ebXML 레지스트리 v2.1 서버 개발

A Development of the ebXML Registry v2.1 Server

조강녕(Kang-Ning Zhao)**, 이규철(Kyu-Chul Lee)***

초 록

ebXML은 UN/CEFACT와 OASIS가 주축으로 제정한 전자상거래 국제 표준으로 모든 거래 당사자들에서 상호 운용적이며 안전하고 일관성 있는 방법으로 XML 기반의 인프라(infrastructure)를 제공하여 글로벌 e-비즈니스가 가능하도록 하는 것을 목적으로 한다.

ebXML 레지스트리는 거래대상자들의 비즈니스 정보들을 저장하는 저장소를 가지며, 클라이언트가 이 저장된 정보에 접근할 수 있도록 다양한 레지스트리 서비스를 제공한다.

이러한 레지스트리 서비스들은 ebXML 레지스트리 정보 모델(RIM)에 정의된 구조를 따르기 때문에, 하부 데이터베이스 시스템에 따라 서비스들을 처리하는 방법이 달라진다. 본 논문에서는 관계형 데이터베이스를 기반으로 ebXML 레지스트리 서버 구현 방법을 제안한다.

ABSTRACT

ebXML is a global electronic business standard that is sponsored by UN/CEFACT and OASIS. ebXML is an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties. ebXML Registry provides a stable store where the business information submitted by partners is made persistent and a set of registry services that enable clients of the Registry to access them.

These services are tied to the structures defined in ebXML Registry Information Model(ebRIM), so we need a way to process them according to underlying database systems. In this paper, we present the result of developing the ebXML Registry Server based on Relational Database Systems.

키워드 : ebXML, 전자상거래, XML, 메타데이타 관리

ebXML, Electronic, Commerce, XML, Metadata Management

본 논문은 충남대학교 소프트웨어연구센터와 BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음.

** LG CNS China Inc.

***충남대학교 컴퓨터공학과 교수

1. 서 론

최근 e-비즈니스는 B2C에서 B2B로 발전하고 있으며 이에 따라 많은 회사와 단체들은 자신들끼리 비즈니스 정보를 효율적으로 교환, 관리하기 위해서 여러 가지 e-비즈니스 표준들을 개발하고 있다. ebXML(electronic business eXtensible Markup Language)은 "Creating A Single Global Electronic Market" 이라는 기치 아래 그 동안 국제 EDI 표준을 추진해 왔던 UN/CEFACT와 OASIS가 주축이 되어, XML을 이용한 인터넷 기반의 e-비즈니스가 가능하도록 제정한 표준이다. ebXML은 1999년 11월부터 만들어지기 시작하여 그 후 18개월의 짧은 기간 안에 만들어졌지만, 기존의 다른 전자 상거래 표준과는 달리 인터넷 환경에 맞게 개방되고 유연한 거래 프레임워크를 정의하고 있으며, 또한 기업 내부의 비즈니스 프로세스를 모델링 할 수 있는 방법론을 제공하는 특징들을 가지고 있다.

특히, ebXML의 모든 메타 데이터는 레지스트리 서버에 저장되며 레지스트리 서버는 클라이언트들이 이 레지스트리 서버에 저장된 정보를 공유할 수 있도록 여러 가지 서비스를 제공하기 때문에 레지스트리 서버는 ebXML에서 가장 중요한 역할을 한다. 최근에는 기존의 ebXML 레지스트리 버전 1.0에서 부족했던 웹 서비스 기능을 보완하여 버전 2.0이 나왔으며, 2002년 6월에는 버전 2.0에서의 에러들을 수정하여 버전 2.1을 발표하였다. 버전 2.0부터는 스키마가 DTD에서 XML Schema로 바뀌었으며, 웹 서비스의 기능을 지원하기 위해서 서비스와 관련된 모델들을 추가하고, 거래 상대자들 사이에 메시지를 교환할 때 ebXML MS(Messaging Service)만을 이용하는 것이 아니라 SOAP(Simple Object Access Protocol)을 이용한 방법도 지원하고 있다.

ebXML 레지스트리 서버에서는 크게 두 가지 서비스를 제공한다. 하나는 레지스트리에 객체를 저장 또는 삭제하는 연산과 같이 객체들의 생명주기를 관리하기 위한 Life Cycle Management 서비스와 레지스트리에 저장된 정보들에 대해서 다양한 질의 방법을 제공하는 Query Management 서비스가 존재한다. 또한 Query management 서비스는 RIM(Registry Information Model)에 기반하여 간단한 XML 형태의 질의를 제공하는 Filter Query와 복잡한 질의를 제공하는 SQL Query로 나뉘어진다.

본 논문에서는 이러한 ebXML 레지스트리에서 핵심이 되는 레지스트리 서버를 설계하고 구현한 결과를 제시한다. 이를 위해 레지스트리 정보 모델에 따르는 Life Cycle Management 서비스와 Query Management 서비스 처리 메커니즘을 제안한다. 특히 본 연구에서 구현된 시스템[6]은 관계형 데이터베이스 위에서 동작하기 때문에 XML 형태의 Filter Query를 실행하기 위해서 SQL Query로 변환하는 메커니즘이 필요하며 본 논문에서 이 알고리즘을 설계하고 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 현재까지 다른 회사에서 ebXML 레지스트리 버전 2.x에 대한 구현 상황을 알아본다. 3장에서 레지스트리의 각 서비스를 설계 및 구현하는 방법을 소개하고 4장에서 본 연구가 다른

관련 연구와의 비교 상황을 설명하며 5장에서 결론을 맺는다.

현재까지 국내외에서 구현 발표된 ebXML Registry 버전 2x 서버는 다음과 같은 것들이 있다.

2. 관련 연구

2.1 OASIS ebXML Registry Reference Implementation Project (ebxmlrr)[7]

2002년 4월에 Sun사와 홍콩대학교가 같이 개발하여 발표한 OASIS ebXML Registry Reference Implementation Project(ebxmlrr)이다. 기반 DBMS는 IBM Software사의 Cloudscape[8]을 사용하였고, Java 기술을 기반으로 구현되었으며, 구현을 목표로 한 표준은 RIM v2.0과 RS v2.0이다. 지난 7월에 ebXML 레지스트리 2.1 버전의 발표에 따라서 ebxmlrr은 2.1 버전으로 업데이트 했다.

ebxmlrr 레지스트리 서버는 RIM을 기반으로 Life Cycle Management와 Query Management 두 가지 서비스를 구현하였으며 메시지의 전송 프로토콜은 SOAP을 사용하였다.

