

관계 DBMS에 기반한 UDDI 3.0 레지스트리 서버의 개발

유수진^o, 박송희, 김영선, 이경하, 이규철
 충남대학교 컴퓨터공학과
 {sjryu^o, shpark, yskim, bart, kclee}@ce.cnu.ac.kr

A Development of UDDI 3.0 Registry based on Relational DBMS

Su-Jin Ryu^o, Song-Hee Park, Young-Sun Kim, Kyong-Ha Lee, Kyu-Chul Lee
 Dep't. of Computer Engineering, Chungnam National University

요 약

UDDI는 OASIS에 의해 표준화가 진행 중인 웹 서비스를 위한 레지스트리의 표준으로 최근에 버전 3.0이 공개되었다. 이 UDDI 3.0은 웹 서비스 기술을 구성하는 중요 기술 요소임에도 불구하고 그 표준의 복잡성으로 인하여 아직까지 국내외에서 구현된 사례가 전무하였다. 또한 국내에서의 웹 서비스 기술 이용이 급속히 증가함에 반해 이러한 기반 기술들에 대한 국내의 개발 활동은 아직까지도 미비한 실정이다. 이에 따라 본 논문에서는 플랫폼에 중립적이면서 최근의 UDDI 3.0 표준을 지원하는 e-비즈니스 레지스트리 시스템을 개발함으로써 해당 기술의 국산화와 국내 e-비즈니스 시장의 활성화를 유도할 수 있도록 한다.

1. 서론

UDDI[1]는 OASIS에 의해 표준화가 진행 중인 웹 서비스를 위한 레지스트리의 표준으로 최근에 버전 3.0이 공개되었다. 이 UDDI 3.0은 웹 서비스 기술을 구성하는 중요 기술 요소임에도 불구하고 그 표준의 복잡성으로 인하여 아직까지 국내외에서 구현된 사례가 전무하였다. 또한 국내에서의 웹 서비스 기술 이용이 급속히 증가함에 반해 이러한 기반 기술들에 대한 국내의 개발 활동은 아직까지도 미비한 실정이다. 이러한 이유로 인해 국내 기관이나 업체들이 웹 서비스의 제반 기술을 실정하여서 국내 e-비즈니스 시장의 활성화 하는데 많은 저해하는 요인이 되고 있다.

이에 본 논문은 국내 e-비즈니스 시장의 활성화하는데 있어 중요 기술 요소인 UDDI 3.0 레지스트리 개발에 대하여 설명을 한다.

2. UDDI 3.0 레지스트리

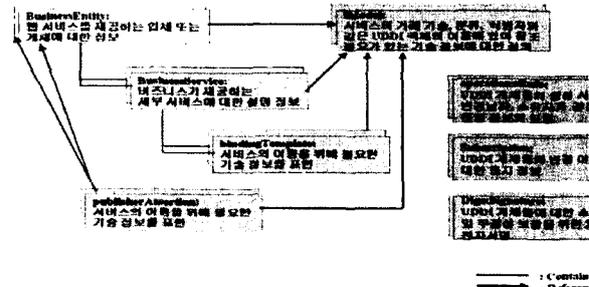
2.1 UDDI 3.0 자료 구조

UDDI의 자료 구조[2]는 레지스트리에 등록할 서비스와 비즈니스 정보를 표현하기 위해 사용되는 자료 구조로 XML 스키마를 통해 정의되어 있다. 현재 UDDI 3.0에서의 자료 구조는 <그림 1>와 같이 7가지의 구성 요소로 정의되어 있다.

