제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

CWM 기반의 메타데이터 레파지토리 설계 및 구현

백운집*, 임정은**
*고려대학교 대학원 컴퓨터학과
e-mail : wjbaik@naver.com*, jelim@software.korea.ac.kr**

CWM Based Metadata Repository Design and Implementation

Woon-jib Baik*, Jung-eun Lim**
*Dept. of Computer Science, Korea University

요 약

데이터 웨어하우스가 발전함에 따라 통합된 메타데이터는 구현뿐만 아니라 활용측면에서도 중요성이 부각되면서 이제는 전략적인 비즈니스 자산으로 여겨지고 있다. 이러한 메타데이터 표준으로 OMG(Object Management Group)에서는 CWM(Common Warehouse MetaModel)을 웨어하우스와 BI(Business intelligence)의 표준으로 채택하였다. 그러나 소프트웨어 개발 업체들간의 메타데이터 상호교환 중심으로 구현됨으로써 CWM 을 사용한 메타데이터의 활용 및 저변확대가 안되고 있다. 이러한 문제점을 개선하기 위해서 CWM 을 기반으로 한 레파지토리(Repository)를 설계 및 구현함으로써 CWM 로 생성된 메타데이터를 저장, 보관하여 비즈니스적인 활용이 가능하도록 하였다. 또한 이러한 연구를 통하여 데이터 웨어하우스 분야에서도 MDA(Model Driven Architecture)기반의 설계 및 구현이 될 것으로 전망된다.

1. 서론

최근 많은 분야에서 데이터 웨어하우스가 구축되고 활용되면서 표준화된 메타데이터(Metadata)의 중요성이 부각되고 있다. 기업 내에 산재되어 있는 각종 데이터를 전사적인 데이터 저장소 내에 통합하여 관리하고자 하는 필요에 의해서 데이터 웨어하우스가 시작되었고 이러한 정보에 대한 정보를 관리하는 것이 메타데이터이다[1]. 이러한 메타데이터를 통일되게 저장하고 데이터의 지도 역할을 수행하는 것이 메타데이터 레파지토리이다.

데이터 웨어하우스 내의 메타데이터 사용자는 최종사용자, IT 사용자(구축자, 관리자) 등으로 나눌 수 있다[2]. 데이터 웨어하우스가 발전함에 따라 최종사용자의 비정형 질의가 급격히 증가하게 되였고 이를 효율적으로 수행하기 위한 메타데이터의 중요성이 부각되고 있을 뿐만 아니라, 메타데이터 자체가 전략적인 비즈니스 자산으로 여겨지고 있다. IT 분야에서도 데이터 웨어하우스와 관련된 다양한 제품이 등장하면서 각 제품별로 존재하는 메타데이터 레파지토리를 통합하여 연관성 및 통일성을 높이고, 서로 다른 제품간의 메타데이터 중복을 최소화하고, 유지보수 효율성과 확장을 높이는데 메타데이터의 중요성이 강조되고 있다. 레파지토리는 이러한 비즈니스 메타데이터와 기술적 메타데이터를 통합 관리하여 ISC(Information supply chain) [3]을 구현하는 중심 역할을 수행한다.

그간 메타데이터 표준화에 대한 많은 노력이 있었고, 분산환경 하에 웨어하우스의 표준 메타데이터 모델로서 CWM(Common Warehouse MetaModel)이 자리잡게 되었다. 그러나 다양한 요구사항을 만족시키는 통일된 형태의 메타데이터 레파지토리를 처음부터 구축하는 것은 많은 어려움이 따르기 때문에 지금까지는 개발된 제품간의 메타데이터 교환이 가능한 형태로 구현되어 왔다. 그러나 메타데이터의 일대일 교환 형태만으로는 메타데이터를 전략적으로 이용하려는 비즈니스적인 요구사항을 만족시킬 수 없다. 그러므로 제안한 CWM 기반의 레파지토리를 통하여 표준화된 메타데이터로 접근 및 분석이 용이하도록 하고 중앙

집중의 효율적인 메타데이터의 상호교환이 가능하도록 하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 데이터 웨어하우스 메타데이터에 대한 표준화된 내용 및 CWM 의 구성을 살펴본다. 3 장은 CWM 기반의 레파지토리의 모델 및 설계내용을 설명한다. 4 장은 3 장에서 제안된 레파지토리를 구현하여 메타데이터를 저장, 보관, 배포 및 조회할 수 있는 모듈들을 보여준다. 마지막으로 5 장에서는 본 논문을 정리하고 향후 연구방향을 제시한다.

