관계 DBMS에 기반한 UDDI 3.0 레지스트리의 개발

A Development of UDDI 3.0 Registry based on Relational DBMS

유수진(Su-Jin You)*, 이경하(Kyong-Ha Lee)**, 이규철(Kyu-Chul Lee)***

초 록

최근에 들어 웹 서비스를 이용한 e-비즈니스를 활용사례가 증가하고 있으며 또한 많은 기업들이 웹 서비스를 이용한 e-비즈니스 시스템의 개발과 보급에 많은 노력을 기울이고 있다. 이렇게 웹 서비스를 개발하는 기업들의 가장 큰 관심사는 개발한 비즈니스 서비스를 사용자가 쉽게 검색하고, 이용하도록 하는 데 있으며, 이에 따라 웹 서비스의 등록과 검색의 지원을 위한 표준을 필요로 하게 되었다. 이러한 요구에 따라 OASIS에서는 웹 서비스에 대한 레지스트리 표준인 UDDI를 정의하게 되었으며, 최근에는 3.0 표준을 공개하였다. 그러나 이 표준의 복잡성으로 인하여, 아직 국내에서는 UDDI 3.0 표준을 지원하는 레지스트리 시스템이 존재하지 않고 있으며, 외국에서도 웹 서비스 표준에 주도적인 역할을 수행해 온 몇몇 업체만이 테스트 버전만을 공개해 놓은 상태이다. 이에 따라 본 논문에서는 UDDI 3.0 표준을 지원하는 레지스트리 시스템을 개발하였다. 개발된 시스템은 SQL 표준을 지원하는 관계형 DBMS를 지원하고 있으며, UDDI의 모든 기능을 충실히 지원함으로써 웹 서비스 기반의 e-비즈니스 프레임워크에서 중요 기반 기술로 활용될 것으로 기대된다.

ABSTRACT

Recently, it is to increase rapidly in people who do a private business through Web service. So, many companies have been developing part of businesses using Web Services. The greatest interest of companies developing Web Services was that users were easily discovering and using information on theirs. So, companies have been taking interest in a standard that supports registrations of information on Web service and discoveries of a registered Web services. Therefore UDDI 3.0 standard of the registry for Web service standardized by OASIS is in the spotlight. But, because of the complexity of it, there are no implementations developed in the inside of the country yet. And a development activity of it is faint. In outside of the country, IBM, Microsoft and SAP has developed it, but they do not perfectly support a standard of UDDI 3.0.

This paper shows the development of the Registry System which supports perfectly a standard of UDDI 3.0.

키워드 : UDDI, 웹 서비스, e-비즈니스, XML
UDDI, Web services, e-Business, XML

본 연구는 대학 IT 연구센터 육성 지원 사업의 연구결과로 수행되었음.

* 오름정보 지식정보화사업본부
** 충남대학교 대학원
***충남대학교 공과대학 교수

1. 서 론

최근에 웹 서비스를 통해 e-비즈니스를 수행하고자 하는 요구가 증가함에 따라 개발된 웹 서비스의 효과적인 검색과 이용의 필요성이 증대되어 왔다. 또한 웹 서비스는 많은 기업들에 의해서 개발될 수 있는 만큼 이들에 대한 등록과 검색은 특히 e-비즈니스 수행에 있어 중요한 기술 요소로 인식되고 있다. 이러한 이유로 IBM, Microsoft, SAP과 같은 IT 업체들은 UDDI.org라는 단체를 결성하고 웹 서비스를 위한 분산 레지스트리 표준인 UDDI(Universal Description, Discovery and Integration) 기술을 공동 개발하였으며, 웹 서비스 시장에 성공적으로 진입하게 되었다. 최근에 이 UDDI 기술은 OASIS로 표준화 활동이 이관되어 버전 2.0이 OASIS 표준으로 제정되었으며 3.0이 위원회 명세서로 공개되어 있다. 하지만 UDDI 3.0 표준은 많은 요구사항을 충족시키려다 보니 표준 자체 또한 방대하고 또한 복잡하게 되었다. 이러한 이유로 아직까지 국내에서는 개발된 사례가 전무하고 개발 활동도 아직까지 미비한 실정이다. 한편 외국에서는 UDDI 표준화 활동에 초기부터 참여해 왔던 IBM, SAP, Microsoft만이 3.0 버전을 지원하는 레지스트리를 개발, 공개하였으나 아직까지 테스트 버전에서 그치고 있으며 3.0 표준을 완벽하게 지원하지는 못하고 있는 실정이다. 이에 본 논문에서는 UDDI 3.0 표준을 충실히 지원하며 SQL 표준을 지원하는 일반적인 관계형 DBMS를 하부 저장 구조로 이용할 수 있는 레지스트리 시스템을 개발하고 이를 통해 웹 서비스 기반 e-비즈니스 프레임워크를 구축하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 UDDI 3.0으로 개발된 레지스트리 시스템의 개발 환경과 제공되는 기능 등에 대해서 알아보고, 3장에서는 UDDI 3.0 레지스트리 시스템의 설계와 구현 내용을 설명하도록 한다. 4장에서는 관련 연구를 통하여 현재까지 공개된 UDDI 3.0 레지스트리 시스템과 비교 분석을 통해 본 논문에서 개발한 레지스트리 시스템을 기능적으로 평가하도록 한다.

2. 관련 연구

현재 공용 UDDI 레지스트리 시스템은 IBM, Microsoft, SAP, NTT-com에서 운영 중이고 이 시스템들은 UDDI 2.0을 기반으로 개발되어 있다. 또한 IBM, Microsoft, SAP에서는 UDDI 3.0을 기반으로 한 레지스트리를 개발하여 현재 베타 테스트를 진행 중에 있다.

2.1 IBM의 UDDI 3.0 레지스트리 시스템

IBM사에서 개발, 발표된 3.0 레지스트리[1]는 UDDI 데이터의 저장을 위해 DB2 버전 8을 사용하고 있으며 레지스트리 서비스를 위한 웹 서버로는 IBM에서 개발한 WAS(WebSphere Application Server) 버전 5에서 구동되고 있다. 또한 개발 언어는 Java를 사용해 구현하였으며, 구현에 있어 XML 메시지를 파싱하기 위한 파서로 Apache Xerces 라이브러리, SOAP 메시지의 통신을 위해 Apache Axis 라이브러리를 사용한다. 이

레지스트리 시스템은 운영의 기반 시스템을 IBM사에서 개발한 시스템들을 이용하고 있어서 시스템에 종속적인 문제점이 있다.