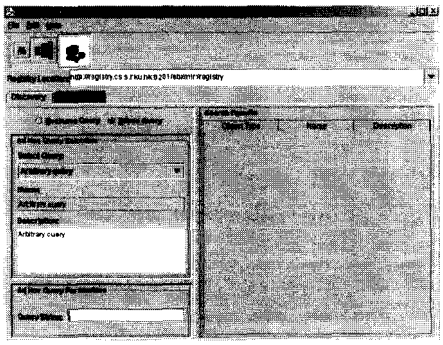
<그림 1>은 ebxmlrr 레지스트리 브라우저이다. ebxmlrr의 레지스트리 서버는 표준에 따라서 Life Cycle Management 서비스와 Query Management 서비스를 전부 지원한다.

2.2 GoXML Registry(XMLGlobal co. Ltd.)[9]

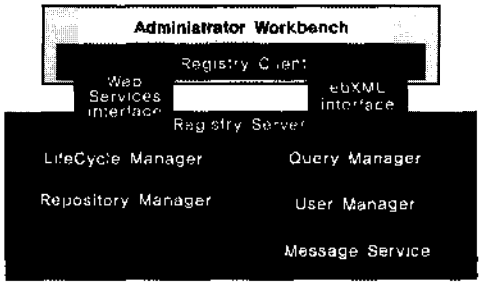
XMLGlobal에서 개발한 ebXML 레지스트리이다. 기반 DBMS는 MySQL 3.23을 사용하였고, Java로 구현되었으며, 구현을 목표로 한 표준은 RIM v2.0과 RS v2.0이다.

<그림 2>는 GoXML Registry의 시스템 구조이다. 레지스트리 서버의 핵심 컴포넌트는 LifeCycle Manager와 Query Manager, Repository Manager, User Manager 그리고 Message Service로 이루어져 있다.

- LifeCycle Manager: 저장 또는 삭제 등 연산들을 컨트롤한다.
- Query Manager: 레지스트리에 저장된



<그림 1> ebxmlrr 레지스트리 브라우저



<그림 2> GoXML Registry의 시스템 구조

정보들에 대한 검색을 가능하게 한다.

- Repository Manager: 레지스트리 객체들의 물리적인 저장소를 다루는 역할을 한다.
- User Manager: 어떤 사용자가 이 레지스트리를 사용할 수 있는지, 그리고 어떤 특권을 가질 수 있는지를 관리한다.
- Message Service: ebXML 메시지의 전송을 처리한다.

그러나 개발된 서비스는 Life Cycle Management 서비스와 Query Management 서비스 모든 표준을 전부 지원하지 않고 일부 서비스만 구현하였다. 예를 들어서 Life Cycle Management 서비스 중의 AddSlots와 RemoveSlots 서비스를 지원하지 않고, Query Management 서비스 중의 복잡한 질의를 가능하게 하는 SQL Query를 지원하지 않는다.

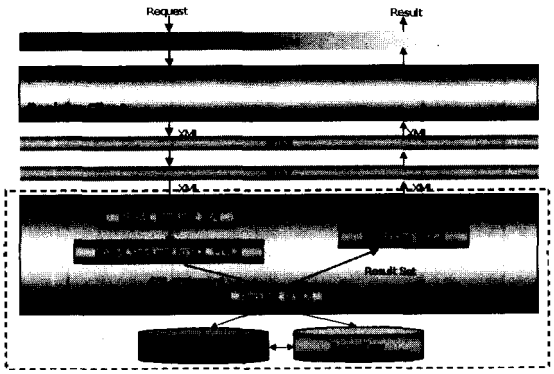
2.3 기타 ebXML 레지스트리

위에 설명한 두 개의 레지스트리뿐만 아니라 ebXMLsoft[10] 회사에서 ebXMLsoft Registry 2.1을 구현했고 국내에서는 KTNET(Korea Trade Network)[11]에서 KTNET ebXML Registry v2.0을 개발하였으며 POSDATA사에서 Biz Binder RR (Registry/Repository)[12]를 구현했다. 하지만 이 회사에서 구현한 레지스트리들은 구현 방법과 내용에 대해서 자세하게 알리고 있지 않기 때문에 본 연구에서 직접적인 비교는 힘들다.

3. 시스템 설계 및 구현

3.1 전체 시스템 구조

본 시스템의 구현 환경은 운영체제로는 Sun Solaris 2.7을 사용하였고 개발 언어는 JDK 1.3.1을 사용했으며 파서는 JAXP1.1를 사용했다. 기반 DBMS는 관계형 데이터베이스인 Oracle 8.1.7을 사용했는데 JDBC를 이용해서 개발하였으므로, 다른 RDBMS위에서도 동작할 수 있다. 〈그림 3〉은 ebXML 레지스트리의 전체 구조로서 본 연구에서는 점



〈그림 3〉 ebXML 레지스트리 전체 시스템 구조

선 안의 Registry Server부분을 설계, 구현하였다.

사용자가 레지스트리 클라이언트를 통해서 레지스트리 서버로 서비스를 요청하면 서버는 이 메시지를 파싱한 후 SQL문으로 변환하게 된다. 변환된 SQL문이 레지스트리에서 실행된 후 검색 실행 결과는 다시 XML 형태로 클라이언트로 보내지게 된다. 다음은 레지스트리 서버의 각각의 모듈에 대한 설명이다.

▣ ebXML Request 파서
클라이언트에서 요구한 XML 형태의 서비스를 파싱하는 모듈

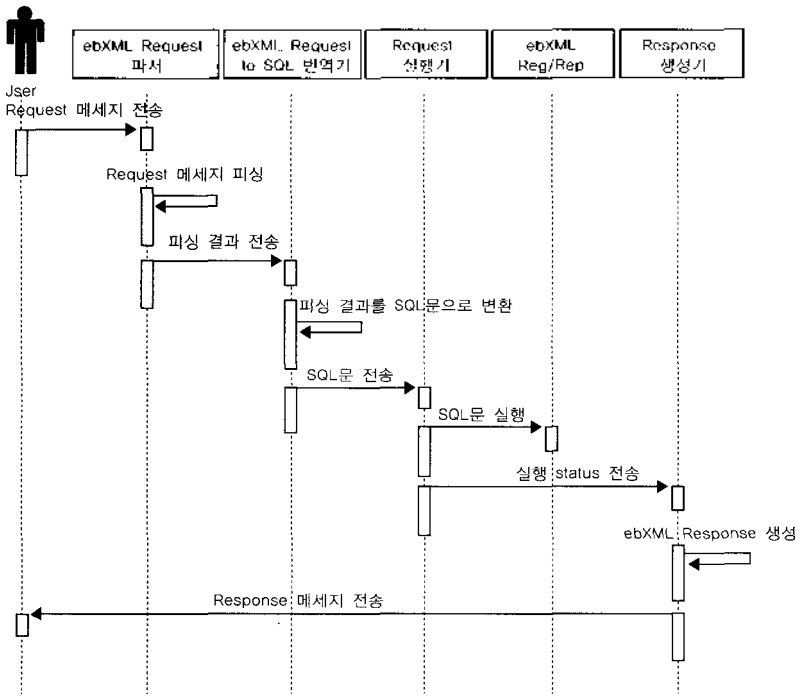
▣ ebXML Request to SQL 번역기
ebXML Request Parser에 의해 파싱된 결과를 사용하여 각 서비스에 따라 레지스트리가 인식 가능하도록 SQL문으로 변환하는 모듈

