e-비즈니스 레지스트리 통합 질의 시스템 설계 및 구현

김 계 용* · 이 규 철*

Design and Implementation of Integrated Query System for e-Business Registries

Kye-Yong Kim* · Kyu-Chul Lee*

Abstract

With the spread of Internet, e-business using Internet technology is being actively developed and operated. Currently, on behalf of the International e-business framework, International standard such as ebXML and Web Service is being advanced. We are able to publish and search business information through ebXML and Web Service and then actual trading between partners is accomplished.

By the way, it is Registry that play an important part in e-business. Registry is an e-business infrastructure for enabling building, deploying, and the discovery of business information. We can do e-business through Registries dynamically and share the resources. Representatives of Registry are ebXML and UDDI used as an international standard.

We will meet with some problems when using Registries. ebXML is focused on B2B collaborations and Web Service is focused on application integrations. So we must use ebXML and Web Service characteristically, and sometimes use all at need. ebXML and Web Service contain business information by Registry specific way. So When using the services offered by ebXML and Web Service, we should access each Registry by using Registry specific tools. This thesis intended to integrate business information from ebXML Registry and UDDI Registry to accomplish e-business conveniently.

This thesis defined the common data model as well as integrated query language for integrated access to ebXML and Web Service along with design and implementation of the system.

Keywords : Integrated Query, Registry, e-Business, ebXML, UDDI

※ 본 논문은 소프트웨어연구센터(SOREC)와 BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음.
* 충남대학교 컴퓨터공학과

1. 서 론

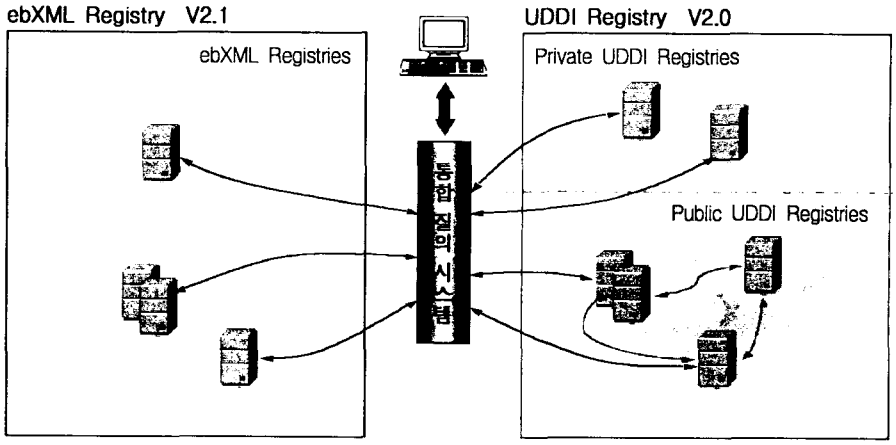
인터넷을 통해 전자거래가 활발히 진행되고 있는 가운데, 전 세계적으로 인터넷 기반 전자상거래 표준을 만들기 위한 노력이 활발히 진행되고 있다. 현재는 e-비즈니스 서비스 관련 표준 프레임워크로 ebXML과 웹 서비스와 같은 국제 표준이 제정되어 개발 운용 중에 있다. 이러한 ebXML과 웹 서비스를 이용해서 거래 상대자의 비즈니스 정보를 검색하고, 검색된 내용을 바탕으로 거래 당사자들 간의 실제적인 거래가 이루어지게 된다. 즉 e-비즈니스를 수행함에 있어 비즈니스 메타 정보를 관리하여 비즈니스 정보를 등록, 배포, 발견할 수 있도록 하는 기술이 필요한데, 이러한 일을 담당하는 e-비즈니스의 기반 기술이 레지스트리이다. 즉, 레지스트리는 e-비즈니스 거래를 원활하게 하는 중립적인 제 삼자라고 할 수 있다. 대표적 레지스트리로 ebXML 레지스트리와 UDDI 레지스트리가 국제 표준인 레지스트리로 제정되어 있다.

ebXML은 B2B 협업의 측면에, 그리고 웹 서비스는 여러 응용 간의 통합에 중점을 두고 있기 때문에 비즈니스 정보를 저장하거나 검색할 경우 이러한 특성에 맞게 각각의 서비스를 이용해야 한다. 따라서 현재까지는 전자거래에 필요한 비즈니스 정보를 검색하고자할 경우 ebXML과 웹 서비스를 개별적으로 접근해야만 한다. 이러한 방법은 각각의 서비스를 이용할 수 있도록 하는 전용 도구를 이용해서 레지스트리를 개별적으로 접근해야 한다. 따라서 ebXML이나 웹 서비스가 제공하는 서비스를 개별적으로 이용하지 않고 한 개의 통합 시스템으로 여러 레지스트리를 통합 접근하여 e-비즈니스 서비스를 이용할 경우 거래 당사자는 최적의 검색 결과를 쉽게 얻을 수 있을 것이다.

한편, ebXML 레지스트리의 경우 현재까지는 각각의 레지스트리가 개별적으로 운용되고 있다. UDDI 레지스트리는 버전 2.0부터는 공용 레지스트리 간의 데이타 복제(replication)가 지원되어 이들 간에는 항상 동일한 데이타가 유지된다. 그러나 사설 UDDI 레지스트리는 복제가 이루어지지 않은 상태에서 운용된다. 통합 시스템은 ebXML 레지스트리 간의 통합이나 UDDI 레지스트리 간의 통합과 같이 동질의 레지스트리 통합뿐만 아니라, ebXML 레지스트리와 UDDI 레지스트리간의 통합과 같은 이기종 간의 통합이 가능해야 한다.

(그림 1)은 본 논문에서 제안하고 있는 통합 질의 시스템이 ebXML 레지스트리 또는 UDDI 레지스트리와 상호 관계를 맺고 있는 모습을 보여주고 있는 그림이다. 통합 질의 시스템은 각각의 ebXML 레지스트리와 개별적인 연결을 맺어 질의를 수행하고 레지스트리로부터 반환된 결과를 통합한다. UDDI 공용 레지스트리에 통합 질의를 할 경우 이들 간에는 복제가 지원되기 때문에 공용 레지스트리 중 하나의 레지스트리에만 질의를 수행해도 각각 질의를 한 것과 같은 결과를 가져온다. 사설 UDDI 레지스트리에 통합 질의를 할 경우 ebXML 레지스트리와 마찬가지로 개별적으로 질의를 수행하고 그 결과를 통합 질의 시스템에서 통합한다. 또한 통합 질의 시스템은 ebXML 레지스트리와 UDDI 레지스트리와 같은 이기종의 레지스트리에 대한 통합 질의를 지원한다.

서로 다른 형태의 레지스트리 간의 통합을 위해서는 각각의 레지스트리의 데이타 모델을 포괄할 수 있는 새로운 공통 데이타 모델을 정의해야 한다. 이러한 공통 데이타 모델을 통하여 공통 데이타 모델과 각각의 레지스트리의 데이타 모델 간의 매핑 관계가 정의 되어진다. 뿐만 아니라 통합 질의를 위해서는 공통 데이타 모델에 기반을 둔 통합 질의 언어를 정의해야 한다.



(그림 1) 통합 질의 처리기와 레지스트리의 상호 관계

e-비즈니스 서비스를 위해 국제 표준 프레임워크인 ebXML과 웹 서비스가 이용되고 있는데, 본 논문에서는 ebXML과 웹 서비스에 대한 통합 질의를 위해 ebXML 레지스트리 버전 2.1과 UDDI 레지스트리 버전 2.0에 대한 공통 데이타 모델로서 JAXR 데이타 모델을 이용했다. 통합 질의를 위해 공통 데이타 모델을 바탕으로 통합 질의 언어를 정의하였다. 그리고 이러한 내용을 바탕으로 통합 질의 시스템을 설계하고 구현하였다.

2. 관련 연구

대표적인 e-비즈니스 표준 프레임워크로는 국제 표준으로 제정된 ebXML과 웹 서비스(Web Service)가 있다. 이러한 ebXML과 웹 서비스의 비즈니스 정보를 저장 검색할 수 있도록 하기 위해 레지스트리를 이용하는데 이들은 각각 ebXML 레지스트리와 UDDI 레지스트리를 이용한다. IBM에서는 다양한 레지스트리를 통합 접근 하고자 하는 취지에서 UDDI 레지스트리의 통합 검색을 지원하는 USML이라는 도구를 개발하였다. 이번 장에서는 ebXML과 웹 서비스 그리고 USML에 대해 알아보겠다.