비즈니스, 서비스, 기술 모델 및 비즈니스 개체간 관계성 등록 등의 기본적인 자료 구조들에 대해서는 등록, 수정, 삭제가 가능하다. 하지만 자료 구조의 등록에 있어서 3.0에 추가된 등록자 키 할당 및 전자 서명 지원 기능을 지원하지 않는다. 그리고 레지스트리의 특징으로는 비즈니스, 서비스, 기술 모델 정보의 등록 시 같이 입력될 수 있는 분류, 식별 코드의 값이 유효한 값인지를 검증하기 위해 외부 서비스에 요청할 때 사용하는 Value Set API의 기능을 UI(user interface)로 처리하였는데, 예를 들면 카테고리 정보 저장 시 트리 형태의 선택 메뉴에서 직접 값을 선택할 수 있게 하는 방식으로 Value Set API를 사용하지 않고 레지스트리 서버에 저장된 유효한 데이터 값에 대해서만 사용자가 선택할 수 있게 함으로써 잘못된 값의 입력을 피하고 있다.

UDDI에서 정의된 검색 방법은 일반, 상세, 브라우즈 검색 이 세가지가 있는데 IBM에서 개발된 레지스트리는 브라우즈 검색을 제외한 일반, 상세 검색을 지원하고 있다. 또한 상세 검색은 3.0에서 요구하는 모든 기능을 지원하고 있지 않다. 예를 들면 bindingSubset, combineCategoryBags검색 조건을 지원하지 않는다.

bindingSubset과 combineCategoryBags은 비즈니스, 서비스 및 기술 모델을 검색하는 Inquiry API의 설정 파라메터인 findQulaifer의 값이다.

2.2 Microsoft의 UDDI 3.0 레지스트리 시스템

Microsoft 사에서 운영 중인 Microsoft 레지스트리[3] 시스템의 기반은 DBMS로는 SQL 서버를, 웹 서버로는 IIS를 이용하고 있다. 또한 SOAP 메시지 통신을 위해 .NET Framework class library를 이용한다. 이 시스템은 IBM과 같이 Microsoft의 플랫폼군를 이용하고 있어서 시스템에 종속적인 문제점을 가지고 있다. 그리고 등록 기능의 구현 여부, 레지스트리의 특징들 모두 IBM 레지스트리 시스템과 동일하다.

반면에 검색은 UDDI에서 지원 요청하는 일반, 상세, 브라우저 검색 모두를 지원하고 있지만, 상세 검색은 UDDI 3.0 표준이 요구하는 모든 사항을 지원하지 않는다. 예를 들면 비즈니스 검색의 결과를 비즈니스 이름의 오름/내림 차순 또는 생성 날짜 오름/내림차순으로 상세히 설정을 할 수 있는데 비해 이 시스템은 검색 사용자가 상세히 환경 설정할 수 없도록 되어있다. 또한 비즈니스 상세 검색에서 식별자 검색은 지원하지 않는다.

2.3 SAP의 UDDI 레지스트리 시스템

현재 SAP사에서 운영중인 SAP 레지스트리[5] 시스템은 DBMS와 웹 서버 기능을 담당하는 SAP Web Application Server를 기반으로 하고 있다. 개발 언어는 Java를 사용해 개발되었으며, XML 파서와 SOAP 메시지 통신을 위해 SAP NetWeaver를 이용해 구현되어 역시 시스템에 종속적인 문제점을 가지고

있다. 이 공용 레지스트리는 UDDI 개체에 대한 등록하는 사용자의 인터페이스는 없고 SOAP access points를 통해서만 접근이 가능하여 많은 불편한 점이 있다. 또한 비즈니스 개체간의 관계성 등록 기능이 없어 비즈니스 개체간의 관계성 정보를 등록을 할 수 없는 문제점이 있다.

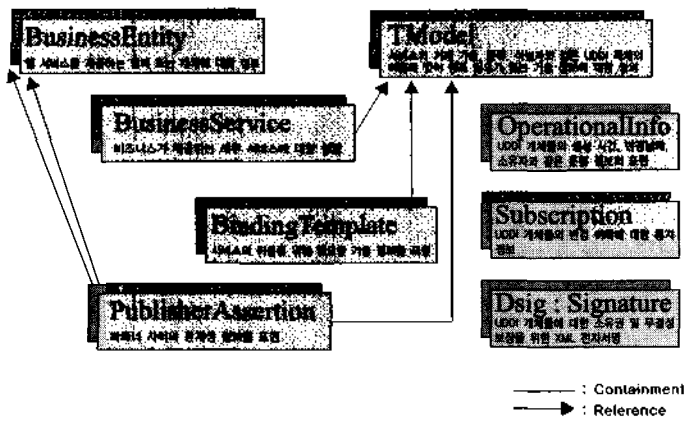
현재 SAP에서 지원하는 레지스트리의 검색 인터페이스는 UDDI 2.0 표준을 지원하는 인터페이스이다. 또한 이 검색 인터페이스는 비즈니스, 서비스, 기술모델에 대한 이름, 식별자 및 카테고리 조건에 대한 검색만 가능한 문제점을 가지고 있다.

3. UDDI 3.0 레지스트리 설계 및 구현

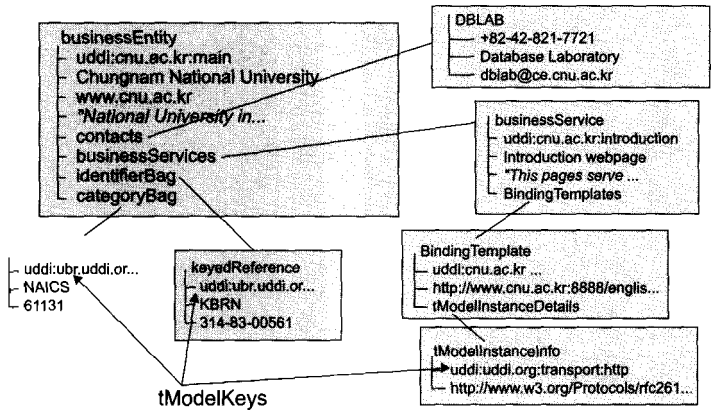
3.1 UDDI 3.0 자료 구조

UDDI 3.0의 자료 구조[4]는 레지스트리에 등록할 비즈니스와 그 비즈니스 정보를 표현하며 XML 스키마를 통해 정의되어 있다. UDDI 표준에서는 XML 스키마로 정의된 자료 구조를 기반으로 UDDI 정보를 기술하고 검증할 수 있도록 할 뿐만 아니라 다양한 데이터 타입을 제공한다는 장점으로 인하여 XML 스키마를 UDDI 메시지에 대한 스키마 정의 언어로 사용한다. 현재 UDDI 3.0의 자료 구조와 각 계층간 계층 관계는 〈그림 1〉과 같다.

〈그림 2〉는 위 자료 구조를 통하여 "Chungnam National University"라는 이름의 비즈니스 개체의 예를 보이고 있다. 이 비즈니스 개체는 "Introduction Webpage"라는 서비스를 제공하고 있으며, 이 서비스는 접근 방법으로 기술 모델로 정의된 바인딩 정보를 제공하고 있다. 비즈니스 개체에서 분류 체계와 식별자의 이용은 비즈니스 개체 자료 구조에 정의되어 있는 categor-yBag과 identifierBag을 통하여 정의한다. 또한 이 categoryBag과 identifierBag은 기술 모델을 이용하여 정의하고 있고 식별자 및 분류 체계의 이름을 지정하기 위한 keyName과 코드 값을 지정하기 위



〈그림 1〉 UDDI 3.0 자료 구조



〈그림 2〉 UDDI 3.0 데이터 예

한 keyValue로 구성된다.

