XML 문서 저장 방법 비교

김경래, 하상호
순천향대학교 정보기술공학부

Comparison of the Storage of XML Document

Kyungrea Kim, Sangho Ha
Dept. of Information Technology, SoonChunHyang University
E-mail : krkim@java.sch.ac.kr

요 약

XML은 강력한 데이터 표현능력을 인정받아 전자상거래와 같은 데이터 처리 분야에 적극적으로 도입되고 있다. 전자상거래는 인터넷의 확산과 더불어 급속도로 발전되었고, B2C의 성공은 기업간 전자상거래를 위한 비즈니스 모델들을 창출하였다. 이 비즈니스 모델들은 새로운 문서 기술 언어인 XML로 작성되고, 그 정보들은 각각의 기업의 데이터베이스에 저장된다. 저장에 필요한 데이터베이스로서 관계형 데이터베이스가 가장 일반적으로 사용되고 있으나, 관계형 데이터베이스의 단점을 보완한 객체지향형 데이터베이스가 개발되었고, 관계형 데이터베이스와 객체지향형 데이터베이스의 장점을 살린 혼합형이라 할 수 있는 객체 관계 데이터베이스가 개발되었다. 본 논문에서는 XML 문서의 저장에 관계형 데이터베이스와 객체 관계 데이터베이스를 사용하여 저장의 효율성을 비교한다.

1. 서론

인터넷의 확산과 더불어 B2C의 수요가 크게 증가하면서, 각각의 기업들에서는 B2B의 중요성을 깨닫게 되었고, 이를 위한 표준 거래 환경을 제정하기 위한 노력을 하고 있다[1]. 이와 같은 노력에 의해 제안된 프레임워크들은 기업간의 상품에 대한 정보교환을 위해 XML[2]에 기반하여 각각의 고유한 상품카탈로그를 제공하고 있다. 이러한 모든 상품 정보를 통합적으로 표현할 수 있는 XML기반 통합 상품 표현 모델[3]이 지난 연구에서 제안된 바 있다. 이 상품 표현 모델은 XML에 기반하여 상품 정보를 효과적으로 통합하여 기술할 수 있는 표준 상품 표현 모델로서, 전자상거래에서 취급되는 대부분의 상품을 기술할 수 있으며, 상품 정보를 속성별로, 단계적 세분화를 통해서 기술 가능하며, 상품 정보상에 중복될 수 있는 데이터를 피하는 것이 가능하다. XML은 웹 상의 데이터를 표현하고 교환하기 위한 새로운 표준으로 빠른 속도로 받아들여지고 있다. XML 데이터는 그 구조를 자체적으로 기술 가능하며, 따라서 XML 문서를 해석하고 그 문서를 조작하는 응용 프로그램을 작성하는 것이 가능하다. 이러한 프로그램은 내용에 기반 하여 문서를 필터링하고 필요에 따라 문서를 재구성하는 것을 포함한다. 이런 특징에 의해 XML에 기반한 상품카탈로그는 각 기업에서 출시하는 상품에 대한 정보를 표현하기 위한 구조를 가지고 있고, 표현되는 각각의 모든 상품들은 공통적인 정보와 고유의 정보를 가지고 있다. 이러한 정보들은 기업간에 교환을 통해 각 기업의 데이터베이스에 저장된다. 데이터베이스는 현재 일반적으로 가장 많이 사용되는 관계형 데이터베이스와 관계형 데이터베이스의 단점을 보완하여 데이터베이스에 객체를 도입한 객체지향형 데이터베이스, 그리고 관계형 데이터베이스와 객체지향형 데이터베이스의 장점을 혼합한 객체 관계 데이터베이스가 있다. 관계형 데이터베이스는 사용자가 데이터의 물리적 저장형태를 의식하지 않고, 행과 열로 구성된 테이블 형태의 데이터 표현을 사용하여 자연스럽게 데이터 항목을 다룸으로써 조작성과 유연성이 뛰어나다는 장점을 가지고 있다. 하지만, 컴퓨터 기술의 발달과 더불어 사회 전 분야에 걸친 정보화 요구는 관계형 데이터베이스만으로는 충족시킬 수 없는