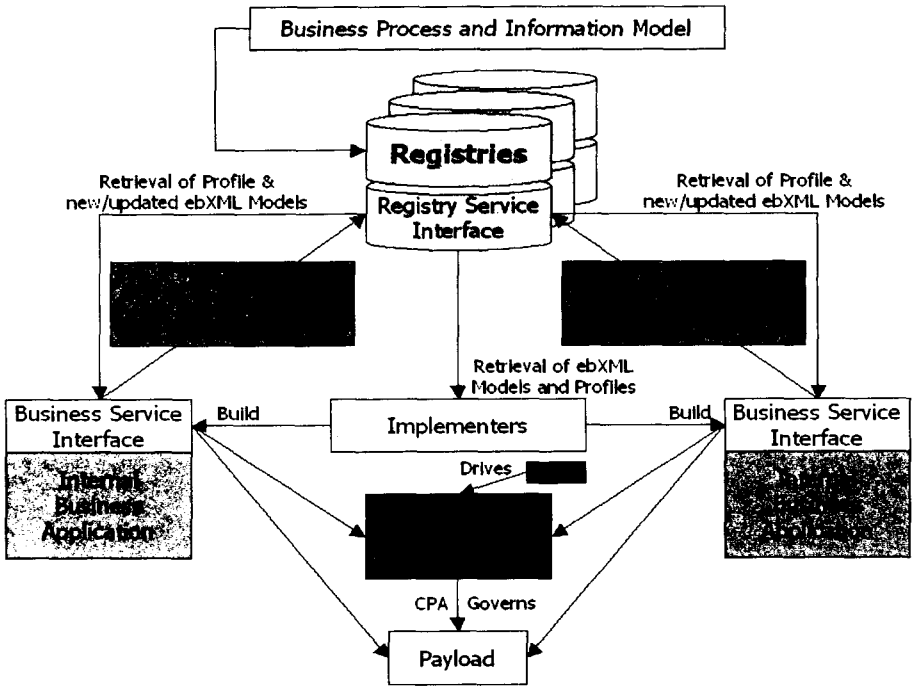
2.1 ebXML

ebXML은 UN/CEFACT와 OASIS가 주축이 되어 'Creating A Single Global Electronic Market'이라는 기치 아래 1999년 11월부터 시작하여 2001년 5월에 표준을 제정한 표준 전자상거래 프레임워크이다.

ebXML 레지스트리는 Business Process(BP), Collaboration Protocol Profile(CPP), Collaboration Protocol Agreement(CPA)와 같은 정보뿐만 아니라 Organization, Service와 같은 비즈니스 관련 객체들을 저장하고 검색할 수 있다.

(그림 2)은 ebXML의 전체적인 구조를 보여주고 있다. 거래 당사자는 다른 거래 당사자가 이해할 수 있도록 자신의 Business Process와 Business Service Interface 정보를 나타내는 CPP 문서를 생성하고 레지스트리에 저장한다. 거래 당사자가 거래를 원하면 다른 거래 당사자와의 CPP들의 교차점을 찾아내고 그들간의 상호 동의를 나타내는 CPA를 생성하고 레지스트리에 저장한다. 상호 동의가 이루어지면 두 당사자 간에 직접적인 거래가 이루어진다.

현재 ebXML 레지스트리의 개발 상황을 살펴보면, 국외에서는 홍콩대학과 SUN 사가 협력하



(그림 2) ebXML의 구조

여 ebxmlrr을 개발하여 발표하였다. 그 외에 eb XMLsoft Inc.와 xmlglobal.com에서 개발 또는 운용 중에 있다. 국내에서는 ETRI, KTNet 그리고 충남대학교 컴퓨터공학과 데이터베이스 연구실에서 ebXML 레지스트리를 개발하였다.

2.2 웹 서비스(Web Service)

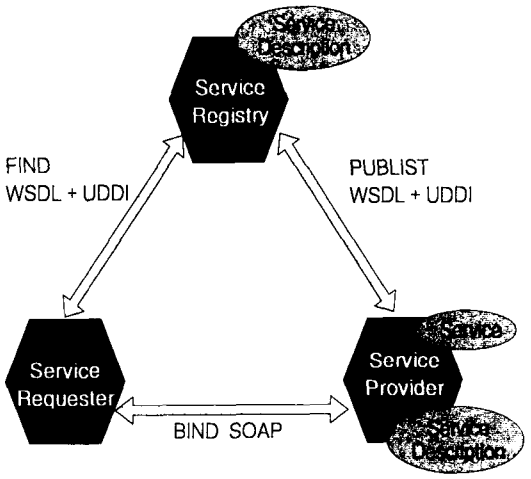
웹 서비스는 회사와 그 회사가 제공하는 서비스 등에 대한 정보를 저장, 검색하는 기능을 제공한다. 이러한 웹 서비스는 XML 메시지를 기반으로 하며 HTTP, SMTP, SOAP 등의 다양한 인터넷 프로토콜들을 통해 다른 응용들과의 직접적인 상호작용을 지원한다. 따라서 플랫폼이나 개발 언어에 독립적이라는 특징이 있다.

UDDI(Universal Description, Discovery, and Integration) 레지스트리는 웹 서비스를 등록하거나 검색하는데 이용되는 레지스트리이다. UDDI 레지스트리는 서비스 제공회사에 대한 정보뿐만 아니라 서비스 자체에 대한 정보를 제공한다. 현재 UDDI 레지스트리 V2.0에서는 다른 레지스트리의 정보를 복사(replication)함으로써 각각의 레지스트리는 항상 최신의 이용 가능한 정보를 제공한다.

(그림 3)은 웹 서비스 구성요소 간의 상호 작용을 보여주고 있다. 서비스 제공자는 서비스를 생성해서 서비스 레지스트리에 등록을 한다. 일단 서비스가 등록이 되면 서비스 요구자는 UDDI 인터페이스를 통해 서비스를 찾을 수 있다. 서비스 레지스트리는 서비스 요구자에게 WSDL 서비스 기술과 서비스 자체를 가리키고 있는 URL을 제공한다. 그 다음에 서비스 요구자는 서비스에 직접 바인딩하기 위해 이 정보를 이용할 수 있다.

UDDI 레지스트리는 크게 공용 레지스트리와 사설 레지스트리의 두 개의 부분으로 나누어 볼

수 있다. 공용 레지스트리는 IBM, Microsoft, SAP 그리고 NTT Com에서 운용하고 있다. 또한 사설 레지스트리는 우리나라의 동양그룹에서 운용 중에 있다. 그밖에 IBM에서 사설 IBM UDDI 레지스트리를 개발해 배포하고 있으며, Microsoft에서 사설 UDDI 레지스트리를 .NET 플랫폼의 일부로 제공한다.



(그림 3) 웹 서비스 구성요소의 상호작용

2.3 USML

다중 UDDI 레지스트리를 질의할 수 있도록 IBM에서 USML(UDDI Search Markup Language)이라는 도구를 개발하였다. USML은 다양한 방법으로 서로 다른 레지스트리를 검색하는 질의들을 조합한 형태이다. 또한 USML은 한번의 질의 요구로 복잡한 질의를 수행할 수 있는 표준 인터페이스를 제공한다. 질의와 응답 메시지는 모두 XML 인스턴스의 형태를 취한다. USML은 다중 질의의 Business, Service 그리고 ServiceType의 질의 결과를 조합한다.

(그림 4)은 USML의 질의 메시지의 예로서 여러 UDDI 레지스트리에 질의를 수행할 수 있음을 보여주는 그림이다. URL이 http://wsbi10/services/uddi/inquiryAPI인 사설 UDDI 레지스트리에서 'Microsoft'라는 이름을 갖는 Business를 찾아 반환하고, URL이 http://wsbi5/services/uddi/servlet/uddi인 사설 UDDI 레지스트리에서 'KEPEX'라는 이름을 갖는 Business를 찾아 반환한다. 그런데, AggOperator를 통해서 통합 연산자가 OR임을 나타내기 때문에 각각의 레지스트리에서 반환된 Business 정보를 최종 결과로서 모두 반환하게 된다.

```
<?xml version = "1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
    <ProcessId>9999</ProcessId>
    <Query>
        <Source>Private UDDI</Source>
        <SourceURL>http://wsbi10/services/uddi/
                    inquiryAPI</SourceURL>
        <BusinessName>Microsoft</BusinessName>
        <FindBy>Business</FindBy>
    </Query>
    <Query>
        <Source>Public UDDI</Source>
        <SourceURL>http://wsbi5/services/uddi/
                    servlet/uddi</SourceURL>
        <BusinessName>KEPEX</BusinessName>
        <FindBy>Business</FindBy>
    </Query>
    <AggOperator>OR</AggOperator>
</Search>
```

(그림 4) USML의 질의 메시지의 예

(그림 5)은 위의 질의문에 대한 응답 문으로서 Business 이름이 'Microsoft Corporation'인 회사에 대한 정보를 보여주는 예이다. 반환되는 Business는 자신의 여러 가지 정보를 보여주게 된다. 즉, 'Microsoft Corporation'이라는 이름을 갖는 Business는 자신의 id키 값으로 'fadc7f22-bab7-4364-b36f-facedc52553a'을 갖고, Business에 대한 설명은 갖고 있지 않다. 그리고 URL

과 Operator에 대한 정보를 가지고 있다. 또한 Contact 정보로서 email 주소와, 연락 대상에 대한 이름과 설명들이 포함된다.