2. 관련 표준 메타데이터 및 레파지토리 연구

CWM 은 1997 년 약 800 여 개 관련 업체들의 컨소시엄으로 구성한 OMG(Object Management Group)에 의해서 주창 되었다. 다른 메타데이터 모델 표준으로 존재하던 MDC(Meta Data Coalition)의 OIM(Open Information Model)이 2000 년 9 월 CWM 에 병합됨으로써 CWM 이 완벽한 데이터 웨어하우스 분야의 표준 모델로 자리잡게 되었다. [4]

CWM 은 OMG 의 3 가지의 표준사항 인 UML (Unified Modeling Language), MOF (Meta Object Facility), XMI (XML Metadata Interchange)을 기반으로 구성되어 있다. UML 은 객체 모델을 설계하고 표현하기 위한 표준 모델로써, CWM 은 이러한 UML 의 객체 모델 개념을 포함하고 있으며, CWM 의 모델도 UML 의 다이어그램으로 표현하고 있다. 또한 메타모델 을 정의하기 위한 사양인 MOF 에 근거하여 CWM 메타모델이 구성되어 있다. MOF 는 메타모델을 정의하기 위한 메타모델의 메타모델로서 메타모델을 작성하기 위한 최상위 단계의 모델이며, 이에 근거하여 UML, CWM 등의 표준 메타모델이 만들어졌다. 세 번째로는 CWM 의 핵심인 메타데이터의 상호교환을 위해서는 XML 기반의 XMI 를 통하여 메타데이터의 상호교환을 수행한다. XMI 는 MOF 의 기반으로 만들어진 메타모델 혹은 UML 이나 CWM 등의 메타모델에 의해 생성된 메타데이터를 교환하기 위한 표준 형태를 정의한다. XMI 는 XML 을 기반으로 구성되어 있으므로 일반 XML 파서를 사용한 어플리케이션이나 API 의 개발이 가능하다.

CWM 은 전체기능을 5 개의 계층으로 분류하고 총 21 개의 기능을 가진 패키지로 구성되어 있다. [5]

Object Model 계층의 경우 MOF 을 기반으로 하여 객체 지향 기술과 접목하기 위한 것으로 상위의 다른 CWM 패키지들이 사용하기 위한 기본 계층이다. Foundation 계층은 상위계층의 패키지들이 공통으로 사용할 서비스를 정의한 것으로, CWM 뿐만이 아닌 일반적인 목적으로 설계된 패키지이다. Resource 계층은 CWM 의 물리적 대상을 기술한 것으로 파일, 관계형 데이터베이스, XML 등 데이터 웨어하우스의 물리적인 구성요소를 기술한다. Analysis 패키지는 Resource 계층에 정의되어 있는 데이터들에 대한 기능과 서비스를 기술하는 계층이다. 데이터 웨어하우스 내에 사용될 논리적인 비즈니스 개념을 정의하는 부분으로 최종사용자에게 유용한 정보를 제공한다. Management 계층은 데이터 웨어하우스의 관리와 운영을 위한 환경과 프로세스등을 기술하는 계층이다.[6]

Management	Warehouse Process			Warehouse Operation		
Analysis	Transformation	OLAP	Data Mining	Information Visualization	Business Nomenclature	
Resource	Object	Relational	Record	Multidimensional	XML	
Foundation	Business Information	Data Types	Expressions	Key and Indexes	Software Deployment	Type Mapping
Object Model	Core	Behavioral	Relationships	Instance		

< 표 1 > CWM 계층별 Package 분류

CWM 은 객체 지향 기술을 기반으로 구성되어 있으므로 상속을 기반으로 한 메타데이터의 확장 및 재사용을 가능하게 한다. 이러한 기능을 통하여 각 제품에 맡는 특별한 속성을 표현할 수 있으며 모델 자체를 확장할 수 있다. 상속 이외에도 UML 의 스테레오타입이나 꼬리 값(TaggedValues)을 사용하여 쉽게 확장할 수 있다.