위 그림에 나온 비즈니스 개체는 식별자로써 KBRN을 이용하고 있으며, 분류 체계로는 NAICS를 이용하고 있다. NAICS는 모든 UDDI 레지스트리가 반드시 제공해야 할 정규 tModel로 UDDI 레지스트리에 tModel의 한 인스턴스로써 저장되며, 이에 대해 유일한 URI를 tModelKey로 부여함으로써 여러 비즈니스 또는 서비스가 이를 참조하여 분류 체계 또는 식별자를 이용할 수 있도록 한다.

3.2 UDDI 데이터 저장

실제 UDDI 데이터 저장 시 이용할 수 있는 저장 하부구조로는 크게 파일, LDAP 및 데이터베이스로 분류할 수 있다.

파일 구조는 질의 처리와 다중 사용자 지원이 용이하지 않고 또한 데이터에 대한 일관성 유지하기 위하여 별도의 처리가 필요로 하는 단점이 있다.

LDAP 구조는 UDDI 3.0부터 다중어 지원이 필수 요소인데 다중어 지원이 안 되는 단점이 있다. 데이터베이스 구조는 위 두 구조가 가지고 있는 단점이 없고 또한 저장 된 데이터에 대한 백업 및 회복 기능도 있어 트랜잭션 처리 실패 시 복구가 쉬운 장점을 가지고 있다. 이에 본 논문에서는 데이터베이스 구조를 UDDI 데이터를 저장하는 구조로 삼았다.

또한 UDDI의 데이터를 저장하는 방법은 크게 eXcelon, Tamino와 같은XML 저장 DBMS을 이용하여 데이터를 저장하는 방법, XML 타입과 같은 전용 타입을 이용하여 데이터를 저장하는 방법 및 XML 스키마를 스키마의 구조와 의미를 쪼개어 기존의 관계형 데이터베이스의 테이블에 데이터를 저장하는 방법으로 분류할 수 있다.

첫번째 XML 전용 DBMS는 현재 구현 완료성 및 시스템 안정성에 대한 연구와 구현 사례가 없어 신뢰할 수 없는 문제점이 있다.

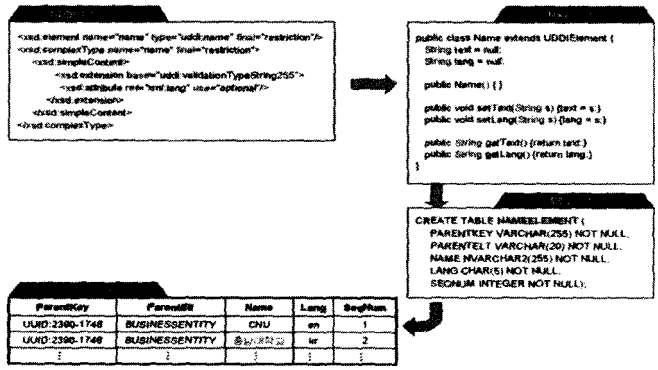
두 번째 XML 타입과 같은 전용 타입을 이용하여 데이터를 저장하는 방법은 오라클9i와 같이 기존의 RDBMS에 XML 데이터를 저장할 수 있는 방법을 적용한 방법이다. 하지

만 이 방법은 특정 DBMS에 종속적인 문제점을 가지고 있다.

마지막으로 XML 스키마를 스키마의 구조와 의미에 따라 나누어 관계형 데이터베이스의 테이블에 데이터를 저장하는 방법은 웹 서비스에 대한 정보 검색 시에 많은 조인 연산을 갖는다는 단점을 가지지만, 기존SQL 데이터베이스와 연동이 용이하며, 스키마의 변경이 없는 경우 충분한 성능을 보이는 특징이 있다. 이에 본 논문에서는 기존 데이터베이스와 연동이 쉽고 범용적으로 사용되고 있는 마지막 방법을 UDDI 데이터 저장 방법으로 채택하였다. 하지만 이 방법은 실제 UDDI 데이터 저장과 검색 시에 적용되어야 하는 UDDI 자료 구조와 데이터베이스 사이의 사상 관계가 요구된다. 〈그림 3〉은 UDDI 3.0 명세에 정의된 "name"이라는 엘리먼트 구조로부터 데이터베이스 스키마로 매핑이 되는 과정을 보이고 있다.

먼저 UDDI 3.0 자료 구조로부터 XML 자바 바인딩 클래스를 정의한다. 클래스의 정의 방법은 먼저 엘리먼트 이름은 클래스 이름으로 매핑시키고, 스키마에 정의된 애트리뷰트와 엘리먼트 값은 클래스의 멤버 변수로 정의한다. 다음으로 데이터 정의 언어(DDL)로 RDBMS의 테이블을 생성한다. 테이블 생성에서 이름은 비즈니스, 서비스 및 기술 모델의 이름 등 다양한 자료 구조에서 쓰일 수 있도록 정의한다. 또한 하나의 개체에 다중의 이름이 입력 가능하도록 순서 정보인 seqName이라는 컬럼을 추가 하였다. 이렇게 생성된 데이터베이스 테이블에 실제 데이터를 저장한 예가 〈그림 3〉의 하단의 테이블이다.

이와 같은 방법으로 UDDI의 자료 구조들을 서로 다른 별도의 테이블로 정의하고 각 테이블들은 서로 다른 키를 통해 참조하도록 설정된다. 각 테이블들의 키는 3.0 표준 에 따라 URI(Uniform Resource Identifier)키로 식별이 되며, 2.0과의 호환성을 위해 내부적으로 UUID(Universally Unique Identifier)키 값을 갖도록 하여 내부적인 테이블간의 참조는 UUID키를 통하여 참조한다. 또한 한 자료 구조에서 반복되어 올 수 있는 엘리먼트들은 별도의 테이블로 관리한다. "name", "description" 등과 같이 여러 자료 구조에서 사용될 수 있는 엘리먼트들은 별도 테이블로 분리하며,



〈그림 3〉 UDDI 데이터 저장 예

UUID 키 값을 통해 상위 엘리먼트를 지정하고 상위 엘리먼트의 종류를 구별하기 위해서 별도의 PARENTELT 컬럼과 같은 UDDI 3.0 자료 구조에 없는 컬럼을 고려하여 UDDI 3.0 자료 구조와 데이터베이스 스키마의 사상 관계를 설정하였다. (예: NAMEELEMENT, DESCELEMENT 등).