그러나, USML은 몇 가지 제한점을 가지고 있다. 첫째, USML은 검색 대상이 되는 레지스트리가 UDDI 레지스트리로만 제한되어 있다. 따라서 ebXML이나 그 밖의 레지스트리에 대한 검색이 불가능하다. 둘째, USML은 Business, Service, ServiceType에 대한 세 가지 객체에 대한 검색만 지원한다. 셋째, 레지스트리에 대한 연산자는 AND와 OR 두 개의 연산만 지원하기 때문에 사용자에게 좀 더 다양한 질의를 제공하지 못한다. 넷째, USML은 중첩 질의를 제공하지 못하는 한계를 가지고 있다. 따라서 본 논문에서는 USML이 가지고 있는 이러한 한계점을 극복하였다.

```
<?xml version = "1.0"?>
<USMLResponse>
    <Business>
        <BusinessName> Microsoft Corporation
                </BusinessName>
        <BusinessKey> fadc7f22-bab7-4364-b36f-
                        facedc52553a </BusinessKey>
        <Description> null </Description>
        <URL> http://uddi.rte.microsoft.com/
                        discovery?businessKey=
                        fadc7f22-bab7-4364-b36f-
                        facedc52553a</URL>
        <Operator>Microsoft Corporation</Operator>
        <Contacts>
            <Contact>
              <Email> uddiask@microsoft.com
                        </Email>
             <PersonName>UDDI Team
                        </PersonName>
             <DefaultDescription>Inquiries, bug
                        reports, and account upgrade
                        requests</DefaultDescription>
            </Contact>
        </Contacts>
    </Business>
</USMLResponse>
```

(그림 5) USML의 응답 메시지의 예

3. 공통 데이터 모델 정의

서로 다른 이질 레지스트리 간의 통합 질의를 위해서는 먼저 레지스트리 각각의 데이타 모델에 대한 이해가 필요하다. 레지스트리에 어떤 객체들이 정의되고 그 객체가 어떤 일을 담당하는지, 그리고 그들 객체들이 어떤 관계를 가지고 서로 관련되는지를 이해해야 한다. 레지스트리에 분석 후에 각각의 데이타 모델들을 기반으로 각각의 데이타 모델을 포괄할 수 있는 공통 분모격의 공통 데이타 모델을 정의해야 한다. 이러한 공통 데이타 모델을 정의하고 나면 각각의 데이타 모델에 대해 알 필요가 없고 공통 데이타 모델에 대해서만 이해하면 된다.

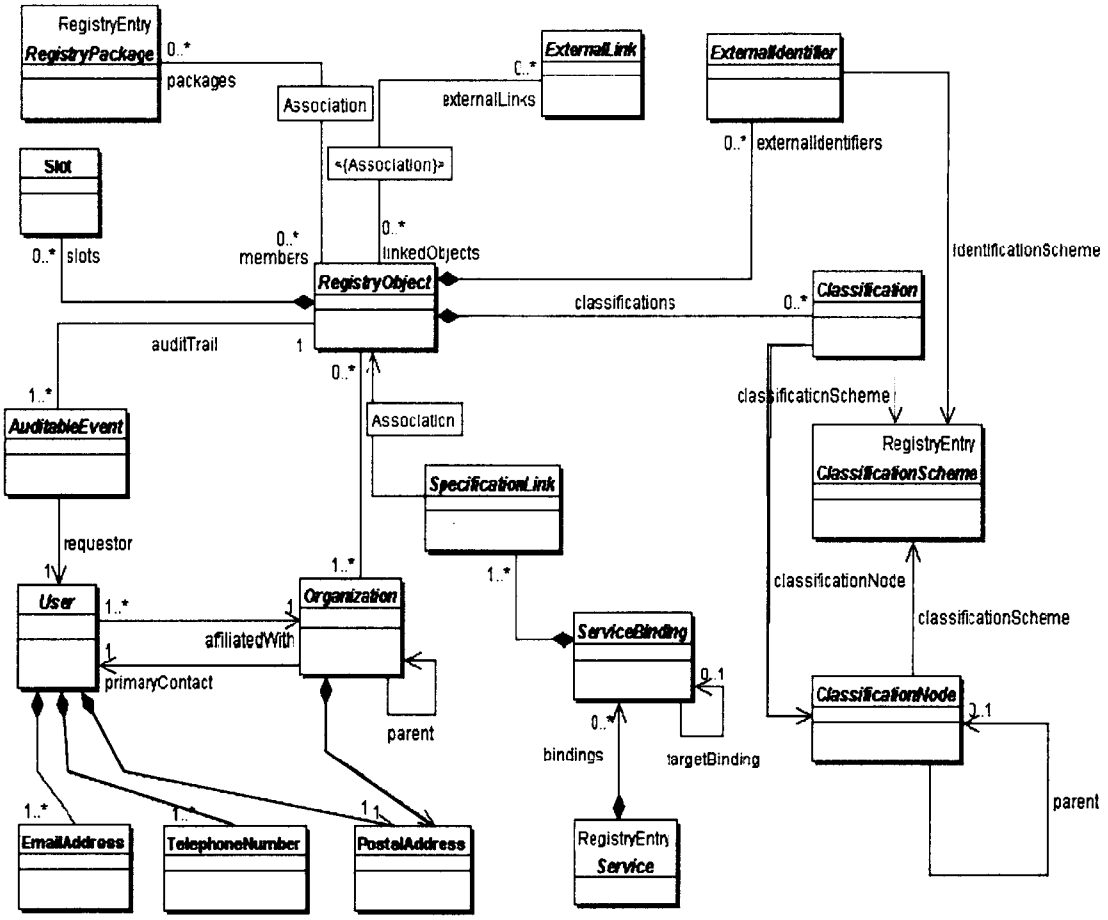
마찬가지로 ebXML과 UDDI 레지스트리의 통합 질의를 위해서는 ebXML 레지스트리와 UDDI 레지스트리를 기반으로 공통 데이타 모델을 정의해야 한다. 공통 데이타 모델이 정의되면 이 공통 데이타 모델에 기반을 두어 통합 질의 언어를 생성하게 된다. 물론 내부적으로 그 질의를 각각의 데이타 모델에 매핑시켜주는 단계를 거치게 된다.

3.1 ebXML 데이타 모델

(그림 6)은 ebXML 레지스트리 V2.1에서 각각의 객체들과 그들 간의 관계를 보여주는 그림이다.

ebXML의 데이타 모델은 Business에 대한 객체들, 그들 간의 관계를 맺기 위한 객체들, 그리고 어떤 분류 체계를 제공하기 위한 객체들이 유기적으로 관련되어 있다.

Organization은 기업에 대한 정보를 제공하는 RegistryObject이다. Service는 Organization이

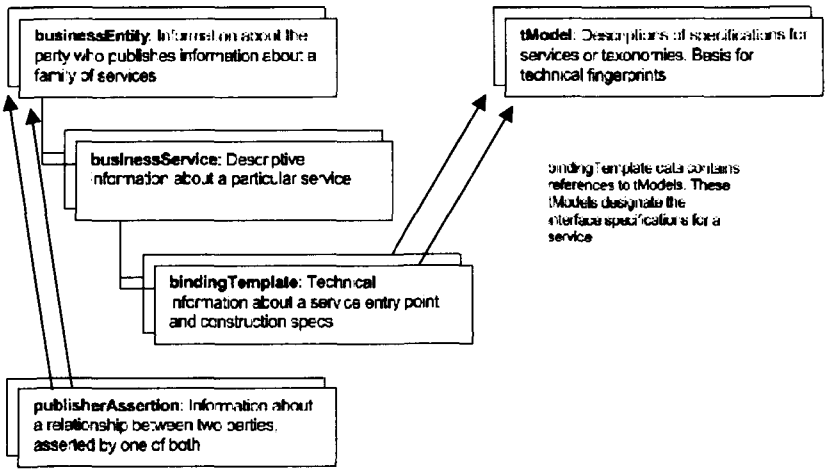


(그림 6) ebXML의 데이타 모델

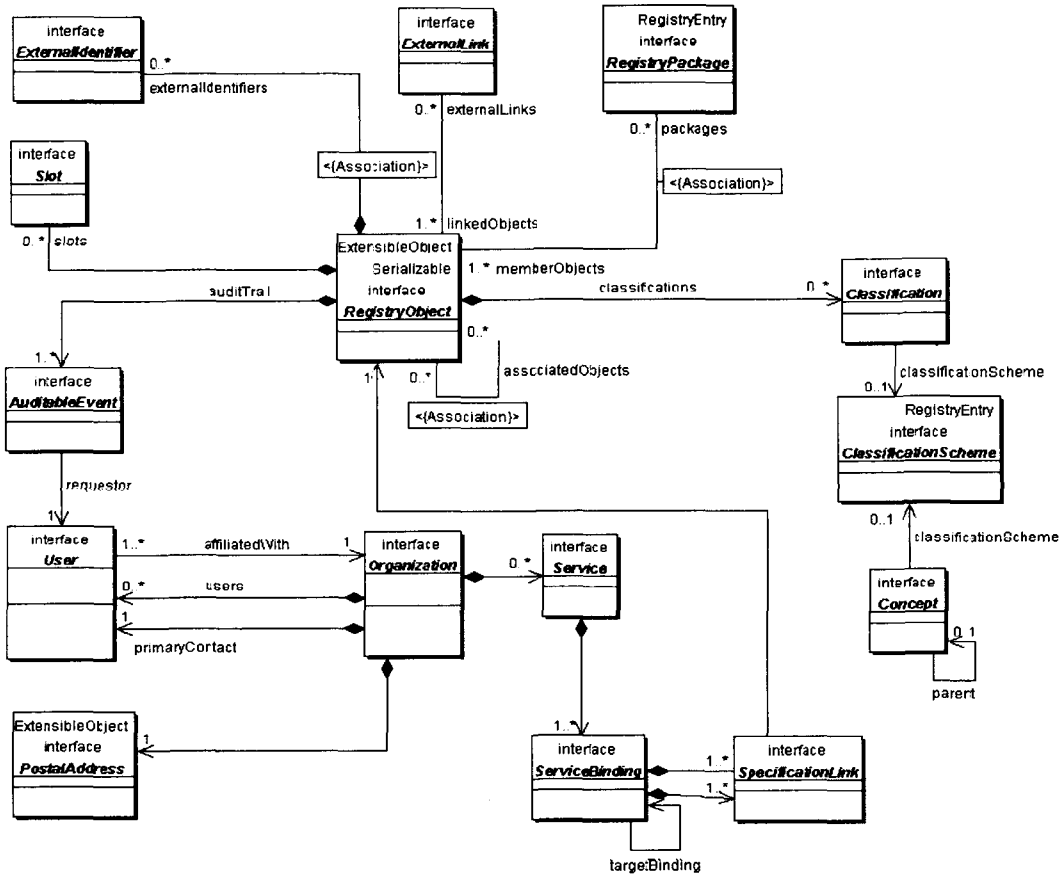
제공하는 서비스이며, ServiceBinding은 Service에 의해 제공되는 인터페이스를 접근하는 방법에 대한 기술적인 정보를 제공한다. SpecificationLink는 ServiceBinding과 그 Service Binding을 가지고 서비스를 어떻게 이용할 수 있는지를 기술하는 기술 명세들 중 하나와의 연결고리를 제공한다. User는 레지스트리 내부에 등록된 사용자에 대한 정보를 제공한다. PostalAddress는 우편 주소의 속성들을 정의하는 클래스이며, EmailAddress는 전자우편 주소의 속성들을 정의하는 클래스이다.