▣ Request 실행기
레지스트리에 SQL문을 보내 실행하여 결과를 가져오는 모듈

▣ Response 생성기
서비스 결과를 가지고 ebXML 서비스 응답 메시지를 생성하여 보내는 모듈

3.2 Life Cycle Management 서비스

이 서비스는 레지스트리에 객체를 저장 또는 삭제하는 연산들과 같이 객체들의 생명주기를 관리하는 역할을 하는 7 가지 서비스를



〈그림 4〉 Life Cycle Management 서비스 순차 다이어그램

가진다. 이 중에서 SubmitObjects 서비스는 레지스트리에 정보를 저장하고 UpdateObjects 서비스는 레지스트리에 저장된 정보들을 업데이트하며 RemoveObjects 서비스는 레지스트리에 저장된 정보들을 삭제는 역할을 한다. ApproveObjects 서비스와 DeprecateObjects 서비스는 레지스트리에 저장된 정보들을 승인하거나 deprecate한다. AddSlots 서비스는 객체에 표준에서 정의되지 않는 정보들을 추가하게 하고 RemoveSlots 서비스는 추가한 Slot을 삭제한다.

〈그림 4〉는 클라이언트가 Life Cycle Management 서비스를 요청할 때의 일련의 처리 과정을 순차 다이어그램으로 나타낸 것이다

사용자는 클라이언트 인터페이스에 따라 Request Message를 생성한 후 서버로 전송한다. 서버의 ebXML Request 파서는 사용자가 전송한 XML 인스턴스를 ebXML Registry/Repository에 저장하기 위해 데이터를 추출하는 과정을 수행한다. 파싱 과정을 거쳐 추출된 데이터를 가지고 ebXML Request to SQL 번역기는 요청 서비스에 따라 해당 SQL문으로 변환한다. Request 실행기는 SQL문을 실행한 결과를 ebXML Reg/Rep에 저장한다.

이 과정에서 발생 가능한 모든 예외 상황들이 처리된다. 모든 수행이 완료되면, 서버의 Registry Response 클래스는 request가 성공적으로 완료되었는지의 여부를 나타내는 RegistryResponse 메시지를 클라이언트로 전송한다. 성공적으로 수행되었을 경우에 Response 메시지의 status는 Success가 되며, 실패 시에는 Failure가 된다.

Life Cycle Management 서비스가 포함한 7가지 서비스는 다음과 같은 SQL문으로 변환해야 하며 SQL문에서 밑줄 있는 단어들은 프로그램 인자들이다.

- SubmitObjects Service
  INSERT INTO table [(column [, column...])] VALUES (value,[, value...]);
- UpdateObjects Service
  UPDATE table SET column = value [, column = value] [WHERE condition];
- RemoveObjects Service
  DELETE FROM table WHERE condition;
- ApproveObjects Service
  UPDATE table SET status = 'Approved' [WHERE condition];
- DeprecateObjects Service
  UPDATE table SET status = 'Deprecated' [WHERE condition];
- AddSlots Service
  INSERT INTO Slot [(column [, column...]) VALUES (value, [, value...]);
- RemoveSlots Service
  DELECT FROM Slot WHERE condition;

〈그림 5〉는 위의 변환 규칙을 이용해서 SubmitObjects를 SQL로 변환한 예이다. SubmitObjectsRequest 메시지의 a 부분은 id가 "CNU-DBlab"인 조직의 정보들을 보여주고 b 부분은 id가 "KNZhao"인 사용자의 정보들을 보여준다. 조직과 사용자의 정보들을 관계형 데이터베이스에 저장하기 위해서 각

```
- <SubmitObjectsRequest>
  - <rim:LeafRegistryObjectList>
    - <rim:Organization id="CNU-DBLab" primaryContact="KNZhao">
      - <rim:Name>
          <rim:LocalizedString lang="en-us" value="ChungNam Univ. Database Lab." />
        </rim:Name>
      - <rim:Description>
          <rim:LocalizedString lang="en-us" value="ChungNam Univ. Database Lab." />
        </rim:Description>
        <rim:Address city="DaeJeon" postalCode="305-764" country="Korea" />
        <rim:TelephoneNumber number="042-821-7721" />
      </rim:Organization>
    - <rim:User id="KNZhao" organization="CNU-DBLab">
      - <rim:Name>
          <rim:LocalizedString lang="en-us" value="Kang-Ning Zhao" />
        </rim:Name>
      - <rim:Description>
          <rim:LocalizedString lang="en-us" value="Kang-Ning Zhao" />
        </rim:Description>
        <rim:EmailAddress address="knzhao@ce.cnu.ac.kr" type="MadeUp" />
        <rim:PersonName firstName="Kang_Ning" lastName="Zhao" />
        <rim:Address city="Daejeon" postalCode="305-764" country="Korea" />
        <rim:TelephoneNumber number="042-822-0523" />
      </rim:User>
  </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

a

b

```
INSERT INTO Organization (id, primaryContact) VALUES('CNU-DBlab','KNZhao');
INSERT INTO Name (lang, value, parent)
            VALUES('en-us','ChungNam Univ. Database Lab.','CNU-DBlab');
INSERT INTO Description (lang, value, parent)
            VALUES ('en-us', 'ChungNam Univ. Database Lab.','CNU-DBlab');
INSERT INTO PostalAddress (city, postalCode, country, parent)
            VALUES('Daejeon','305-764','Korea','CNU-DBlab');
INSERT INTO TelephoneNumber (number,parent)
            VALUES('042-821-7721','CNU_DBlab');
```

A

