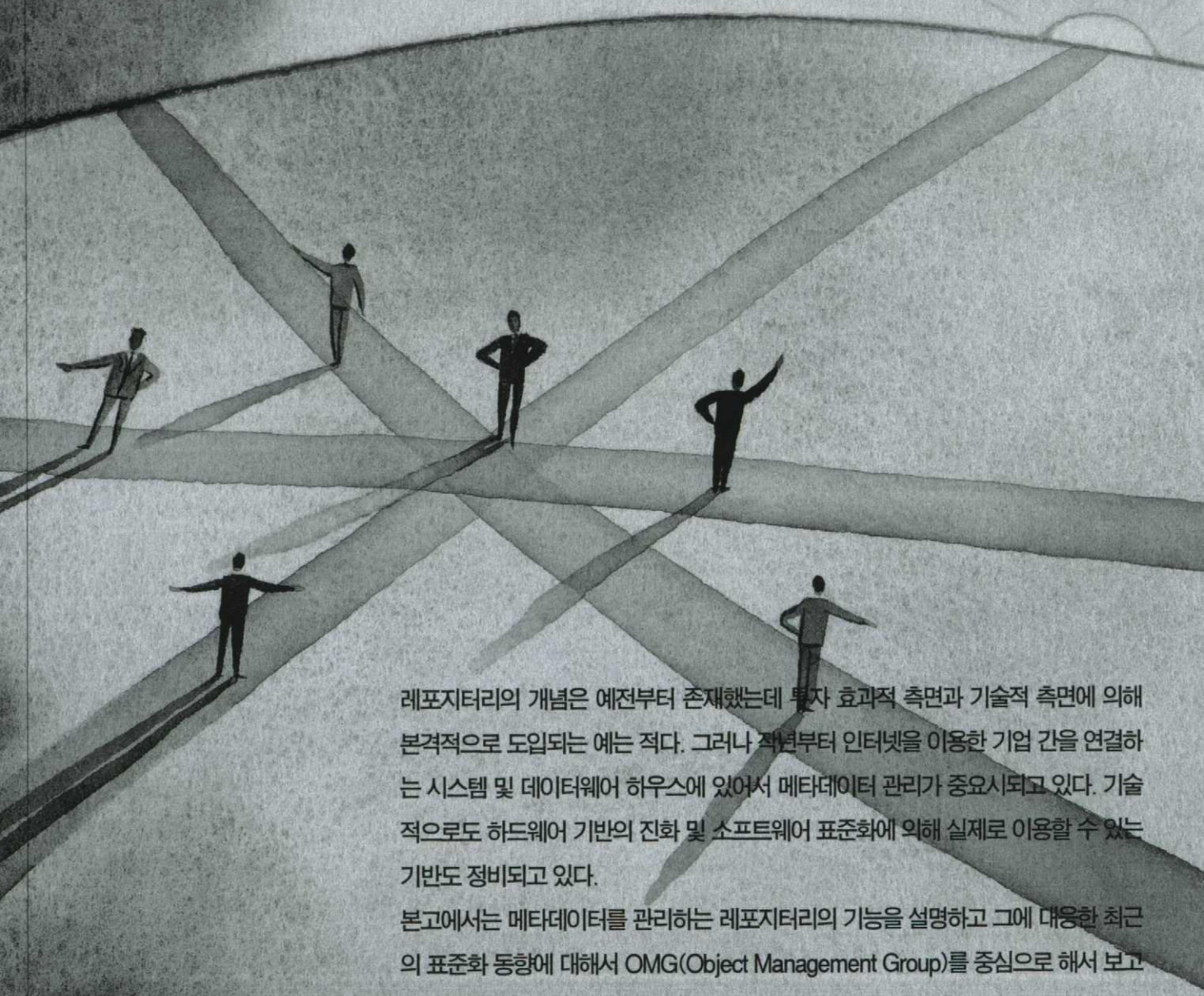
메타 데이터의 개념은 상대적인 것

기반기술인 레포지터리의 중요성

유키나리 아츠시 (行成 敦)



레포지터리의 개념은 예전부터 존재했는데 투자 효과적 측면과 기술적 측면에 의해 본격적으로 도입되는 예는 적다. 그러나 작년부터 인터넷을 이용한 기업 간을 연결하는 시스템 및 데이터웨어 하우스에 있어서 메타데이터 관리가 중요시되고 있다. 기술적으로도 하드웨어 기반의 진화 및 소프트웨어 표준화에 의해 실제로 이용할 수 있는 기반도 정비되고 있다.