Association은 객체들간의 다대다 관계를 정의하기 위해 이용된다. ExternalIdentifier는 RegistryObject에 추가적인 고유 신원을 제공한다. ExternalLink는 레지스트리에 의해 관리되지 않는 컨텐츠에 고유한 URI를 형성해준다. RegistryPackage는 논리적으로 연관된 RegistryObject들을 하나의 그룹으로 묶는다.

ClassificationScheme은 RegistryObject를 분류하는 구조화된 방법을 기술한다. ClassificationNode는 ClassificationScheme 하위의 트리구조를 정의하기 위해 이용된다. Classification은 다른 RegistryObject들을 분류하기 위해 이용된다.



(그림 7) UDDI의 데이타 모델



(그림 8) JAXR의 데이타 모델

3.2 UDDI 데이타 모델

(그림 7)은 UDDI 레지스트리 V2.0의 데이타 모델을 보여주는 그림이다.

businessEntity는 회사에 대한 정보나 그 회사가 제공하는 서비스를 등록하는 회사 또는 개체에 대한 모든 정보를 나타낸다. businessService는 회사가 제공하는 서비스에 대한 기술적인 정보를 나타낸다. 각각의 businessService는 하나의 businessEntity의 자손이다. bindingTemplate은 서비스에 대한 기술적인 설명을 수용한다. bindingTemplate은 서비스에 대한 기술적인 엔트리 포인트를 결정하도록 한다. 각각의 bindingTemplate은 하나의 논리적인 businessService를 부모로 갖는다. tModel은 분류법이나 서비스에 대한 명세의 설명을 가지고 있다. publishAssertion은 두 당사자 사이의 관계에 대한 정보를 포함한다.

3.3 공통 데이타 모델

SUN에서 ebXML과 UDDI 등의 레지스트리에 대한 표준화된 공통 데이타 모델로 JAXR (Java API for XML Registries) 데이타 모델을 제안하였다. JAXR은 UDDI 레지스트리와 JAXR의 데이타에 대한 매핑과, ebXML 레지스트리와 JAXR과의 데이타 매핑을 정의하고 있다.

(그림 8)은 JAXR의 데이타 모델을 보여주는 그림이다. JAXR 데이타 모델은 ebXML 데이타 모델과 비슷한 구조를 가지고 있다. Organization과 Service와 같은 비즈니스 관련 객체들이 있고, 레지스트리에 저장되는 객체들의 체계적인 분류 체계를 지원하기 위한 Classification, Concept과 같은 객체들, 그리고 레지스트리 내의 객체들 사이의 연관 관계를 맺기 위한 Association과 같은 객체들이 존재한다.

3.3.1 비즈니스 관련 객체들

Organization	• Submitting Organization과 같은 기업에 대한 정보 • 각각의 Organization은 부모 Organization에 대한 참조를 가짐 • Service의 집합을 가짐
Service	• Organization에 의해 제공되는 서비스에 대한 정보 • ServiceBinding의 집합을 가짐
ServiceBinding	• Service에 의해 제공되는 인터페이스를 접근하는 방법에 대한 기술적인 정보 • SpecificationLink의 집합을 가짐
Specification Link	• ServiceBinding과 서비스를 어떻게 이용하는지를 기술하고 있는 기술 명세(technical specification)와의 연결고리를 제공
User	• 레지스트리 내에서 등록된 사용자에 대한 정보 • 각각의 User는 하나의 Organization에 소속 • RegistryObject에 대한 감사에서 이용
PostalAddress	• 우편 주소의 속성을 정의 • User와 Organization의 주소 정보를 제공하기 위해 이용

3.3.2 분류체계 관련 객체들

Classification Scheme	• RegistryObject를 분류하기 위해 이용되는 분류법 • 트리구조의 root
Classification	• 분류 체계를 이용해서 RegistryObject를 분류 • 레지스트리 내에서 RegistryObject를 빠르게 발견할 수 있게 함
Concept	• 계층적인 트리 구조나 분류 체계의 세부적인 요소들을 정의 • 트리구조의 하위 노드

3.3.3 객체 사이의 관계를 위한 객체들

Association	• 객체들 간의 다대다 관계를 정의
Registry Package	• 논리적으로 관련된 RegistryObject들을 묶는데 이용 • 임의 개수의 RegistryObject를 포함 • 임의 개수의 RegistryPackage의 멤버
External Identifier	• RegistryObject에 추가적인 신원정보 제공 • 잘 알려진 신원 체계, 소유자의 신원 체계 이용 가능 • ClassificationScheme 재사용 할 수 있음
ExternalLink	• 외부 컨텐츠에 URI를 이용해서 외부 컨텐츠에의 연결을 제공

3.3.4 그 외 객체들

ExtensibleObject	• Slot을 찾거나 추가하고 삭제하는 방법을 제공
Slot	• 런타임시 RegistryObject에 임의의 속성을 추가할 수 있는 동적인 방법을 제공
AuditableEvent	• RegistryObject에 감사를 제공

4. 통합 질의어 설계

공통 데이타 모델이 정의 되면 공통 데이타 모델을 기반으로 통합 질의어를 설계해야 한다. 본 논문은 e-비즈니스 레지스트리에 대한 통합 질의어(EBRIQuery)를 정의했다. EBRIQuery는 Organization과 Service 등과 같은 비즈니스 레벨의 객체에 대한 검색을 지원하며, 중첩 질의가 가능하다. 또한, 레지스트리의 반환 객체들에 대한 통합을 위해서 통합 연산자를 정의하고 이를 이용한다.

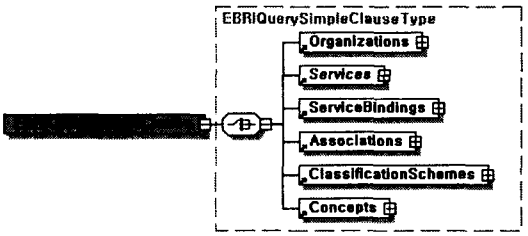
4.1 지원 질의

EBRIQuey에서 지원하는 질의는 다음과 같은 4가지가 있다.

4.1.1 비즈니스 레벨의 객체 질의

통합 질의 시스템에서 검색 가능한 객체는 Organization, Service, ServiceBinding, ClassificationScheme, Association, Concept의 6가지이다. (그림 9)은 통합 질의 시스템에서 검색 가능한 객체들을 질의하기 위한 질의 구조를 보여주고 있다. 각각의 객체들은 EBRIQuery SimpleClause의 하위 요소이다. EBRIQuery SimpleClause는 6가지 질의 객체 중에 질의 하고자 하는 객체를 하위 요소로 포함한다.

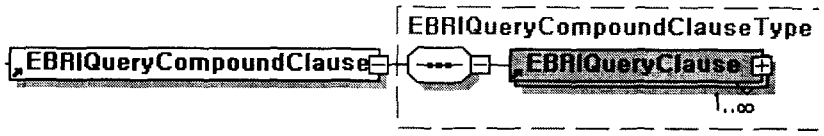
EBRIQuerySimpleClause는 url이라는 속성을 갖는데, 이 url 속성에 질의 하고자 하는 레지스트리의 URL을 지정한다. 즉, EBRIQuerySimpleClause는 대상 레지스트리를 지정하고 그 레지스트리에서 질의 객체 중 하나를 지정하여 질의를 수행한다.