```
INSERT INTO User (id, Organization,email, PersonName_firstname, PersonName_lastname)
            VALUES ('KNZhao','CNU-Dblab','knzhao@ce.cnu.ac.kr','Kang-Ning', 'Zhao');
INSERT INTO Name (lang, value, parent) VALUES ('en-us','Kang-Ning Zhao','KNZhao');
INSERT INTO Description (lang, value, parent)
            VALUES ('en-us', 'Kang-Ning Zhao','KNZhao');
INSERT INTO EmailAddress (address, type, parent)
            VALUES('knzhao@ce.cnu.ac.kr','MadeUp','KNZhao');
INSERT INTO PostalAddress (city, postalCode, country, parent)
            VALUES('Daejeon','305-764','Korea','KNZhao');
INSERT INTO TelephoneNumber (number,parent) VALUES('042-822-0523','KNZhao');
```

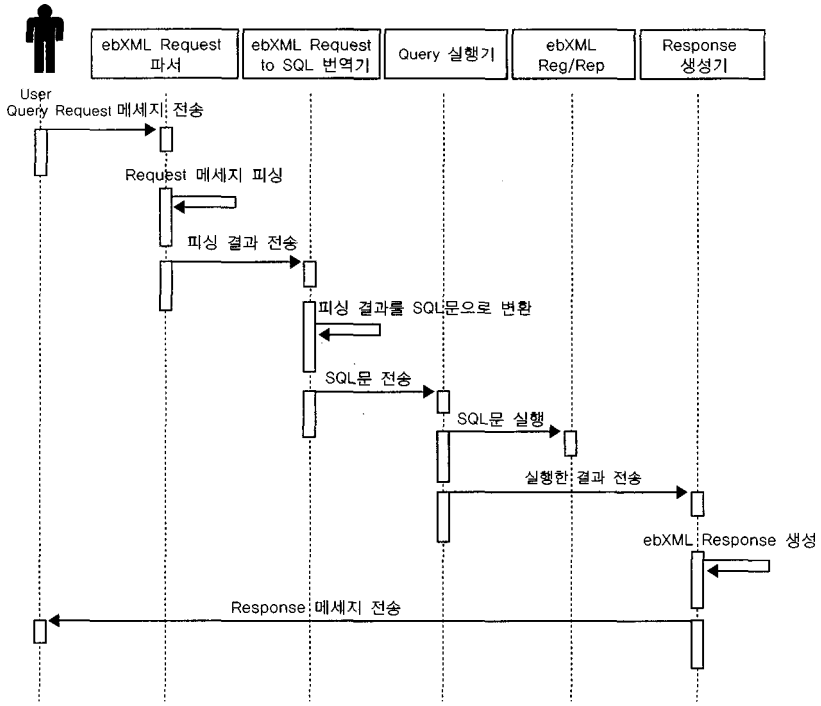
B

〈그림 5〉 SubmitObjects Request 메시지의 SQL 변환 예

각 A 박스와 B박스에 있는 SQL문으로 변환 해야 한다.

3.3 Query Management 서비스

이 서비스는 레지스트리에 저장된 정보들에 대해서 다양한 질의 방법을 제공한다. 주



〈그림 6〉 Query Management 서비스 순차 다이어그램

로 3 가지의 질의 서비스를 제공하는데 Filter Query 서비스와 SQL Query 서비스, 그리고 Content Retrieval 서비스를 포함한다.

3.3.1 Query Management 서비스 구현

〈그림 6〉은 질의가 처리되는 일련의 과정을 순차 다이어그램으로 나타낸 것이다.

서버는 클라이언트에서 보내 온 Request 메시지를 받게 되면 ebXML Request 파서는 Request 메시지를 파싱해서 SQL로 변환하기 위해서 ebXML Query to SQL 번역기에 보낸다. ebXML Query to SQL 번역기는 전달받는 메시지를 바탕으로 각 질의 서비스에 따라 SQL로 변환한다. 변환된 SQL는 Query 실행기에 전달되어 질의를 처리한 후 Result Set을 Response 생성기에 보낸다. 마지막에 Response 생성기는 받은 Result Set을 가지고 Registry Response Message를 생성해서 클라이언트 쪽으로 보낸다.

3.3.2 Filter Query 서비스

Filter Query 서비스는 ebXML 레지스트리 서비스에서 필수적으로 제공해야 하는 질의 서비스로서 레지스트리 정보 모델에서 정의된 구조 정보를 기반으로 XML 구문을 이용해서 메타데이터를 검색할 수 있는 간단한 질의 서비스이다.