본고에서는 메타데이터를 관리하는 레포지터리의 기능을 설명하고 그에 대응한 최근의 표준화 동향에 대해서 OMG(Object Management Group)를 중심으로 해서 보고하겠다.

필자 주

21세기를 맞이하여 컴퓨터가 점점 친밀하게 됨과 동시에 그것을 이용한 E 비즈니스의 근간을 이루고 있는 것은 인터넷인데, 실제 비즈니스에 이용할 수 있는 시스템으로서 운용될 수 있게 하기 위해서는 데이터에 관한 데이터 메타데이터를 관리해, 공개함에 따라 처음으로 다른 기업 간에 데이터 교환에 의한 업무가 가능하게 된다.

본고에서는 메타데이터의 개념을 해설함과 동시에 메타데이터를 관리하기 위한 툴 레포지터리 기능을 설명하고 그에 얽힌 최근의 동향을 보고하기로 한다.

1. 데이터의 계층과 메타데이터

본장에서는 일반적으로 기업에서 관리되는 정보의 계층을 고찰하고 메타데이터라는 것은 무엇을 정의하고 레포지터리가 커버하는 범위를 명확하게 설명하겠다.

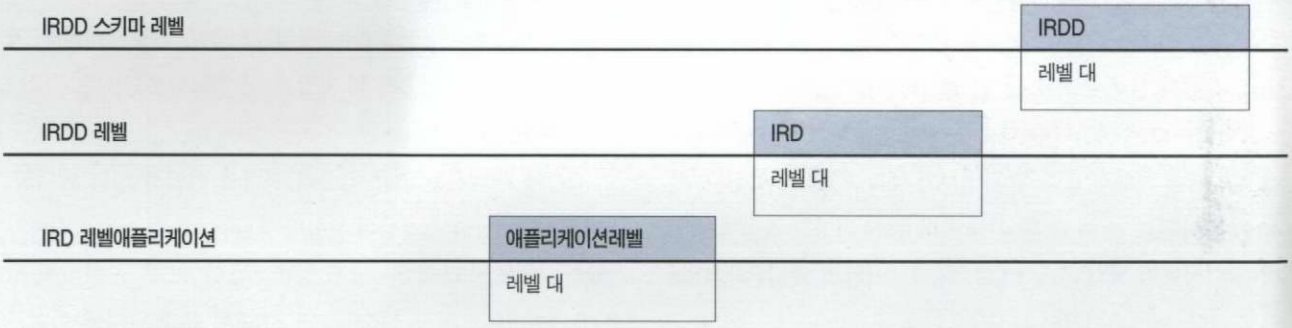
기업에서 관리되는 정보는 몇 가지의 계층으로 되어 있는데, ISO IRDS로 제안하고 있는 참조모델의 개념을 이용해서 설명하겠다.

(1) ISO IRDS 참조모델과 메타데이터

ISO IRDS(Information Resource Dictionary System: 정보자원사전 시스템)를 참조 모델에서는 정보관리라는 관점에서 정보를 레벨이라 불리는 4계층으로 분류하고 있다.

상위층부터 IRDD 스키마레벨, IRDD 레벨, IRD 레벨, 애플리케이션 레벨이라 불린다(그림1). 인접한 레벨은 상위층이 하위층의 스키마를 기술하고 있고 레벨대라 불린다. 구체적으로는 다음 그림 2-4에 표시된 것처럼 정보가 각 레벨에 격납된다. 다만 이것은 개념을 도시한 것이고 IRDS의 규격으로 이처럼 정의되어있다는 것은 아니다.

