RDB 대상 메타데이타 교환을 위한 XML기반 번역기 설계 및 구현

(Design and Implementation of an XML-based Translator for Metadata Interchange in RDBs)

이 월 영 † 이 기 호 ††
(WolYoung Lee) (Kiho Lee)

요 약 XML은 여러 분야에서 다양한 타입의 데이타를 효과적으로 관리하기 위한 표준으로 사용되고 있다. 이러한 XML의 한 응용으로서 데이타베이스 시스템, 파일 시스템, OLTP시스템, OLAP시스템, 데이타웨어하우징을 위한 ETL 시스템, 데이타마이닝 시스템 등을 포함하는 다양한 타입의 데이타 처리 시스템 사이에서 메타데이타를 교환하기 위하여 XML을 이용하여 메타데이타를 표준화하는 것이다. 본 연구는 MDC(MetaData Coalition)에서 제안한 일반적인 데이타 처리 시스템을 위한 메타데이타 교환 스펙인 MDIS(MetaData Interchange Specification)를 따르도록 설계한 관계형 데이타베이스 시스템의 메타데이타 교환을 위한 사양으로서 R-MDIS(MetaData Interchange Specification for RDBs) 모델을 제안하고 이에 따라 표현된 자료를 상호 데이타베이스 시스템 사이에서 교환할 수 있는 Export 함수와 Import 함수를 포함하는 번역기를 개발한다. 이 번역기는 메타데이타의 일관성을 유지하면서도 메타데이타의 양방향 흐름을 지원할 수 있도록 하는 정보를 포함하고 있다. 본 연구는 구체적인 관계형 데이타베이스 시스템에서의 메타데이타 교환을 위한 실제적인 번역기를 구현함으로써 다양한 종류의 데이타 처리 시스템들 사이에서 메타데이타 불일치 문제를 해결할 수 있는 근거를 제시하였다. 둘째, XML을 이용하여 메타데이타 교환 사양을 설정함으로서 구조화된 데이타나 구조화되지 않은 데이타 모두에 대하여 일관된 방법으로 데이타 교환이 가능하도록 하여 효율적인 정보 교환을 할 수 있다는 것을 보여 주었다. 셋째, 본 연구에서 개발한 번역기는 메타데이타 교환에 의해 데이타들을 공유하게 함으로서 효율적으로 관리하지 못하던 데이타들을 효율적으로 사용할 수 있게 함으로서 데이타 관리 문제에 해결책이 될 것이다.

Abstract XML is an emerging standard for efficient data managements. One very important application of XML is the exchange of metadata among data-processing systems of various types, including database systems, file systems, OLTP systems, OLAP systems, data warehousing ETL systems, data mining systems, etc. This research proposes R-MDIS(Meta Data Interchange Specification for RDBs) based on the MetaData Interchange Specification (MDIS) proposed for metadata interchange by the Meta Data Coalition (MDC). The MDIS is a metadata interchange specification for general data-processing systems. However the R-MDIS that we proposed is metadata interchange specification among the relational database systems. Also a translator is developed that integrates the functions of export and import to exchange metadata between a pair of RDBs. This translator is designed to supports bi-directional interchange of metadata and consistency of data exported and imported in compliance of the metadata exchanged. The contributions of this research include solutions for metadata inconsistency problem among various data processing systems, methods for efficient information exchange of structured data and semistructured data by the XML consistently, and solutions for the data managements through the useful data sharing.

† 학생회원 : 이화여자대학교 컴퓨터학과
wylee@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
khlee@mm.ewha.ac.kr
논문접수 : 2000년 4월 26일
심사완료 : 2000년 12월 11일

1. 서 론

인터넷의 발전으로 나날이 늘어가고 있는 웹 데이타를 유용한 정보로 활용하기 위하여 여러 기관에서는 이

들 자료에 대한 메타데이타를 eXtensible Markup Language(XML)를 이용하여 표준화를 시도하고 있다. 대표적인 예로서 RDF(Resource Description Framework)는 사이트 맵, content ratings, 스트림 채널 정의, 검색 엔진 데이타 모음(web crawling), 전자 도서 모음, 분산 저작 등을 포함하는 다양한 웹 기반의 메타데이타 액티비티들을 통합하기 위하여 XML을 사용하여 메타데이타 교환을 위한 문법을 표현하고 있다[12]. 그 외에도 XML은 고유한 자신만의 마크업 언어를 정의할 수 있기 때문에, Dublin Core[1], Warwick Framework, Resource Platform for Internet Content Selection (PICS) [11]과 같은 여러 형태의 메타데이타를 표현하는데 응용될 수 있다. XML을 이용하여 표준화시킨 자료는 서로 다른 시스템들 간에 아무런 정보의 손실 없이 효율적으로 전송하고 저장할 수 있으며 상호 교환하여 사용할 수 있고[10] 웹 데이타와 같은 구조화되지 않은(semistructured) 자료뿐 아니라 구조화된 환경에서 구축된 데이타베이스의 자료라 할지라도 일관된 방식으로 통합하여 사용할 수 있도록 하는 표준이 되고 있다[14]. 이들 자료를 XML로 표준화한다면 어떠한 환경으로든 이주가 가능하고 유용한 정보로 활용할 수가 있다.