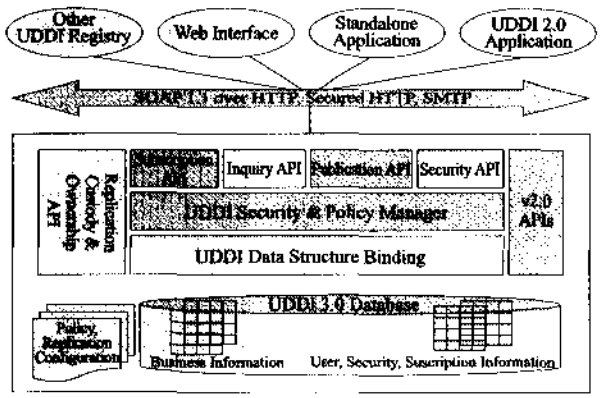
3.3 UDDI 3.0 레지스트리 시스템 구조

본 논문에서 개발한 UDDI 3.0 레지스트리 시스템은 XML 형식의 메시지와 관계형 데이터베이스 간의 사상을 위한 UDDI 자료 구조에 대한 바인딩 모듈[5]이 존재하며 이를 통해 API 기능을 수행하도록 시스템을 설계하였다. <그림 4>은 본 연구에서 개발한 UDDI 3.0 레지스트리의 전체 시스템 구조를 보여주고 있는 그림이다. 시스템은 비즈니스, 서비스 및 기술모델을 등록하는 Publication API, 등록된 정보를 검색하기 위해 사용되는 Inquiry API, 특정 노드와 토폴로지를 구성한 노드 또는 레지스트리들 간의 데이터 복제하는데 사용하는 Replication, Custody & OwnerShip API, 사용자가 관심 있어 하는 비즈니스, 서비스 정보에 대한 변경 사항이 발생할 경우 이를 사용자에게 통지하기 위해 사용되는 Subscription API, 사용자 인증을 위해 사용되는 Security API, 2.0 호환을 위한 목적으로 UDDI 2.0 표준을 따르는 API 및 암호화와 시스템 설정을 관리하는 Manager API 들로 구성이 되어있다. 또한 HTTP, HTTPS 및 SMTP 위에 SOAP을 이용하여 다른 레지스트리, 클라이언트 요청, 독자적인 응용프로그램 및 2.0을 요청하는 응용 프로그램들을 처리하도록 구현하였다.

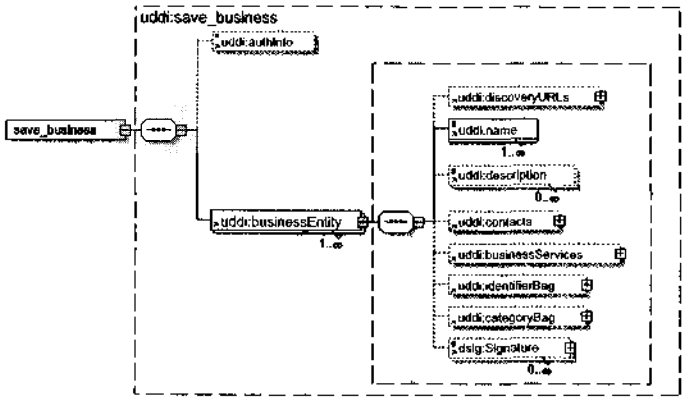
그 외의 레지스트리 운영에 있어 필요한 정책이나 데이터 복제에 관련된 설정들은 별도의 XML 파일로써 레지스트리에 유지하도록 하며 이를 참조하여 레지스트리의 보안과 정책을 설정하도록 시스템을 설계하였다.

3.4 UDDI 3.0 API 설계 및 구현

3.4.1 Publication APIs



<그림 4> UDDI 3.0 시스템 구조



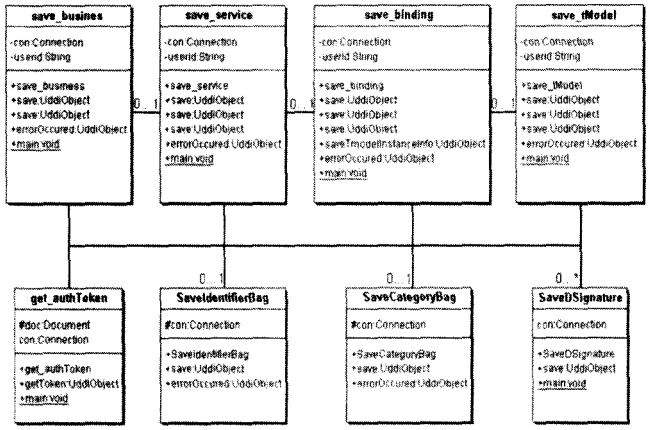
〈그림 5〉 save_business 메시지 자료 구조

〈그림 5〉은 Publication API 중에서 대표적인 비즈니스 정보를 등록하는 데 사용하는 API인 save_business API의 메시지 구조이다.

위 save_business API는 내부적으로 해당 요청을 처리할 때 다른 Publication API들과 기타 식별자, 분류자 및 전자서명 메시지 등의 추가적인 저장 모듈들이 연관되어 지는데 관련된 클래스들과의 상관 관계는 아래의 〈그림 6〉에 보이는 클래스 다이어그램과 같다. 비즈니스의 하부 자료 구조인 서비스, 바인딩 및 기술모델의 등록 기능을 담당하는 API들이 save _business에서 재사용할 수 있도록 클래스를 구성하였다. get_authToken 클래스는 데이터를 저장 시 인증된 사용자만 가능하도록 인증토큰 발행과 검사 기능을 한다. SaveIdentifierBag 클래스는 비즈니스와 기술모델의 식별자를 저장하는 기능을 하고, SaveCategoryBag 클래스는 비즈니스, 서비스, 바인딩 및 기술 모델의 분류정보를 저장하는 기능을 한다. 마지막 SaveDSignature 클래스는 모든 UDDI 개체의 전자 서명을 저장하는 기능을 한다. 이 네 모듈들은 독립적인 모듈로 작성해 다른 API에서 사용하도록 하여 재사용성을 높였다.

모든 등록 API들은 사용자가 요청한 메시지 처리 과정이 매우 유사하다. 먼저 API들은 등록하려고 하는 사용자가 인증된 사용자인지 검사하고 각 메시지의 입력 된 값이 유효(validate)한지 검사를 한 후에 실제 사용자가 요청한 메시지를 처리한다. 요청한 메시지에 하부구조, 예를 들어서 비즈니스인 경우에, 서비스, 바인딩, 식별체계, 분류체계 및 전자 서명에 관한 정보가 있으면 각각 해당하는 클래스를 호출하여 처리한다. 이와 같은 흐름으로 사용자가 요청한 비즈니스 정보를 등록하는 과정을 보여주고 있는 것이 〈그림 7〉의 save_business 시퀀스 다이어그램이다.