〈그림1〉 ISO IRDS 참조 모델



〈그림2〉 IRDD 레벨 대

IRDD 스키마 레벨

IRDD 레벨

관리표	
이름	코드
RDB 표	M0001
RDB 예	M0001

관리예	
이름	표코드
이름	M0001
코드	M0001
이름	M0002
표 코드	M0002

〈그림3〉 IRD 레벨 대

IRDD 레벨

IRD 레벨

RDB 표	
이름	코드
사원표	0001
소속표	0002

RDB 예	
이름	표코드
이름	0001
사원번호	0001
사원번호	0002
소속표	0002

〈그림4〉 애플리케이션 레벨 대

IRD 레벨	사원표		소속표	
	이름	사원번호	사원번호	소속코드
애플리케이션	스즈키	11111	11111	9901A
	마스오	22222	22222	9902B
	아오키	33333	33333	9901C

통상 기업에서 관리하는 정보는 애플리케이션에 속해, 그 스키마 정보가 IRD 레벨에 격납된다. 예를 들면 애플리케이션에서 정의한 표는 사원표 및 소속표인데, 이들 표에 관한 정보는 IRD레발에 격납된다. 좀더 말하자면 스키마 정보를 관리하는 정보는 IRDD 레벨에 속한다. IRDD레벨에 격납된 정보는 위의 예에서는 표나 예에 관한 정보인데 이들의 정보는 관계 데이터베이스라는 특정의 테크놀로지에 속하는 저의 정보이다.

예를 들면 어브젝트 지향 데이터 베이스를 사용한 시스템의 정보를 격납할 경우, IRDD레벨에서는 클레스나 소속에 관한 정보를 관리하게 된다. 최상위층이 IRDD 스키마 레벨이고, 이것은 고정된다. ISO IRDS에서는 IRDD 스키마 레벨의 구조와 IRDD레벨의 일부 구조를 규격으로 규정하고 있다.

메타데이터는 데이터를 기술한 데이터로 상기의 그림에서는 각 데이터에 대해서 표나 예의 정보가 메타데이터라 불린다. 예를 들면 사원표, 소속 표 및 이들 표에 관한 정보가 기업에서 관리하는 정보의 메타데이터이다. 메타데이터의 개념은 상대적인 것이다. 메타데이터의 특징은

- 이용자의 데이터베이스에 비해서 데이터 양이 비교적 소량이라는 것
- 개개의 데이터 간에 복잡한 관련을 갖는 것
- 어느 빈도로 동적으로 변경되는 것

이다. 메타데이터를 관리 · 격납하기 위한 데이터 격납고가 레포지터리이다.

(2) 레포지터리

본 고에서는 ISO IRDS로 정의된 계층 중, 상위 3계층의 정보를 관리하는 격납고를 리포지트라 부른다.

격납된 정보는 상위 층으로 갈 정도의 정보량은 적은데, 스키마 구조는 복잡하다는 특징이 있다. 예를 들면 상기의 예에서는 관계 데이터베이스의 정보를 관리하기 위해서는 표나 예에 관한 정보에 덧붙여 그들 정보간의 관련, "외부 키"등의 시멘틱 정보 등을 관리할 필요가 있고, 정보의 복잡성은 늘어난다. 또 상위 층의 변경은 하위 층의 광범위하게 영향을 미치기 때문에 정의, 보수하는 곤란함이 지적되고 있다.

기업의 시스템에서 레포지터리의 활용이 예견되어 온 배경에는 이러한 곤란함과 함께 다음과 같은 원인을 들 수 있다.

- 투자효과를 볼 때, 본격적인 레포지터리를 작성함으로서 데이터 베이스 관리 시스템 등을 이용해서 그 시스템에 특화한 메타데이터를 격납 . 관리하는 것이 유효하다고 생각되었다는 것.
- 규격으로서의 표준화는 진행되었지만, 레포지터리를 지원하는 기반기술의 미 발달로 실제로 이용되는 프로젝트가 없었다는 것.

전자에 대해서는 최근 컴퓨터 환경에 있어서 멀티 밴더화, 인터넷을 이용한 시스템의 분산화에 따른, 그 환경을 관리하기 위한 레포지터리의 필요성이 높아지고 있다. 특히 SCM(Supply Chain Management)처럼 문화가 다른 기업간에 데이터를 교환할 경우, 어디에 어떠한 정보가 어떠한 형태로 존재하는 가 관리해서 이용자에게 공개할 필요가 있어, 레포지터리에 대한 수요는 높다.

후자에 대해서는 어브젝트 지향 기술의 발달과 함께 복잡한 구조를 가진 메타데이터를 관리하는 기술기반이 정비되고 그에 따른 프로젝트가 준비되어 있다. 또 복잡한 데이터를 정의하기 위해서 시스템 전체에 관한 정보를 정의하는 것이 아니라, "도메인"마다 레포지터리 구조를 정의해서 데이터 교환에 의한 시스템 제휴를 도모하기 위한 기반기술이 정비되고 있다.