이러한 시대적인 흐름에 발맞추어 일반적인 데이타 처리 시스템을 대상으로 하는 메타데이타 표준화에 대한 연구는 있어 왔지만 구체적으로 관계형 데이타베이스 시스템 사이에서의 메타데이타 교환을 위하여 소스 메타데이타를 추출하여 타겟 데이타베이스 시스템으로 메타데이타를 전송함으로서 데이타를 교환할 수 있는 실제적인 번역기 개발에 대한 연구는 아직 미비한 상태이다. XML 표준화 작업 이전의 많은 자료는 표준이 없는 상태에서 구축이 되었기 때문에 구조나 형식 또한 다르게 구성되어 있어 서로의 정보 자료는 교환이 불가능한 상태로 남아 있다[3]. 이로 인해 자료의 양이 기하급수적으로 늘어나고, 또한 이들 자료를 관리하는 데이타에 대한 데이타가 되는 메타데이타도 급증하고 있어 많은 사용자는 여러 가지 면에서 고통을 당하고 있다.

따라서 우리는 관계형 데이타베이스 시스템에 저장되어 있는 데이타를 상호 교환하여 사용할 수 있도록 하기 위하여 메타데이타 교환을 위한 사양을 설정한 R-MDIS(MetaData Interchange Specification for RDBs)모델을 제시한다. 또한 제시된 모델에 따라 서로 다른 관계형 데이타베이스 시스템 사이의 메타데이타를 교환할 수 있도록 메타데이타 Import, Export 기능을 지원하고, 메타데이타 일관성을 유지하면서 메타데이타의 양방향 흐름을 지원할 수 있는 정보를 포함하는 번역기를 개발한다. 우리가 제안한 R-MDIS모델은 텍스트 문서로 되어 있어 문서의 파싱을 통해 어떠한 시스템으로도 이주가 가능하고 어떠한 응용에도 자유 자재로 활용될 수가 있다. 또한 우리가 개발한 번역기는 보통의 JDBC가 다루는 일반적인 메타데이타 뿐만 아니라 자바소프트에서 별도로 제공하고 있는 API를 이용하여 관계형 데이타베이스의 추가적인 메타데이타를 추출한다. 여기서 대상으로 하고 있는 메타데이타는 관계형 데이타베이스 시스템이 설치될 때 발생되는 시스템 자체에 대한 메타데이타는 그대로 두고 사용자가 데이타베이스를 생성할 때 발생되는 메타데이타를 대상으로 하는데 테이블과 컬럼에 대한 권한이나 그들의 속성, 제한점, 기본키, 외부키 등과 인덱스 정보 등을 교환한다. 이러한 메타데이타들은 관계형 데이타베이스 시스템에 공통으로 존재하고 있는 것들이라 하나의 관계형 시스템에서 추출한 메타데이타는 또 다른 관계형 시스템에 아무런 문제없이 이식이 가능하다. 따라서 각 데이타베이스 시스템에 따라 유사하지만 구문이나 구조에 있어 다르기 때문에 발생하는 메타데이타의 불일치 문제를 근본적으로 해결하고 있다. 우리는 이러한 방법으로 메타데이타를 이식한 후 실제 데이타를 이주시킴으로서 호환이 불가능한 많은 양의 자료를 상호 교환하여 사용할 수 있도록 하고 있다. 이와 같이 데이타 처리 시스템 사이에서의 메타데이타 교환을 위한 메타데이타 번역기 개발은 비즈니스 어플리케이션의 상호 운영성의 실제적인 방법을 보여 주고 있다.

본 논문은 1장의 서론에 이어 2장에서는 XML의 특징을 보고 메타데이타와의 관계를 분석하여 어떻게 응용될 수 있는지 살펴본다. 제 3장에서는 XML기반의 메타데이타 교환을 위한 사양을 정의한 R-MDIS 모델을 제안하고 이를 통해 메타데이타를 교환하는 번역기를 설계한다. 제 4장에서는 3장에서 설계한 번역기를 토대로 실제 구현을 하고 제 5장에서는 결론을 맺는다.

2. 관련 연구

2.1 메타데이타와 표준화

ISO description 11179에서는 메타데이타를 이해 가능하고 공유 가능한 데이타 세트를 만드는 정보와 문서로서 데이타에 대한 데이타로 묘사하고 있다. Meta Data Coalition(MDC) Open Information Model(OIM)에서는 메타데이타를 데이타와 어플리케이션 구조와 의미에 대한 명세적인 정보와 데이타를 조정하는 프로세스라고 정의하고 있다[9].

이러한 정보를 명세할 때 공통의 언어를 사용하여 표현한다면 보다 명백히 의사 소통을 할 수 있고 시스템과 프로그램은 의미를 갖고 상호 의사 소통을 할 수가 있다. 그러나 같은 의미의 데이타를 서로 다른 언어를 사용하여 각기 자신의 파일과 데이타베이스를 유지하고 있다면 데이타가 중복되고 또한 이것들을 변화 시켜야 서로의 정보 공유가 가능해지기 때문에 막대한 시간과 사람을 필요로 하게 된다. 공통의 언어로 정의된 메타데이타는 모든 사람, 시스템, 그리고 프로그램 사이에서 서로 밀접하게 교통할 수 있도록 해준다. 이것이 표준화의 중요성이다[3]. 따라서 XML을 이용한 메타데이타의 표준화는 시스템의 환경이 다름으로서 유용한 자료들을 교환하여 사용하지 못하고 각 시스템에 중복하여 유지하던 과거의 데이타 유지관리 방법을 개선하고 공통의 데이타 버전으로 통합하게 함으로서 데이타 처리비용을 절감시킬 수 있다.

2.1.1 메타데이타 표준화를 위한 노력들

여러 기관에서는 메타데이타 표준화의 중요성을 인식하고 메타데이타 표준화 방안을 내놓았고 XML을 이용한 메타데이타 표준화 작업들도 다양한 형태로 진행되고 있다[15]. 구조화된 데이타에 대한 표준화로서 데이타웨어하우스와 관련된 메타데이타의 표준화는 메타데이타 모델 부분과 메타데이타 교환 메소드 부분으로 구분된다.