이 과정에서 필요한 것이 해당 XML로 이루어진 UDDI 메시지에 따른 SQL문을 생성해 주는 작업이 필요하다. 예로 〈그림 8〉에서 save_business 메시지에 대한 SQL 변환 예를 보여준다. 오른쪽에 있는 SQL문은 비즈니스에 관련된 정보를 저장하는save_business 클래스에서 생성한 것이다. 또한 아래 SQL문은



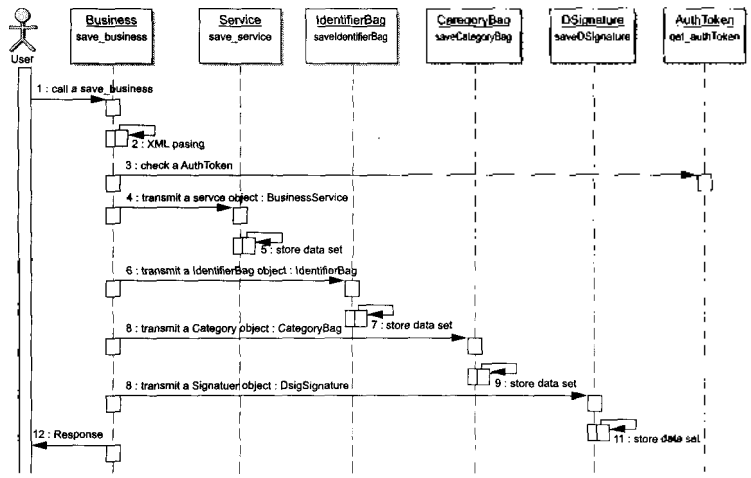
〈그림 6〉 save_business의 클래스 다이어그램

각각 식별 체계와 분류 체계를 저장하는 SaveIdentifierBag 클래스와 SaveCategoryBag 클래스에서 생성한 SQL문이다. 이와 같은 SQL문을 통해서 실제 테이블에 등록된 예가 〈그림 3〉의 아래 있는 테이블이다. 마지막으로 서비스를 요청한 요청자에게 등록 처리 결과에 대한 내용을 XML메시지 형태로 보낸다.

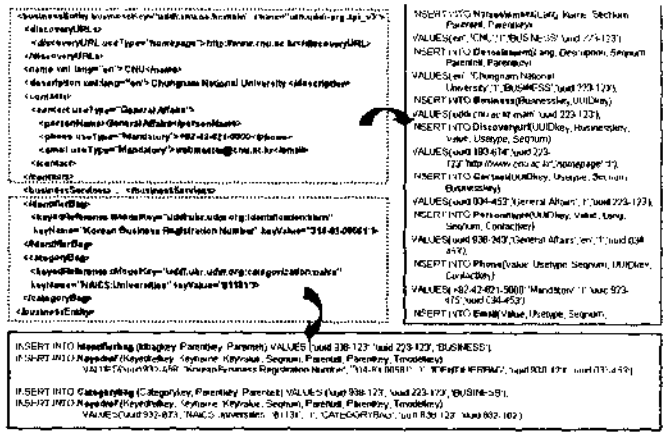
3.4.2 Inquiry APIs

〈그림 9〉은 Inquiry API 중에서 대표적인 비즈니스를 검색하는 find_business API의 구조이다. 이 find_business API의 구조는 비즈니스를 검색하는데 사용하는 검색 조건과 검색에 필요한 설정으로 구성되어 있다.

〈그림 9〉의 자료구조를 바탕으로 비즈니스



〈그림 7〉 save_business의 Sequence 다이어그램

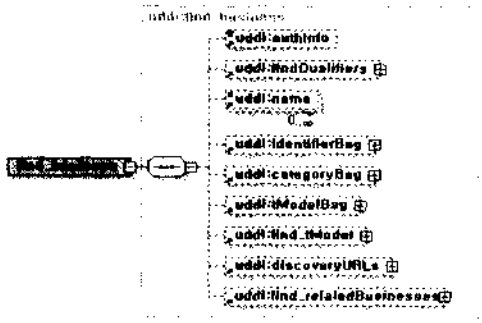


〈그림 8〉 save_business의 SQL 변환

의 이름이 "CNU"이거나 "Chungnam National University"인 비즈니스를 검색하는 예가 아래와 같다. 이 예는 검색조건 설정은 approximateMatch와 caseInsensiti-veMatch이고 검색결과 설정은 sortByNa-meAsc과 sortByDateAsc로 구성이 된다.

```
<find_business listHead = "1" maxRows = "10"
                    xmlns="urn:uddi-org:api_v3"
>
  <findQualifiers>
   <findQualifier>approximateMatch</findQualifier>
   <findQualifier>caseInsensitiveMatch</findQualifier>
   <findQualifier>sortByNameAsc</findQualifier>
   <findQualifier>sortByDateAsc</findQualifier>
   </findQualifiers>
   <name xml:lang = "en">CNU</name>
   <name xml:lang = "en">
         Chungnam National University
   </name>
  </find_business>
```

위 예에서 보는 바와 같이 findQualifiers는 검색 결과의 정렬, 검사 대상의 한정 및 검사 조건의 설정 등의 값을 가진 한 개 이상의 findQualifier의 조합으로 구성된다. 하지만 어떤 findQualifier들은 서로 같은 findQualifiers를 구성할 수 없는 findQual-ifier가 있다. 서로 같은 findQualifiers를 구성할 수 없는 findQualifer 조합은 〈표 1〉과 같다. 위와 같은 조합이 검색조건에 있는 서비스 요청자에게 유효하지 못한 findQualifier의 조합이라는 에러를 통보한다. 따라서 검색조건을 처리하기 전에 findQualifiers의 조합이 유효한지 검사해야 한다. 또한 findQualifiers 조합의 유효성 검사가 반복적으로 검사되어야 한다. 예를 들면 exactMatch는 approximateMatch와 조합할



〈그림 9〉 find_business API 자료구조

〈표 1〉 유효하지 않는 findQualifer의 조합

유효하지 않는 조합의 findQualifer			내용
andAllKeys	orAllKey	orLikeKeys	한 개 이상의 조건의 조합 방법
sortByNameAs	sortByNameDesc		검색 결과의 이름정렬
sortByDateAs	sortByDateDesc		검색결과의 날짜 정렬
bindingSubse	combineCategoryBags	serviceSubse	비즈니스, 서비스 및 기술모델의 검색 적용 범위 설정하는 조건
exactMatch	approximateMatch		정확한 일치하는 검색와 유사 검색
exactMatch	caseInsensitiveMatch		정확한 일치하는 검색과 대소문자 구별않는 검색
diacriticSensitiveMatch	diacriticInsensitiveMatc		발음 기호 검색
exactMatch	diacriticInsensitiveMatc		정확한 일치하는 검색과 발음기호 구별않는 검색
caseSensitiveSor	caseInsensitiveSort		대소문자를 정렬
caseSensitiveMatch	caseInsensitiveMatch		대소문자 구별 검색
binarySort	UTS-10		유니코드 정렬

수 없고 또한 caseInsensitive-Match와 diacriticInsensitiveMatch하고도 조합할 수 없으므로 검색조건 유효성 검사를 반복적으로 할 수 있는 방안이 필요하다.