```
- <AdhocQueryRequest>
    <ResponseOption />
  - <FilterQuery>
    - <RegistryEntryQuery>
      - <NameBranch>
        - <LocalizedStringFilter>
          - <Clause>
            - <SimpleClause leftArgument="Value">
                <StringClause stringPredicate="Equal">XYZ</StringClause>
              </SimpleClause>
            </Clause>
          </LlocalizedStringFilter>
        </NameBranch>
      - <RegistryEntryFilter>
        - <Clause>
          - <SimpleClause leftArgument="objectType">
              <StringClause stringPredicate="Equal">CPP</StringClause>
            </SimpleClause>
          </Clause>
        </RegistryEntryFilter>
      </RegistryEntryQuery>
    </FilterQuery>
  </AdhocQueryRequest>
```

〈그림 7〉 Filter Query 예

이 질의 서비스는 RegistryObject, RegistryEntry, Association, AuditableEvent, Classification, ClassificationNode, ClassificationScheme, RegistryPackage, ExtrinsicObject, Organization, Service 의 11 개의 독립적인 클래스 질의로 이루어져 있다. 하나의 클래스 질의 내에서는 ebRIM에서 정의하는 클래스들이 바인딩되어 각 클래스에 대해 질의가 수행된다.

〈그림 7〉은 Filter Query의 예로써 레지스트리에 등록되어 있는 이름의 Value 값이 "XYZ"이면서 objectType이 "CPP"인 RegistryEntry들을 찾기 위한 질의다

Filter Query의 질의 형태는 XML 문서를 기반으로 제공되기 때문에 관계형 데이터베이스 상에 ebXML 레지스트리를 구현하고자 하면 XML 형태로 된 Filter Query를 SQL 형태로 변환하는 것이 필요하다. 본 논문에서는 Filter Query의 SQL Query 변환 알고리즘을 구현하였으며 다음과 같은 알고리즘을 사용한다.

(ㄱ) 기본 알고리즘

하나의 Filter Query는 하나의 클래스로 구성되어 있고 하나의 클래스는 여러 개의 Filter로 구성되어 있다는 특징을 고려해서 하나의 Filter Query를 SQL문으로 변환하려면 다음과 같은 기본 알고리즘이 필요하다.

- 하나의 클래스마다 하나의 SQL문 생성
- 하나의 서브 클래스마다 하나의 SQL문 생성
- 하나의 Filter에 하나의 subquery 생성
- Filter와 Filter 사이는 "AND"로 연결

(ㄴ) 세부 알고리즘

① 질의 대상 클래스의 이름은 전체 질의테이블의 이름으로 변환

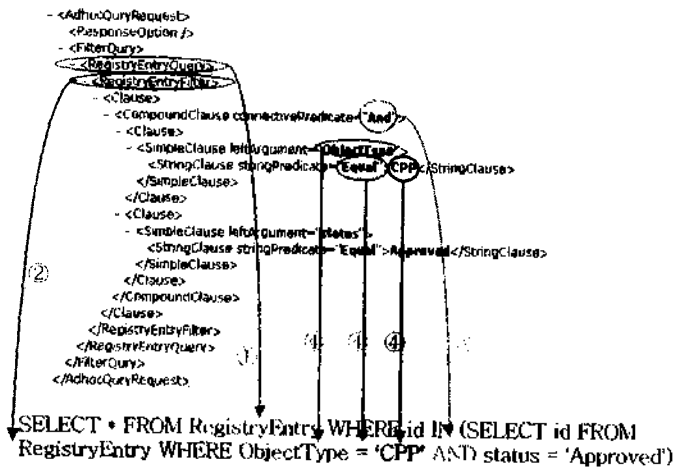
② 각 Filter와 서브 클래스의 이름이 subquery의 테이블의 이름으로 변환

③ CompoundClause의 connectivePredicate

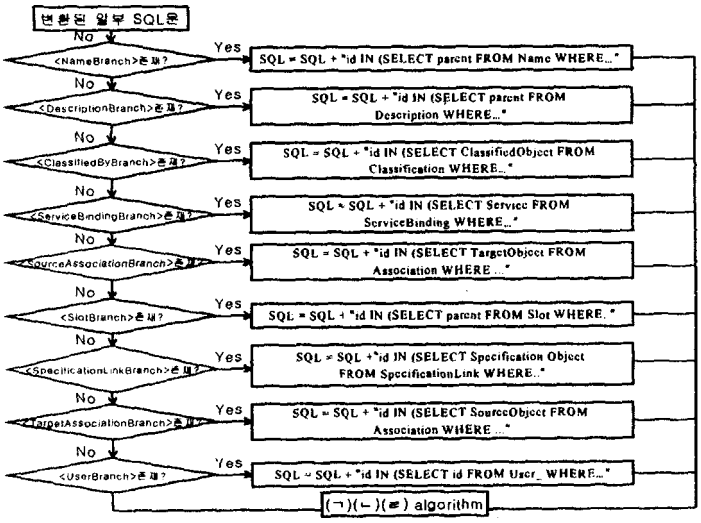
attribute의 값을 연결 연산자(AND, OR)로 변환

④ SimpleClause의 leftArgument attribute의 값을 subquery의 WHERE clause의 Column name으로 변환한다. 그리고 BooleanClause, RationalClause, StringClause의 각각의 attribute들의 값을 SQL의 연산자로 변환하고, 그 사이의 값을 WHERE 절에 적용

〈그림 8〉은 세부 알고리즘의 적용 예로 "objectType"이 "CPP" 이면서 "status"가 "Approved"인 Registry Entry을 찾기 위한 해서 세부 알고리즘을 적용해서 Filter Query을 SQL로 변환 예이다.



〈그림 8〉 세부 알고리즘 적용 예



〈그림 9〉 Branch 변환 알고리즘

(ㄷ) 각 Branch의 변환 알고리즘

Filter Query중의 Branch는 한 클래스의 서브 클래스들을 식별한다. 존재하는 각 Branch에 대한 변환 알고리즘은 〈그림 9〉와 같다.

〈그림 10〉은 〈NameBranch〉에 대한 변환 적용 예이다. 이름의 값이 "XYZ"인 Organization을 찾기 위한 질의이다.

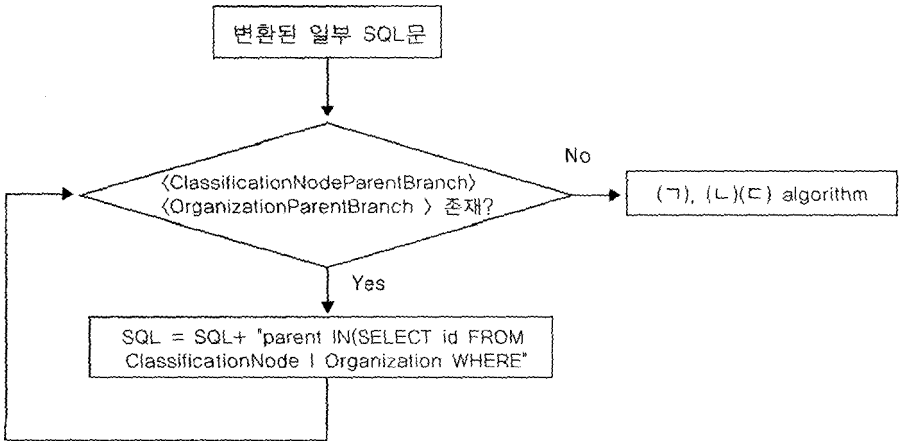
(ㄹ) Recursive 질의를 위한 알고리즘

Filter Query중에서 recursive한 질의가 있다. ClassificationNodeQuery 중에서 〈ClassificationNodeParentBranch〉 엘리먼트가 나오거나 OrganizationQuery 중에서 〈OrganizationParentBranch 〉 엘리먼트가 나오는 경우에 다음과 같은 알고리즘이 필요하다.

```
- <AdhocQueryRequest>
    <ResponseOption />
  - <FilterQuery>
    - <OrganizationQuery>
      - <NameBranch>
        - <LocalizedStringFilter>
          - <Clause>
            - <SimpleClause leftArgument="Value">
                <StringClause stringPredicate="Equal">XYZ</StringClause>
              </SimpleClause>
            </Clause>
          </LlocalizedStringFilter>
        </NameBranch>
      </OrganizationQuery>
    </FilterQuery>
  </AdhocQueryRequest>
```

SELECT * FROM Organization WHERE id IN (SELECT parent FROM Name WHERE Value = 'XYZ')

〈그림 10〉 Branch 변환 알고리즘의 적용 예

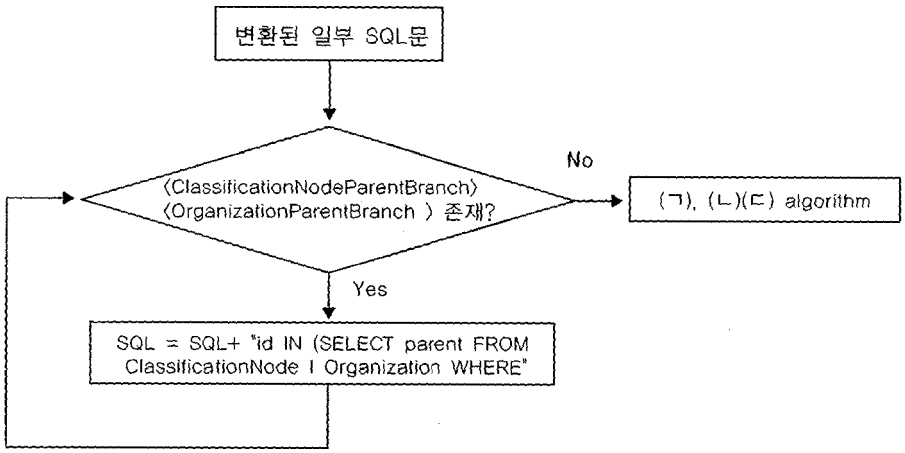


〈그림 11〉 Recursive 질의를 위한 알고리즘

또한 ClassificationNodeQuery 중에서 〈ClassificationNodeChildrenBranch〉 엘리먼트가 나오거나 OrganizationQuery 중에서 〈OrganizationChildrenBranch〉 엘리먼트가 나오는 경우에 다음과 같은 알고리즘이 필요하다.

〈그림 13〉은 Recursive 질의를 위한 알고리즘 적용 예이다. Parent 이름의 값이 "ABC"인 Organization을 찾기 위한 질의이다.

하나의 FilterQuery를 SQL로 변환하기 위해서 위에서 제안한 (ㄱ) 기본 알고리즘과 (ㄴ) 세부 알고리즘, (ㄷ) Branch를 위한 알고리즘, 그리고 (ㄹ) Recursive 질의를 위한 알고리즘을 전부 적용해야 되고 〈그림 14〉와 같은 다이어그램으로 표시한다.

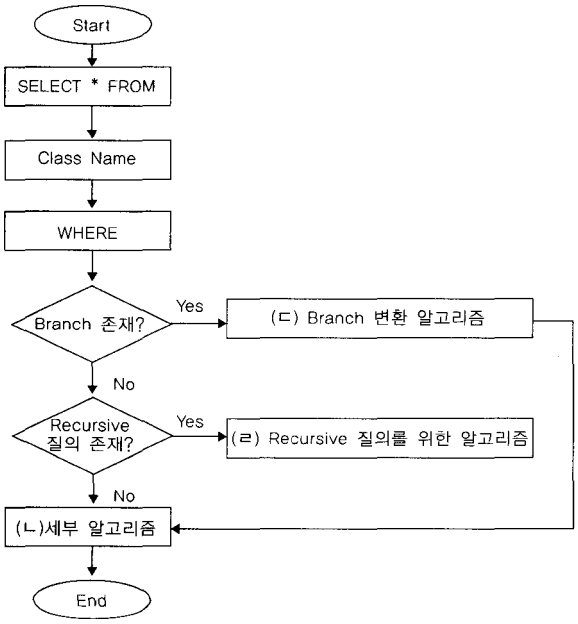


〈그림 12〉 Recursive 질의를 위한 알고리즘2