이중 UDDI 3.0 표준에서 새로이 추가된 자료 구조는

* 본 연구는 정보통신부의 IT연구센터(ITRC) 지원을 받아 수행되었음

OperationalInfo, Subscription, Signature가 존재한다. OperationalInfo는 UDDI 개체들에 대한 생성, 변경과 같은 운영 정보를 가지고 있는 자료 구조이다. Subscription은 서비스 사용자들이 자신이 관심이 있는 UDDI 개체들에 대한 변경 이력에 대한 통지 정보를 다루는 자료 구조이다. 또한 Signature는 각 사용자가 자신이 발행하는 비즈니스, 서비스 정보에 대한 XML 전자 서명을 표현하기 위한 자료 구조이다. 이를 통해 UDDI 3.0에서는 사용자 권한과 보안 및 정보에 대한 무결성 보장 기능을 강화하였다.



<그림 1> UDDI 3.0 데이터 구조

2.2 UDDI 3.0 레지스트리 설계

XML 정보에 대한 저장 매체로는 크게 파일, DB, LDAP 등이 이용될 수 있다. 본 논문에서는 저장 데이터의 회복, 복구가 용이한 일반적인 관계형 DBMS를 UDDI 자료 구조에 대한 저장 매체로 이용하였다. 또한 기존의 XML 정보에 대한 저장 방법은 크게 3가지로 분류될 수

있는데, 첫 번째 방법은 eXcelon, Tamino 등과 같은 XML 전용 DBMS을 이용하여 정보를 저장하는 방법이다. 두 번째는 Oracle9i에서 지원하고 있는 XMLType과 같은 XML 저장을 위해 확장된 데이터 타입을 이용하는 방법이다. 또한 XML 데이터 바인딩[3]이라고 불리는 DTD 또는 XML Schema에 고정적으로 DB 스키마를 설계하는 방법이 존재한다.

비록 XML 데이터 바인딩 방법으로 설계한 DB 스키마는 추후의 스키마 변경 시 DB 스키마의 변경이 요구되는 문제점을 가질 수 있으나, 본 논문에서는 UDDI 표준에 대한 XML Schema 정의가 제공되며 또한 이 정의는 고정적인 특징을 가지고 있으므로 XML 데이터 바인딩 방법을 이용하여 XML Schema 정의에 고정적인 DB 스키마를 구성하였다. 따라서 XML 스키마로 정의된 UDDI 자료 구조 정의와 관계형 데이터베이스 스키마 간의 사상관계를 설정하고 UML의 관계형 데이터베이스를 위한 프로파일[4]을 이용하여 데이터베이스를 설계하였다.

다음은 메시지는 'uddi-org:types'에 대한 tModel의 XML 메시지의 예이다.

```
<tModel tModelKey=" uddi:uddi.org:categorization:types" >
  <name>uddi-org:types</name>
  <description>UDDI Type Category System</description>
  <overviewDoc>
    <overviewURL useType=" text" >
      http://uddi.org/pubs/uddi_v3.htm#UDDITypes
    </overviewURL>
  </overviewDoc>
</tModel>
```

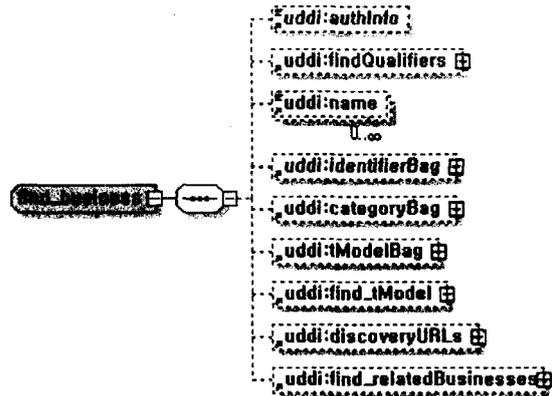
위와 같이 XML 스키마로 정의된 UDDI 자료 구조 정의와 데이터베이스 스키마 간의 사상 관계에 맞게 다음과 같이 DB 스키마를 정의한다.

```
CREATE TABLE TMODEL (
  TMODELKEY VARCHAR(255),
  UUIDKEY VARCHAR(255) NOT NULL UNIQUE,
  LANG CHAR(5) NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  VALID CHAR(5),
  CHECKED CHAR(5),
  CONSTRAINT TMODEL_IDX PRIMARY KEY (UUIDKEY));
```