표 1 메타데이타 표준화[15]

메타데이타 모델		메타데이타 교환 방법	
표준	주도기관	표준	주도기관
MOF	OMG	XMI	OMG
MSR/OIM	Microsoft	MDIS	MDC

이 중 본 연구와 가장 유사한 연구로서 1998년 OMG에서 제안한 XMI(XML Metadata Interchange)는 XML을 사용하여 객체에 대한 정보를 교환하는 것이고[18], 여러 업체[7]가 참여하고 있는 MDC(Meta Data Coalition)에서 제안한 ER모델에 기반한 메타데이타 교환에 관한 연구인 MDIS(Meta Data Interchange Specification)[8]가 있다. MDIS는 툴들 사이에서 메타데이타를 공유하기 위하여 필요한 가장 기본적이고 일반적인 것들만을 언급하고 있다. 즉, MDIS는 객체 유형, 데이타베이스 이름, 스키마, 파일, 관계 등을 정의하고 있다. 그러나 본 연구가 제안하는 R-MDIS모델은 구체적으로 관계형 데이타베이스 시스템을 대상으로 관계형 데이타베이스의 데이타를 교환하기 위하여 필요한 구체적이고 세세한 메타데이타들을 정의하고, 이러한 메타데이타들을 서로의 관계형 데이타베이스들 사이에서 교환하기 위해서는 XML을 이용하여 어떻게 표현하는가에 대한 방법을 제시한 모델이다. R-MDIS는 관계형 데이타베이스 시스템들에 공통으로 존재하는 메타데이타, 즉 관계형 데이타베이스의 테이블 정보나 컬럼 정보, 인덱스 정보 등을 추출하여 또 다른 관계형 데이타베이스 시스템에 이식시킨 후, 하나의 데이타베이스에서 실제 데이타들을 또 다른 데이타베이스로 이주시킴으로서 메타데이타 불일치로 인하여 서로의 관계형 데이타베이스들 사이에서 데이타를 공유하지 못했던 문제를 해결할 수 있는 번역기를 개발한다.

2.1.2 XML을 이용한 메타데이타 표현

XML DTD 파일은 XML문서의 전체적인 구조를 정의[4,17]하기 때문에 메타데이타를 XML 문서로 표현하고 이에 대한 구조로서 DTD를 구성한다. 예를 들어 다음과 같은 관계형 데이타베이스 테이블에 대한 컬럼 정보를 위해 메타데이타와 내용을 구성한다면 다음과 같다. 여기의 예는 컬럼과 테이블에 대한 정보를 나타내는데 데이타 타입은 숫자에 따라 문자형, 숫자형, 긴 문자형 등을 나타내는데, 이 중 4와 같은 숫자는 정수형을 뜻하고 이것에 대한 type name이 int identity인 것은 정수형이면서 primary key로 정의되었다는 의미이다. 그 외 컬럼의 크기, 카탈로그의 이름, 스키마의 이름이나 테이블의 이름 등에 대한 정보를 나타낸다.

표 2 메타데이타 예

COLUMN	Column_Name	EmployeeID
	Data_Type	4
	Type_Name	int identity
	Column_Size	10
TABLE	Catalog_Name	Northwind
	Schema_Name	dbo
	Table_Name	employees

이것에 대한 XML 문서는 다음과 같이 표현한다.

```
<COLUMN>
  <Column_Name>EmployeeID</Column_Name>
   <Data_Type>4</Data_Type>
   <Type_Name>int identity</Type_Name>
   <Column_Size>10</Column Size>
</COLUMN>
<TABLE>
   <Catalog_Name>Northwind</Catalog_Name>
   <Schema_Name>dbo</Schema_Name>
   <Table_Name>employees</Table_Name>
</TABLE>
```

이 때 < >를 마크업 태그라 하고 실제로 표현된 내용(contents)인 메타데이타를 설명하는 부분이다. 여기서 보는 것처럼 <COLUMN>의 자세한 정보는 <Column_Name>, <Data_Type>, <Type_Name> 등을 통해 쉽게 알 수 있다. 또한 DTD(Data Type Definition)를 이용해 어떤 태그가 있어야 하는지 아니면 옵션인지를 명시할 수 있고 어떤 태그는 서브 태그를 갖는지 등을 나타낼 수 있다.

3. 메타데이타 교환을 위한 XML기반 번역기 구조 및 R-MDIS

3.1 번역기 구조

사용자가 하나의 데이타베이스를 생성하면 이에 따르는 정보가 될 수 있는 형태의 데이타인 메타데이타가 리파지토리에 저장된다[16]. 리파지토리에 저장되는 메타데이타는 똑같은 기본 데이타에 대해서도 시스템에 따라, 사용하는 툴에 따라 다른 형태와 다른 종류의 메타데이타를 생성하고 이로 인해 상호간 데이타의 호환도 불가능하다. 따라서 의미가 같은 데이타라 할 지라도 메타데이타의 불일치로 인하여 시스템간에 서로 데이타를 공유할 수 없고 DBMS의 종류가 다를 경우에는 중복하여 데이타를 유지하여야만 한다. 본 연구에서는 이러한 현실에서 상호 데이타베이스 간 데이타 교환을 위해 필요한 메타데이타 사양들을 XML을 이용하여 표현한 모델을 정의하고 이를 기반으로 메타데이타를 추출하고 또한 다른 데이타베이스로 이식할 수 있는 함수를 포함하는 번역기를 설계한다. 이를 위한 번역기의 개념적인 설계도는 다음과 같다.

