XML을 기반으로 한 관계형 데이터베이스 메타데이터 리파지토리 설계 및 구현

권 은 정[†] · 용 환 승[††]

요 약

메타데이터는 데이터에 관한 데이터로 데이터를 관리하기 위하여 사용된다. 데이터베이스를 기반으로 하는 응용프로그램이 증가하면서 메타데이터를 관리하기 위해 XML(eXtensible Markup Language)형식의 메타데이터 모델 및 메타데이터 교환모델이 제안되고 있지만 XML형식의 메타데이터를 관계형 데이터베이스에 저장하는 것이 일반적이다. 따라서 본 논문에서는 관계형 데이터베이스의 메타데이터를 객체지향 데이터베이스에 저장하기 위해 메타데이터 모델과 메타데이터 교환모델을 설계하여 메타데이터를 관리하는 새로운 리파지토리 시스템 구현을 목적으로 한다. 관계형 데이터베이스의 메타데이터를 XML문서로 변형 후 객체지향 데이터베이스의 XML 데이터 서버인 eXcelon에 저장하여 XQL(XML Query Language)에 의해 질의함으로써 관계형 데이터베이스 시스템들의 메타데이터의 검색, 편집을 가능하도록 하고 XSL(extensible StyleSheets Language)의 적용을 통하여 다양한 양식으로 웹 브라우저 상에 메타데이터를 출력함으로써 메타데이터의 속성을 면밀히 파악할 수 있다.

Design and Implementation of XML Based Relational Database Metadata Repository

Eun-jung Kwon[†] · Hwan-Seung Yong[††]

ABSTRACT

Metadata is data about data that is used to manage data itself. As applications based on DBMS are increased, it is suggested that metadata model and metadata interchange model to manage metadata in DBMS. but metadata which is in the form of XML (eXtensible Markup Language) document is generally stored into RDBMS. Therefore In this paper, as for the method to store metadata of RDBMS into OODBMS, we design metadata model, metadata interchange model and implement new repository system. The metadata of RDBMS is translated into in the form of XML Document and integrated into XML Data Server on OODBMS, eXcelon and executes retrieval metadata information about RDBMS by XQL(XML Query Language). So It is possible to search, edit a metadata. The metadata of XML documents stored in eXcelon is easily made to be printed in web browser by applying a XSL (extensible StyleSheets Language). So we can have a detail information about property of metadata in DBMS.

키워드 : 메타데이터(Metadata), 데이터베이스(Database), 리파지토리(Repository), 메타데이터 교환모델(Metadata Interchange Model), XML, XQL

1. 서 론

최근 정보사회라고 일컬을 만큼 폭발적으로 증가하는 데이터를 관리하기 위한 연구가 활발히 진행 중이다. 산재된 정보를 중앙 저장소로 저장하면서 발생하는 데이터의 생성 또한 데이터에 대한 정보를 지니기 때문에 관리할 필요성이 필요하며 데이터웨어하우스를 구축하는 예가 바로 이러하다.[1, 3, 7, 8] 이런 정보에 관한 정보를 지닌 데이터를 '메타데이터(metadata)' 라고 한다.

메타데이터를 관리해서 얻을 수 있는 이득은 곳곳에 산재되어 있는 메타데이터를 공유함으로써 대량의 데이터 관리, 유지 보수가 용이해 진다는 것이다. 이런 메타데이터를 관리해 주는 저장고를 메타데이터 리파지토리(repository)라고 한다[2, 4]. 메타데이터가 데이터에 대한 지도 역할을 한다면 이러한 메타데이터를 저장해 주는 리파지토리 시스템은 시스템 개발자에게 데이터베이스의 유지 보수나 확장 시에도 절대적으로 필요하다[1, 8]

메타데이터 리파지토리 기술은 메타데이터를 데이터베이스시스템에 저장하고 메타데이터는 표준화된 메타데이터 교환모델에 맞게 상호 교환을 가능하게 한다[7, 8]. 이러한 메

† 정 회 원 : 한국전자통신연구원 연구원
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
논문접수 : 2001년 4월 16일, 심사완료 : 2001년 11월 27일

타데이터 표준화는 XML 형식으로 이루어지고 있고 XML 형식의 메타데이터 교환 모델에 따라 데이터베이스시스템(DBMS)에 저장하는 것이 일반적이다[10, 11].

본 논문에서는 데이터베이스 기반의 응용 프로그램을 개발하면서 발생되는 데이터베이스의 카탈로그 정보를 메타데이터로 그 의미를 정의하여 데이터베이스 구조가 다른 관계형 데이터베이스의 메타데이터를 공유하기 위해 메타데이터 교환모델에 맞게 메타데이터를 XML형태로 생성하여 객체지향 데이터베이스인 eXcelon에 저장한다. 저장된 메타데이터는 물리적인 관점과 논리적인 관점으로 분류하여 상세 검색을 통해 메타데이터의 관리를 용이하게 한다. 즉 메타데이터의 물리적인 관점으로 데이터베이스를 구성하는 테이블 목록, 데이터의 물리적인 속성, 인덱싱 방법, 최종 갱신 시간, 사용 권한 등으로 구분하고 논리적인 관점으로는 테이블간의 관계 정보를 통해 데이터베이스 기반의 응용프로그램 개발이나 데이터베이스의 유지, 보수의 편리성을 제공한다.

XML로 정의된 메타데이터의 객체 지향적이며 계층적인 속성이 객체지향 데이터베이스의 구조와 맞기 때문에 XML 문서를 저장해주는 데이터베이스 시스템이 증가하고 있지만 XML 문서형태의 메타데이터를 관계형 데이터베이스에 저장하기 위해서는 관계형 데이터베이스의 구조에 맞게 변환시키는 과정이 필요하다[5, 6]. 따라서 본 논문은 XML문서를 저장해 주는 객체지향 데이터베이스를 XML 기반 리파지토리 저장엔진으로 선택하였다. 그러므로 관계형 데이터베이스에 맞게 구조의 변환이 불필요하다[9]. 이와 같은 장점을 고려하여 본 논문에서는 메타데이터 리파지토리를 객체지향 데이터베이스인 eXcelon을 메타데이터 저장소로 결정하고 메타데이터를 표현하는 XML문서를 구성하는 규칙은 XML-Data 스키마를 이용하여 메타데이터 교환모델을 제시한다. 또한 저장된 XML 형식의 메타데이터는 XSL(XML StyleSheet Language)을 이용하여 웹 인터페이스를 통해 보여지며 XQL(XML Query Language)을 통해 메타데이터를 편집, 검색 작업을 수행할 수 있다.