그리고 findQualifiers는 비즈니스, 서비스 및 기술 모델등과 같이 여러 곳에서 사용하는 엘리먼트이다. 이런 사항을 고려하여 클래스를 구성한 것이 〈그림 10〉의 클래스 다이어그램으로 각 클래스의 상관 관계를 볼 수가 있다.

FindQualifers 클래스는 fndQualifier 엘리먼트가 가질 수 있는 모든 값을 하나의 Boolean 타입의 멤버 변수로 구성한다. 이렇게 findQualifier의 값을 하나의 멤버 변수로 만든 이유는 유효성 검사가 끝난 후 사용자가 요청에 맞는 검사 조건을 설정하고 검사 결과를 생성하는데 사용하기 위해 findQualifier의 값을 저장해 둘 필요가 있기 때문이다. 또한 이것이 사용자가 요청한 findQualifers 조합이 유효한 조합인지 검사하는 문제점을 해결해 준다. 그 해결 방법은 아래와 같다.

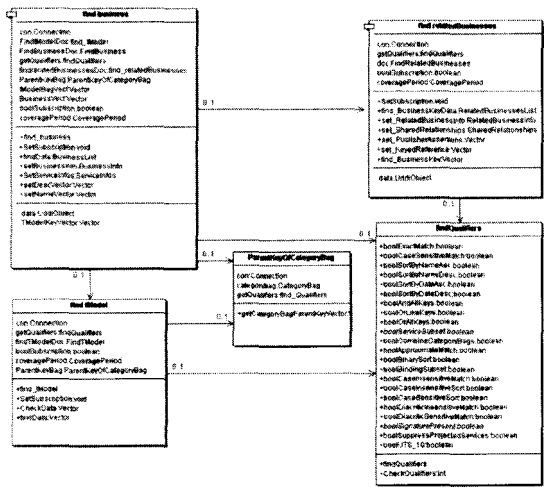
1. 모든 검색 API, 즉 예로 find_business, 가 모든 초기 값이 false인 findQualifiers 클래스를 생성한다.
2. 서비스 요청자가 조합한 findQualifiers 엘리먼트의 findQualifier 값을 대응하는 멤버 변수의 값을 true로 순차적으로 변경하면서 위에 나온 유효하지 못한 조합의 조건에 맞는 멤버 변수가 동시에 true인지 검사한다. 만약 동시에 true이면 서비스 요청자에게 에러 메시지를 보낸다.

또한 bindingSubset, serviceSubset 및 combineCategoryBags와 같은 findQualif-ier는

카테고리 검색조건과 조합하여 카테고리 검색조건에 대한 검색의 대상을 한정하여 다른 검색결과가 된다. 예를 들면 카테고리 검색조건과 findQualifier의 검색조건 값이 serviceSubset인 비즈니스 검색인 경우는 비즈니스에 등록된 서비스의 카테고리가 검색조건에 주어진 카테고리와 일치하는 비즈니스를 검색하도록 검색대상을 서비스로 제한한다. 그와는 달리 검색조건이 combineCategoryBags인 비즈니스 검색인 경우는 비즈니스와 비즈니스 하위 개체의 카테고리가 검색조건과 일치하는 비즈니스를 검색하도록 검색대상을 변경한다. 이와 같이 findQualifier의 값에 따라 검색대상을 변경해야만 하는 문제점이 있다. 앞에서 데이터베이스 스키마 설계할 때 하나 이상의 개체에서 사용되는 엘리먼트는 별도의 테이블로 만들고 상위 엘리먼트를 지정할 목적으로 UUID 키 값을 가지는 parentKey와 상위 엘리먼트의 종류를 구별할 목적으로 parentElt라는 속성을 테이블에 추가했다. 또한 카테고리를 소유한 비즈니스, 서비스 및 기술 모델의 구별은 'BUSINESS', 'SERVICE' 및 'BINDING'의 값을 가진 parentElt를 통해 구별된다. 따라서 검색조건 findQualifer의 값에 따라 검색을 요청하는 질의문중 parentElt의 조건을 변경하는 것으로 검색대상을 변경하는 문제점을 해결할 수 있다.

UDDI 3.0부터는 비즈니스와 서비스 검색시 중첩질의를 지원하여 더 상세한 검색을 할 수 있는 방안을 제시하고 있다. 예를 들면 find_tModel과 find_relatedBusiness은 사용자가 독립적으로 요청할 수 있는 API이지만 위 find_business 메시지의 자료 구조에서는 하나의 엘리먼트로 사용되고 있다. 이에 본 논문에서는 〈그림 10〉과 같이 독립적인 클래스로 구성해 재사용성을 높였다.

그리고 find_business와 같은 사용자가 요청하는 서비스 처리하는 작업 대부분이 데이터베이스에 질의할 SQL 질의문을 생성하는 것



〈그림 10〉 find_business API의 클래스 다이어그램

이다. 이 SQL 질의문 생성하는데 있어 가장 큰 문제점은 한 개 이상의 검색조건이 있을 경우에 질의문이 매우 복잡하게 되는 것이다. 다음은 비즈니스 이름이 "My Company Name"이고 카테고리가 "Witch-doctors or voodoo services"인 비즈니스를 검색하는 예이다.

```
<find_business xmlns="urn:uddi-org:api_v3">
        <findQualifiers>
          <findQualifier>
                combineCategoryBags
          </findQualifier>
        </findQualifiers>
        <name>My Company Name/name>
        <categoryBag>
        <keyedReference keyValue="85.14.15.01.00"
            tModelKey="uddi:ubr.uddi.org:catego
                rization:unspsc"/>
        </categoryBag>
</find_business>
```

위 예는 검색조건 findQualifier의 값으로 combineCategoryBags을 가지고 있다. 따라서 "My Company Name"라는 비즈니스 이름을 가지고 있고 또한 그 비즈니스나 그 하위구조의 카테고리가 "Witch-doctors or voodoo services"로 분류된 비즈니스를 검색하는 질의문을 만들어야 한다. 이 질의문은 아래와 같이 각 검색 조건에 대한 하나의 서부 질의문 형식의 정형화된 질의문를 만들 필요가 있다. 왜냐하면 정형화되지 않은 SQL 질의문은 보기도 어렵고 에러 발생 시 발견하기도 어려운 단점을 가지고 있기 때문이다.

```
Select businessKey
From business
Where a.uuidKey in (
(Select parentKey
From nameElement
Where parentElt = 'BUSINESS'
   and name = 'My Company Name')
intersect
(( Select a.parentKey
From CategoryBag a, Keyedref b, tModel c
Where a.parentElt = 'BUSINESS'
      and a.categoryBagKey = b.parentKey
      and b.parentElt = 'CATEGORYBAg'
      and b.KeyValue = '85.14.15.01.00'
      and b.tModelKey = c.uuidKey
      and c.tModelKey='uddi:ubr.uddi.org:categorization:
                        unspsc')
Union
(Select a.parentKey
From CategoryBag a, Keyedref b, tModel c
Where a.parentElt = 'SERVICE'
   and a.categoryBagKey = b.parentKey
   and b.parentElt = 'CATEGORYBAg'
   and b.KeyValue = '85.14.15.01.00'
   and b.tModelKey = c.uuidKey
  and c.tModelKey= 'uddi:ubr.uddi.org:categorization:
                        unspsc' )
Union
(select a.parentKey
from CategoryBag a, Keyedref b, tModel c
Where a.parentElt = 'BINDTEMPLATE'
      and a.categoryBagKey = b.parentKey
      and b.parentElt = 'CATEGORYBAg'
      and b.KeyValue = '85.14.15.01.00'
      and b.tModelKey = c.uuidKey
      and c.tModelKey= 'uddi:ubr.uddi.org:categorization:
                        unspsc' )))
```