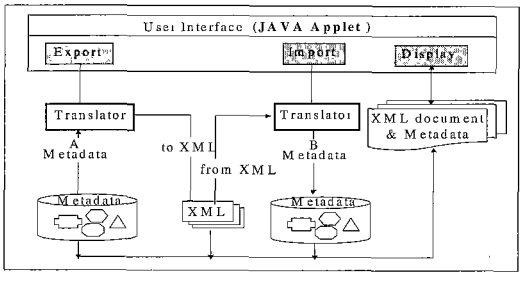


그림 1 번역기 구성도

그림에서 보는 바와 같이 메타데이타 교환을 위한 번역기는 크게 두 부분으로 구성된다.

① XML을 이용하여 의미와 형식을 지니고 있는 교환될 메타데이타에 대한 사양을 정의하는 부분이다. 여기서는 XML을 이용하여 관계형 데이타베이스의 메타데이타 교환을 위한 사양을 정했다는 의미에서 R-MDIS(MetaData Interchange Specification for RDBs)로 명명한다. 또한 버전은 1.0으로 정한다.

② 메타데이타를 Export하고 Import하기 위한 두 개의 파일기반의 함수로서 각각 소스 메타데이타를 XML 문서로 변환하는 기능과, XML문서의 메타데이타를 타겟 데이타베이스에 이식시킬 수 있는 형태로 변환하는 기능을 포함한다.

3.2 R-MDIS(MetaData Interchange Specification for RDBs) 모델

우리가 제안한 메타데이타 교환 사양인 R-MDIS에는 메타데이타 일관성을 유지하는 반면 메타데이타 양방향 흐름을 지원하기 위하여 세 가지 파일을 정의하고 각각 세 가지 타입의 정보를 포함하고 있다. 즉, 메타데이타의 정상적인 흐름에 대한 정보를 위한 구성 프로파일과 지원할 수 있는 객체에 대한 정보를 포함하고 있는 툴 프로파일, R-MDIS에서 지정하는 사양에 맞게 교환 파일을 구성하기 위하여 교환 파일의 헤더 내용을 정의하는 헤더 파일을 갖는다. 이들 파일은 각각 Export와 Import 함수에 의해 생성되고 소비되는 파일이다. 이 세 개의 파일을 구문 구조로 표현하면 아래와 같은 문법 사양을 가진다.

Filespec1::='<CONFIGURATION>'configspec'</CONFIGURATION>'

Filespec2::='<TOOL>' toolspec '</TOOL>'

Filespec3::='<HEADER>' headerspec '</HEADER>' *objects**

이 때 Filespec1은 구성 프로파일을 의미하고, Filespec2는 툴 프로파일이며, Filespec3는 교환 파일이다. 따라서 이 세 가지 파일의 구문 구조는 다음과 같이 정의된다.

첫째, configspec은 메타데이타의 정상적인 흐름을 나타내는 구성 프로파일로서 하나의 XML 기반 파일로 만들어져야 한다.

configspec ::=
'<TargetToolName>'#PCDATA'</TargetToolName>'
'<TargetToolVersion>'#PCDATA'</TargetToolVersion>'
'<TargetToolInstance>'#PCDATA'</TargetToolInstance>'
'<SourceToolName>'#PCDATA'</SourceToolName>'
'<SourceToolVersion>'#PCDATA'</SourceToolVersion>'
'<SourceToolInstance>'#PCDATA'</SourceToolInstance>'
'<R-MDISVersion>'#PCDATA'</R-MDISVersion>'
'<Objects>'#PCDATA'</Objects>'

둘째, toolspec은 직접적으로 표현하거나 업데이트하는 데이타 엘리먼트 타입이 무엇인지 나타내는 툴에 대한 프로파일로서 툴의 이름과 버전 정보 등을 담고 있으며 데이타베이스, 서브스키마, 차원, 레코드, 관계, 레벨, 뷰 등을 지원하는지에 대한 정보와 Import하고 Export하는 파일에 대한 정보를 포함하고 구성프로파일처럼 표현한다.

셋째, headerspec은 교환 파일의 가장 상위부분에 나타나는 헤더에 대한 사양으로서 Export되는 메타데이타를 표현하는 XML문서에서 어떤 언어로 표현하는지, 또한 교환파일이 생성되는 날짜와 시간 등을 표현한다. 그리고 objects*은 본 연구에서 대상으로 하고 있는 메타데이타에 대한 요소들로서 0번 이상 나타나는 것을 뜻한다. 이러한 객체는 카달로그 정보, 테이블 정보, 컬럼 정보, 기본키 정보, 외부키 정보와 인덱스 정보 등을 포함한다. 이들 요소 각각에 대한 세부적인 메타데이타에 대해서는 구현부분에서 설명한다.

objects::='<CATALOG>'catalogelem'</CATALOG>'|
'<TABLE>'tableelem'</TABLE>'
'<TPREVILEGE>'tprivilegeelem'</TPREVILEGE>'|'<RECORD>'recordelem'</RECORD>'|'<COLUMN>'columnelem'</COLUMN>'
'<CPREVILEGE>'cprivilegeelem'</CPREVILEGE>'
'<PRIMARYKEY>'primarykeyelem'</PRIMARYKEY>'
'<FORIEGNKEY>'foriegnkeyelem'</FORIEGNKEY>'
'<RELATIONSHIP>'relationshipelem'</RELATIONSHIP>'|'<INDEX>'indexelem'</INDEX>'

4. 메타데이타 교환을 위한 XML기반 번역기 설계 및 구현

4.1 메타데이타 교환을 위한 번역기 설계

본 연구에서 설계한 번역기는 하나의 데이타베이스로부터 메타데이타를 추출하여 이것을 메타데이타 교환을 위한 형태의 XML문서로 출력하고 이 생성된 XML문서를 파싱하여 어떠한 관계형 데이타베이스에도 이식될 수 있는 형태로 변환(translate)하여 타겟 데이타베이스로 Import 시킨다.