2. 기존의 메타데이터 리파지토리 구조 및 한계점

리파지토리는 소프트웨어 생명주기 동안 모아진 시스템 정보를 관리하기 위해 각 도구들, 개발 단계별, 사용자 정보들, 응용 프로그램 사이의 시스템 정보를 공유할 수 있도록 하는 정보 저장소이다[1]. 특히 데이터웨어하우스를 구축하거나 관계형 데이터베이스 기반의 응용 프로그램을 개발할 때 생성되는 메타데이터를 관리하기 위한 메타데이터 리파지토리는 메타데이터를 관리하기 위해 특별한 소프트웨어 설치를 하고 저장 엔진은 관계형 데이터베이스를 채택하고 있는 것이 일반적이다[1, 11].

2.1 기존 메타데이터 리파지토리의 대안

본 절에서는 기존 메타데이터 리파지토리의 대안으로 본 논문에서 제안하고 있는 대안을 세 가지로 요약할 수 있다. 첫 번째는 메타데이터 저장 방법, 두 번째는 메타데이터 교환모델, 세 번째는 사용자 인터페이스로 분류하여 대안을 제시한다.

2.1.1 메타데이터 저장 방법

메타데이터 리파지토리의 필요성이 대두되면서 마이크로소프트, 플라티늄, IBM, Metadata Integration에서는 표준으로 제시한 메타데이터 모델을 기반으로 리파지토리를 개발하였다.

〈표 1〉 리파지토리 시스템별 아키텍쳐 비교

구 분	저장 엔진	인터페이스	메타데이터 교환모델
PLATINUM	DB2	Windows Based GUI, WEB Interface	XIF
MS Repository	SQL Server	SQL DBMS	XIF
Prism	ORACLE, Informix, DB2	Windows Based GUI, WEB Interface	CDIF
Meta Integration	MS Access	Windows Based GUI	XIF,XMI

<표 1>에서 보는 것과 같이 메타데이타 리파지토리 저장 엔진은 관계형 데이타베이스를 사용하고 메타데이타 교환모델은 XML문서 구조를 따르는 것이 일반적이다. 즉 XML문서 형식의 메타데이타 교환모델을 통해 XML문서의 객체지향적인 특성인 계층 구조를 다시 관계형 데이타베이스에 맞게 구조를 변환해야 하는 작업이 필요하다. 이러한 이유로 전자상거래 애플리케이션 개발을 위해 자바의 오브젝트와 메소드를 메타데이터로 정의하고 저장, 관리를 위해 객체지향 데이터베이스인 O_2 기반의 리파지토리가 Ardent Software에 의해 개발되기도 했다[2].

2.1.2 메타데이터 교환모델

메타데이터 교환모델에 대해 언급하기 앞서 리파지토리를 구축하기 위한 요소로써 메타데이터 모델과 메타데이타 교환모델에 대한 정의가 필요하다. 특히 본 논문에서는 메타데이터를 XML문서형으로 생성하는데 있어서 XML문서의 정의는 DTD가 아닌 XML-Data 스키마를 통해 메타데이터를 생성하였다.

① 메타데이터 모델

메타데이터의 범위 및 메타데이터 간의 관계를 설정하기 위해 제안되고 있는 메타데이타 모델은 메타데이터를 관리하여 응용할 수 있도록 하기 위해 UML(Unified Modeling Language)을 이용한 메타데이터 모델링 방법론과 구현을 위해 API를 제공한다. MDC(Meta Data Coalition)의 OIM(Open Information Model), OMG

(Object Management Group)의 MOF(Meta Object Facility)가 표준화된 메타데이타 모델에 속한다.

② 메타데이터 교환모델

메타데이터 교환모델은 상이한 구조를 가진 데이터 양식을 메타데이터 교환 모델을 설정하여 메타데이터를 교환함으로써 정보의 손실 없이 데이터의 통합된 관리가 목적이다. 메타데이터 교환모델로는 MDC의 MDIS (Meta Data Interchange Specification), OMG의 XMI (XML-based Metadata Interchange), 마이크로소프트의 XIF(XML Interchange Format)가 있다.

③ 메타데이터 교환모델로서의 XML의 역할

문서의 교환은 문서의 구조와 관계되는 것으로 XML 문서 정의형을 구성하는 DTD 또는 XML-Data 스키마에 의해서 표현할 수 있으며 XML형태의 구조화된 데이터 교환의 시도는 표준화 기구를 통해 이미 검증을 거친 상태이다.

2.1.3 사용자 인터페이스

메타정보를 획득하기 위해서 본 논문에서는 웹 인터페이스를 통해 인터넷 환경 어디서나 메타데이터를 접근, 편집, 검색 할 수 있다.

3. 관계형 데이터베이스의 메타데이터 표현 및 메타데이터 교환모델 설계

본 논문에서는 관계형 데이터베이스의 메타데이터를 표현하기 위해서 UML을 이용하였다. UML의 사용 이유는 관계형 데이터베이스 내의 스키마 관계의 복잡성을 UML을 통해 시각적 모델링 방법을 통해 복잡한 메타데이터의 관계를 표현하기에 적합하기 때문이다[12]. 본 절에서는 UML을 통해 설계된 메타데이터 모델구조를 기반으로 관계형 데이터베이스의 메타데이터를 XML-Data 스키마에 맞게 XML문서로 표현함으로써 메타데이터 리파지토리 구현을 위한 하부 구조를 설명한다.