CWM 은 공통된 표준모델을 통한 서로 다른 시스템간의 메타데이터 교환에 초점을 맞추고, 패턴기반의 접근을 통하여 CWM 의 범용적인 사용이 가능하도록 하고 있다. 또한 메타데이터 전송 아키텍쳐로 Point-to-point 교환에서 레파지토리 기반의 상호교환형태까지 다양한 형태의 구현이 가능하도록 지원하고 있다.

CWM 는 아직 데이터 웨어하우스 관련 소프트웨어 개발 업체간에 메타데이터 교환을 위한 업체표준형태로만 사용되는 수준이지만, CWM 이 발전하면 이를 기반으로 한 많은 응용 기술들이 나올 것으로 예상된다.

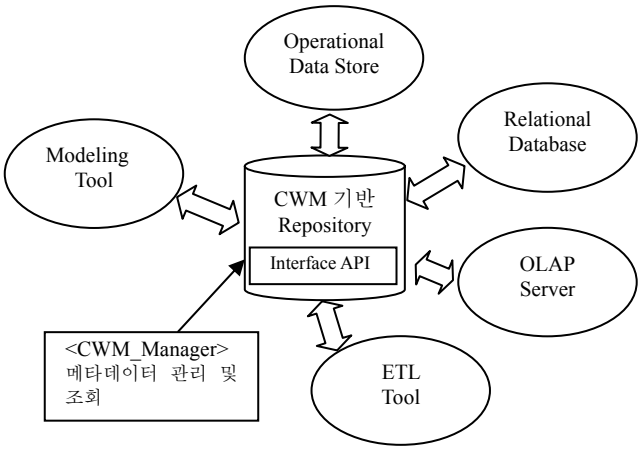
국내에서는 아직 데이터 웨어하우스 관련 프로젝트에 CWM 을 적용한 사례는 없으나, CWM 이 데이터 웨어하우스와 관계된 영역의 모든 프로세스를 기술하고 있으므로, 이를 메타데이터 관리 및 데이터웨어하우스의 개발방법에 적용을 한다면 MDA 기반의 표준화된 데이터 웨어하우스 프로젝트가 가능할 것으로 예상된다.

3. CWM 기반의 메타데이터 레파지토리 설계

CWM 이 메타데이터 교환에 초점을 맞춤으로써 메타데이터의 비즈니스적인 응용측면에서는 미흡한 것이 사실이다. 본 연구에서는 이러한 측면을 보완하기 위해 레파지토리 기반의 아키텍쳐를 설계하였다. 도구들 사이의 메타데이터 교환을 위해서는 중앙집중 방식을 사용하는 것이 상호 연결성이 뛰어나며 메타데이터의 생명주기를 관리할 수 있으므로 이에 초점을 맞추었다. 또한 레파지토리를 통하여 최종 사용자가 데이터 웨어하우스 사용 시 비정형분석을 위한 메타데이터 조회를 원활히 수행할 수 있고 데이터 웨어하우스 전체 프로세스를 파악할 수 있도록 하였으며, 메

타데이터의 관리가 용이하고 수정된 내용에 대해 시스템간의 상호 동기화가 가능하도록 하였다.

특히 데이터 웨어하우스 분야에서 모델 주도형 구조 (MDA) 구현을 지원하기 위해서 표준 메타모델인 CWM 의 형태를 최대한 유지하고, 표준 인터페이스를 통한 상호운영성과 플랫폼 독립적인 시스템을 구축할 수 있도록 레파지토리를 설계하였다.