4.1.1 XML 문서 생성을 위한 Export 함수

소스 데이타베이스로부터 메타데이타를 추출하기 위한 함수로서 export될 객체 타입, export되는 객체 인스턴스, XML문서(교환 파일)로 출력되는 파일의 패스이름, 메타데이타를 export하는 툴을 정의하는 ToolInstanceID를 입력 매개변수로 한다. Export 프로그램을 위한 알고리즘은 다음과 같다.

main(interchange_file,object_type,object_instance)
/* Export_UserTable_Index program */
{
1. Parse input parameter;
2. Read configuration profile;
3. **if**(object_type objects of configuration profile)
4. exit(1);
5. Read private metadata dictionary for request object_instance;
6. **if** (object_instance not found)
7. exit(1);
8. Open R-MDIS file;
9. **if** (open error)
10. exit(1);
11. Format and write R-MDIS header information; //R-MDIS구문에 따라 기록
12. **if** (write error)
13. exit(1);
14. Format and write metadata of requested object with tag ;//R-MDIS구문 참조
15. **if** (write error)
16. exit(1);
17. Format and write metadata of relationship objects;// R-MDIS 구문 참조
18. **if** (write error)
19. exit(1);
20. Close interchange file;
21. exit(0);
22.} /* Export_UserTable_Index program end*/

4.1.2 XML 문서 파싱과 변환을 위한 Import 함수

XML 문서를 파싱하여 대상 데이타베이스에서 메타데이타를 Import할 수 있도록 변환하는 함수를 수행하기 위해 Import되는 객체 타입의 이름, 교환 파일(XML 파일)에 포함된 객체의 인스턴스 ID, Import될 메타데이타를 포함하고 있는 파일의 패스이름를 입력 파라미터로 취한다. Import 프로그램을 위한 알고리즘은 다음과 같다.

main(interchange_file,object_type,object_instance)
/* Import_UserTable_Index program */
{
1. Parse input parameter;
2. Read configuration profile;
3. **if** (object_type or object_instance object of configuration profile)
4. exit(1);
5. Read interchange file header;
6. **if** (file not found)
7. exit(1);

```
8. switch {
9. case invalid or unknown tag: exit(1);
10.case invalid source tool generated file: exit(1);
11 .case invalid or not supported character set:
        exit(1);
12. case invalid or not supported R-MDIS version:
        exit(1);
13. }
14. Read private tool metadata dictionary;
15. if(date of source>date of interchange file)
16.     exit(1);
17. Read and parsing interchange file
18. if  (error parsing)
19.    exit(1);
20. /* Import할 데이타베이스의 메타데이타 형식으로 변환
    한다 */
21. Translate metadata of object in interchange file
    21-1.{Search the metadata including the identity
    property;
21-2. Analyze the data type of all metadata;
21-3. Analyze the constraint property;
21-4. Search the metadata including primary key
    property;
21-5. Search the metadata including foreign key
    property;
21-6. Convert the searched metadata into string;
21-7. Concatenate the strings; }
22. Store translated string into array
23. Close interchange file;
24.    exit(0);
25. }
```

4.2 구현

메타데이타 교환을 위한 XML기반의 번역기는 Windows NT 4.0 환경에서 JDK1.2.1버전으로 구현하였다. 메타데이타 교환을 위한 사양을 정의한 XML문서 버전은 1.0 형식에 맞도록 설정하였고 문서를 파싱하는 파서는 IBM에서 제공하는 xml4j2.0.15내에 포함된 SAXDriver를 사용하였다[6]. 대상으로 하고 있는 DBMS는 MSSQL 7.0과 Oracle 8.15의 메타데이타를 추출하였다.

표준 SQL은 데이타 구조와 그에 대한 메타데이타를 DDL을 사용하여 생성하기 위해 데이타베이스에 있는 테이블이나 뷰, 그 외의 요소들을 생성할 수 있다. 그러나 그 구조에 대한 정보가 되는 메타데이타는 DBMS에 어떻게 저장해야 할 지에 대해서는 기술하지 않는다. 각 벤더마다 유사한 정보를 담고 있긴 하지만 그 구문이나 구조가 다른 것이 메타데이타의 불일치 문제이다. 그런데 보통의 JDBC는 가장 일반적인 메타데이타만을 다루고 있어 추가적인 메타데이타를 위해서는 별도의 API를 사용하여야만 원하는 메타데이타 정보를 검색할 수 있다[13]. 따라서 본 논문에서는 자바소프트에서 별도로 제공하고 있는 DatabaseMetaData[2,5] 인터페이스를 사용하여 데이타베이스의 메타데이타 정보를 검색한다. DatabaseMetaData 객체는 다양한 메소드의 활용을 위해서 Connection객체로부터 반환되고 Connection 객체는 이를 수행하기 위해서 getMetaData() 메소드를 사용한다.