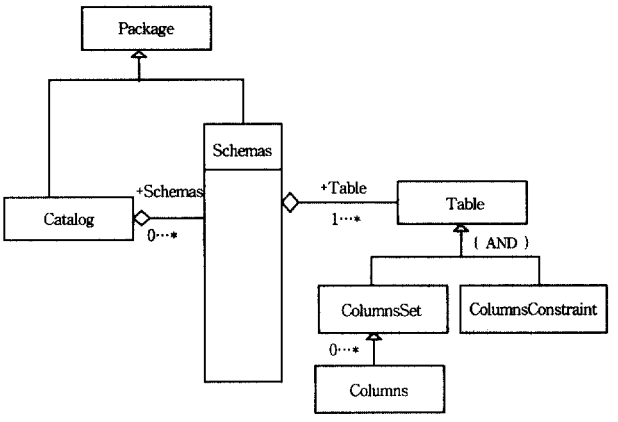
〈표 2〉 UML의 구성 요소

구 분	표 현	의 미
Association	———	하나의 객체와 다른 객체들 사이가 관련이 되었을 때 그 특징을 구체화하기 위해서 사용한다.
Aggregation	◇———	Association의 한 형태로 하나의 객체와 다른 객체간의 관계가 전체와 부분의 관계를 표시하기 위해서 사용한다.
Composition	◆———	Aggregation의 한 형태로 하나의 객체와 다른 객체간의 관계가 전체 부분에 대해 부분이 강한 소속감을 가지고 동일한 생명 기간을 가질 때를 나타낸다.
Refinement	◀·····	동일한 것에 대하여 다른 추상화 레벨에서 기술할 때 나타낸다.
Generalization	◁———	일반적인 것과 이 일반적인 것에서 특화 된 것 사이의 관계를 나타낼 때 사용한다. 즉 객체지향 언어에서 흔히 볼 수 있는 상속의 의미와 동일하다.
Dependency	<·····	하나의 특징이 변화함에 따라 다른 하나에 영향을 미칠 때의 관계를 표시할 때 사용한다.

3.1 메타데이터 표현

본 논문에서 표현한 메타데이터 간의 관계의 이해를 돕기 위해서 UML에서 사용되는 기본 표기법을 알아보도록 한다.

<표 2>의 UML 구성 요소를 기반으로 작성된 관계형 데이터베이스의 메타데이터는 다음 (그림 1)과 같다.



(그림 1) UML을 통한 관계형 데이터베이스의 메타데이터 표현

위의 (그림 1)에서 메타데이터 구성 요소로 사용되는 Package, Catalog, Schema에 대한 정의는 다음과 같다.

- Package

  하나의 클래스가 아닌 시스템에서 확장 가능한 모델링 구성 요소로써 UML로 선언된 다른 Package 엘리먼트와 그룹화가 가능하도록 제공되는 요소이다[12].

- Catalog

  데이터베이스를 정의하는 메타데이터로 테이블의 구성 및 테이블간의 관계 정보를 담고 있는 상위 레벨 메타데이터 저장하고 있는 구성 요소이다.

- Schemas

  테이블의 메타데이터를 구성하는 것으로 테이블 내에 정의된 다수의 컬럼 속성을 포함한다. 즉, (그림 1)의 Table 구성 요소 내에 정의된 다수의 Column을 포함하는 ColumnsSet과 테이블의 속성 정보를 가지는 ColumnConstraint 구성 요소를 전체 포함하는 구성 요소이다.

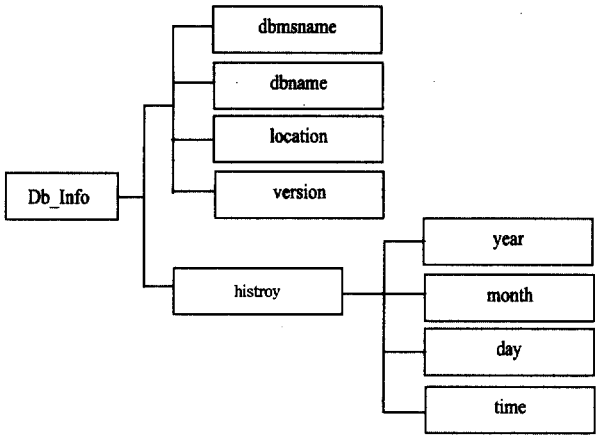
3.2 XML-Data를 이용한 메타데이터 교환모델 설계

메타데이터 교환모델을 설계하기 위해 관계형 데이터베이스의 속성에 따라 DBMS 메타데이터 교환모델 및 데이

터베이스 메타데이터 교환모델로 구분하여 XML-Data 스키마 규격에 맞게 메타데이타를 XML 문서로 생성한다.

3.2.1 메타데이터 교환모델 구조

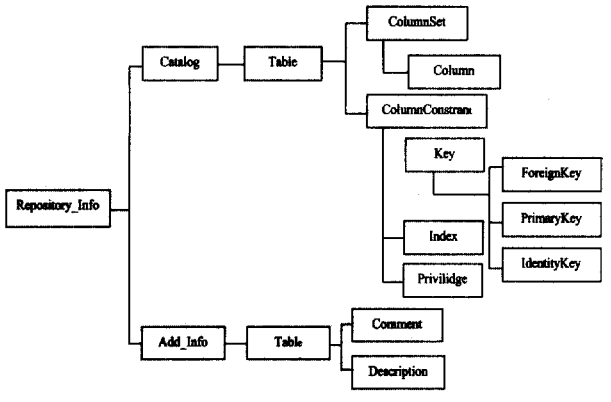
- DBMS 메타데이터 교환모델 구조



(그림 2) DBMS 메타데이터 교환모델 구조

(그림 2)는 DBMS의 속성을 기술하기 위한 메타데이터 구조로써 메타데이터의 히스토리 정보, 물리적인 위치의 속성, 호스트 이름, 특정 DBMS의 버전 정보에 관한 메타데이터를 XML문서로 생성하기 위한 구조이다.

- 데이터베이스 메타데이터 교환모델 구조



(그림 3) 데이터베이스 메타데이터 교환모델 구조

(그림 3) 데이터베이스 메타데이터 교환모델 구조는 데이터베이스의 속성 및 데이터베이스가 포함하고 있는 테이블의 속성을 나타내기 위한 구조이다. Repository_Info 엘리먼트를 루트로 시작하여 Catalog 엘리먼트의 서브 엘리먼트는 테이블의 속성을 표현하고 Add_Info 엘리먼트는 웹 브라우저를 인터페이스를 통해 메타데이터의 정보를 접근하여 수정할 경우 추가되는 엘리먼트이다.

3.2.2 메타데이터 교환모델 설계

(그림 2)와 (그림 3)을 기반으로 XML-Data 스키마에 맞게 메타데이터 교환모델을 설계한다.