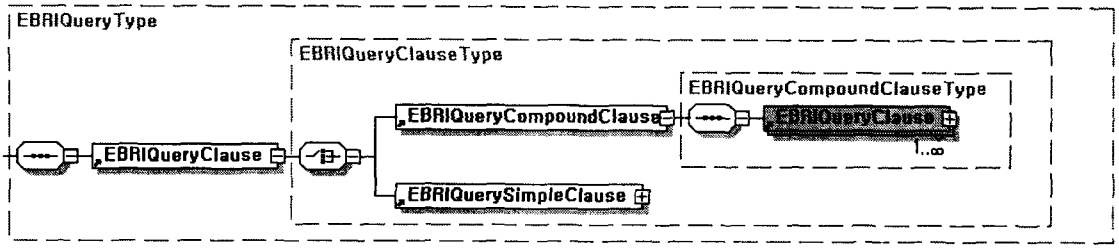
(그림 9) 통합 질의 시스템의 질의 객체

4.1.2 다중 레지스트리 통합 질의

(그림 10)은 통합 질의 시스템에서 다중 레지스트리의 통합 질의를 위한 구조를 보여주고 있다. EBRIQueryCompoundClause는 여러 개의 EBRIQueryClause를 하위 요소로 포함한다. EBRIQueryClause는 하나의 EBRIQuerySimpleClause를 하위 요소로 갖기 때문에 EBRI QueryCompoundClause는 여러 개의 EBRIQuerySimpleClause를 갖는 의미가 된다. 따라서 EBRIQuerySimpleClause를 통해 각각의 레지스트리에 실행되는 질의가 EBRIQueryCompound



(그림 10) 다중 레지스트리 통합 질의 구조



(그림 11) 중첩 질의 구조

Clause를 통해 하나로 묶여진다. 이 묶여진 질의들은 통합 연산자를 통해 결과를 조합하게 된다.

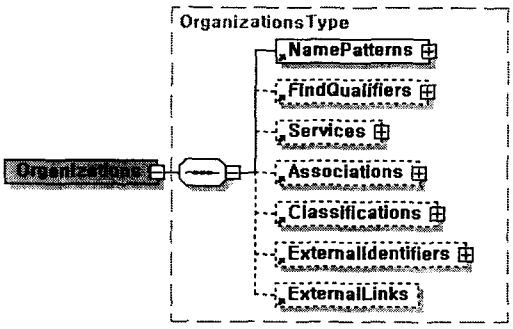
4.1.3 중첩 질의

(그림 11)은 통합 질의 시스템에서 중첩 질의를 지원하기 위한 구조를 보여주고 있다. 통합 질의 시스템에서 중첩 질의는 EBRIQueryClause가 EBRIQueryCompoundClause를 포함하고 EBRIQueryCompoundClause가 다시 EBRIQueryClause를 포함한다. 이런 순환 구조는 EBRIQueryClause가 EBRIQuerySimpleClause를 하위 요소로 만날 때까지 계속된다. 즉, EBRIQueryClasue는 EBRIQueryCompoundClause와 EBRIQuerySimpleClause를 하위 앨리먼트로 가질 수 있는데, EBRIQueryCompoundClause를 가질경우 계속적인 순환이 이루어지게 되고, EBRIQuerySimpleClause를 가질 경우 순환이 종료되어 레지스트리에 직접적인 질의가 이루어지게 된다.

4.1.4 객체 관계성 등의 조건을 이용한 질의

질의 객체 7가지를 질의 할 경우 NamePattern과 FindQualifier와 같은 단순한 조건만을 명시하여 질의를 할 수도 있지만, 공통 데이타 모델에 정의된 객체들 간의 관계를 이용하여 질의 조건을 부여할 수도 있다. NamePattern은 질의하고자 하는 객체들의 이름을 지정하기 위해 이용된다. FindQualifier는 그러한 NamePattern 값들의 관계, 즉 정렬 방법이나 대소문자 구별과 같은 조건들을 지정하기 위해 이용된다. 객체 관계성을 이용한 질의는 공통 데이타 모델에 정의된 Association이나 Classification과 같은 객체들 간의 관계를 이용한다.

(그림 12)은 Organization을 검색할 경우 Association, Classification, ExternalLink, ExternalIdentifier 등의 관계를 나타내는 조건을 이용



(그림 12) 관계성을 이용한 질의 구조의 예

할 수 있음을 나타내고 있다.

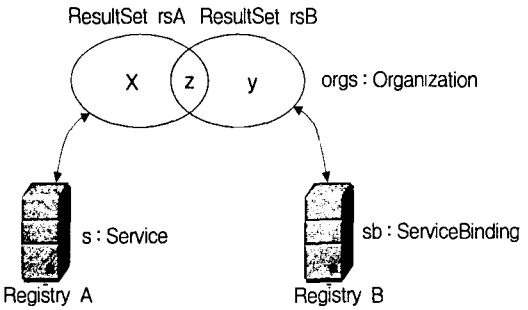
4.2 통합 연산자

통합 연산자는 각각의 레지스트리에 질의를 실행하고, 그 반환된 결과 값들을 통합 처리하기 위한 연산자이다. 이러한 통합 연산자는 UNION, INTERSECT, DIFFERENCE의 세 가지 연산을 지원하다. 이러한 통합 연산자가 결과를 통합하기 위해서는 최종적으로 어떠한 반환 값을 반환할 것인지 알아야 한다. 따라서 EBRIQueryCompoundClause는 통합 연산자를 지정하기 위한 속성과 최종적으로 반환해야 할 객체를 지정하기 위한 속성을 가지고 있다. 반환 객체의 지정은 최상위 EBRIQueryCompoundClause에서만 지정 되어야 한다.

비록 EBRIQuerySimpleClause에서 어떤 객체를 결과 값으로 반환하도록 지정되어 있다고 하더라도, EBRIQueryCompoundClause에서 지정된 반환 값을 EBRIQuerySimpleClause에서 최종적으로 반환을 해야만 한다. 질의를 위해 여러 개의 레지스트리가 이용된다고 하더라도 결국 모든 레지스트리는 EBRIQueryCompound Clause에서 지정된 반환 객체를 반환하게 된다.

4.2.1 UNION

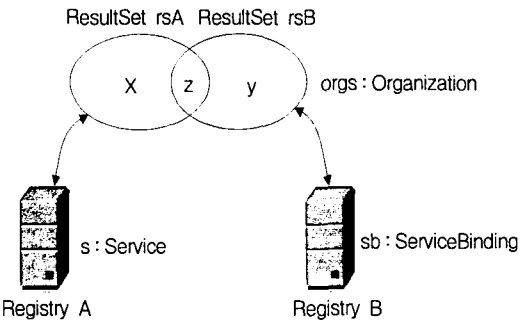
통합 연산자 UNION은 반환된 객체들의 중복된 이름을 제거한 상태에서 모든 객체들을 반환한다. (그림 13)은 UNION 통합 연산의 예를 보여주는 그림이다. Registry A에서 Services를 갖는 Organization을 질의하고 Registry B에서는 ServiceBinding sb를 갖는 Organization을 질의한다. 그리고 각각의 레지스트리에서 반환된 Organization을 UNION 연산을 취한다. 여기서 Organization의 이름들을 비교해서 같은 이름을 갖는 Organization은 중복을 배제하고 하나의 Organization만을 취한다. 결과적으로 반환되는 결과는 x, y, z영역에 해당하는 Organization이 된다. 반환 값은 rsA ∪ rsB = {x, y, z} 이다.



(그림 13) 통합 연산자 UNION의 예

4.2.2 INTERSECT

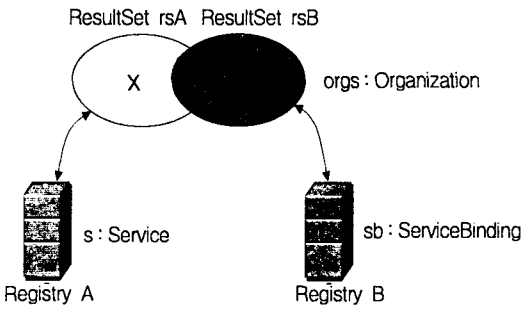
통합 연산자 INTERSECT는 각각의 레지스트리에서 반환된 결과들의 이름을 비교해서 같은 이름을 갖는 객체들을 반환한다. (그림 14)은 INTERSECT 통합 연산자의 예를 보여주고 있다. Registry A에서 Service s를 갖는 Organization을 반환한 값 rsA와, Registry B에서 ServiceBinding sb를 갖는 Organization을 반환한 값 rsB를 반환하여 그 반환된 Organization들의 이름을 비교하여 같은 이름을 갖는 Organization을 반환한다. 결과 값은 rsA ∩ rsB = {z} 이다.



(그림 14) 통합 연산자 INTERSECT의 예

4.2.3 DIFFERENCE

통합 연산자 DIFFERENCE는 하나의 레지스트리로부터의 반환 결과에서 또 다른 레지스트리의 반환 결과에 포함된 부분을 제외한 결과를 반환한다. (그림 15)은 DIFFERENCE 통합 연산자의 예를 보여주고 있다. Registry A에서 Service s를 갖는 Organization의 반환 값에서 Registry B에서 ServiceBinding sb를 갖는 Organization의 반환 값을 제외한 값이다. Registry A에서 Service s를 갖는 Organization의 반환 값 rsA에서 Registry B에서 ServiceBinding sb를 갖는 Organization인 반환 값 rsB를 제외한 반환 값이다. 결과 값은 rsA - rsB = {x}이 된다.