```
- <AdhocQueryRequest>
    <ResponseOption />
  - <FilterQuery>
    - <OrganizationQuery>
      <OrganizationParentBranch>
        - <NameBranch>
          - <LocalizedStringFilter>
            - <Clause>
              - <SimpleClause leftArgument="value">
                  <StringClause stringPredicate="Equal">ABC</StringClause>
                </SimpleClause>
              </Clause>
            </LocalizedStringFilter>
          </NameBranch>
        </OrganizationParentBranch>
      </OrganizationQuery>
    </FilterQuery>
  </AdhocQueryRequest>
```

SELECT * FROM Organization WHERE parent IN
(SELECT id FROM Organization WHERE id IN
(SELECT parent from Name WHERE value='ABC'))

〈그림 13〉 Recursive 질의를 위한 알고리즘 적용 예



〈그림 14〉 전체 알고리즘 다이어그램

〈표 1〉 Filter Query의 질의 predicate를 SQL연산자로 바꾼 테이블

Predicate	FilterQuery	SQL operator
Connective	AND	AND
	OR	OR
Boolean	True	True
	False	False
Logical	LE	〈=
	LT	〈
	GE	〉=
	GT	〉
	EQ	=
	NE	〈 〉
String	Contains	LIKE
	-Contains	NOT LIKE
	startswith〈Str〉	LIKE 'Str%'
	-startswith〈Str〉	NOT LIKE 'Str%'
	equal	=
	-equal	〈 〉
	endswith〈Str〉	LIKE '%Str'
	-endswith〈Str〉	NOT LIKE '%Str'

```
- <AdhocQueryRequest>
    <ResponseOption />
    <SQLQuery>SELECT eo.id FROM ExtrinsicObject eo, Name nm, WHERE nm.value LIKE %DBLab% AND
     eo.id = nm.parent AND eo.majorVersion > =1</SQLQuery>
  </AdhocQueryRequest>
```

〈그림 15〉 SQL Query 예문

```
- <AdhocQueryRequest>
    <ResponseOption />
    <SQLQuery> SELECT eo.id FROM ExtrinsicObject eo, Name nm, WHERE nm.value LIKE %DBLab% AND
    eo.id = nm.parent AND eo.majorVersion > =1 </SQLQuery>
  </AdhocQueryRequest>
```

```
SELECT eo.id FROM ExtrinsicObject eo, Name nm, WHERE nm.value LIKE '%DBLab%'
AND eo.id = nm.parent AND eo.majorVersion > =1
```

〈그림 16〉 SQL Query Request 메시지의 SQL 변환 예

◈ 연산자 매핑 테이블

Filter Query의 질의 predicate들을 SQL 연산자로 바꾸기 위해서 다음 〈표 1〉에서 정의한 테이블을 이용한다.

3.3.3 SQL Query 서비스

SQL Query 서비스는 ebRIM에서 제공하는 관계 스키마를 기반으로 SQL을 사용하여 복잡한 질의가 가능하도록 하는 선택적인 서비스이다. 각 ebRIM에 정의되어 있는 인터페이스와 클래스 이름, 애트리뷰트는 정해진 표준에서 정의하는 SQL Query를 위한 관계 스키마로 바인딩 되어야 한다. SQL Query는 여러 개의 서브 질의들로 이루어질 수 있으며, 각 서브 질의 내 SELECT 절의 컬럼 리스트는 하나이어야 한다.

〈그림 15〉는 "이름의 Value가 'DBLab'을 포함하고 majorVersion이 1보다 같거나 큰 Extrinsic Object를 검색하라" 라는 질의 문에 위해 생성된 SQL Query의 예이다. SQL Query Request 메시지를 SQL로 변환하기 위해서는 〈SQLQuery〉 엘리먼트의 값을 추출하면 된다. 〈그림 16〉은 SQL Query Request 메시지를 SQL로 변환한 예이다.

3.3.4 Content Retrieval 서비스

Content Retrieval 서비스는 레지스트리에 저장되어 있는 객체들을 가져올 수 있는 질의 서비스로 모든 질의의 검색 결과 내에서 특정 원하는 객체의 세부 정보를 검색한다. 이를 위해서 검색하고자 하는 id 값을 서버에 전달하면 서버에서 id에 해당하는 검색 결과를 돌려준다.

〈그림 17〉는 "id 값이 각각 'bf7424ee-616e-4844-92a0-eeaf1c261fc1"와 "212c3a76-1290-aa3c-bcc8-1a359ele4fg8' 인 객체들을 가져 오라" 라는 질의 문에 의해 생성된 Content

```
- <GetContentRequest>
  - <ObjectRefList>
      <ObjectRef id="bf7424ee-616e-4844-92a0-eeaf1c261fc1" />
      <ObjectRef id="212c3a76-1290-aa3c-bcc8-1a359ele4fg8" />
  - </ObjectRefList>
- </GetContentRequest>
```

〈그림 17〉 Content Retrieval 예문