- XML-Data를 이용한 문서 규칙

```
<?xml version = "1.0"?>
<Schema xmlns = "schemaname" xmlns : dt
                = "urn : schemas-microsoft-com : datatypes">
    <ElementType name = elementname content = contenttype/>
</Schema>
```

XML-Data 스키마를 위한 문서의 규칙은 XML 버전 1.0을 토대로 데이터 타입을 사용하기 위한 스키마 이름을 선언한다. 이를 토대로 엘리먼트를 선언하여 XML문서에서 사용될 엘리먼트의 규칙과 속성을 정의한다. (그림 1)을 토대로 작성된 DBMS 메타데이터 교환모델의 문서 정의형은 다음과 같다.

```
<?xml version = "1.0"?>
<Schema xmlns = "urn:schemas-microsoft-com:xml-data" xmlns :
          dt = "urn : schemas-microsoft-com:datatypes">
<AttributeType name = "id" dt:type = "string"/>
<ElementType name = "dbmsname" dt:type = "string"
<ElementType name = "dbname" dt:type = "string"
<ElementType name = "location" dt:type = "string"
<ElementType name = "year" dt:type = "datetime"/>
<ElementType name = "month" dt:type = "datetime"/>
<ElementType name = "day" dt:type = "datetime"/>
<ElementType name = "time" dt:type = "datetime"/>
<ElementType name = "history" content = "eltOnly" order = "one">
  <group order = "seq">
    <element type = "year"/>
    <element type = "month"/>
    <element type = "day"/>
    <element type = "time"/>
  </group>
 </ElementType>
 <ElementType name = "DB_Info" content = "eltOnly">
   <attribute type = "id" required = "yes"/>
   <element type = "dbmsname"/>
   <element type = "dbname"/>
   <element type = "location"/>
   <element type = "history"/>
 </ElementType>
</Schema>
```

위의 작성된 DBMS 메타데이터 스키마는 history 엘리먼트 내의 서브 엘리먼트 데이터타입을 datetime로 설정하여 history 엘리먼트 내에 한번만 나타내도록 하였다. 나머지 엘리먼트는 모두 string데이터 타입으로 설정하고 DB_Info 엘리먼트의 서브 엘리먼트에 포함된다.

(그림 2)를 토대로 Repository_Info를 루트로 시작해서 Catalog 엘리먼트는 데이터베이스 내의 테이블과 테이블의 컬럼 정보를 획득함으로써 데이터베이스의 테이블의 관계 정보를 파악할 수 있다. 나머지 Add_Info 엘리먼트는 웹 기반 인터페이스인 메타데이터 뷰어를 통해 관리의 편리성을 위해 메타데이터 정보에 대한 Comment 및 Description을 추가할 수 있도록 한다. (그림 2)를 토대로 설계된 XML-Data 스키마로 작성된 XML문서 정의형은 (그림 1)을 토대

로 작성된 DBMS 메타데이터 교환모델과 같은 맥락이므로 생략한다.

4. 시스템 구현

본 장에서는 논문에서 제안한 메타데이터 모델과 메타데이터 교환모델에 맞게 관계형 데이터베이스의 메타데이터를 추출하여 XML문서로 생성한다. 생성된 XML문서는 객체지향 데이터베이스인 eXcelon에 저장 후 웹 인터페이스를 통해 메타데이터를 접근하여 메타데이터 정보의 세부검색을 지원하는 예를 보인다.

4.1 시스템 구현 환경

본 장에서는 논문에서 제안한 메타데이터 모델 및 교환모델에 맞게 관계형 데이터베이스의 메타데이터를 추출하여 XML문서로 생성한다. 생성된 XML문서는 메타데이터 브리지를 통해 객체지향 데이터베이스인 eXcelon에 저장 후 웹 인터페이스를 통해 메타데이터를 접근하고 메타데이터 정보의 세부 검색을 지원하는 예를 보인다.

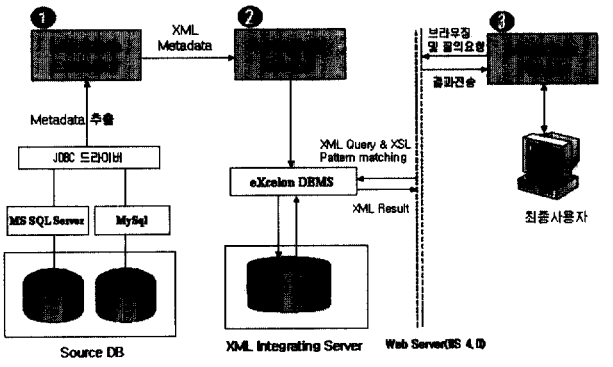
〈표 3〉 시스템 구현 환경

서버 측 운영 체제	Windows NT Server 4.0
XML Data Server	Object Design사의 eXcelon 2.0
웹 서버	Internet Information Server 4 (IIS4)
웹 클라이언트	Microsoft Internet Explorer 5.0 이상
개발 도구 및 언어	MS SQL Server 7.0, MySQL JDBC Driver, JAVA 1.1.6 SDK Extensible Style Sheet Language (XSL) XQL(XML Query Language) Active Server Page

JDBC를 지원하는 관계형 데이터베이스의 메타데이터는 JDK(Java Developer' Kit)를 통해서 메타데이터를 추출할 수 있다. XML기반 리파지토리를 구현하기 위한 프로토타입 시스템은 서버측에는 Windows NT 기반의 객체 데이터 서버인 Object Store사의 eXcelon을 사용하였다. eXcelon은 XML 문서를 관계형 데이터베이스에 저장할 때 행과 열을 위한 변환의 추가 작업 없이 문서 그 자체를 저장하기 때문에 관계형 데이터베이스처럼 런 타임 시조인 연산을 하지 않고 질의 할 수 있다. XML문서 형식의 메타데이터는 Remote Java API를 통해 eXcelon으로 저장되고 저장된 XML문서는 IIS 4.0 웹 서버를 기반으로 XSL을 통해 XML문서를 HTML 형식으로 결과를 보여 주는 웹 인터페이스를 제공한다. 이에 사용자의 질의 검색을 위해서는 XQL(XML Query Langugae)를 사용하였다.

4.2 시스템 구조

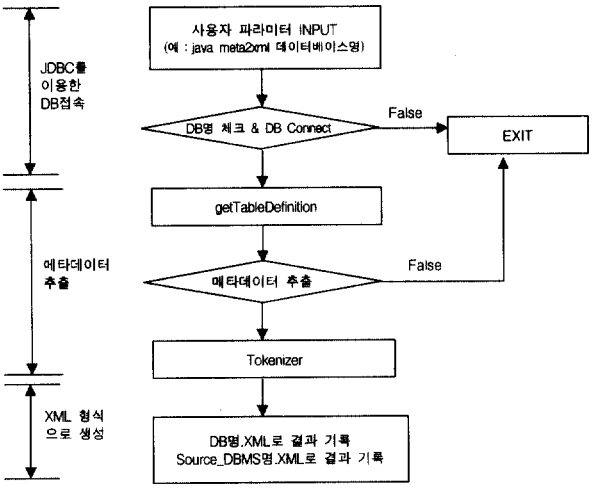
시스템 전체 구조는 크게 3개의 모듈로 나누어지며 각 부분의 해당 알고리즘을 기술하도록 한다. 첫 번째 모듈은 JDBC를 지원하는 관계형 데이터베이스의 메타데이터를 추출하여 메타데이터 교환모델에 맞게 XML문서로 생성하기 위한 Metadata Extractor 모듈, 두 번째 모듈은 생성된 메타데이터 결과물인 XML문서를 eXcelon으로 저장하는 Metadata Bridge모듈, 세 번째 모듈은 eXcelon에 저장된 메타데이터를 웹 인터페이스에 보여 주는 Metadata Viewer모듈로 구성이 된다.