(그림 15) 통합 연산자 DIFFERENCE의 예

4.3 통합 질의의 XML 스키마

(그림 16)은 통합 질의 구조를 XML 스키마를 이용하여 표현한 내용의 일부이다. EBRIQuery는 EBRIQueryType을 갖는 엘리먼트이다. EBRIQueryType은 EBRIQueryClause를 하위 엘리먼트로 갖는다. EBRIQueryClause는 마찬가지로 EBRIQueryClauseType을 갖는다. EBRIQueryClauseType은 EBRIQueryCompoundClause와 EBRIQuerySimpleClause를 하위 엘리먼트로 갖는데, 두 개의 엘리먼트 중 하나만 가질 수 있다. 그리고 EBRIQueryCoumpoundClause는 EBRIQueryCoumpoundClauseType을 타입으로 갖는다. EBRIQueryCoumpoundClauseType는 여러 개의 EBRIQuery

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd : schema xmlns : xsd = "http://www.w3.org/
                              2001/XMLSchema">
  <xsd : element name = "EBRIQuery"
                 type = "EBRIQueryType"/>
  <xsd : complexType name = "EBRIQueryType">
    <xsd : sequence>
      <xsd : element ref = "EBRIQueryClause"/>
    </xsd : sequence>
  </xsd : complexType>
  <xsd : element name = "EBRIQueryClause"
               type = "EBRIQueryClauseType"/>
  <xsd : complexType name = "EBRIQuery
                             ClauseType">
    <xsd : choice>
      <xsd : element ref = "EBRIQueryCompound
                            Clause"/>
      <xsd : element ref = "EBRIQuerySimple
                            Clause"/>
    </xsd : choice>
  </xsd : complexType>
  <xsd : element name = "EBRIQueryCompound
                         Clause"
        type = "EBRIQueryCompoundClauseType"/>
  <xsd : complexType name = "EBRIQuery
                             CompoundClauseType">
    <xsd : sequence>
      <xsd : element ref = "EBRIQueryClause"
                   maxOccurs = "unbounded"/>
    </xsd : sequence>
      <xsd : attribute name = "aggOperator"
                   use = "required">
        <xsd : simpleType>
          <xsd : restriction base = "xsd : string">
            <xsd : enumeration value = "UNION"/>
            <xsd : enumeration value =
                   "INTERSECT"/>
            <xsd : enumeration value =
                   "DIFFERENCE"/>
          </xsd : restriction>
        </xsd : simpleType>
      </xsd : attribute>
      <xsd : attribute name = "returnType" use = "optional">
    ...
```

(그림 16) 통합 질의의 XML 스키마의 일부

Clause를 하위 엘리먼트로 갖는다. 이러한 구조에서 알 수 있듯이 EBRIQueryClause가 EBRIQueryCoumpoundClause가 EBRIQueryCoumpound Clause를 포함하고 다시 Clause를 포함하기 때문에 EBRIQuery는 중첩이 가능한 질의의 형태를 취하고 있다.

또한 EBRIQueryCoumpoundClause는 두 개의 속성을 가지고 있다. 하나는 통합 연산자를 지정하기 위한 aggOperator이고 다른 하나는 반환 객체에 대한 형태를 지정하기 위한 returnType이다. aggOperator는 UNION, INTERSECT, DIFFERENCE의 3가지 연산자 중 하나를 선택해야 한다. aggOperator는 EBRIQueryCoumpoundClause에 반드시 지정되어야 하며, returnType는 선택 사항으로 최상위 EBRIQueryCoumpoundClause에만 이용된다.

4.3.1 질의 메시지의 예

(그림 17)은 질의 메시지를 XML 질의 스키마에 유효하게 만든 XML 통합 질의 메시지의 예이다. 이 질의는 두 개의 서로 다른 레지스트리에 대해 통합 질의를 하는 예이다. http://registry.csis.hku.hk : 8201/ebxmlrr/registry의 URL을 갖는 레지스트리에서 'Software'라는 이름의 Service를 갖는 Organization을 검색하고, http://uddi.ibm.com/ubr/inquiryapi의 URL을 갖는 레지스트리에서 'Hardware'라는 이름의 Service를 갖는 Organization을 검색한다. EBRIQueryCompoundClause의 aggOperator 속성에 통합 연산자가 *UNION*으로 지정되어 있기 때문에 두 레지스트리에서 반환된 Organization 객체들을 모두 최종 결과로서 반환을 한다.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<EBRIQuery xmlns : xsi = "http://www.w3.org/2001/
                XMLSchema-instance"
    xsi : noNamespaceSchemaLocation =
        "EBRIQuery.xsd">
    <EBRIQueryClause>
        <EBRIQueryCompoundClause aggOperator = "UNION"
            returnType = "Organizations">
            <EBRIQueryClause>
                <EBRIQuerySimpleClause url = "http://registry.
                    csis.hku.hk : 8201/ebxmlrr/registry">
                    <Services>
                        <NamePatterns>
                            <NamePattern np = "Software"/>
                        </NamePatterns>
                    </Services>
                </EBRIQuerySimpleClause>
            </EBRIQueryClause>
            <EBRIQueryClause>
                <EBRIQuerySimpleClause url = "http://uddi.ibm.
                    com/ubr/inquiryapi">
                    <Services>
                        <NamePatterns>
                            <NamePattern np = "Hardware"/>
                        </NamePatterns>
                        <FindQualifiers>
                            <FindQualifier fq = "CASE_
                                SENSITIVE_MATCH"/>
                        </FindQualifiers>
                    </Services>
                </EBRIQuerySimpleClause>
            </EBRIQueryClause>
        </EBRIQueryCompoundClause>
    </EBRIQueryClause>
</EBRIQuery>
```

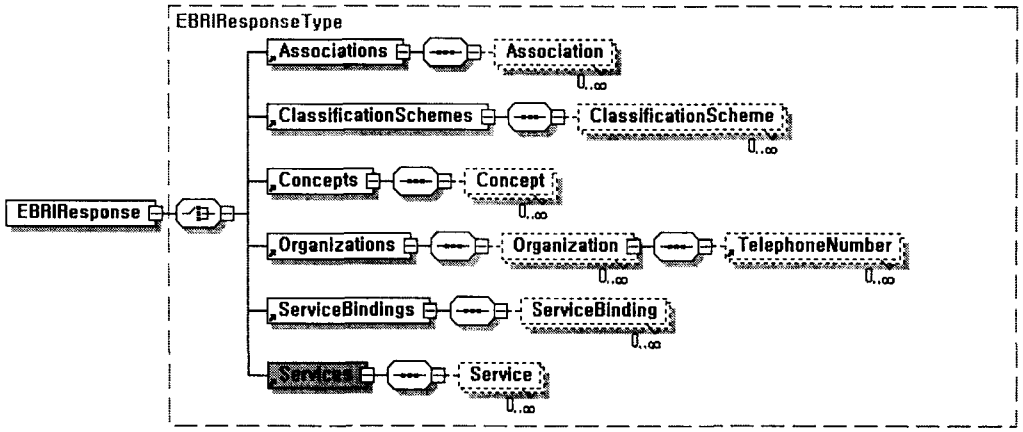
(그림 17) XML 통합 질의 메시지의 예

4.4 응답 구문의 설계

응답 구문은 EBRIQuery의 반환 객체를 가질 수 있어야 한다. 즉, Organization, Service, ServiceBinding, Association, ClassificationScheme, Concept의 6가지 객체를 포함한다. 그리고 Organization의 경우 TelephoneNumber를 하위 요소로 가진다.

4.4.1 응답 구문의 구조

(그림 18)은 응답 구문의 구조를 나타내는 그



(그림 18) 응답 구문의 구조

림이다. EBRIQuerySimpleClause는 검색 가능한 객체들의 집합, 즉 Associations, ClassificationSchemes, Concepts, Organizations, Service Bindings, Services의 6개를 하위 요소로 갖는다. 그리고 각각은 여러 개의 반환 객체들을 가지고 있다. 또한 Organization은 주소 정보로 TelephoneNumber를 하위 요소로 갖는다.

4.5 응답 구문의 XML 스키마

(그림 19)은 응답 구문의 구조를 XML 스키마를 이용하여 표현한 내용의 일부이다. 최상위 엘리먼트인 EBRIResponse는 EBRIResponse Type의 타입을 갖는다. EBRIResponse은 Organizations, Services, ServiceBindings, Associations, ClassificationSchemes, Concepts와 같은 엘리먼트들을 하위 엘리먼트로 갖는다. 또한, Organizations는 여러 개의 Organization을 하위 엘리먼트로 갖는다. 마찬가지로 Associations는 Association을, ClassificationSchemes는 ClassificationScheme을, Concepts는 Concept을, Services는 Service를, ServiceBindings는 ServiceBinding을 자신들의 하위 엘리먼트로 갖게 된다.

또한 Organization은 TelephoneNumber를 하위 엘리먼트 가진다. TelephoneNumber는 Organization에 대한 연락 방법을 제공해주기 위해 이용된다. Organization은 TelephoneNumber와 같은 엘리먼트를 하위 엘리먼트로 갖지만, 그 외에도 자신의 속한 레지스트리의 주소를 나타내기 위한 url, 자신의 id, 이름, 설명에 대한 값을 지정하기 위한 속성들, 자신의 부모 Organization을 지정하기 위한 속성, 그리고 주 연락 대상에 대한 이름이 지정되는 속성 등을 갖는다.