2. 레포지터리로서 필요시 되는 기능

본 장에서는 메타데이터를 관리하는 관점에서 그 격납고인 레포지터리에 필요시 되는 기능에 대해서 설명하겠다. 기능은

[정보 모델], [레포지터리 서비스 기능], [시스템으로서의 요건], [데이터 교환 기능]으로 나뉜다.

(1) 정보 모델

정보모델이라는 것은 레포지터리에 격납된 스키마 정보를 가리킨다. ISO IRDS 참조 모델에 비교시키면 IRDD 스키마 레벨 및 IRDD 레벨에 격납된 정보를 의미한다.

IRDD 스키마 레벨에 격납된 정보는 표준규격으로서 ISO IRDS 및 OMG (Object Management Group)에서 규정되어 있다.

따라서 스키마 레벨에 격납된 사용자의 메타데이터를 널리 공유할 수 있는 가는 IRDD 레벨의 정보가 어떻게 표준화되는 가에 달려있다.

이 레벨에 격납된 정보는 관계 데이터 베이스나 UML(Unified Modeling Language)과 같은 컴퓨터 기술에 기초한 스키마, PDM(Product Data Management)과 같은 어떤 애플리케이션 분야로 특화된 모델, 사용자가 특정의 목적을 위해서 정의한 모델 등이 포함된다.

레포지터리에 필요시 되는 요건으로서는 표준규격에 기초한 IRDD 레벨의 정보가 얼마나 많이 제공되는 가가 중요하다.

(2) 레포지터리 서비스 기능

레포지터리 서비스는 레포지터리에 격납된 정보를 취급하기 위해서 제공된 기능을 가리킨다. 복잡한 메타정보를 관리할 필요 때문이다. 지금까지 제안되어 온 주요 기능은 아래와 같다.

- 메타데이터의 복잡한 구조나 관련을 관리하기 위한 기능
- 메타 데이터의 버전 관리와 구성관리를 위한 기능
- 정보 모델의 동적인 변경 . 확장을 가능케 하기 위한 기능
- 트랜젝션 관리 (특히 롱 트랜젝션)

이들 기능 중에는 데이터베이스 관리 시스템(이하 DBMS라 약칭한다)이 이미 지원하고 있는 기능도 포함되어 있다. 이하 관계 데이터베이스, 어브젝트 지향 데이터베이스와 비교를 하면서 상세히 설명하겠다.

1) 복잡한 구조나 관련의 관리

일반적으로 레포지터리의 정보는 복잡한 관련을 갖는다. 그것을 그림3을 이용해서 설명하기로 한다. 그림3의 예는 관계 데이터베이스의 표 정보를 관리하기 위한 것인데, 정보 자원의 관점에 보면 다음과 같은 조회가 필요하다.