(그림 4) 시스템 구조

4.2.1 메타데이터 추출 모듈(Metadata Extractor Module)

JDBC를 지원하는 관계형 데이터베이스의 메타데이터 추출은 JDK(Java Developer Kit)의 DatabaseMetaData 오브젝트에서 제공하는 메타데이터 메소드(Method)에 의해 얻어질 수 있다. 본 논문에서 사용된 DBMS인 SQL Server 7.0에 접근하기 위한 JDBC 드라이버는 Ashna사에서 제공하는 jturbo4.0을 사용하고 MySQL DBMS에 접근하기 위해서는 Matthews의 MySQL JDBC Driver Version 0.9를 사용하였다. 메타데이터를 추출하는 순서도는 다음과 같다.



(그림 5) 메타데이터 추출 모듈 흐름도

위의 (그림 5)처럼 데이터베이스 접속, 메타데이타 교환모

델에 맞는 메타데이터 추출, 추출된 메타데이터의 XML 문서로의 생성 과정을 거친다. XML구조로 저장되는 XML파일은 메타 데이터를 읽어 들인 'source_DBMS명.XML' 파일과 파라미터로 넘겨준 DB명을 argument 변수로 받아서 'DB명.XML'파일로 저장이 된다. 추출된 관계형 데이터베이스의 메타데이터를 표현하는 XML문서를 토대로 구현의 예를 제시한다.

4.2.2 메타데이터 연결 모듈(Metadata Bridge Module)

메타데이터 연결 모듈은 생성된 XML파일을 eXcelon에 저장시켜 주는 모듈로 XML파일과 eXcelon의 연결 다리 역할을 한다. eXcelon2.1 Data Server의 새로운 추가 기능은 Remote JAVA API를 제공함으로써 클라이언트에서 eXcelon Data Server에 연결하여 생성된 XML파일을 저장할 수 있도록 한다.

```
(1) import eXcelon Version 2.1 Public API ;
(2) meta2bridge(생성된XML파일) {
(3) Connect to the eXcelon Data Server ;
(4)       If (Connection error)
                exit (1) ;
(5)       DatabaseCreate(session s) ;
(6) DatabaseCreate(session s)
    {
(7)      if (repository Directory is not exist) { //repository 디렉토
         리가 존재하지 않으면 생성createDirectory ("repository") ;
           // Create a directory
(8)        if(CreateDirectory error)
              exit(1) ; //디렉토리 생성이 error가 나면 exit
         }
(9)        createXMLDocument("생성된XML파일") ; // Put a file
                              in the directory
(10)       Copy a xml file from this directory into the eXcelon
                              file system ; //파일 복사
(11)       Close Session) //열려진 session 닫음
       }
     }
```

위의 알고리즘은 eXcelon 2.1에서 제공하는 eXcelon Version 2.1 Public API 중 원격 데이터서버 제어를 위해 (1)라인 에서는 Class Import를 위해서 com.odi.excelon.client, com.odi.excelon. filesystem을 import하고 import된 클래스 중 (3)라인에서 XlnClientSessionFactory 클래스에서 유도되는 remote client eXcelon session을 (6)라인에서 핸들링 하여 (7)라인에서는 repository 데이터베이스가 존재하지 않는다면 createXMLStore 메소드를 통해 데이터베이스를 만든다. (9)라인에서는createDirectory 메소드를 이용해서 XML파일을 복사하기 위해 데이터 서버 내에 디렉토리를 만들고 (10)라인에서 파일 시스템에 존재하는 XML파일을 파라미터 값으로 전달하여 데이터 서버에 복사를 해 둔다. 마지막으로 (11)라인에서 열려 있는 세션을 닫는다.

4.2.3 메타데이터 뷰어 모듈(Metadata Viewer Module)

메타데이터 뷰어는 웹 인터페이스를 통해 DBMS정보 및 데이터베이스 별 테이블 및 테이블의 속성(Property)를 파악하고 사용자가 원하는 리포팅 형태에 따라서 같은 데이터를 XSL을 적용하여 관점에서 브라우징 할 수 있도록 한다. 메타데이터 뷰어 모델에서 제공하는 기능은 크게 메타정보의 브라우징 기능과 편집 및 검색 기능으로 구분된다. 브라우징 기능은 DBMS별 카탈로그 정보, 테이블 별 속성정보, 테이블간의 관계 정보가 있다. 그리고 메타데이터의 검색 기능으로는 데이터베이스 검색, 테이블 검색, 컬럼 검색, 세부 검색으로 나누어서 검색 패턴을 다양하게 하였다.

● 메타데이터 브라우징 및 편집 기능

XML문서로 저장된 메타데이터는 사용자에게 HTML형태로 변환되어 보여지도록 XSL의 Pattern Matching를 통해 최종 결과가 사용자에게 보여지게 된다. XML 기반의 리파지토리 시스템을 위해 총 4개의 XSL 문서로 구성되고 메타데이터의 추출된 결과물인 XML 문서와 XSL의 결합되어지는 원리는 XML문서 내에 XSL문서를 참조하는 형태와 eXcelon에서 제공하는 XSL문서 참조 형태로 구분해 볼 수 있다.