4.2.1 사용자 테이블에 대한 메타데이타

사용자가 작성한 데이타베이스 테이블을 대상으로 추출해야 하는 메타데이타는 Java API에서 지원하는 메타데이타이기 때문에 관계형 데이타베이스이면 동일한 사항이다. 그 내용으로는 테이블에 대한 접근 권한 정보로서 테이블에 접근 허가 명령을 내리는 자(Grantor), 테이블에 접근 허락을 받은 자(Grantee), 허가된 접근의 유형(privilege), 허가 여부(Is_Grantable) 등이 있다.

```
while(notdone) {
 notdone=rs.next();
   if(notdone) {
      ...
      cpi.setColumn_Name(rs.getString(4));
      cpi.setGrantor(rs.getString(5));
      cpi.setGrantee(rs.getString(6));
      cpi.setPrivilege(rs.getString(7));
      cpi.setIs_Grantable(rs.getString(8));
      CPI_List.addElement(cpi);
      ...
      }//if
} //while
```

컬럼 정보를 위한 메타데이타는 컬럼 이름, 데이타 타입, 타입 이름, 컬럼 크기, 소수점 이하 자리수, 기수, Null값 허용 여부, 컬럼을 설명해 주는 주석, 컬럼의 디폴트 값, 컬럼의 최대 바이트 수, 테이블 내에서의 컬럼의 위치, Null 허용 값 등이 있다.

```
while(notdone) {
 notdone=rs.next();
 if(notdone) {
Column_Infos ci=new Column_Infos();
   ...
ci.setColumn_Name(rs.getString(4));
ci.setData_Type(rs.getShort(5));
ci.setType_Name(rs.getString(6));
ci.setColumn_Size(rs.getInt(7));
ci.setDecimal_Digits(rs.getInt(9));
```

```
ci.setNum_Prec_Radix(rs.getInt(10));
ci.setNullable(rs.getInt(11));
ci.setRemarks(rs.getString(12));
ci.setColumn_Def(rs.getString(13));
ci.setSQL_Data_Type(rs.getInt(14));
ci.setSQL_DateTime_Sub(rs.getInt(15));
ci.setChar_Octet_Length(rs.getInt(16));
ci.setOrdinal_Position(rs.getInt(17));
ci.setIs_Nullable(rs.getString(18));
System.out.println(ci);
 } //if
}//while
```

이 외에도 기본키(primary key)를 위한 키 이름, 키의 일련 번호 등의 메타데이타와 외부키(foreign key)를 위한 외부키 이름, 외부키의 일련 번호, 외부키로 지정된 컬럼명, 연결되어 있는 외부 테이블명 등에 대한 메타데이타를 추출할 수 있다. 이와 같이 추출한 메타데이타들을 다른 데이타베이스로의 이식을 위해 중간 형태의 XML 문서로 변환하고 Import 함수에서는 이 XML 문서를 파싱하여 여기에 나타나는 메타데이타들을 하나의 긴 문자열로 변환한다. 이 문자열은 CREATE TABLE 명령에 의해 타겟 데이타베이스에 테이블을 생성할 수 있도록 다음과 같은 과정을 통해 변환한다.

①XML문서에 나타나는 추출된 컬럼의 이름을 모두 배열에 저장한다.

②XML 문서에서 텍스트가 "int identity"인 것을 찾아 만일 존재한다면 그 때의 컬럼에 해당하는 컬럼 이름에 identity속성을 지정하여 변환한다.

③데이타 타입 정보에 따라 Import될 DB타입에 맞도록 변환하기 위하여 태그가 Type_Name이면서 크기가 고정되어 있는 날짜 타입이나, 정수, 실수, 이미지, 비트 타입 등을 골라 알맞은 형태로 변환한다. 또한 문자 타입과 같이 크기를 지정할 수 있는 타입은 별도로 분리하고 크기 정보도 추출하여 문자 타입과 크기를 연결해 준다.

④각 컬럼의 속성이 Nullable인지 아닌지를 체크해 YES이면 그 컬럼 이름에 NULL을 지정하고 NO이면 NOT NULL을 지정하여 import할 때 그 컬럼 정보로서 사용한다.

⑤기본 키 속성을 가진 컬럼이 있는지 찾아 그 컬럼에 PRIMARY KEY를 지정하여 import할 때 그 컬럼의 정보로서 사용하도록 한다.

⑥외래 키 속성을 지닌 컬럼을 찾아 그 해당 컬럼에 FOREIGN KEY REFERENCES를 지정하고 그 컬럼이 참조하고 있는 테이블 이름을 찾아 참조하는 테이블로 설정한다.

이렇게 생성된 문자열을 executeUpdate하면 타겟 데이타베이스 테이블의 스키마를 형성한다.

4.2.2 인덱스를 위한 메타데이타

JDBC 버전에 따라 인덱스가 5개를 초과하면 실행시에 오류가 걸린다. 따라서 인덱스의 수가 5개가 넘지 않는 데이타베이스 테이블을 대상으로 인덱스에 대한 메타데이타를 추출하고 이것에 대한 XML 문서를 생성한다. 인덱스에 대한 메타데이타를 추출하기 위하여 인덱스를 위한 번역기의 입력자료로서 객체타입, 객체 인스턴스, XML 파일명, ToolInstanceID 등을 입력한다. 테이블을 위한 인덱스들은 getIndexInfo() 메소드를 통해 알아낼 수 있다. getIndexInfo() 메소드를 위한 인수들과 반환되는 결과 타입은 다음과 같다.

public abstract ResultSet getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate) throws SQLException

여기서 사용되는 인수 중 boolean 타입의 unique는 유일한 식별자인지 아닌지를 가리는 것이고, approximate는 근사값을 허용하는지에 대한 것을 나타내는 boolean 타입의 인자이다. 또한 getIndexInfo()의 리턴되는 결과값은 인덱스의 중복 여부를 나타내는 NON_ UNIQUE, 인덱스 카탈로그인 INDEX_QUALIFIER, 인덱스 이름을 뜻하는 INDEX_NAME, 인덱스 유형의 TYPE, 인덱스 내에서의 컬럼 일련 번호인 ORDINAL_ POSITION, 인덱스되는 컬럼 이름의 COLUMN_ NAME, 컬럼 정렬 순서인 ASC_OR_DESC, CARDINALITY, PAGES, 그리고 FILTER_CONDITION 등이고, 이러한 메타데이타를 사용자 테이블에서 행한 것과 같은 방법으로 추출하여 XML 문서로 표현한다. 이렇게 표현된 XML 형식의 메타데이타를 파싱하고 이를 문자열로 변환하여 CREATE INDEX를 실행시키면 대상 데이타베이스로 이식된다.