(그림 1) CWM 기반의 레파지토리 아키텍쳐

본 논문에서 제안한 CWM 레파지토리 시스템은 크게 4 개 부문으로 구성된다. CWM 를 정의하는 XMI 를 사용하여 레파지토리를 생성하는 CWM_Creator 모듈과 인터페이스 API 로 CWM 표준에 따라 생성된 XMI 파일을 레파지토리 내로 가져오는 CWM_Importer 모듈, 레파지토리 내의 내용을 XML 파일로 내보내는 CWM_Exporter 모듈, 레파지토리 내의 메타데이터를 조회 및 관리하는 CWM_Manager 모듈 등 4 가지 모듈로 구성된다.

CWM_Exporter 를 통하여 모델링 도구에서 OLAP 도구까지 생성된 CWM 기반의 XMI 파일을 XML 파서를 통하여 레파지토리 내에 저장할 수 있으며 CWM_Importer 를 통하여 레파지토리 내의 정보를 XMI 파일로 생성하여 다른 도구들이 생성된 XMI 파일을 전송 받아 사용할 수 있도록 하였다.[7]

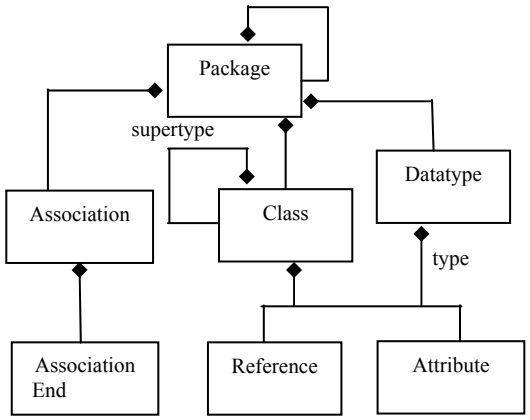
그리고 CWM_Manager 를 통하여 최종 사용자가 웹을 통하여 메타데이터 검색, 조회 및 관리가 가능하도록 하였으며, 레파지토리가 가져야 할 생명주기 관리, 동시 제어, 접근 권한 관리, 버전 관리 등의 기능이 포함되도록 하기 위하여 레파지토리 내에 자체 관리 기능을 추가하였다.

4. 시스템 구현

CWM 기반의 레파지토리 저장소로서 상용의 관계형 데이터베이스(RDBMS)를 선택하였다. CWM 이 객체지향 기반으로 구성되었으므로 객체지향 데이터베이스를 사용하는 것이 저장 및 보관에 용이할 수 있으나, 상용화되지 못하여 메타데이터 사용에 어려움이 많아 활용성이 떨어지고 비즈니스적 요구사항을 만족시키기 어려움으로 관계형 데이터베이스를 저장소로 선택하였다. 상용화된 데이터베이스를 사용함으로써 SQL(Structured Query Language) 기반의 비정형 질의 도구들을 사용할 수 있으며, 메타데이터를 응용한 프로그램 개발 등을 통하여 메타데이터를 사용하기에 용이하다. 또한 검증된 관계형 데이터베이스 기술로 메타데이터 관리 및 유지가 원활하다.

기본적으로는 객체 개념의 CWM 을 관계형 데이터베이스로 변환하기 위해 O-R(Object-Relation) 매핑 개념을 사용하였다. 그러나 UML 로 표현된 다이어그램을 O-R 매핑 방식으로 적용하는 기존의 방법으로 레파지토리를 구성할 경우 CWM 모델이 변경되거나 확장될 때 레파지토리를 일일이 수작업으로 재구성해야 하므로 확장성이 떨어진다. 그래서 본 연구에서는 CWM 을 정의하는 XMI 파일을 기반으로 하여 관계형 데이터베이스의 데이터정의어(Data Definition Language)를 생성하여 레파지토리를 구축하였다. 레파지토리 생성 시 CWM 을 정의하는 XMI 파일을 사용함으로써 CWM 이 변경되거나 확장될 경우, 변경내용을 CWM 정의 XMI 파일에 반영함으로써 레파지토리의 변경이 가능하도록 함으로써 레파지토리의 확장성을 높였다.