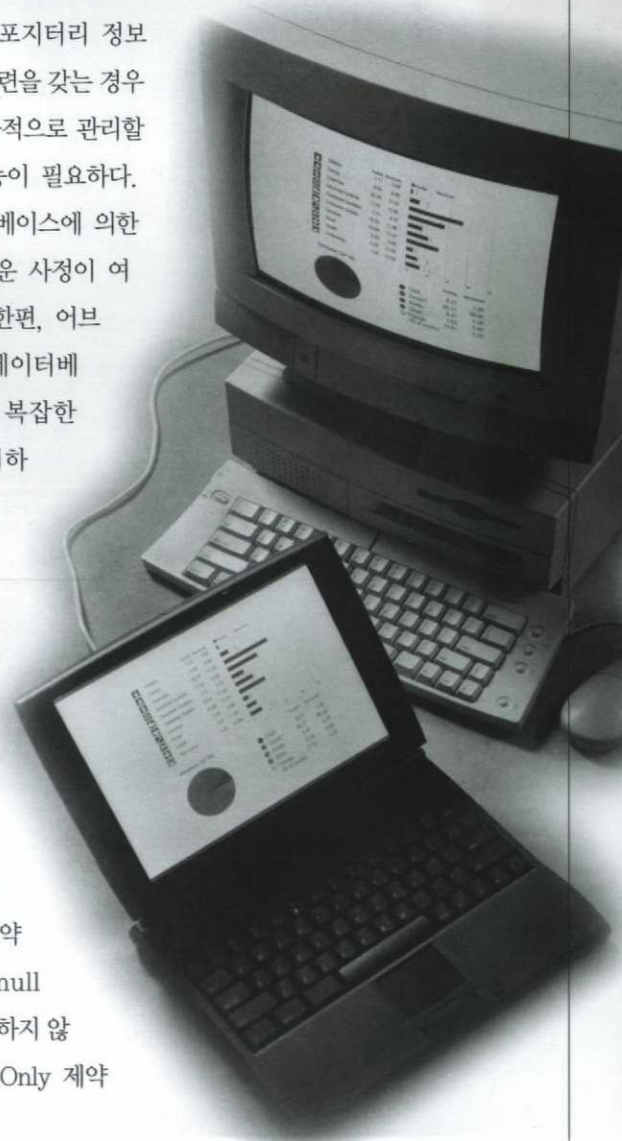
① 각각의 표는 어떠한 예로부터 되는 가

② 어느 표와 관계 있는 표에는 어떠한 표가 있는 가

① 의 조회는 RDB 표와 RDB예를 join한 것에 의해 검색할 수 있다.

② 의 조회를 위해서는 추가정보가 필요하다. 위의 예에 있어서 사원표를 변경할 경우 어느 표에 영향이 있는 가를 생각한다. 이를 위해서는 사원 표와 소속 표의 사원번호가 동일하다는 정보를 관리할 필요가 있고, 이 정보를 기초로 join할 필요가 있다.

어느 쪽의 경우도 정보 간에 관련을 갖고 있고, 관련 자신에 대한 정보도 필요해, 비즈니스 정보에 비해서 복잡한 관련 정보를 관리, 조작할 필요가 있다.



이처럼 레포지터리 정보는 복잡한 관련을 갖는 경우가 많아 효율적으로 관리할 수 있는 기능이 필요하다. 관계 데이터베이스에 의한 관리가 어려운 사정이 여기에 있다. 한편, 어브젝트 지향 데이터베이스에서는 복잡한 관련을 관리하는 기능은 일반적으로 갖추어져 있다.

그러나 관련의 의미 제약 예를 들면 일의성 제약, 필수제약 (그 관련이 null인 것을 허용하지 않는다), read Only 제약

등은 DBMS에 의해서 지원 정도의 차가 있다. 필요한 제약이 지원되어 있지 않은 경우 제약 체크를 메서드 등으로 유저가 정의할 수 있는 기능이 필요하다.

2) 버전 관리와 구성관리

소프트웨어의 자산관리를 할 경우, 버전 관리는 중요하다. 기업 시스템은 항상 변경 · 개량 되는 것이고, 그것들의 이력을 기록하는 것이 소프트웨어의 보수 및 장래의 개발에 있어서 중요하기 때문이다.

구성 관리라는 것은 의미가 있는 단위를 하나의 그룹으로 해서 관리하는 기능을 말하고 버전 관리와 밀접한 관계가 있다. 예를 들면 실행 가능 모듈과 소스 파일의 관계를 하나의 프로젝트로 생각해 관리하는 경우, 하나의 소스파일의 버전 변경이 프로젝트 전체의 버전 변경으로 생각할 필요가 있는데 이와 같은 정보의 관리를 레포지터리에 기대하고 있다.

버전관리 및 구성관리도 복잡한 데이터 구조를 취급할 필요가 있어 어브젝트 지향 기술이 가장 유력시되고 있다.

3) 동적인 변경 · 확장

레포지터리도 일반적인 데이터 베이스와 마찬가지로 어느 주기로 모델 변경이 행해진다. 그 경우, 레포지터리 이용자가 변경 내용에 의존하니 않고 접근할 수 있는 기능이 필요하다.

예를 들면 그림3에서 [사원 표]에 관한 정보의 일람을 표시하는 것을 생각할 수 있다. [RDB 표]의 표에는 [이름]과 [코드]에 관한 정보가 격납되어 있는데 [뷰 표]인가 아닌가를 판단하는 정보가 첨가된다고 한다. 이 경우 [RDB표]에 정의되어 있는 속성에는 무엇이 있는 가를 검색해서 [사원 표]에 관한 그것들의 속성치를 모두 검색할 수 있는 것이 필요하다.

