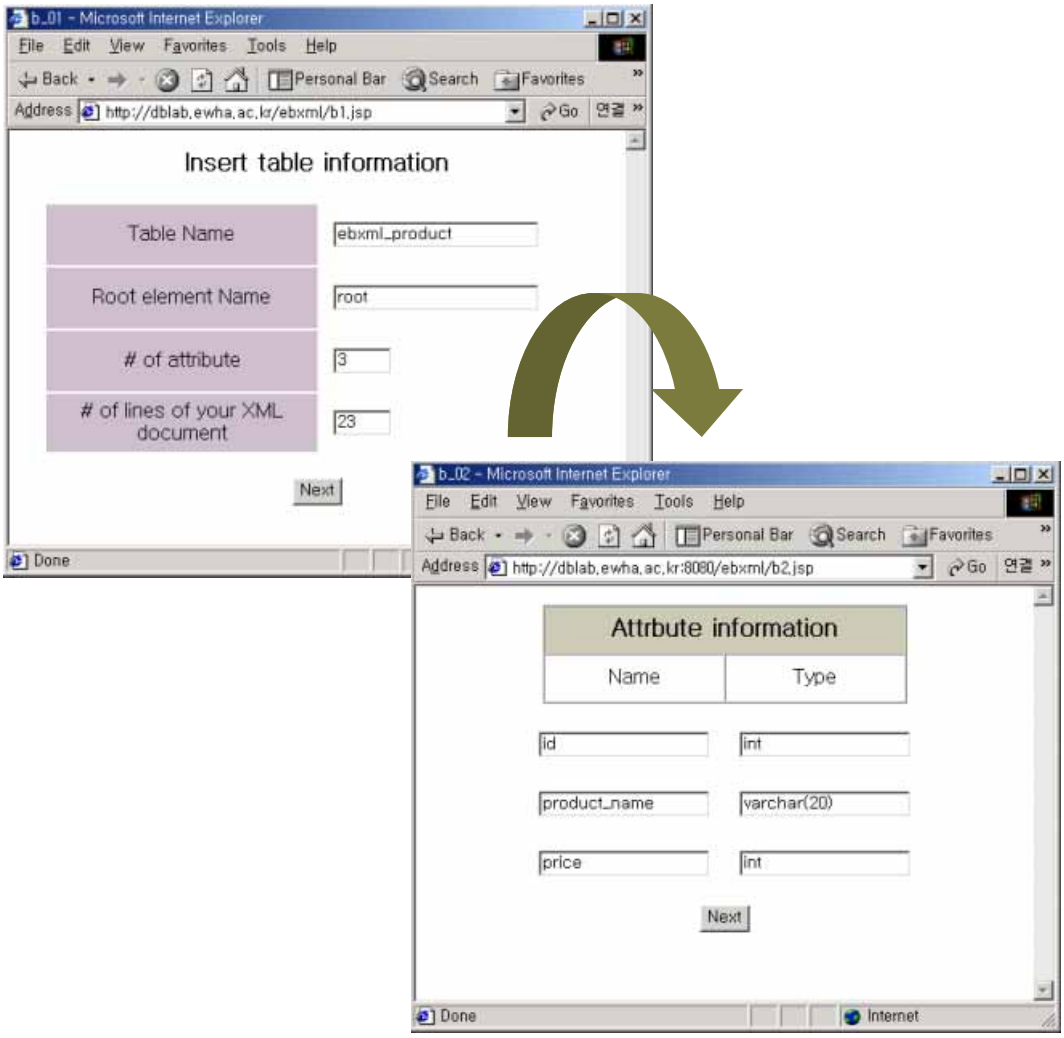
기업 B는 메일 수신 어플리케이션을 통해 SOAP 봉투와 적재 문서를 파일 형태로 시스템의 하드에 저장한다. 예에서는 C:/Purchase_Order/ 위치에 저장하는 것으로 하겠다. 먼저 SOAP 봉투는 프로그램 내부에서 파서를 통해 추출된 텍스트 데이터를 테이블에 insert하는 질의로 데이터베이스에 저장한다. 저장된 테이블은 다음 그림과 같다.


Open Table - OLAP.ebxml.dbo.purchase_order(read-only)

	From	To	CPAId
1	happyid@hanmail.net	schooler@ewha.ac.kr	happyid@hanmail.netschooler@ewha...
2	jungedun@freechal.net	fighting@ebxml.com	jungedun@freechal.netfighting@ebxm...
3	priest@catholic.com	jesus@heaven.com	priest@catholic.comjesus@heaven.com
4	lki@university.ac.kr	wowbook@wowbook.com	lki@university.ac.krwowbook@wowbo...
5	petitsfrance@hanmail.net	sushibar@ewha.ac.kr	petitsfrance@hanmail.netsushibar@e...