4.3 구현 결과

본 연구에서 대상으로 하고 있는 메타데이타는 크게 사용자테이블에 대한 테이블 정보나 컬럼 정보, 접근 권한 정보, 컬럼에 대한 제한(constraint), 기본키나 외부키 등에 대한 메타데이타와 인덱스를 위한 메타데이타를 다루고 있다. 이와 같은 메타데이타는 자바에서 별도의 API를 제공하여 DatabaseMetaData라는 객체를 통해 어떠한 관계형 데이타베이스끼리도 교환이 가능하지만 저장 프로시저(Stored Procedure)와 같은 메타데이타 경우 현재까지 지원되고 있는 자바 어플리케이션에서의 메타데이타는 제한적이어서 완전한 CREATE PROCEDURE문장을 만들어 내기가 어렵다.

4.3.1 메타데이타를 위한 Export와 Import함수 처리 과정 및 결과 예

하나의 관계형 데이타베이스에서 사용자가 생성한 테이블에 대한 메타데이타를 Export하여 XML 문서를 만든 경우 그림 2와 같은 형태의 .xml파일이 생성된다. 그리고 이 문서를 파싱하여 그림 3과 같은 형태의 문자열로 변환한다.

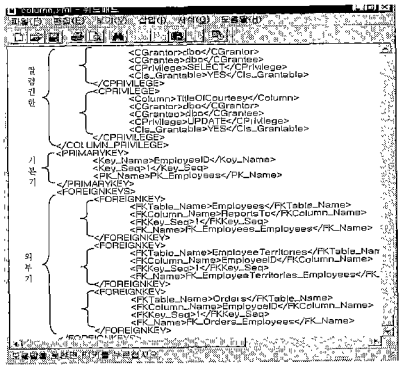
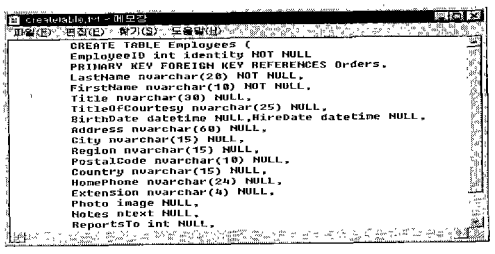


그림 2 Export수행결과



```
CREATE TABLE Employees (
EmployeeID int identity NOT NULL
PRIMARY KEY FOREIGN KEY REFERENCES Orders.
LastName nvarchar(20) NOT NULL,
FirstName nvarchar(10) NOT NULL,
Title nvarchar(30) NULL,
TitleOfCourtesy nvarchar(25) NULL,
BirthDate datetime NULL,Hiredate datetime NULL,
Address nvarchar(60) NULL,
City nvarchar(15) NULL,
Region nvarchar(15) NULL,
PostalCode nvarchar(10) NULL,
Country nvarchar(15) NULL,
HomePhone nvarchar(24) NULL,
Extension nvarchar(4) NULL,
Photo image NULL,
Notes ntext NULL,
ReportsTo int NULL,
```

그림 3 Import 함수 처리 과정 중 메타데이타를 문자열로 변환한 결과

변환된 문자열에 대하여 자바 명령어의 하나인 executeUpdate를 수행하여 다른 관계형 데이타베이스로 Import시킨다. 결과는 그림 4와 같이 원래 데이타베이스 테이블에 대한 스키마가 추출되어 옮겨간다. 그림 4는 원래 데이타베이스의 스키마이다.

```
SQL> desc Employee
 이름                                      널?      유형
 ----------------------------------------- -------- ------------
 EMPLOYEEID                                NOT NULL NUMBER(3)
 LASTNAME                                           VARCHAR2(20)
 FIRSTNAME                                          VARCHAR2(10)
 TITLE                                              VARCHAR2(30)
 TITLEOFCOURTESY                                    VARCHAR2(30)
 BIRTHDATE                                          DATE
 HIREDATE                                           DATE
```

그림 4 오라클에서의 원래 스키마

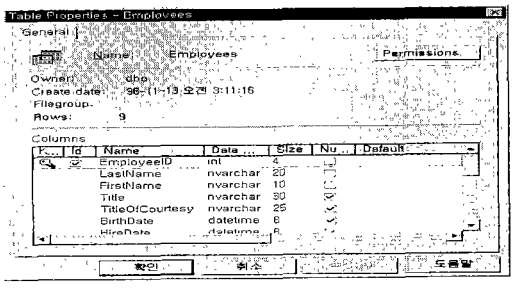


그림 4 MS-SQL로 메타데이타를 옮긴 후의 결과

4.3.2 실제 데이타의 이주

소스로부터 타겟으로 데이타를 이주시키는 것은 메타데이타를 옮기고 난 후, SELECT 명령으로 소스데이타를 Export하고 INSERT INTO 명령으로 Import 시킨다. 반드시 모든 데이타 타입을 다 수용할 수 있는 JDBC를 사용해야 하고 다음은 실제 데이타가 이동된 한 예이다.

```
SQL> select * from Employee
  2  ;

EMPLOYEEID LASTNAME             FIRSTNAME  TITLE
---------- -------------------- ---------- ------------------------------
TITLEOFCOURTESY           BIRTHDAT HIREDATE
------------------------- -------- --------
         1 Davolio              Nancy      Sales Representation
Ms.                       48/12/08 92/03/01

         2 Fuller               Andrew     Vice President
Dr.                       52/02/19 92/08/14

         5 Buchnan              Steven     Sales Representation
Mr.                       67/01/02 92/10/17
```