UDDI 3.0에서는 URI(Uniform Resource Identifier)를 키로 이용하나 기존 2.0에서는 UUID를 키로 이용하므로 이 둘 간의 호환성을 위해서 본 DB 스키마 설계는 내부적으로 UUID 키를 이용하였으며, SQL VIEW를 이용하여 UDDI 3.0에 대한 DB 스키마에서 2.0에 대한 뷰를 제공하고 이를 통해 2.0 API와의 호환성을 지원하도록 하였다.

2.2 UDDI 3.0 API 구현

트리틀을 이용하는데 있어 필요한 API와 레지스트리의 운영 및 레지스트리 간의 상호작용에 필요한 API들로 분류할 수 있다(Inquiry, Publication, Replication, Security Policy, Custody and Ownership Transfer, Value Set, Subscription). UDDI 3.0 API부터는 2.0 API 표준에는 제공하지 않던 중첩 질의 기능이 추가되었다. 중첩 질의가 추가된 API로는 find_binding, find_business, find_service가 있고 이 API들은 파라미터로 find_tModel 또는 find_relatedBusinesses API를 가질 수 있다. <그림 2>는 UDDI 3.0 스키마 표준에 따르는 find_business API 메시지이다.



<그림 2> find_business API의 XML 스키마

이에 본 레지스트리 개발에서는 해당 중첩 질의를 처리하기 위해서 아래 find_business 클래스 정의와 같이 find_tModel, find_relatedBusiness 클래스를 멤버 변수로 두어 find_business 메시지의 상태에 맞게 클래스를 생성할 수 있도록 하였다.

```
public class find_business
{
  Connection      con = null;
  find_Qualifiers Qualifiers = null;
  FindBusiness    FindBusinessDoc = null;
  find_tModel     findTModelDoc = null;
  find_relatedBusinesses findrelatedBusinessesDoc = null;

  Vector          tModelBagVect = null;
  Vector          BusinessVect = null;
}
```

find_Qualifiers 클래스는 Inquiry API에 있는 모든 findQualifier를 멤버 변수로 설정하고 각 멤버 변수는 기본적으로 false로 설정이 되어 있다. findQualifier는 각 Inquiry API의 설정에 맞게 조합이 되었는지 검사를 한 후에 find_Qualifiers의 멤버 변수의 값을 true로 설정을 변경한다.

아래 예와 같이 'MS Internet Explorer' 인터페이스로 구현된 웹 서비스 바인딩을 가진 비즈니스 이름이

UDDI API는 서비스 요청자, 서비스 제공자가 레지스 '충남대학교' 를 찾기 위한 find_business 메시지의 예이다.

```
<find_business xmlns="urn:uddi-org:api_v3">
  <findQualifiers>
    <findQualifier>uddi:uddi.org:findQualifier:caseInsensitiveSort
  </findQualifier>
</findQualifiers>
<name> Chungnam National University</name>
<find_tModel>
  <findQualifiers>
    <findQualifier>
      uddi:uddi.org:findQualifier:approximateMatch
    </findQualifier>
    <findQualifier>
      uddi:uddi.org:findQualifier:caseInsensitiveMatch
    </findQualifier>
  </findQualifiers>
  <name>MS Internet Explorer%</name>
</find_tModel>
</find_business>
```