① XML문서 내에 XSL문서를 참조하는 형태

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "source_info.xsl"?>
<Repository_Info xmlns = "x-schema : schema.xml">
.....
</Repository_Info>
```

② eXcelon에서 제공하는 XSL문서 참조 형태

▶ HTML문서에서 직접 XML문서와 XSL적용

```
http : //localhost/scripts/xlnisapi.dll/repository/source_MSSQL
.xml?xslsheet=repository/source_info.xsl
```

▶ ASP 문서 내에서 eXcelon의 Session획득 후 XSL적용

```
'Session 획득
.....
OptArgs = "xslsheet = 'repository/ source_info.xsl.xsl'"
temp = "repository/source_MSSQL.xml"
set datares = sess.GetFileData(temp, OptArgs)
response.write datares
```

eXcelon에서 제공하는 XSL문서 참조 형태를 사용하게 되면 XQL의 사용을 통한 질의 결과에 따라 XSL문서를 적용할 수 있기 때문에 XML문서 내에 XSL문서를 참조하는 형태 보다 동적으로 XML문서를 처리 할 수 있다.

● 메타데이터 검색 기능

XML 문서의 데이터베이스, 테이블, 컬럼별 검색 및 세부 검색을 지원하기 위해서 XML문서를 질의할 수 있는 XQL을 사용한다. XQL질의 결과를 보이기 위해서는 eXcelon에서 COM 클라이언트 API기술인 Active Server Page를 사

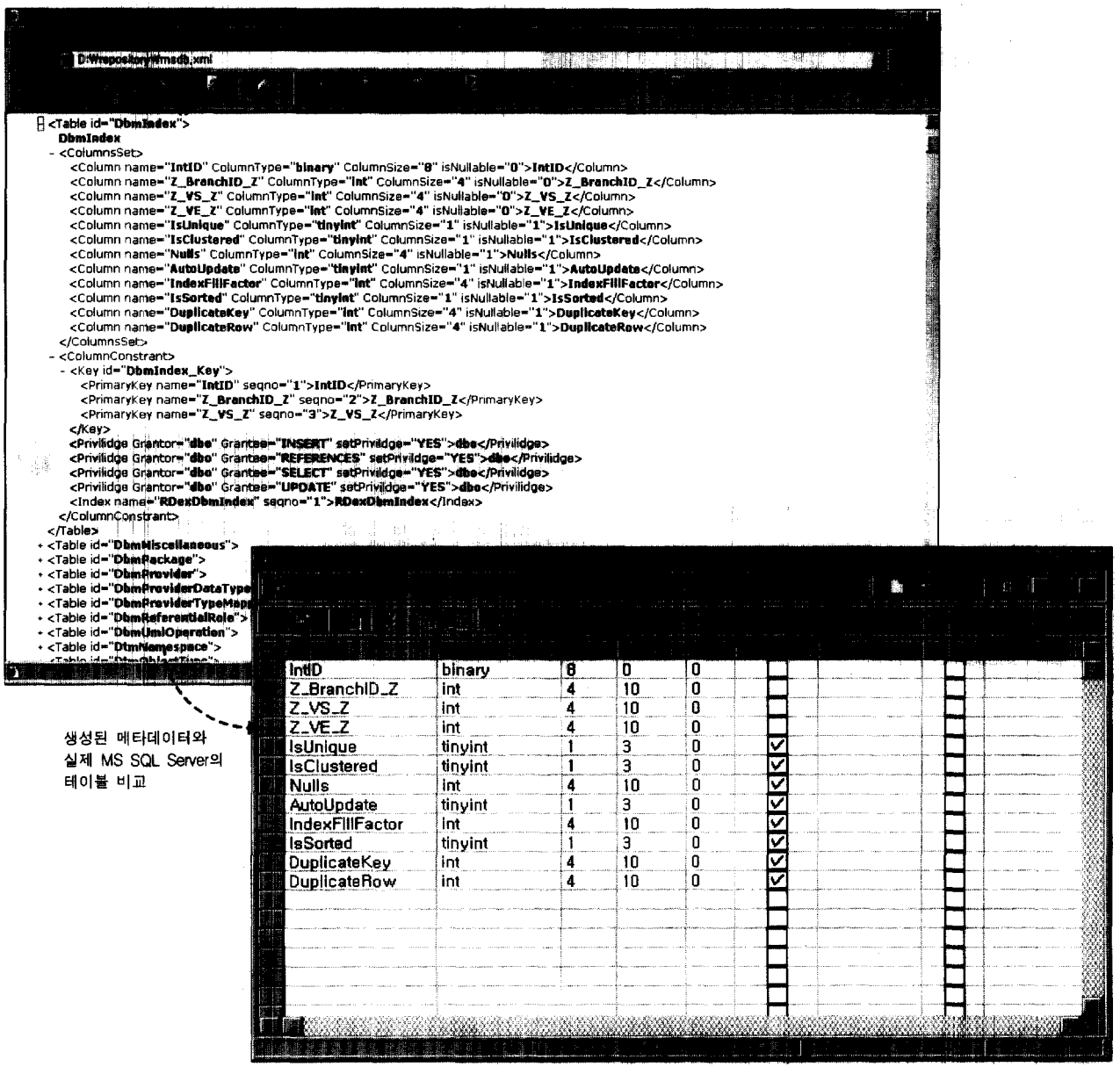
용하여 eXcelon서버와 연동 하여 검색을 위한 Application을 작성하였다. 검색을 통해 사용자는 실제 원하는 데이터베이스가 어떤 DBMS에 존재하는지, 테이블이 어떤 데이터베이스 내에 존재하는지, 그 테이블을 구성하는 컬럼의 속성들은 어떠한지 파악 할 수 있다. 실제 Active Server Page에서 XQL을 통한 질의문을 수행하기 위한 과정은 (1) eXcelon에 접근하기 위한 session 획득, (2) session획득 후 Query 수행, (3) XQL로 질의결과에 XSL문서를 적용 후 사용자에게 질의 결과가 최종적으로 보여진다.

```
(1) Function getSession
        Set sess = Application("eXcelon")
        If (sess is nothing) then
            Set fact=Server.Object("ODLeXcelon")
            Set sess = fact.GetSession(" ",0)
            Set Application("eXcelon") = sess
        End if
        Set getSession = sess
    End function
(2) Query
     Set sess = getSession()
     Set qry = sess.Query("repository : /data/source_MSSQL.XML",
              "//DB_Info/dbname")