```
- <GetContentRequest>
  - <ObjectRefList>
      <ObjectRef id="bf7424ee-616e-4844-92a0-eeaf1c261fc1" />
      <ObjectRef id="212c3a76-1290-aa3c-bcc8-1a359ele4fg8" />
  - </ObjectRefList>
- </GetContentRequest>

SELECT blobdata FROM ObjectContent WHERE id = bf7424ee-+2+e-4844-
92a0-eeaf1c261fc1 OR id = '212c3a76-1290-aa3c-bcc8-1a359ele4fg8'
```

〈그림 18〉 Content Retrieval SQL 변환 예

Retrieval 질의의 한 예이다.

Content Retrieval 메시지를 SQL로 변환하려면 GetContentRequest 메시지를 파싱해서 검색할 id 정보를 추출한다. 그 다음에 추출한 id 정보를 가지고 데이터베이스 테이블에 적용해서 SQL문으로 변환한다. 이 중에서 id가 하나 이상인 경우 "OR" 연산을 적용한다. 〈그림 18〉은 Content Retrieval 질의의 SQL 변환 예를 보여 준다.

4. 관련 연구와의 비교 고찰

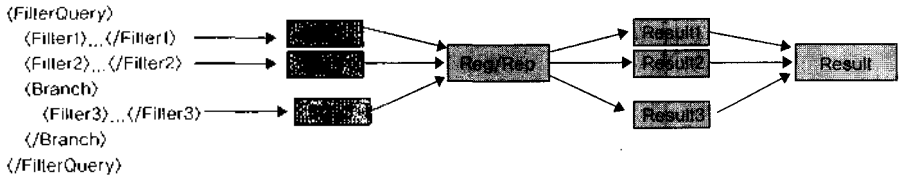
4.1 관련 연구와의 기능 비교

〈표 2〉는 본 연구에서 설계, 구현한 시스템과 관련 연구에서 소개하였던 Sun사와 홍콩 대학교가 같이 개발한 ebxmlrr, 그리고 XMLGlobal에서 개발한 GoXML Registry들을 비교 분석한 것이다. 비교 항목은 구현을 목표로 한 표준과 시스템의 구현 환경, 그리고 시스템이 지원하는 서비스를 하였다.

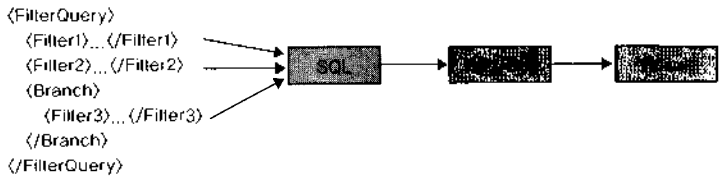
Sun 사와 홍콩 대학교가 같이 개발한 ebxmlrr는 RIM v2.1과 RS v2.1을 목표로 하여 구현하였으며, 기반 DBMS로는 Cloudscape을 사용하였고, 개발 언어는 Java이다. 지원하는 서비스는 Life Cycle Management 서비스와 Query Management 서비스를 표준에 따라 전부 지원한다. XMLGlobal에서 개발한 GoXML Registry는 RIM v2.0과 RS v2.0을 기준으로 구현하였으며, 기반 DBMS로는 MySQL3.23을 사용하였고 개발 언어는 Java이다. 이 시스템은 표준에 따라 서비스를 전부 지원하지 않고 부분 서비스만 지원한다. 이 중에서 Life Cycle Management 서비스 중

〈표 2〉 관련 연구와의 기능 비교

비교 항목			[illegible]	GoXML Registry	본 시스템
구현한 표준			RIM v 2.1, RS v 2.1	RIM v 2.0, RS v 2.0	RIM v 2.1, RS v 2.1
구현환경		DBMS	Cloudscape (RDBMS)	MySQL 3.23 (RDBMS)	Oracle8.17 (RDBMS)
		개발언어	Java	Java	Java
지원서비스	LM	SubmitObject	지원	지원	지원
		UpdateObject	지원	지원	지원
		RemoveObject	지원	지원	지원
		ApproveObject	지원	지원	지원
		DeprecateObject	지원	지원	지원
		AddSlots	지원	지원하지 않음	지원
		RemoveSlots	지원	지원하지 않음	지원
	QM	Filter Query	지원	지원	지원
		SQL Query	지원	지원하지 않음	지원
		Content Retrieval	지원	지원	지원



〈그림 19〉 ebxml/rr에서 Filter Query에 대한 구현방식



〈그림 20〉 본 시스템에서 Filter Query에 대한 구현 방식

의 AddSlots 서비스와 RemoveSlots 서비스를 지원하지 않고 Query Management 서비스 중의 복잡한 질의를 수행할 수 있는 SQL Query 서비스를 지원하지 않는다.

이에 비해 본 연구에서 개발한 시스템은 RIM v2.1과 RS v2.1에 따라 Oracle 8.1.7위에 Java 언어로 구현하였다. 또한 표준에 따라 Life Cycle Management 서비스와 Query Management 서비스를 전부 지원한다.

4.2 ebxmlrr과의 처리방식 비교

Sun사와 홍콩대학교가 같이 개발한 ebxmlrr 시스템은 본 연구에서 개발한 시스템과 같이 관계형 데이터베이스 위에 Java 언어를 사용해서 개발했으며, ebXML 레지스트리 표준에 따라 레지스트리 서비스를 전부 지원하기 때문에 전체적인 시스템 구조뿐만 아니라 지원되는 서비스들이 스펙에 명시된 방법에 따른다. 하지만 filter query를 구현하는 부분에 있어서 질의 처리 방법에 따라서 다음과 같은 차이가 있다.

〈그림 19〉에서 ebxmlrr의 Filter Query 구현 방식을 보여준다. ebxmlrr에서는 각 Filter마다 하나의 SQL 문으로 변환하고 RegRep에 액세스해서 하나의 결과를 생성한다. 각 Filter에 대한 실행한 결과들을 최종 결과로 합친다. 이렇게 해서 Filter가 많은 경우 데이터베이스를 여러 번 검색해야 되고 각각의 결과들을 합치는 과정도 거쳐야 한다.

본 연구에서 개발한 시스템이 Filter Query에 대한 구현 방식은 〈그림 20〉과 같이 나타난다. 여러 Filter들을 하나의 SQL 문으로 변환하고 데이터베이스에 한 번만 액세스해서 실행하면 결과를 생성한다.

ebxmlrr와 본 시스템의 구현 방식을 비교해 보면, 하나의 Filter Query에 대해서 ebxmlrr에서는 Filter의 수에 따라 여러 개의 SQL 문으로 변환한 다음에 데이터베이스를 여러 번 액세스해서 생성한 여러 개의 결과를 하나의 결과로 합친다. 하지만 본 시스템에서는 여러 개의 Filter들을 하나의 SQL 문으로 변환한 다음에 데이터베이스를 한 번만 액세스해서 실행한 결과를 바로 서버에서 생성해야 하는 최종 결과이다. 하나의 Filter Query에는 여러 개의 Filter들과 Filter들을 정의하는 서브클래스, 그리고 Branch들이 존재한다. 이런 Filter Query의 특성상 본 시스템의 구현 방식으로 하나의 SQL문으로 변환해서 실행하는 방식이 더 유리하고 효율적이다.