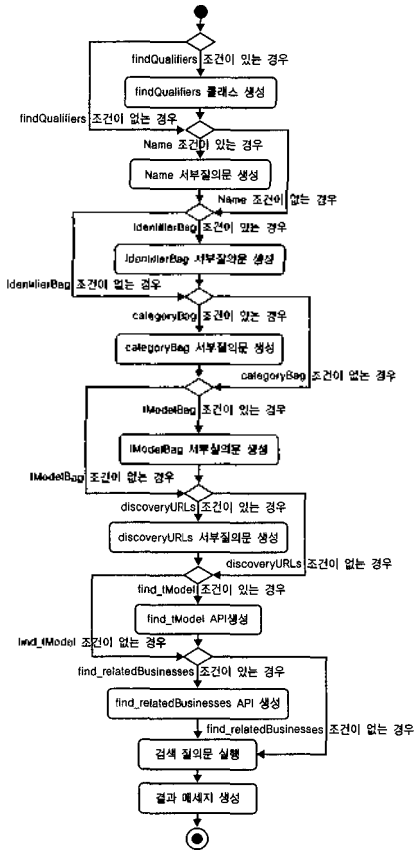
이에 본 논문에서는 아래 〈그림 11〉와 같은 Activity 다이어그램과 같은 흐름의 SQL 질의문을 변환 메커니즘을 만들게 되었다. 먼저 findQualifiers의 검색조건이 있는 경우에 findQualifiers 클래스를 생성하여 검색조건의

설정을 저장한다. 다음으로 각 검색조건을 설정에 맞게 서부 질의문을 생성한다.

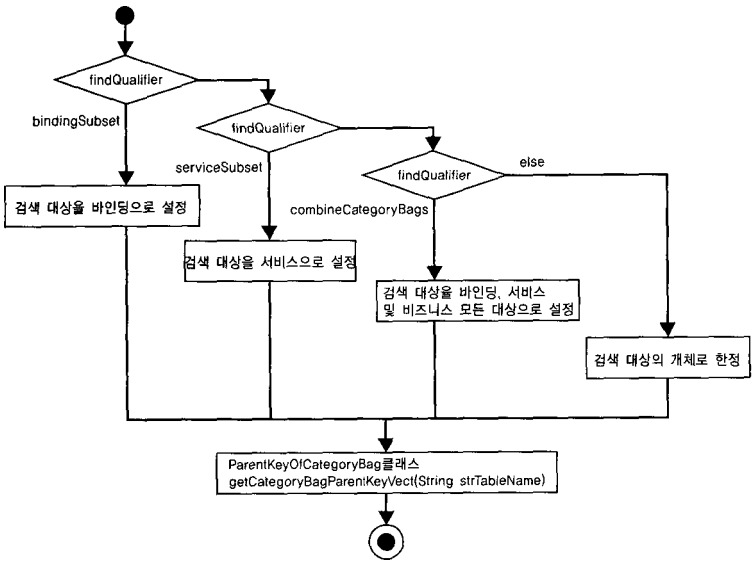
특히 find_tModel, find_relatedBusinesses API는 독립적인 서비스 요청인 경우와 다른 API의 검색 조건인 경우로 분류할 수 있다. 독립적인 서비스 요청인 경우에는 위 흐름에 따라 처리한 결과를 서비스 요청자에게 반환해 준다. 반면에 다른 API의 검색 조건인, 즉 중첩질의 경우에는 흐름에 따라 처리한 결과가 원 API의 검색조건이 되어 다른 검색조건과 같이 검색 결과를 생성한다.

또한 findQualifer의 값에 따라 카테고리 검색조건의 검색대상을 변경하는 bindingSubset, serviceSubset 및 combineCategoryBags 검색조건은 보다 간단한 질의문을 생성하기 위해서는 〈그림 12〉과 같은 별도의 처리가 필요하다. 단, 검색조건에 카테고리 조건이 있는 경우

1.bindingSubset인 경우 검색대상, 즉 검색 테이블을 바인딩으로 설정하여 검색한다.
2.serviceSubset인 경우 검색대상, 즉 검색 테이블을 서비스로 설정하여 검색한다.
3.combineCategoryBags인 경우 검색대상, 즉 검색 테이블을 바인딩, 서비스 및 비즈



〈그림 11〉 SQL 변환 Activity 다이어그램



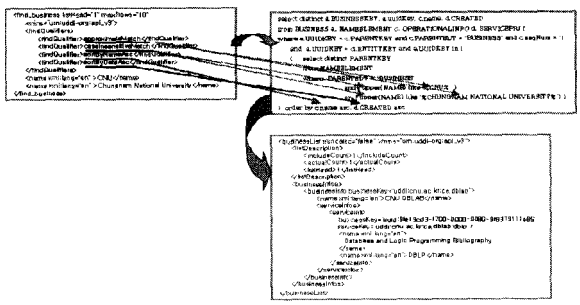
〈그림 12〉 bindingSubset, serviceSub-set 및 combineCategoryBags의 SQL문 생성 알고리즘

니스로 설정하여 순차적으로 검색한다.

위 3가지의 조건이 없고 카테고리 조건만 있을 경우에는 검색 요청하는 해당 개체만을 검색한다. 즉 find_binding, find_service 및 find_business API 각각 바인딩, 서비스 및 비즈니스의 카테고리만을 검색한다.

위에서 제시된 방안을 이용하면 앞에 예로 제시된 find_business API는 다음과 같이 보다 간단한 SQL 질의문으로 변경할 수 있다.

```
Select businessKey
From business
Where a.uuidKey in (
        (Select parentKey
        From nameElement
        Where parentElt = 'BUSINESS'
          and name = 'My Company Name')
```



〈그림 13〉 find_business의 변경된 SQL 질의문과 그 결과 메시지

```
Intersect
(Select from parentKey
from CategoryBag
where CategoryBagKey = 'uuid:9fe19cd3-
        f700-0000-0080-9f8379111a85'
    or CategoryBagKey = 'uuid:619f86d3-
        f700-0000-0080-9f8379111a85' ))
```

이와 같은 SQL 질의문을 생성하는 정형화 방법을 따라 find_business API를 처리하여 SQL 질의문 생성하고 그 결과를 보여주는 것이 〈그림 13〉이다.