그 외의 Association, ClassificationScheme, Concept, Service, ServiceBinding 엘리먼트에 대해서도 비슷한 구조를 가지며 자신들의 id, 이름, 설명 등 여러 가지 속성들과, 소속된 레지스트리의 주소를 나타내기 위한 url 속성을 포함해서 최종 결과로 반환한다.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd : schema xmlns : xsd = "http://www.w3.org/2001/
        XMLSchema">
  <xsd : element name = "EBRIResponse"
      type = "EBRIResponseType"/>
  <xsd : complexType name = "EBRIResponseType">
    <xsd : choice>
      <xsd : element ref = "Associations"/>
      <xsd : element ref = "ClassificationSchemes"/>
```

```
        <xsd : element ref = "Concepts"/>
        <xsd : element ref = "Organizations"/>
        <xsd : element ref = "ServiceBindings"/>
        <xsd : element ref = "Services"/>
      </xsd : choice>
  </xsd : complexType>
  <xsd : element name = "Organizations">
    <xsd : complexType>
      <xsd : sequence>
        <xsd : element name = "Organization"
              minOccurs = "0"
              maxOccurs = "unbounded">
          <xsd : complexType>
            <xsd : sequence>
              <xsd : element
                  ref = "TelephoneNumber"
                  minOccurs = "0"
                  maxOccurs = "unbounded"/ >
            </xsd : sequence>
            <xsd : attribute name = "url"
                  type = "xsd : string"
                  use = "required"/>
            <xsd : attribute name = "id"
                  type = "xsd : string"
                  use = "required"/>
            <xsd : attribute name = "name"
                  type = "xsd : string"
                  use = "optional"/>
            <xsd : attribute name = "description"
                  type = "xsd : string"
                  use = "optional"/>
            <xsd : attribute name = "parent"
                  type = "xsd : string"
                  use = "optional"/>
            <xsd : attribute name = "primaryContact"
                  type = "xsd : string"
                  use = "optional"/>
          </xsd : complexType>
        </xsd : element>
      </xsd : sequence>
    </xsd : complexType>
  </xsd : element>
<xsd : element name="Services">
...
```

(그림 19) 응답 구문의 XML 스키마의 일부

4.5.1 응답 메시지의 예

(그림 20)은 XML 응답 스키마에 유효하게 만든 XML 응답 메시지의 예이다. Organization의 id가 '388678e9-40df-4412-ac65-c42bbb265e-7d' 이고 이름이 'ebxmlrr test registry'인 Organization과 id가 '80dc6270-eb0c-11d6-b618-000629dc0a53'이고 이름이 'IBM'인 Organization, 그리고 id가 'fdfdbba0-a7d3-11d5-a30a-002035229c64'이고 이름이 'IBM WSTK Tutorial'인 세개의 Organization이 최종 결과로 반환되었다. 그리고 이름이 'ebxmlrr test registry'인 Organization은 두 개의 TelephoneNumber가 하위 엘리먼트 값으로 반환되어 있다.

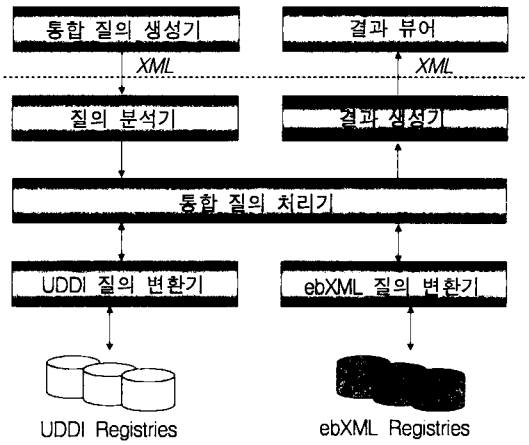
```
<EBRIResponse xmlns : xsi = http://www.w3.org/2001/
          XMLSchema-instance
xsi : noNamespaceSchemaLocation ="EBRIResponse.xsd">
  <Organizations>
    <Organization
      url = "http://registry.csis.hku.hk:8201/ebxmlrr/
            registry/soap"
      id = "388678e9-40df-4412-ac65-c42bbb265e7d"
      name = "ebxmlrr test registry"
      description = "ebxmlrr test registry">
      <TelephoneNumber areaCode = "781"
        countryCode = "1" number = "442-0704"
        type = "fax"/>
      <TelephoneNumber areaCode = "781"
        countryCode = "1" number = "442-0703"
        type = "office"/>
    </Organization>
    <Organization url = "http://uddi.ibm.com/ubr/
                  inquiryapi"
      id = "fdfdbba0-a7d3-11d5-a30a-002035229c64"
      name = "IBM WSTK Tutorial"
      description = "IBM WSTK Tutorial" />
  </Organizations>
</EBRIResponse>
```

(그림 20) XML 응답 구문 메시지의 예

5. 통합 질의 시스템 설계 및 구현

(그림 21)은 본 논문에서 제안한 통합 질의 시스템의 전체적인 구조를 보여주고 있다. 사용자로부터 질의에 필요한 정보를 입력받아 XML 질의 메시지를 생성하고, 질의 분석기에서는 그

질의를 분석한다. 통합 질의 처리기에서 분석된 질의를 바탕으로 질의를 처리하는데, 대부분의 질의 처리에 관련된 일을 통합 질의 처리기에서 관리를 한다. 통합 질의 처리기에서 최종적으로 반환된 결과를 결과 생성기에서 XML 메시지로 생성하고, 그 메시지를 결과 뷰어에 넘겨주면 결과 뷰어는 최종적으로 사용자에게 결과를 보여준다.



(그림 21) 통합 질의 시스템 구조

5.1 시스템 모듈

• 통합 질의 생성기

클라이언트로부터 질의 조건들을 입력 받아 XML 질의 메시지를 생성한다.

• 질의 분석기

통합 질의 생성기로부터 질의 메시지를 입력받아 질의문을 분석한다. 또한 질의 분석기는 XML 질의 스키마를 이용해 입력받은 XML 인스턴스의 유효성 검사를 수행한다.

• 통합 질의 처리기

통합 질의 처리기는 통합 질의 시스템의 핵심 부분이다. 통합 질의 처리기에서는 질의의 중첩 구조를 관리하고, 통합 연산자 및 반환 타입에 대한 관리를 위해 Aggregation 연산 처리의 일을 한다. 또한 통합 질의 처리기는 각각의 레지스트리에의 Connection을 관리한다.

• ebXML 질의 변환기

ebXML 질의 변환기는 EBRIQuerySimple Clause의 질의 내용을 ebXML 질의로 매핑시키고 ebXML 질의로 변환한다. 그리고 변환된 질의를 ebXML 레지스트리에 실행을 하고 그 결과를 통합 질의 처리기에 반환하다.

• UDDI 질의 변환기

UDDI 질의 변환기는 EBRIQuerySimple Clause의 질의 내용을 UDDI 질의로 매핑시키고 UDDI 질의로 변환한다. 그리고 변환된 질의를 UDDI 레지스트리에 실행을 하고 그 결과를 통합 질의 처리기에 반환한다.

• 결과 생성기

통합 질의 처리기가 최종적인 결과를 얻으면 그 결과를 결과 생성기에 넘겨준다. 결과 생성기는 통합 질의 처리기로부터 받은 결과를 분석하여 XML 응답 스키마에 유효한 응답 메시지를 생성한다. 그리고 그 생성된 XML 스키마를 결과 뷰어에 넘겨준다.

• 결과 뷰어

결과 뷰어는 결과 생성기로부터 넘겨받은 결과 메시지를 클라이언트에 보여주는 역할을 한다.

5.2 구현 환경

다음은 통합 질의 시스템의 구현 환경이다.

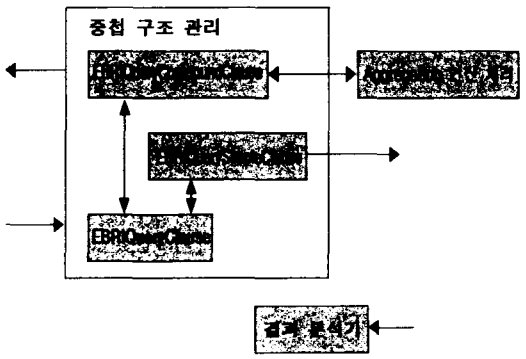
• 운영체제
  - Solaris 2.7

• 개발 언어
  - J2SDK 1.4.1

- 운용 환경
  - Jakarta Tomcat 4.0.4
  - JAXR 1.0

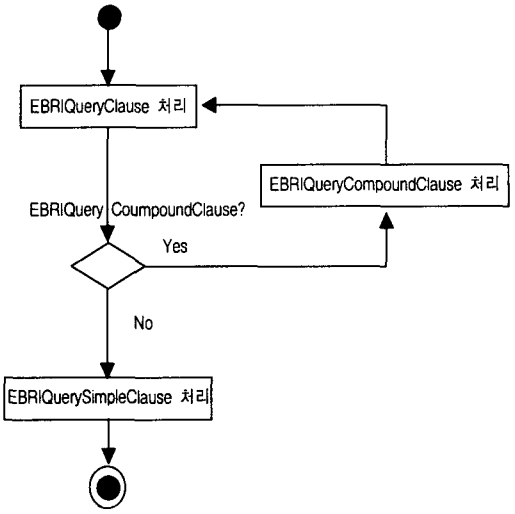
5.3 통합 질의 처리 알고리즘