ebxmlrr의 소스를 보면 Query Management 서비스 중에서 필수 지원해야 하는 서비스인 Filter Qeury의 구현 방식은 본 연구의 구현 방식과 다르다. 이런 비교를 통해서 ebxmlrr에 비해서 본 연구에서 개발한 시스템은 더 효율적이라고 알 수 있다. XMLGlobal에서 개발한 GoXML Registry는 소스 공개를 하지 않아서 소스 분석을 통한 성능 비교를 할 수 없었다.

5. 결 론

XML을 이용하여 인터넷 기반의 B2B e-비즈니스가 가능하도록 하는 전자상거래 국제 표준인 ebXML이 국제 기구인 UN/CEFACT

와 OASIS가 공동 추진하여 2001년 5월에 표준으로 제정되었다. ebXML에서 핵심 역할을 담당하는 레지스트리에서 ebXML의 모든 메타 데이터는 레지스트리 서버에 저장되며 레지스트리 서버는 클라이언트들이 이 레지스트리 서버에 저장된 정보를 공유할 수 있도록 여러 가지 서비스를 제공하기 때문에 레지스트리 서버는 ebXML 레지스트리에서 가장 중요한 역할을 한다.

최근에는 기존의 ebxml 레지스트리 버전 1.0에서 부족했던 웹 서비스 기능을 보완하여 버전 2.0이 나왔으며, 2002년 6월에는 버전 2.0에서의 에러들을 수정하여 버전 2.1을 발표하였다. 본 논문에서는 표준 RIM v2.1 과 RS v2.1을 기준으로 ebXML Registry 서버를 설계하고 개발한 결과를 설명하였다.

레지스트리 서비스에는 객체들의 생명주기를 관리하는 Life Cycle Management 서비스와 저장된 객체들에 대한 메타 데이터들을 검색하고 Content Retrieval을 할 수 있는 Query management 서비스가 존재한다. 본 논문에서 개발한 ebXML 레지스트리 서버는 두 가지 레지스트리 서비스 모두를 표준에 따라 완벽하게 지원한다. Sun사와 홍콩대학교가 함께 개발한 ebxmlrr도 표준에 따라 서비스를 모두 지원하지만 본 시스템과 ebxmlrr의 개발 방식에 대한 비교를 통해서 본 연구는 더 효율적이라고 알수 있다. 그리고 XMLGlobal에서 개발한 GoXML Registry는 부분 서비스만 지원한다.

본 시스템은 기반 DBMS로 Oracle 8.1.7을 사용하였다. 그러나 JDBC를 이용해서 개발하였으므로, 다른 RDBMS위에서도 동작할 수 있다. 본 연구에서 구현된 시스템은 관계형 데이터베이스 위에서 동작하기 때문에 XML형태의 Filter Query를 실행하기 위해서 SQL Query로 변환하는 메커니즘이 필요하며 이 알고리즘을 설계하고 구현하였다. 제안한 알고리즘에 대해 표준에 명시된 예제들을 실행하여 타당성을 검증하였다.

향후 연구과제로는 현재 표준 작업이 진행중인 ebXML Registry v3.0의 추가 기능을 구현할 예정이다.

참 고 문 헌

[1] UN/CEFACT & OASIS, ebXML Technical Architecture Specification v1.04, http://www.ebxml.org/specs/index.htm#technical_specifications/ebTA.doc, 2001

[2] UN/CEFACT & OASIS, Registry Services Specification v2.1, http://www.ebxml.org/committees/regrep/document/2.1/specs/ebrs.pdf, 2002

[3] UN/CEFACT & OASIS, Registry Information Model v2.1, http://www.ebxml.org/committees/regrep/document/2.1/specs/ebrim__v2.1.pdf, 2002

[4] 전희영, 김계용, 유정연, 이규철, "ebXML 등록기/저장소에서의 객체 질의 관리 시스템 설계 및 구현", 2001 가을 한국정보

과학회 학술발표논문집(I), pp.37-39, 2001.10

[5] 조강녕, 박재홍, 김상균, 이규철, 조현규, "ebXML Registry에 대한 Filter Query의 SQL변환 기법", 2001 가을 한국정보과학회 학술발표논문집(I), pp.40-42, 2001.10

[6] 조강녕, 이병현, 김상균, 이규철, "ebXML Registry v2.0 서버 개발", 2002 전자거래학회 e-Korea를 위한 전자거래 종합학술대회 논문집, pp.356-363, 2002.09

[7] SUN&HongKong University, ebxmlrr, http://ebxmlrr.sourceforge.net, 2002

[8] IBM Software, Cloudspace, http:// www-3.ibm.com/software/data/cloudscape

[9] XMLGlobal, GoXMLTM Registry, http:// www.xmlglobal.com, 2002

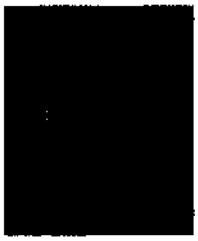
[10] ebXMLsoft, ebXMLsoft Registry 2.1, http://www.ebxmlsoft.com, 2002

[11] KTNET, KTNET ebXML Registry, http://www.ktnet.co.kr, 2002

[12] POSDATA, Biz Binder RR (Registry/Repository), http://www.posdata.co.kr, 2002

저 자 소 개

조강녕 (E-mail : knzhao@lgcns.com.cn)
2001. 경남대학교 컴퓨터공학과(공학사)
2003. 충남대학교 컴퓨터공학과(공학석사)
2003~현재 (주)LG CNS China Inc.
관심 분야 : 데이터베이스, 전자상거래, 웹 서비스, XML



이규철 (E-mail : kclee@ce.cnu.ac.kr)
1984. 서울대학교 공과대학 컴퓨터공학과(공학사)
1986. 서울대학교 공과대학 컴퓨터공학과(공학석사)
1990. 서울대학교 공과대학 컴퓨터공학과(공학박사)
1994. 미국 IBM Almaden Research Center 초빙 연구원
1995~1996. 미국 Syracuse University, CASE Center 초빙 교수
1997~1998. 학술진흥재단 부설 첨단학술정보센터 파견교수
1999~현재 한국정보과학회 논문편집위원
2000~현재 한국 ebXML 전문위원회 위원장
2001~현재 전자상거래 표준화 통합 포럼 전자거래 기반 기술위원회 부위원장
2003~현재 한국전자거래학회 편집이사
현재 충남대학교 공과대학 컴퓨터공학과 교수
관심 분야 : 데이터베이스, XML, 정보 통합, 전자 상거래