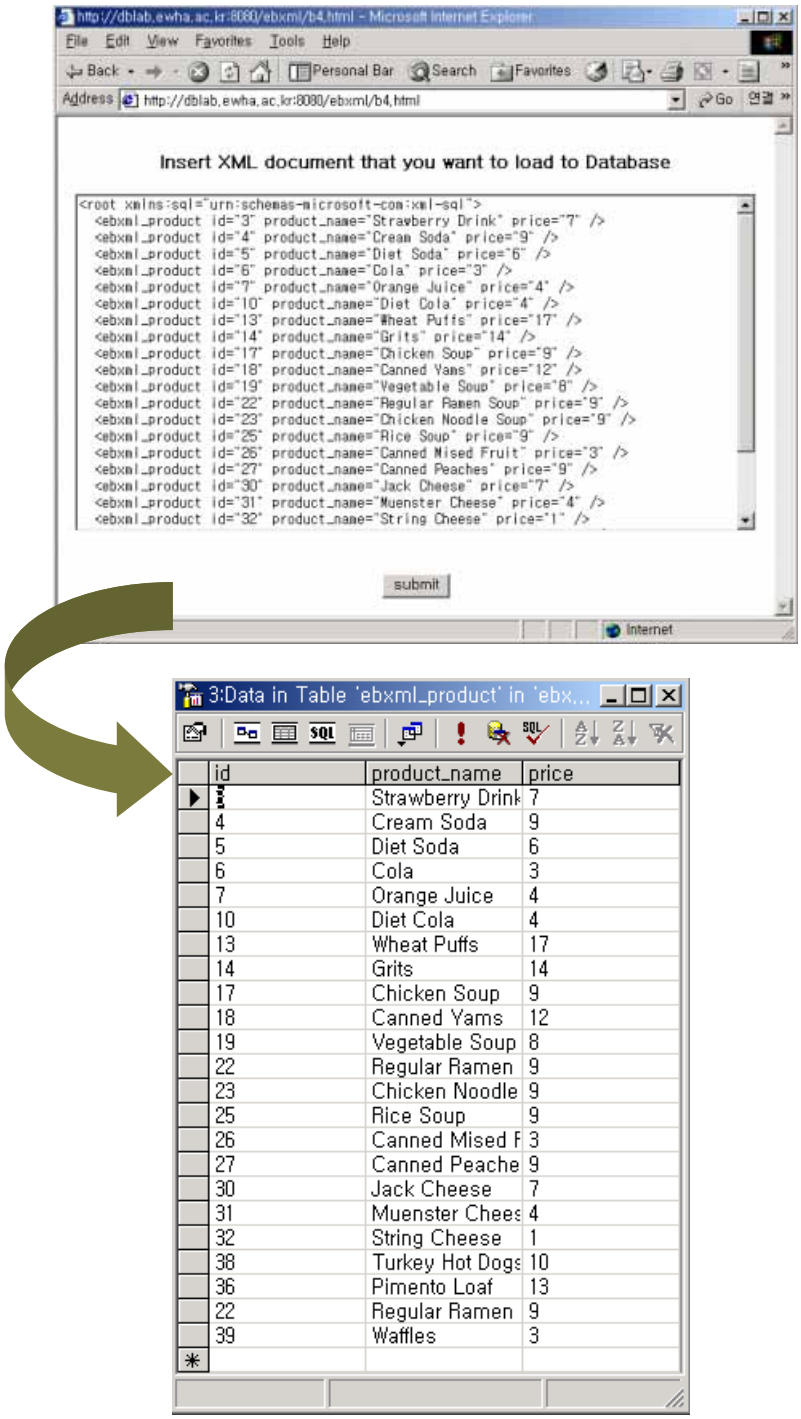
[그림 4.22 테이블에 저장된 SOAP 문서]

다음은 적재 문서를 데이터베이스에 저장하기 위한 화면이다. 적재할 XML 문서의 루트 엘리먼트와 이 문서의 라인 수, 그리고 생성할 테이블 이름, 생성할 테이블의 애트리뷰트 수를 각각 입력한 후에 다음 화면에서 테이블 정보를 상세히 입력한다. 테이블 이름은 ebxml_product 그리고 애트리뷰트는 각각 id int, product_name, varchar(20), price int 와 같이 입력한다. 테이블 디자인을 마치면 프로그램 내부에서 입력한 내용에 해당하는 sql 질의와 XML문서를 데이터베이스에 입력하기 위한 XML 템플릿 파일을 생성한다.



[그림 4.23 적재 문서를 저장할 테이블 디자인]

그리고 html 폼에 XML 문서를 입력하고 submit 버튼을 누르면 템플릿 파일을 통해 sp_update_tablename 프로시저가 수행되면서 폼에 입력된 XML 문서가 데이터베이스에 저장된다. 다음은 XML문서를 html 폼에 입력한 후에 데이터베이스에 저장된 데이터를 보여준다.