그림 2 는 CWM XMI 파일의 관계를 UML 다이어그램으로 표현하였고, 표 2 는 XML 파일 내 요소(Element)들이 관계형 데이터베이스의 어떤 객체와 연관되는지를 나타내고 있다.



(그림 2) CWM 정의 XMI Element 간의 관계

< 표 2 > XMI 요소와 RDBMS 객체간의 매핑

XMI element	DBMS Object	XMI Type
Model.Class	Table	Foreign Key
Model.Association	Table	없음
Model.Attribute	Column	Datatype
Model.Reference	Table	Foreign Key
Model.Association End	Column,	Foreign Key,

이를 구현하기 위한 CWM_Creator 모듈은 CWM v1.0 을 정의한 XMI 파일을 사용 하였으며, 레파지토리 생성 API 는 JAVA 를 기반으로 DOM 파서를 사용하여 XMI 내의 슈퍼타입과 타입의 연결관계 처리가 용이하도록 하였다. CWM 을 정의한 XML 파일을 읽어드린 후에 해당 요소에 맞는 데이터정의어를 생성한 후에 JDBC 를 이용하여 관계형 데이터베이스에 해당 객체를 생성하였다. 추가적으로 레파지토리의 버전 및 생성자 관리를 위하여 최종생성일자, 최종 변경자, 최초 생성일자, 최초 생성자 등을 추가하였다.

```
// CWM 1.0 XMI FILE Parseing 및 Node List 구성
parser.parse("file:///C:/CWM/xmi/cwm_1.0.xmi");
Document d = parser.getDocument();
DocumentTraversal dt = (DocumentTraversal) d;
TreeWalker tw = dt.createTreeWalker(d.getDocumentElement(),
              NodeFilter.SHOW_ALL, new ObjectFilter(), true);
 Node n = tw.nextNode() ;
 // NodeName에 따라 해당되는 DDL을 생성한다.
 while (n != null) {
 if (n.getNodeName().equals("Model:Class") )
       ModelClass_process(n);
  if (n.getNodeName().equals("Model:Attribute") )
       ModelAttribute_process(n, "Attribute");
  if (n.getNodeName().equals("Model:Reference") )
       ModelReference_process(n, "Reference");
  .......
```