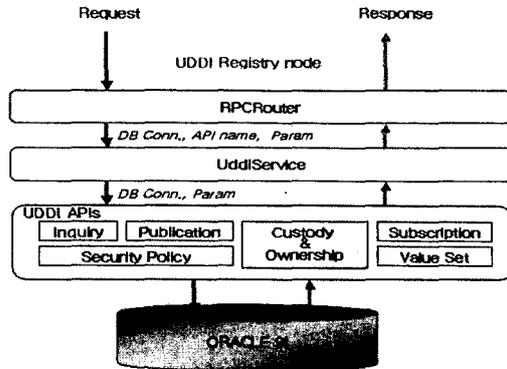
본 논문에서 개발한 find_business API는 첫번째로 메시지의 상태에 맞게 findQualifiers, find_tModel의 멤버 변수를 생성한 후 findQualifier의 조합이 적합한지 검사 한다. 두 번째로 find_tModel 클래스에서 findQualifier의 조합이 적합한지 검사 한 후 이름이 'MS Internet Explorer'인 기술 모델을 검색한다. 마지막으로 find_tModel에서 검색된 기술 모델을 서비스 바인딩으로 가진 비즈니스 이름이 '충남대학교'인 비즈니스를 검색한다. 다음 메시지는 find_business API로 생성한 결과 메시지의 예이다.

```
<find_business xmlns="urn:uddi-org:api_v3">
  <businessInfos>
    <businessInfo businessKey="http://www.cnu.ac.kr">
      <name>Chungnam National University</name>
      <serviceInfos>
        <serviceInfo serviceKey="http://www.cnu.ac.kr:8888/
          businessKey="http://www.cnu.ac.kr">
          <name>HomePage</name>
        </serviceInfo>
      </serviceInfos>
    </businessInfo>
  </businessInfos>
</businessList>
```

2.3 UDDI 3.0 레지스트리 구조 및 개발 환경

UDDI 3.0 레지스트리 서버 시스템의 전체 구조는 <그림 3>와 같다. 본 논문에서는 개발한 레지스트리 서버는 클라이언트 또는 다른 레지스트리간의 SOAP 메시지 통신에서 요청, 응답 메시지의 효율적인 처리를 위해 기존에 개발한 UDDI 2.0 레지스트리에서 이용한 UDDI4j 2.0 라이브러리를 확장하여 UDDI4j 3.0을 개발, 이용하였다. 또한 개발에 이용한 운영체제로는 윈도우 2000서버, 개발 툴킷으로는 자바 기반인 JDK 1.4, 웹 서버로는 Tomcat 4.04를 기반으로 구현하였다. 그 밖에 서비스와 레지스트리 운영에 필요한 데이

터 저장과 관리를 위해 오라클 9i를 사용하였고, XML 전자 서명 처리를 위해 Apache XML-security를 사용하여 개발하였다.



<그림 3> UDDI 3.0 레지스트리 서버 시스템 구조

3. 결론 및 전망

본 논문에서는 e-비즈니스 분야에서 적용 가능한 웹 서비스의 등록, 검색을 위해 요구되는 UDDI 3.0 레지스트리의 개발에 대해 소개하였다. 개발된 레지스트리는 특정 DBMS, 어플리케이션 서버에 종속적이지 않는 시스템을 개발함으로써 다양한 플랫폼, 여러 운영 체계를 지원하도록 개발되었다.

웹 서비스 기반 기술인 UDDI 3.0 레지스트리 개발이 해당 기술의 국산화하는데 공헌하게 되었다. 또한 향후에 본 레지스트리에 대한 기술 이전을 통하여 국내 단체 또는 기업에 보급될 것이며, 이를 통하여 e-비즈니스 시장의 활성화를 유도하는 기반이 될 것이다.

[참고 문헌]

- [1] Arbia Inc. Microsoft Corp. "UDDI Technical WhitePaper", http://www.udi.org/pub/Iru_UDDI_Technical_White_Paper.pdf, uddi.org, September 2000.
- [2] UDDI Version 3.0, Published Specification, <http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf>, uddi.org, July 2002.
- [3] Ronald Bourret, "XML Data Binding Resources", <http://www.rpbouret.com/xml/XMLDataBinding.htm>, May 2003.
- [4] David Carlson, "Modeling XML Applications with UML", Addison-wesley, 2002.