(3) XSL StyleSheets 적용
    Set qry = sess.GetFileData("repository : /data/source_info.xsl", "")
```

4.3 구현된 시스템을 이용한 예제 프로그램

본 논문에서는 3장에서 기술한 메타데이터 및 메타데이터 교환모델에 따라 구현되어진 메타데이터 리파지토리의 기능을 알아본다. 4.2.3의 메타데이터 뷰어 모델의 확장으로



(그림 6) 테이블 메타데이터 정보를 지닌 XML문서의 생성

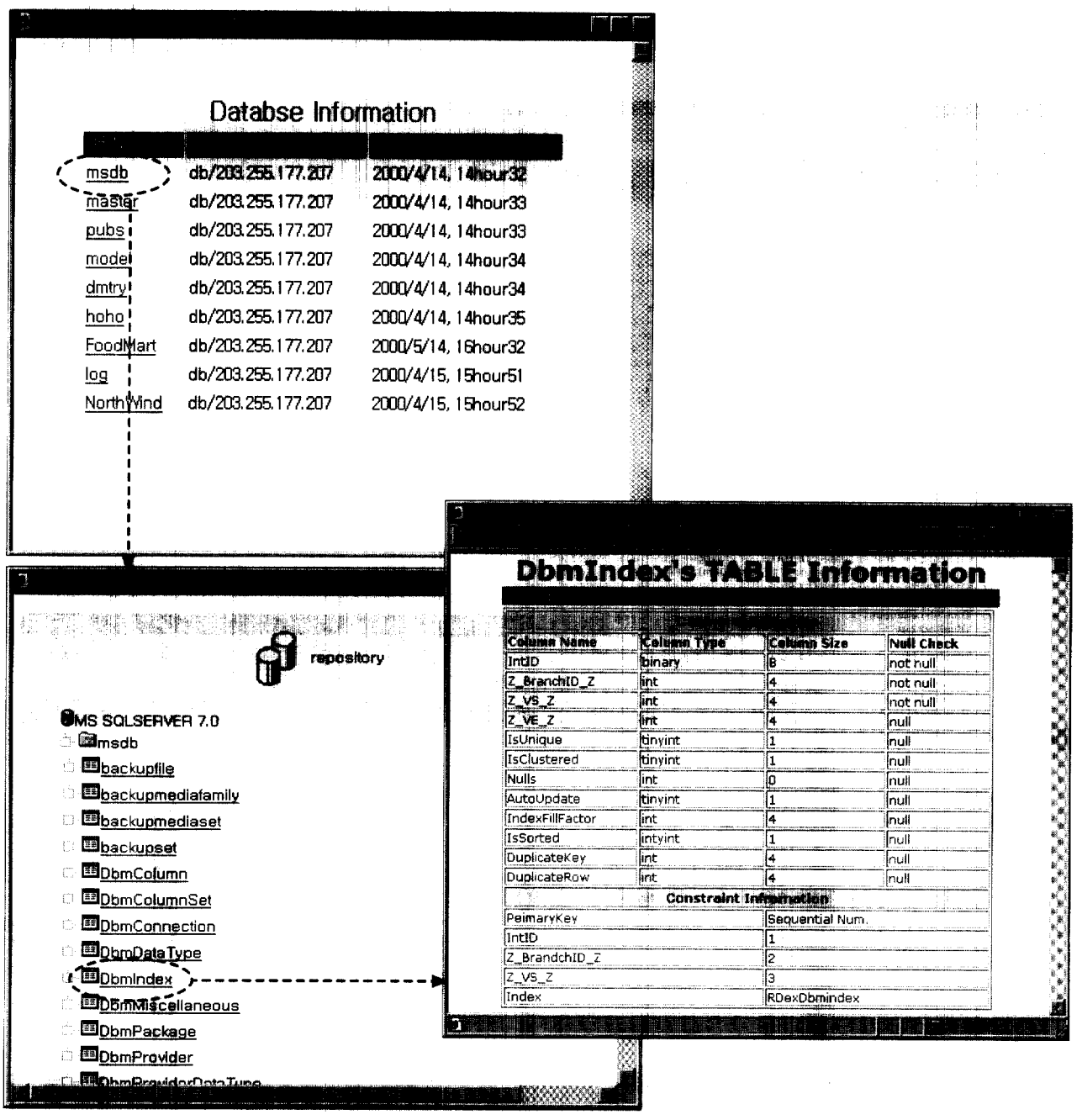
웹 인터페이스를 기반으로 리파지토리의 접근 권한을 가진 이는 메타데이터 접근을 통해 메타데이터의 상세정보를 인터넷 환경 어디서라도 검색, 편집 할 수 있다. (그림 6)은 메타데이터 교환모델에 맞게 관계형 데이터베이스의 메타데이터를 추출하여 XML로 생성된 예이다.

앞 (그림 6)의 관계형 데이터베이스의 XML문서형 메타데이터 표현을 시작으로 DBMS 정보, 데이터베이스 정보, 테이블 정보, 테이블 내의 상세 정보에 대한 접근, 편집 및 검색 기능을 수행한다. 구현 예제의 모든 사용자 인터페이스는 XML문서에 XSL을 적용하여 HTML형식으로 사용자에게 보여진다.

4.3.1 메타데이터 브라우징 및 편집 기능

eXcelon서버에 저장된 XML문서로 표현된 메타데이터를 XSL을 적용하여 DBMS별, 데이터베이스별, 테이블별로 브라우징 할 수 있으며 선택된 메타데이터를 eXcelon COM API를 통해서 편집할 수 있도록 한다. 구현 예제 및 구현 방법은 다음과 같다.

- 메타데이터 브라우징 예제



(그림 7) 특정 데이터베이스의 테이블 목록

● 메타데이터 편집 방법

```
GetSession()          Session을 얻기 위해 eXcelon에 접근하는
                      GetSession 수행
UpdateStr = "<?xml version = " "1.0" "standalone =" "yes"
            "?> <xlnupdate version =" "1.0" ">"
UpdateStr = update_string & "<update select =" "/Repository
            _Info" "><element location = ""lastchild"">"
.....
UpdateStr = update_string & "</element></update>"
UpdateStr = update_string & "</xlnupdate>"
set res = sess.Update(Fname, UpdateStr, " ")
'Fname : Update시킬 파일명의 변수값
```

eXcelon COM API를 이용하여 Session을 획득하고 변수 'UpdateStr'에 XQL 문장인 Update문을 실행함으로써 XML로 표현된 메타데이터의 편집이 가능하다.

4.3.2 메타데이터 검색 기능

메타데이터를 효율적으로 관리하고 검색하기 위해서 네 가지 검색 기능을 제공함으로써 원하는 컬럼이 소속된 테이블과 데이터베이스 목록을 파악할 수 있도록 한다. 검색을 위한 XQL을 사용함으로써 다중 XML문서 검색, 질의 구간의 제약을 통한 특정 엘리먼트의 하위 검색(subquery), 어트리뷰트 검색 등이 가능하다. 네 가지 검색 기능 중 대표적인 예로 테이블 검색을 통해 검색하고자 하는 테이블이 어떤 데이터베이스에 포함이 되고 테이블의 속성은 어떠한지 질의 방법을 알아본다. XML문서 검색을 위해서는 검색하고자 하는 대상의 XML문서를 다중 XML문서 정보를 가진 XML문서에 포함 시켜야 한다.

● DBMS 메타데이터 정보를 지닌 XML문서의 다중 XML 문서 표현