(그림 3) CWM_Creator Java API 의 예

CWM_Importer 는 CWM 기반으로 생성된 XMI 파일을 파싱하여 해당 태그 및 엘리먼트에 매핑되는 테이블을 찾아 삽입을 수행한다. CWM_Exporter 는 테이블간의 연결관계(Relationship)를 이용하여 CWM XMI 을 생성한다. CWM 기반의 레파지토리는 CWM 정의 XMI 구조를 최대한 유지하고 있으므로 XMI 파일의 저장 및 생성이 용이하다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <XMI timestamp="Fri Feb 13 23:34:03 KST 2004" xmi.version="1.1" xmlns:CWM="Core" xmlns:CWMRDB="Relati
  - <XMI.header>
    - <XMI.documentation>
        <XMI.exporter>CWM Based Repository CWM_Exporter</XMI.exporter>
        <XMI.exporterVersion>1.0</XMI.exporterVersion>
      </XMI.documentation>
    </XMI.header>
  - <XMI.content>
    - <CWM:Package xmi.id="X1028" name="TEST_PROJECT" visibility="public">
      - <CWM:Namespace.ownedElement>
        - <CWMRDB:Schema xmi.id="X1031" name="SCOTT" visibility="public" namespace="X1028">
          - <CWM:Namespace.ownedElement>
            - <CWMRDB:Table xmi.id="X1050" name="TB_HRA_QUAL100" visibility="public" isAbstract="false"
                isTemporary="false" isSystem="false" namespace="X1031">
              - <CWM:Namespace.ownedElement>
                - <CWM:Description xmi.id="X1063" visibility="public" body="직원자격상세" namespace="X1050">
```

(그림 4) 생성된 XMI 파일의 예

CWM_Manager 는 관계형 데이터베이스를 지원하는 다양한 환경에서 구현이 가능하나 웹 환경을 위해 JSP 를 사용하여 CWM_Manager 를 구현하였다. CWM 기반으로 생성된 레파지토리는 객체지향모델에 기반하여 구축되어 있기 때문에 테이블간의 연결 관계가 다소 복잡하여 비즈니스 사용자가 이해하기 어렵기 때문에, 연결관계를 미리 정의한 뷰를 생성하여 사용자가 직관적으로 이해할 수 있도록 하였다.

<표 3> 시스템 구현 환경

운영체제	Windows XP
레파지토리 DBMS	Oracle 사의 Oracle9i 9.2
Java API 개발 도구	Oracle 사의 Jdeveloper 9.0.3 Oracle JDBC Apache Xerces XML Parser

5. 결론 및 향후 과제

본 논문에서 제안한 CWM 레파지토리의 경우 다른 연구들에 비하여 다음과 같은 특징을 갖는다.

첫째, 도구들간의 메타데이터 상호교환의 효율성을 높이고 메타데이터를 최종 사용자가 사용하기 용이한 CWM 기반의 표준 레파지토리를 생성하였다.

둘째, 객체기술을 상용의 RDBMS 에 접목함으로써 기능적으로는 객체기술을 활용하면서 저장공간은 상용화된 관계형 데이터베이스를 사용함으로 메타데이터에 대한 다양한 접근이 가능해졌다.

셋째, 고정화된 레파지토리가 아닌 CWM 메타데이터를 정의하는 XMI 에 의하여 레파지토리를 생성함으로, 메타모델의 변경사항이 즉각적으로 반영되도록 하였다.

넷째, Model-Driven 데이터 웨어하우스를 구축하기 위하여 표준 메타데이터 서비스의 제공 및 메타모델의 상호작용을 지원할 수 있는 기반 레파지토리를 구현하였다[8].

<표 4> 타 레파지토리 시스템과의 비교

구분	Metaintegration 사의 MIR [9]	Oracle 사의 Warehouse Builder [10]	제안된 CWM 기반의 Repository
CWM XMI 저장 및 Interchange 지원	지원	부분 지원	지원
레파지토리 저장소	RDBMS	RDBMS	RDBMS
비즈니스 메타데이터 사용의 편리성	하	상	상
CWM 확장, 변경에 따른 레파지토리 변경의 용이성	하	하	상

향후 연구과제로는 CWM 기반의 레파지토리를 이용하여 표준화된 OLAP API 를 설계 및 구현하는 것이다. 또한 레파지토리를 기반으로 하여 실시간으로 메타데이터를 상호 연동할 수 있는 프레임워크를 만드는 것이다.

참고문헌

[1] Sam Anahory, Dennis Murray, “Data Warehousing in the real world” a practical guide for building decision support system pp.125-134, 1997.

[2] Paulraj Ponniah, “ Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals”, 2001.

[3] Kimball, Ralph, The Data Warehouse Toolkit. 1996.

[4] http://www.omg.org/news/releases/pr2000/2000-09-25a.htm, 2000.

[5] John Poole, Dan Chang, Douglas Tolbert, David Mellor, Common Warehouse Metamodel: An introduction to the standard for data warehouse integration, 37-42, 2001.

[6] John Poole, Dan Chang, Douglas Tolbert, David Mellor, Common Warehouse Metamodel for Developer’s Guide, 25-74, 2003.

[7] 정혜선, XMI 기반의 이기종 데이터베이스 간 데이터 변환, 2001.

[8] John Poole, http://www.cwmforum.org/POOLEIntegrate 2003.pdf, 2003.

[9] http://www.metaintegration.net/Products/MIR/

[10] http://otn.oracle.com/products/warehouse/index.html

[11] 권은정, 용환승, “XML 을 기반으로 한 관계형 데이터베이스 메타데이터 리파지토리 설계 및 구현”, 정보처리학회, 2002.