[그림 4.24 XML 문서를 입력하기 위한 html 폼과 테이블에 입력된 예]

V. 결론 및 향후 과제

기업간 거래는 주고 받는 메시지를 통해 이루어지는데 메시지는 카탈로그, 견적서, 주문서, 송장, 영수증 등 다양하게 존재한다. 컴퓨터가 도입되면서 EDI가 기업간 거래의 주 기술로 사용되었는데, 최근 확장성과 호환성에 대한 기술의 필요성이 대두되면서 이를 만족하는 XML 기술이 도입되었으며 전자 거래의 표준화를 형성하기 위해 ebXML 단체가 출범을 하여 전세계를 하나의 전자 거래 기반으로 통일하겠다는 목표를 가지고 있다.

본 논문에서는 ebXML 워킹 그룹중 하나인 TRP에서 지난 2001년 5월에 발표한 ebXML 메시지 표준 1.0 버전을 기반으로 기업간 전자 거래 시스템을 구현하였다. 송신측 기업에서는 ebXML 표준에 근거하여 주문서 등과 같은 메시지를 생성하고 통신 프로토콜을 이용하여 이를 전송한다. 수신측 기업에서는 전송 받은 메시지를 디스크에 저장하여 문서를 분리하여 데이터베이스에 저장하여 관리한다.

본 논문에서 제안한 한 기업간(B2B) 전자 거래 시스템은 기존의 전자 거래 방식과 비교하여 다음과 같은 특징을 갖는다.

- 기업간에 교환하고자 하는 문서 즉, XML 문서에 대해 동일한 메시지 포장 기술을 사용함으로써 표준화 문제를 해결하여 단지 두 기업뿐만 아니라 전세계의 기업과 거래하는데 문제점이 없어졌으며 기존의 EDI 표준과 비교할 때 빠르고 간편하다.

- 교환 문서의 데이터를 관계형 데이터베이스로부터 추출하여 XML문서화 하였으므로 대부분의 기업들이 자체적으로 다수 보유하고 있는 데이터베이스에 대해 변경해 줄 필요가 없으며 데이터베이스화 되지 않은 텍스트 문서도 XML문서로 변경 가능하게 해주어, XML을 기반으로 채택하지 않은 기업들이 XML 기반으로 옮겨가는데 부담을 줄였다.
- 관계형 데이터베이스를 XML문서로 변경하는 부분에 대해 SQL Server 2000의 기능을 이용함으로써 변경 비용을 줄일 수 있다.
- 통신 프로토콜은 SMTP를 이용하여 user friendly 하다.

향후 연구 과제로는 메시지를 수신한 기업에서 메시지를 자체 데이터베이스에 저장하고 처리하기 이전에 메시지 수신 여부 및 에러 발생 여부를 송신 기업 측에 보내줌으로써 B2B 거래가 안정적으로 유지 및 관리 되도록 하는 기술에 대한 구현을 고려하고 있는데 이 시나리오는 TRP 워킹 그룹에서 표준화를 해 놓은 내용이다. 또한 본 논문에서 사용한 통신 프로토콜은 SMTP인데 이는 비동기식 통신만을 지원하고 있기 때문에 ebXML에서 권장하는 또 다른 통신 프로토콜인 HTTP를 사용하여 실시간 메시지 송수신을 실현하는 것이다.

참고 문헌

[1] 김형도, “ebXML의 필요성과 비즈니스 요구사항” , e-Commerce Journal, October 2000

[2] *White Paper*, Enabling Electronic Business with ebXML
http://www.ebxml.org/white_papers/whitepaper.htm

[3] Jayavel Shanmugasundaram, Eugene Shekita, “Efficiently Publishing Relational Data as XML Documents” In Proc. of the VLDB, 2000.

[4] Michael Carey, Jayavel Shanmugasundaram, “XPERANTO: A Middleware for Publishing Object-Relational Data as XML documents”, In Proc of the VLDB, 2000.

[5] Mary Fernandez, Wang-Chiew Tan, “SilkRoute: Trading between Relations and XML” In Proc of the Int. WWW Conference, 2000.

[6] Volker Turau, “Making legacy data accessible for XML application”, FH Wiesbaden,May 1999.

[7] Jayavel Shanmugasundaram, Kristin Tufte, “Relational databases for querying XML documents : Limitations and Opportunities”, In Proc. of VLDB 2000.