바꾸어 말하면 IRD 레벨에 격납된 유저의 메타데이터에 관한 정의 정보를 검색해서 그 수치 자신도 참조할 수 있는 기능이 필요하다. 그에 따라 최초로 동적으로 레포지터리의 변경 및 확장이 가능하게 된다.

4) 트랜젝션 관리(특히 롱 트랜젝션)

일반적인 DBMS가 지원하고 있는 트랜젝션 관리는 일련의 처리를 하나의 단위로서 처리하기 위해서 레포지터리도 필요하다. 그에 덧붙여 레포지터리에서는 롱 트랜젝션도 요구된다.

예를 들면 프로그램 정보를 레포지터리로 격납할 경우에 기존의 프로그램 정보 P1/version1을 변경해서 P1/version2로 바꾸어 쓰는 것을 생각할 수 있다. 통상 프로그램을 변경하는 작업은 1초라든가 2초안에 종료하는 것은 아니다. 이 경우 P1의 변경 중에는 P1/version1은 다른 멤버에게 변경을 허락하지 않는 구조가 필요하게 된다. 장시간에 걸쳐 접근을 제한하는 구조 롱 트랜젝션이 필요하게 된다.

(3) 시스템으로서의 요건

레포지터리 서비스에서 말한 기능이 레포지터리의 "논리적"인 기능인데 반해 다음과 같은 요건은 실 시스템으로서 레포지터리 시스템을 구축할 경우 필요하게 된다.

- 콘카렌트에 복수 유저가 접속할 경우 자세한 레벨로 동시 실행 제어를 행해 높은 콘카렌시를 보증하는 것
- 콘카렌트 수가 증대해도 장애가 없이 극단적인 효율악화를 초래하지 않는 스케러빌리티
- 유저의 역할(롤)을 등록해서 롤마다 어브젝트에 대한 접근을 제어하는 기능

이것들은 현재 DBMS 일반으로 구할 수 있는 것인데 레포지터리 시스템을 미션크리티컬한 기업 시스템으로 사용할 경우도 필요하다.

지금까지의 레포지터리는 어느 쪽인가 하면 정적인 레포지터리, 전형적으로는 소프트 웨어 개발환경에 있어서 레포지터리라는 이미지가 강해 이점에 대해서는 그다지 중시되지 않았다. 그러나 CORBA 환경에 있어서 레포지터리 등, 실행시에 이용되는 기회가 증가됨에 따라 최초의 단계에서 이와 같은 환경을 의식하거나 레포지터리 시스템이 필요하다.

(4) 데이터 교환기능

레포지터리 시스템은 일반적으로 기업 시스템에 관한 정보를 관리하는 것이므로 정보의 "양"은 통상의 데이터베이스에 비해서 적은데 데이터 구조 및 의미는 복잡하다. 따라서 모두를 포함하는 레포지터리를 정의하는 것은 현상태를 분석하는 면에서도 장래에 걸쳐 확장하는 면에서도 보더라도 곤란하다.

어느 도메인에게 특화한 레포지터리를 구축하는 [보텀 업] 어프로치가 현실적이다. 이처럼 복수의 레포지터리 시스템이 공존해서 서로 제휴하는 형태를 [페더레이트]레포지터리라 부른

다. 그 경우 문제가 되는 것은 복수의 레포지터리간의 데이터 제휴이고 표준적인 형식으로 데이터 교환을 행할 수 있는 구조가 필요하다.

3. OMG의 동향…앞으로의 움직임

본 장에서는 앞에서 말한 데이터 계층 및 레포지터리 기능에 관한 OMG의 표준화 동향을 중심으로 설명하겠다. 다른 표준화는 OMG와 제휴하는 형태로 진행되고 있다. 앞부분에서 사용한 참조 모델은 메타데이터를 설명하기 위해 인용된 경우가 많은데 ISO IRDS 규격 자체는 전혀 실장되어 있지 않다.

ISO IRDS에서 정의되고 있는 4계층 모델은 OMG에서도 마찬가지로 정의되고 있어 OMG에서 사용되고 있는 용어와의 대응은 다음과 같다.

ISO IRDS	OMG
IRDD 스키마 레벨	메타메타모델 (M3)
IRDD 레벨	메타모델 (M2)
IRD 레벨	메타데이터/모델 층(M1)
애플리케이션	유저 데이터 층(M0)

1997년 OMG에서 규격화 된 Meta Object Facility(MOF)에서 규정되어 있는 것은 IRDS 최상위 층의 IRDD 스키마 레벨에 대응된다.