4.관련 연구와의 비교

본 논문에서 개발한 UDDI 레지스트리 시스템과 현재 개발되어 운영중인 레지스트리 시스템을 비교한 내용은 〈표 2〉과 같다.

IBM과 Microsoft사가 운영중인 레지스트리 시스템은 구현환경은 다르지만 등록 및 3.0에 추가된 사항에 대한 항목은 동일하다. 두 회사의 레지스트리는 2.0과의 호환성, 등록자 키 할당, 전자서명 메시지의 처리 기능이 없다. 이 레지스트들은 Value Set API를 구현하지 않아도 되는 UI(User Interface)를 제공하고 사용자 입장에선 잘못된 값을 입력하는 오류를 범하지 않는다는 장점이 있지만 항상 유효한 Value Sets을 유지해야 하는 문제점을 가지고 있다. 반면에 검색 및 subscription 항목은 IBM의 레지스트리가 브라워즈 검색을 지원하지 않지만 Microsoft의 레지스트리는 UDDI 표준에서 지원하는 3가지 방법 모두를 지원하고 있다.

SAP에서 운영중인 레지스트리 시스템은 UDDI 개체를 등록하는 3.0을 지원하는 사용자 인터페이스가 없고 단지 SOAP 접근 주소(Access Points)만 제공되고 있어 사용자가 사용하기에는 불편하다. 따라서 SOAP메시지를 통한 등록 및 3.0에 추가된 사항에 대한 테스트는 불가능하다. 반면에 이 레지스트라는 검색 인터페이스는 지원하고 있지만 UDDI 2.0 표준을 준수하고 있어 3.0에 추가된 URI 기반의 키를 가지고 있는 비즈니스, 서비스 및 기술모델을 검색할 수 없는 단점을 가지고 있다.

IBM, Microsoft, SAP에서 운영중인 레지스트리 시스템들은 자사의 데이터베이스 관리 시스템, 웹 서버 등을 사용하기 때문에 시스템 종속적이지만, 본 연구에서 개발한 레지스트리 시스템은 DBMS와 플랫폼에 독립적으로 설계 및 구현하였다. 그리고 다른 시스템들에서 지원되지 않는 기능인 2.0과의 호환성과 전자 서명 메시지 처리, Value Set 지원 , Subscription 및 사용자 인터페이스 같은 항목들을 전부 지원을 하고 있다.

5. 결 론

본 논문에서는 웹 서비스를 개발하는 기업들의 최대 관심 대상인 웹 서비스에 대한 정보를 저장하고 저장된 정보를 사용자들이 쉽게 검색하는 방법을 제공해주는 UDDI 3.0 레지스트리 시스템을 개발하였다. 이 시스템은 현재 운영중인 IBM, Microsoft 및 SAP이 갖고 있던 여러 문제점들을 해결하였다.

본 논문에서 개발한 UDDI 레지스트리 시스템은 특정 DBMS, 어플리케이션 서버에 종

〈표 2〉 UDDI 레지스트리 시스템 비교

비교대상 \ 시스템		IBM	Microsoft	SAP	본 검색시스템
구현환경	DBMS	DB 2 Version 8	SQL Server		RDMS
	웹서버	IBM WebSphere Application Server Version 5	#S	SAP Web Application Server	Tomcat 4.1
	개발 언어	Java(J2EE)	NET framework	Java(J2EE)	Java(J2EE)
	XML파서	Apache Xerces	MSXML	SAP NetWeaver	Apache Xeres
	SOAP 메시지 통신	Apache Axis	NET framework class library		Apache SOAP
등록	비즈니스 개체	0	0	0	0
	비즈니스 서비스	0	0	0	0
	기술 모델	0	0	0	0
	비즈니스 개체간 관계성	0	0	X	0
검색 및 Subscription	일반 검색	0	0	0	0
	상세 검색	0	0	0	0
	브라워즈 검색	X	0	0	0
	모든 FindQualifiers	X	X	X	0
	Subscription	X	X	X	0
버전 3.0 특징	2.0 호환성	X	X		0
	등록자 키 할당	X	X		0
	카테고리 그룹 지원	0	0		0
	전자 서명	X	X		0
	Values Set지원 (API 구현 여부)	0	0		0
사용자 인터페이스		0	0	0	0

속적이지 않은 시스템으로 개발되어서 공용(public) UDDI 레지스트리 뿐만 아니라 조그만 사설(private) UDDI 레지스트리로써 운영이 가능한 장점을 가지고 있다. 또한 UDDI 3.0의 모든 기능을 완벽하게 지원하는 국내 최초의 레지스트리로써 그 활용성과 국내 e-비즈니스 환경에의 파급 효과가 매우 클 것으로 예측된다.

참 고 문 헌

[1] IBM, "IBM Software Solutions Web services by IBM UDDI", http://uddi.ibm.com/beta/registry.html

[2] Microsoft, "Microsoft UDDI Business Registry(UBR) node". http:/uddi.beta.Microsoft.com

[3] OASIS UDDI Specification TC, "UDDI Version 3.0 Technical Committee Specification", http://uddi.org/pubs/uddi-v3.00-published-20020719.htm, July 2002.

[4] SAP, "SAP UDDI Business Registry", http://udditest.sap.com

[5] UDDI4J.org, "UDDI4J", http://www.uddi4j.org/, Nov. 2000.

저 자 소 개

유수진 (E-mail : sjryu@orom.com)
1999. 충남대학교 공과대학 컴퓨터공학과(공학사)
현재 충남대학교 대학원 컴퓨터공학과 석사과정
현재 오롬정보 지식정보화사업본부 사업 4팀
관심 분야 데이터베이스, XML, 웹 서비스

이경하 (E-mail : bart@ce.cnu.ac.kr)
1998. 충남대학교 공과대학 정보통신공학과(공학사)
2000. 충남대학교 대학원 정보통신공학과(공학석사)
현재 충남대학교 공과대학 컴퓨터공학과 박사과정
현재 충남대학교 대학원 연구전문요원
관심 분야 데이터베이스, XML, 객체지향 언어 및 설계, 정보통합

이규철 (E-mail : kclee@cnu.ac.kr)
1984. 서울대학교 컴퓨터공학과(공학사)
1986. 서울대학교 공과대학 컴퓨터공학과(공학석사)
1990. 서울대학교 공과대학 컴퓨터공학과(공학박사)
1994. 미국 IBM Almaden Research Center 초빙 연구원
1996. 미국 Syracuse University, CASE Center 초빙 교수
1997. ~ 1998. 학술진흥재단 부설 첨단학술정보센터 파견 교수
2001. ~ 2003. 한국정보과학회 논문편집위원
현재 한국 ebXML 전문위원회 위원장
관심 분야 데이터베이스, XML, 정보 통합, 멀티미디어 시스템, e-비즈니스 시스템