[8] 김형도, “ebXML TRP 프로젝트”, e-Commerce Journal, December 2000

[9] 김형도, “B2B 기업의 거래능력 정의 및 기업간 거래규약 합의/운영에 관한 ebXML 표준화”, e-Commerce Journal, February 2001

[10] 김형도, “ebXML 등록기/저장소 데이터 모델과 서비스 인터페이스”

e-Commerce Journal, March 2001

[11] http://www.ebxml.org/specs/ebMS.pdf, Message Service Specification . ebXML Transport, Routing & Packaging Version 1.0, 11 May 2001

[12] 김채미, 최학열, 김심석, *마이트 Press*, "전문가와 함께 가는 XML Camp", 2001

[13] http://www.w3c.org/TR/2000/NOTE-SOAP-20000508
Simple Object Access Protocol (SOAP) 1.1 W3C Note 08 May 2000

[14] http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
SOAP Messages With Attachments W3C Note 11 December 2000

[15] http://www.rosettanet.org/, RosettaNet

[16] Kevin Williams, *Wrox*, "Professional XML Databases", pp 533-805, 2000

[17] http://msdn.microsoft.com/library/psdk/xmlsdk/xmlp91b9.htm
XML and Internet Support Overview (msdn online)

[18] http://java.sun.com/products/javamail/javadocs/
JavaMail 1.1 API documentation

[19] http://xml.coverpages.org/ni2001-07-21-b.html
Microsoft Release MSXML Parser 4.0 Beta 2

ABSTRACT

Design and Implementation of the electronic transaction system based on ebXML message specification

Department of Computer Science & Engineering

Ewha Institute of Science and Technology

Hong, Eun Ju

Transactions between companies are accomplished by exchanging messages. Because we use communication technology, extensibility and compatibility would be needed for accomplishing message exchange on the WWW. XML is perfect technology to satisfy this demand.

But single standard had not been formed. Moreover, different suggestions have been made even in the same business area. ebXML is an initiative to gratify this need and it has a goal to unify the electronic business of the whole world in a single standard.

In this paper, we design and implement of the electronic transaction system based on ebXML message specification. A message that is to be sent is composed of header container and payload container and those are packed by

MIME. A company that wants to send the message extract the XML document used as a purchase order from relational database. Then this company sends message using SMTP protocol to the other company that process it. When the other company receives the message, it separates the message into SOAP envelope and payload document. Then these data can be stored in the relational database.

With this ebXML based electronic transaction system that we proposed in this paper, we could solve the standard problem because we used a single standard that is agreed by all companies already. And we have a view that this standard would be used actively in all electronic transaction area.

감사의 글

먼저 많이 부족하지만 저를 지도해 주시고 한 편의 논문이 나올 수 있도록 조언을 아끼지 않으신 용환승 교수님과 논문 심사를 맡아주신 이기호 교수님과 박승수 교수님께 깊은 감사의 마음을 전합니다. 또한 항상 넘치는 에너지로 학생들에게 많은 자극을 주시고 큰 꿈과 포부를 지닐 수 있도록 해주셨던 김원 박사님과 지난 2년의 대학원 생활동안 많은 것을 새로 얻고 깨달을 수 있도록 가르침 주신 컴퓨터학과 모든 선생님께 감사 드립니다.

연구실에서 오랜 시간 같이 보내지 못해 아쉬운 호숙 언니, 나의 대녀 혜림이 엄마인 (김)은정 언니, 예쁘고 언제나 도움을 주려 했던 (권)은정 언니, 힘들 때 조언을 아끼지 않았던 성은 언니, 항상 씩씩하고 시원스러운 면이 부러웠던 주영이, 그녀의 단짝이나 나의 대학 동기인 백선이, 얼굴 몇 번 보지 못한 눈이 동그란 인기, 우리 연구실의 최고 기분파인 보영이, 항상 무언가를 열심히 하느라 분주한 은주, 밥 먹는 모습이 보고싶은 나영이, 보영이 못지않게 엉뚱한 선경이, 그리고 나와 지난 2년을 같이한 대학원 동기 현정, 지현, 현숙이 에게도 감사의 마음을 전합니다.

같이 논문을 쓰며 여러모로 위안이 많이 되었던 새롬이와 힘들 때 마다 적절한 조언을 해준 진한 동기 주영이, 그리고 얼마 전에 결혼한 지선이, 항상 파이팅을 외치며 잘 하라고 격려해준 희정 언니, 논문 쓰는데 필요한 조언을 해준 태연이와 경일이, 잘 해내라고 기도해준 수원 언니와 그 외 성당 전례단 사람들, 그리고 대학 동기들 선영, 충민, 화민, 문기, 화진, 윤희 모두에게 고맙습니다.

마지막으로 저를 항상 사랑하고 아껴주고 믿어주시는 부모님과 착한 동생 정기, 그리고 지금까지 힘들었던 순간을 이겨낼 수 있도록 해주시고 저의 모든 것을 주관하시는 하느님께 감사 드립니다.