실현 하려하는 컨셉은 IRDS와 같은 것인데 기술적인 면에서 큰 차이가 있다. ISO IRDS에서는 IRDD 스키마 레벨 데이터 구조를 관계 데이터 베이스의 말로 정의해서 데이터 조작은 서비스 인터페이스로 해서 함수호출로 제공한다. 한 편, MOF는 어브젝트 지향의 개념을 취해 CORBA IDL을 이용해서 정의하고, 조작은 메서드에 의해 실현하고 있다.

앞에서 말한 '각각의 표는 어떠한 예로부터 되는 가' 처럼 레포지터리에 격납된 정보는 복잡한 구조와 관련을 갖고 있어 어브젝트 지향 기술은 레포지터리를 실현하기 위해서는 필수다.

실 운용 및 장래 확장성의 관점에서는 MOF 쪽이 우수하다. 또 MOF에서는 reflection API로서 현재 참조하고 있는 어브젝트 자신의 정의 정보를 검색하고, 그 저의 정보에 기초해서 어브젝트에 정의 되어 있는 속성을 엑세스하거나 메서드를 기동하는 기능도 넣고 있고, 동적 레포지터리의 기반을 제공한다. 앞으로 버전 관리기능이나 구성관리 기능을 도입해 갈 예정이다.

트렌젝션 관리에 대해서는 숏 트랜젝션은 CORBA의 기능으로서 제공되고 있다. 롱 트렌젝션은 앞으로의 검토과제이다.

MOF 다음으로 표준화가 진행되고 있는 것은 XMI (XML Matadata Interchange)이다. OMG의 기본개념인 인터오페러티빌리티를 지원하기 위해 텍스트 베이스의 데이터 교환 포맷으로 해서 XMI를 정의했다. MOF DTD가 정의되고 또한 MOF 준거 모델 이라면 자동적으로 모델에 대응한 DTD를 생성하는 기능을 갖는다. 이에 따라 다른 벤더의 툴 사이에, 레포지터리 사이에서 데이터 교환이 가능하게 된다. 이 교환 기능을 행하는 [페더레이트] 레포지터리의 구축이 가능하게 된다.

OMG에서는 논리적으로 자세한 엑세스를 제공하고 있고 이들의 요건을 만족시키기 위한 기반을 제공하고 있다.

정보 모델에 관해서는 오브젝트 지향 분석 · 설계를 지원하는 UML 모델, 데이터 웨어하우스의 정보를 관리하기 위한 CWM(Common Warehouse Metamodel)이 규격으로서 채택되어 있다. UML은 널리 인지되어 대응 툴도 시장에 많이 나와 있다. CWM은 앞으로 데이터베이스 벤더를 중심으로 실장이 진행될 예정이다.

더나아가 2001년 2월 OMG는 MDA(Model Driven Architecture)를 발표해 모델 지향의 시스템 구축을 추진했다. MDA에서는 UML에 의해 실장기술에 의존하지 않는 시스템의 모델(Platform Independent Model -PIM)을 기술해 실장단계인 특정의 기술에 의존한 모델(Platform Specific Model - PSM)을 "자동적으로" 생성하는 것을 제창하고 있다. 또한 리베이스 엔지니어처럼 현 상태의 코드에서의 모델 생성도 목표로 하고 있다. 모델 기술 언어로서의 UML, 모델 정보의 격납 기반으로서의 MOF, PIM↔PSM변환 규칙을 격납할 기반으로서의 CWM, 모델 정보교환을 위한 XMI를 중핵으로 해서 기능 확장이 예정되어 있다.

이처럼 글의 앞에서 말한 기능을 만족시키는 방향으로 OMG의 표준화가 진행되고 있어 실장된 제품이 시장에 등장할 것이라 기대된다.

마지막으로 레포지터리 시스템에 대한 요건은 개발환경의 지원에서 랜타임 이용에 크게 시프트하고 있어 그 니즈에 대응되는 규격 및 하드웨어와 소프트웨어 환경이 정비되고 있다. 그리고 MDA를 중심으로 가까운 장래 레포지터리가 본격적으로 이용되리라 기대된다.