그림 5 오라클에서의 원래 데이타

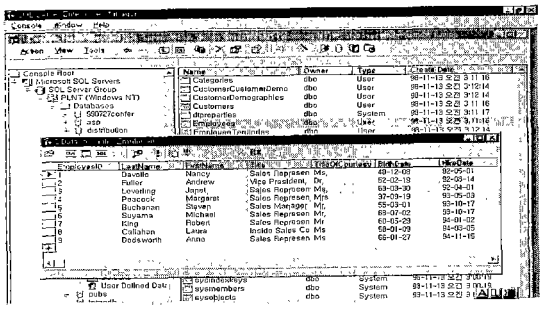


그림 6 MS-SQL로 데이타가 이주된 결과

5. 결 론

본 연구에서는 XML이 웹 데이타와 같이 복잡한 타입의 데이타를 쉽게 표현할 수 있고, 구조화된 데이타 및 구조화되지 않은 문서에 대해 중간자적이고 중립적이며 표준화된 형태로 표현할 수 있음을 보여 주었다.

또한 본 논문에서는 관계형 데이타베이스 시스템들 사이의 메타데이타 교환을 위하여, XML 기반으로 이에 대한 사양을 정의하는 모델로서 R-MDIS를 설정하고 이를 따르는 사양에 따라 사용자 테이블에 대한 메타데이타, 인덱스에 대한 메타데이타를 대상으로 이를 교환하는 번역기를 개발하였다.

본 연구 결과의 의의는 첫째, 여러 가지 종류의 구조화된 데이타를 처리하는 시스템들에서 메타데이타 불일치의 문제 해결 방법을 제시하였다. 둘째, XML을 이용하여 메타데이타 교환 사양에 대한 모델을 제시함으로서 구조화된 데이타나 구조화되지 않은 데이타 모두에 대하여 일관되고 통합된 방법으로 데이타 교환이 가능하도록 하여 효율적인 정보 교환 및 통합의 방법을 보여주었다. 셋째, 본 연구에서 개발한 번역기는 메타데이타 교환에 의해 데이타들을 공유하게 함으로서 효율적으로 관리하지 못하던 많은 데이타들을 효율적으로 사용할 수 있게 함으로서 데이타 관리의 문제에 해결책이 될 것이다.

참 고 문 헌

[1] Dublin Core Metadata Initiative, http://purl.org/DC/index.htm

[2] http://java.sun.com/products//jdk/1.3/docs/api/java/sql/DatabaseMetaData.html

[3] Clive Finkelstein & Peter Aiken, *Building Corporate Portals with XML*, McGraw Hill, 1999

[4] Steven Holzner, *XML Complete*, McGraw Hill, 1999

[5] Brian Jepson, *Java Database Programming*, Wiley Computer Publishing, 1997

[6] Hiroshi Maruyama, Kent Tamura and Naohiko Uramoto, *XML and Java Developing web application*, Addison- Wesley, 1999

[7] Meta Data Coalition, Membership Directory, http://mdcinfo.com/ members.html

[8] Meta Data Coalition, Metadata Interchange Specification (MDIS Version 1.0), http://mdcinfo.com/MDIS/MD IS10.html

[9] Meta Data Coalition, The Open Information Model, http://www.mdcinfo.com/OIM

[10] William J.Pardi, *XML in Action*, Microsoft, 1999

[11] W3C, Platform for Internet Content Selection (PICS), http://www.w3.org/PICS, 1997.12

[12] W3C, Resource Description Framework (RDF), http://www.w3.org/RDF, 1999.8

[13] Jeff Schneider and Rajeev Arora, *Using Enterprise JAVA*, Que, 1998

[14] Dan Suciu, Semistructured Data and XML, *In Proceedings of International Conference on Foundations of Data Organization*, 1998

[15] 홍승길, 방대한 DB 길잡이 메타데이터, http://www.dpc. or.kr/dbworld/document/ 9903/gigo.html

[16] Computer Wire: Data Warehousing Tools Bulletin: Briefing Paper: What is Metadata, http://www.computer wirecom/bulletinsuk/212e_1a6.htm

[17] W3C, Extensible Markup Language (XML), 1998, http:// www.w3.org/XML/

[18] OMG XMI V.1.1, available at http://cgi.omg.org/cgi-bin/doc?ad/99-10-04

이 월 영

1988년 이화여자대학교 전자계산학과(학사). 2000년 이화여자대학교 컴퓨터학과(석사). 2000년 ~ 현재 이화여자대학교 컴퓨터학과 박사과정. 관심분야는 XML, 정보검색, 데이타마이닝, 전자상거래, 데이타베이스

이 기 호

1961년 이화여자대학교 수학과(이학사, 석사). 1968년 ~ 1972년 텍사스 주립대(전산학 석사, 박사 수료). 1981년 서울대학교 대학원(전산학 박사). 1987년 이화여자대학교 전산연구소장. 1988년 캘리포니아 대학교 연구교수. 1973년 ~ 현재 이화여자대학교 컴퓨터학과 교수. 관심분야는 프로그래밍 언어론, 컴파일러, S/W 공학, ICAI 등임.