(그림 22)는 통합 질의 처리기의 중첩 구조를 관리하기 위한 내부 구조의 모습을 보여주고 있다. EBRIQueryCompoundClause는 Aggregation 연산 처리를 통해 각각의 레지스트리의 반환 객체에 대한 통합을 관리한다. 그리고 EBRIQueryClause는 결과 분석기로부터 받은 결과를 질의에 이용할 수 있다. EBRIQueryClause는 EBRIQuerySimpleClause와 EBRIQueryCompoundClause와 관계를 맺고 중첩 구조를 관리한다.



(그림 22) 통합 질의 처리기의 중첩 구조 관리

(그림 23)은 통합 질의 처리기의 중첩 처리 알고리즘을 보여주고 있다. 질의 분석기에서 받은 질의 내용을 EBRIQueryClause가 받으면 그 내용을 처리한다. 처리한 결과가 EBRIQueryCompoundClause의 처리를 필요로 하면 EBRIQueryCompoundClause로 처리를 넘긴다. 여기서 다시 EBRIQueryCompoundClause는 처리를 위해 다시 EBRIQueryClause를 실행한다. 이러한 구조로 반복해서 중첩된 질의를 처리한다. 만약에 EBRIQueryClause가 EBRIQueryCompoundClause의 처리가 아니고 EBRIQuerySimpleClause의 처리를 요구하면 순환은 종료되고 레지스트리에 실제적인 질의를 하게 된다.



(그림 23) 통합 질의 처리기의 중첩 처리 알고리즘

6. 결 론

현재 국제적인 e-비즈니스 관련 프레임워크로 ebXML과 웹 서비스가 운용 중에 있다. 이러한 ebXML과 웹 서비스를 통하여 거래 당사자는 전자거래에 필요한 비즈니스 정보를 저장하고 검색할 수 있다. 현재까지는 비즈니스 정보를 찾고자 할 경우 ebXML과 웹 서비스를 개별적으로 이용하여 원하는 정보를 검색해야 한다. 본 논문에서는 이러한 ebXML과 웹 서비스를 이용하여 비즈니스 정보를 검색하고자 할 경우 비즈니스 정보를 담고 있는 레지스트리를 통합 접근하여 검색할 수 있는 시스템을 제안했다.

현재는 ebXML 레지스트리와 UDDI 레지스트리가 국제 표준으로 채택되어 ebXML과 웹 서비스의 레지스트리로 이용되고 있다. 본 논문은 ebXML 레지스트리와 UDDI 레지스트리의

데이타 모델을 포괄할 수 있는 공통 데이타 모델을 정의했다. 이렇게 함으로써 ebXML 데이타 모델이나 UDDI 데이타 모델을 알 필요가 없고 공통 데이타 모델만 알면 된다. 그리고 공통 데이타 모델을 기반으로 통합 질의어를 정의하여 ebXML 레지스트리와 UDDI 레지스트리를 통합 접근함으로써 한 번의 질의로 원하는 정보를 검색할 수 있게 했다.

본 논문에서 제안한 e-비즈니스 통합 질의 시스템은 크게 다음과 같은 네 가지의 장점을 가지고 있다.

첫째, ebXML 레지스트리와 UDDI 레지스트리에 대한 각각의 인터페이스를 알 필요가 없다. 왜냐하면 ebXML 레지스트리와 UDDI 레지스트리를 바탕으로 공통 데이타 모델을 정의하였기 때문에 공통 데이타 모델만 알면 두 레지스트리를 접근할 수 있게 된다. 그리고 현재까지 ebXML이나 웹 서비스를 이용할 경우 각각의 레지스트리에 의존적인 전용 도구를 이용하여 서비스를 이용해야만 했다. 그러나 공통 데이타 모델에 기반을 둔 통합 질의 시스템을 이용할 경우 전용 도구를 이용하지 않고도 레지스트리에서 제공하는 모든 서비스를 이용할 수 있기 때문에 전용 도구들을 구입할 필요가 없다. 따라서 비용절감의 효과를 가져올 수 있다.

둘째, 통합 질의 시스템은 이질적인 ebXML 레지스트리와 UDDI 레지스트리의 통합 검색을 지원한다. 즉 서비스 정보를 검색하기 위해 ebXML 레지스트리에 맞는 질의문을 생성하고 ebXML 레지스트리에 질의를 실행하여 결과를 얻어온다. 마찬가지로 UDDI 레지스트리에 맞는 질의를 생성하고 UDDI 레지스트리에 질의를 실행하여 그 결과를 가져와야 한다. 그러나 통합 질의 시스템을 이용할 경우 공통 데이타 모델에 기반을 둔 질의를 생성해서 질의를 실행하면 한번의 질의문으로 ebXML 레지스트리와 UDDI 레지스트리의 통합 검색을 제공한다.

셋째, 공용 UDDI 레지스트리에는 분산 질의를 통해 검색 속도의 향상을 가져온다. UDDI 버전 2.0에서는 레지스트리 간에 데이타 복제를 정의하고 있는 Replication 스펙이 있다. 현재는 IBM, Microsoft, SAP 그리고 NTT Com에서 공용 레지스트리를 운용하고 있으며, 하루에 한 번 이들 레지스트리 간에 복제가 이루어져서 이들 공용 레지스트리를 검색할 경우 모두 같은 정보를 얻게 된다. 이러한 특성을 이용해서 공용 UDDI 레지스트리에서 어떤 서비스 정보를 얻고자 할 경우 하나의 레지스트리에 질의를 하는 것보다 질의 조건을 나눠서 다른 레지스트리에 질의를 할 수 있다. 이러한 분산질의를 수행하면 어느 하나의 레지스트리에 치우치는 질의의 부담을 줄일 수 있고 검색 속도의 향상을 가져올 수 있다.

넷째, ebXML 레지스트리의 경우 통합 검색의 효과가 있다. ebXML은 버전 2.1에서 ebXML 레지스트리 간의 데이타 공유나 복제에 대한 내용이 정의 되어 있지 않다. 따라서 아직까지는 ebXML 레지스트리들에 저장된 내용들이 서로 다를 수밖에 없다. 통합 질의 시스템을 이용하면 ebXML 레지스트리 간의 상이한 내용들을 통합 검색할 수 있다.

e-비즈니스 레지스트리 통합 질의 시스템은 분산되어 있는 e-비즈니스 서비스를 이용함에 있어 많은 기여를 할 것으로 기대된다. 특히 하나의 인터페이스를 이용함으로써 여러 e-비즈니스 서비스를 이용할 수 있게 되었으며, 그로 인해 비용절감의 효과도 예상된다. 또한 분산 질의를 통한 검색 속도의 향상도 가져올 수 있다. 따라서 e-비즈니스 레지스트리 통합 질의 시스템은 현재까지의 e-비즈니스 서비스가 가지고 있던 한계들을 극복하고자 했다.

참 고 문 헌

[1] 이석호. 데이타베이스 시스템.

[2] 전희영, 김계용, 유정연, 이규철. (2001). ebXML 등록기/저장소에서의 객체 질의 관리 시스템 설계 및 구현. 2001 가을 학술발표논문집(I). 37-39

[3] 김계용, 황윤영, 유정연, 이규철. (2002). e-비즈니스 레지스트리 통합 질의 시스템 설계. 2002 가을 학술발표논문집(I). 187-189.

[4] ebXML.org. electronic business eXtensible Markup Language. available at http://www.ebxml.org.

[5] w3c.org. Web Service. available at http://www.w3.org/2002/ws.

[6] ebXML.org. ebXML Registry. available at http:/www.oasis-open.org/committees/regrep/.

[7] uddi.org. Universal Description, Discovery, and Integration. available at http://uddi.org.

[8] uddi.org, UDDI Replication Specification. available at http://uddi.org/pubs/Replication-V2.03-Published-20020719.pdf.

[9] IBM. UDDI Search Markup Language (USML). available at http://www.alphaworks.ibm.com/tech/be4ws/.

[10] w3c.org. Extensible Markup Language. available at http://www.w3c.org.

[11] IBM. Web Services Toolkit. available at http://alphaworks.ibm.com/tech/webservicestoolkit.

[12] Microsoft. Windows .NET Server. available at http://www.microsoft.com/windows. netserver/developers/default.mspx.

[13] SUN. Java API for XML Registries. available at http://jcp.org/jsp/detail/93.jsp.

■ 저자소개

김 계 용

Kyeyong Kim is Feb. 2001 Bachelor of Computer Engineering in Chungnam National University. Feb. 2003 Master of Computer Engineering in Chungnam National University. Domain of Interest : e-Business, ebXML Repository, UDDI, Integration of e-business repository.

이 규 철

Kyuchul Lee is Feb. 1984 Bachelor of Computer Engineering in Seoul National University. Feb. 1986 Master of Computer Engineering in Seoul National University. Feb. 1990 Doctor of Computer Engineering in Seoul National University. Feb. 2000~Current : Chairperson of Korea ebXML Speciality Committee. Apr. 2000~Current : Professor of Chungnam National University. Feb. 2001~Current : Vice Chairperson of Electornic Commerce Technology Committee in Integrated Forum on Electronic Commerce. Domain of Interest : Database, Multimedia, XML, Electronic Commerce, Integration of Information.

◈ 이 논문은 2003년 1월 11일 접수하여 1차 수정을 거쳐 2003년 3월 8일 게재확정되었습니다.