```
<?xml version = "1.0" encoding = "UTF-8"?>
<metadatas xmlns : xlnlink = "http : //www.ObjectDesign.
          com/ eXcelon/namespaces/link"
 xlnlink : link = "simple" xlnlink:queryAction = "auto">
        <metadata xlnlink:href = "source_MSSQL.xml/"/>
        <metadata xlnlink:href = "source_MySql.xml/"/>
</metadatas>
```

DBMS의 메타데이터를 정보를 가진 source_MSSQL.xml와 source_MySql.xml 문서를 포함함으로써 xmlns 네임스페이스를 통해 문서 내에 포함된 XML문서의 내용을 한 문서씩 스캔 (scan) 하면서 질의 결과를 통합한다.

5. 결론 및 향후 과제

본 논문에서 제안한 메타데이터 리파지토리 시스템은 기존의 다른 연구들에 비하여 다음과 같은 특징을 갖는다.

● 객체지향 데이터베이스 서버를 저장 엔진으로 채택했다. 메타데이터를 저장해 주는 기존 저장 엔진은 관계형 데이터베이스를 기반으로 하고 있다. 이것은 메타데이터 교환모델은 객체지향적인 성격을 띄고 있으나 저장 엔진은 관계형 데이터베이스를 기반으로 하는 것은 다시 한번 XML문서의 변환 과정이 필요하다. 따라서 본 논문은 XML문서로 생성된 메타데이터 결과물을 직접 저장하기 때문에 변환 과정이 필요하지 않다.

● 메타데이터 정보의 접근이 다양하다. 차세대 인터넷 표준 언어인 XML을 기반으로 생성된 메타데이터의 정보를 웹 기반 사용자 인터페이스를 통해 제공하기 때문에 어디서라도 메타데이터의 정보를 파악 할 수 있다. 현재 대부분의 메타데이터 리파지토리는 특정 소프트웨어나 DBMS에 의존적이기 때문에 메타데이터의 정보를 획득하기 위해서는 특정 소프트웨어에 의존할 수밖에 없다.

● 검색을 위한 연산 비용을 줄일 수 있다. 기존의 연구들은 추출된 XML 형식의 메타데이터를 관계형 데이터베이스에 저장함으로써 상세 검색 시 테이블간의 조인(join)연산이 필요하다. 그러나 본 논문에서는 XML data server를 이용하여 객체를 중심으로 저장 되므로 조인 없이 메타데이터 정보의 검색이 가능하다.

● 다양한 장소에서 메타데이터의 교환 및 통합이 용이하다. XML은 웹 상에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 형식이며 여러 장소에서 생성된 메타데이터 정보의 교환 통합이 용이하다.

● XSL을 이용하여 같은 데이터를 사용자의 관점에 따라 다양하게 출력할 수 있다.

향후 연구 과제로는 본 연구에서 제안한 XML 메타데이터 모델 및 메타데이타 교환모델의 추가 기능으로 메타데이터 정보의 추출 범위를 확대하고 메타데이터 뷰어 모듈에서 테이블간의 관계 정보를 표현할 수 있도록 확장함으로써 좀 더 세밀한 메타데이터 정보를 파악할 수 있도록 한다.

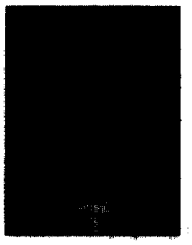
참 고 문 헌

[1] Philip A. Bernstein, Brain Harry, Paul Sanders, "An Overview of Repository Technology," Proceeding of the 20th VLDB Conference, Santiago Chile, pp.705-713, 1994.

[2] J. C. Mamou, T. Milo, "XML repository and Active Views Demonstration," Proceeding of the 25th VLDB Conference, Edinburgh, Scotland , pp.742-745, 1999.

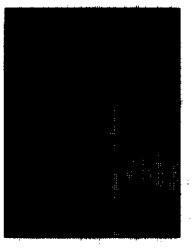
[3] Computerwire, "What is Metadata," Data Warehousing Tools Bulletin, http : //www.computerwire.com/bulletinsuk/

212e_1a6.htm, January, 1996.
[4] Serge Abiteboul, "On Views and XML," Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp.1-9, 1999.
[5] Daniela Florescu, Donald Kossmann, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," INRIA Technical Report, INRIA, No.3680, Mai, 1999.
[6] Daniela Florescu, Alin Deutsch, Mary Fernandez, Alon Levy, David Maier, DanSuciu, "Querying XML data," Data Engineering Bulletin, Vol.22, No.3, pp.27-34, 1999.
[7] Computer Associate, "Putting Metadata to Work in the Warehouse," http : //www.cai.com/products /platinum/wp/wp_meta.htm.
[8] Computer Associate, "Platinum Repository," Computer Associate's Platinum Repository Product Description, http : //www.cai.com/products/descriptions/repository_ pd.pdf.
[9] Vitorio Viarengo, "eXcelon XML Data Server Technical Overview," Object Exchange 98, Object Design User Conference, 1998.
[10] Meta Data Coalition, "driving the evolution of metadata interchange format standards," http : //www.mdcinfo.com/index.html.
[11] Microsoft, "Microsoft SQL Server Meta Data Services," http : //msdn.microsoft.com/repository/oim/overview.asp.
[12] UML Korea, "UML 사용자 지침서", http : //www.uml.co.kr/html/dox/userguide_01.ppt.
[13] Microsoft, "Oepn Information Overview(OIM)," http : //msdn.microsoft.com/oim/overview.asp.



권 은 정

e-mail : ejkwon@etri.re.kr
1997년 동덕여대 전산정보학과 학사
2000년 이화여대 대학원 컴퓨터학과 공학석사
2000년~현재 한국전자통신연구원 연구원
관심분야 : XML, 디지털 방송 기술, 암호학



용 환 승

e-mail : hsyong@ewha.ac.kr
1983년 서울대학교 컴퓨터공학과 학사
1985년 서울대 대학원 컴퓨터공학과 공학석사
1985년~1989년 한국전자통신연구원 연구원
1994년 서울대 대학원 컴퓨터공학과 공학박사
1994년 서울대 컴퓨터신기술공동연구소 특별연구원
1995년~현재 이화여자대학교 컴퓨터학과 부교수
관심분야 : 객체-관계 데이타베이스 시스템, 멀티미디어 데이터베이스, OLAP 및 데이터 마이닝, 바이오정보학