새로운 응용 분야를 등장시키게 되었다. 이들 응용에서는 멀티미디어 정보, 비정형 문서 정보, X선 사진 등의 이미지 정보, 지리정보 등을 다루는데 이러한 정보의 특성은 기존의 관계형 데이터베이스에서는 효과적으로 다룰 수 없는 대용량의 비정형 레코드라는 것이다. 또한 복잡한 모델링 기능을 요구함으로써 비교적 단순한 데이터 모델만을 제공하던 기존의 관계형 데이터베이스만으로는 이러한 요구를 만족시킬 수 없게 되었다. 이러한 단점을 극복하기 위해서 새로운 객체지향 기술을 도입한 객체지향형 데이터베이스에 대한 연구가 진행되었다. 객체지향형 데이터베이스는 프로그래밍 언어 분야에서 파생된 객체지향 모델을 지원하는 데이터베이스이다. 객체지향형 데이터베이스에서는 객체, 객체 식별자, 클래스, 메소드, 복합 객체, 상속, 집합 애트리뷰트 등 복잡한 데이터 모델링 기능과 대용량의 레코드를 저장할 수 있는 기능을 모두 제공한다. 그러나 현재로는 질의어, 보안성, 서버함수, 병렬처리 등의 데이터베이스 상위 기능 등의 제공이 취약한 수준에 있고 또한 객체지향형 데이터베이스는 공통된 질의어가 없고 기존의 관계형 데이터베이스와 호환성이 없다는 단점이 있다. 따라서 복잡 다양한 데이터 유형을 객체지향적으로 관리함과 동시에 업계 표준인 SQL을 확장하여 더욱 효율적으로 만든 질의어를 제공하는 객체관계형 데이터베이스가 등장하게 되었다.

본 논문에서는 지난 연구에서 제안된[3]바 있는 XML기반 상품표현모델을 적용하여 객체관계형 데이터베이스와 일반적으로 널리 사용되고 있는 관계형 데이터베이스에의 XML 문서 저장시스템을 구현하고, XML 문서를 저장하기 위한 스키마 생성의 효율성 및 데이터베이스로의 저장 시간, 특정 키워드에 의한 검색 시간 등을 비교한다.

논문의 순서는 다음과 같다. 2장에서 XML 문서를 관계형 데이터베이스와 객체 관계 데이터베이스에 저장하는 방법에 대해 설명하고, 3장에서 간단한 XML DTD를 예로 들어 데이터베이스 스키마를 생성해본다. 4장에서는 제안된 모델을 적용하여 관계형 데이터베이스와 객체관계형 데이터베이스의 저장 시스템을 구현하고, 구현된 시스템들에 실제 웹상의 상품정보들을 대상으로 두 데이터베이스의 XML 문서 저장의 효율성을 비교해본다. 마지막으로 5장에서 결론을 맺도록 하겠다.

2. XML 문서의 저장

최근에 XML 문서를 데이터베이스에 저장하는 방법[4]에 대해서 많이 연구되어 왔다. 본 장에서는 XML 문서의 저장시 객체 관계 데이터베이스 사용의 잇점에 대해 설명하고, XML 문서로부터 관계형 데이터베이스와 객체 관계 데이터베이스의 스키마를 생성하는 방법을 설명한다.

XML 문서의 가장 큰 장점 중 하나는 문서의 구조를 기술할 수 있다는 것이다. 객체 관계 데이터베이스는 VARRAY나 Nested Table의 지원으로 테이블의 필드로 테이블 또는 객체가 올 수 있다. 이는 XML 문서의 구조를 데이터베이스 상에서 유지할 수 있게 해주는 특징이다. VARRAY는 특정필드에 객체를 배열의 형태로 표현할 수 있는 방법으로, 한 필드에 객체가 여러 번 표현될 수 있다. Nested Table은 이와 유사하나 특정 테이블을 테이블의 필드에 중첩하여 나타낼 수가 있다. 데이터베이스에서 문서의 구조를 유지 할 수 있다는 것은 그 만큼 데이터 베이스 스키마의 생성이 쉽다는 것을 나타낸다. 다음은 관계형 데이터베이스와 객체 관계 데이터베이스의 스키마를 생성하는 방법에 대해 설명한다.

관계형 데이터베이스의 스키마는 기본적으로 DTD상의 최하위 elements나 attribute는 그것을 포함하는 상위 elements의 Fileds를 이룬다. 따라서 상위 elements는 관계형 데이터베이스의 테이블로 표현이 된다. 하지만 (*)나 (+)속성을 가진 elements는 중복하여 표현될 수 있기 때문에 따로 Table로 구성하는 것이 바람직하다. XML 문서의 elements중에는 용이한 문서 구조와 의미 파악을 위한 중간 elements가 존재하기 마련이다. 이와 같은 elements는 삭제 및 적절한 조정이 가능하며, 물론 이와 같은 삭제 및 조정은 DB Deginer 가 Appication 개발자에게 분명하게 언급을 해야만 한다. 삭제할 수 없는 중간 elements는 상위 elements가 이루는 Table과 하위 elements가 이루는 Table간의 적절한 연관을 지어주는 기본키나 외부키의 이름으로 사용될 수 있다.

객체 관계 데이터베이스의 스키마를 위해 우리는 지난 연구에서 XML DTD의 구조에서 객체 관계형 스키마를 생성하는 규칙을 제시한 바 있다[5]. 이 연구에서는 4가지의 규칙을 제시하였고 우리는 이 규칙에 따라 객체 관계형 스키마를 생성하였다. 다음은 제시된 4가지 규칙이다.

(1) 트리-구조에서 최상위 노드는 테이블로 사상한다.
(2) 노드가 잎 노드일 경우에, 상위 노드는 객체 타입

으로 선언하고 잎 노드는 그 객체의 필드가 된다.

(3) 노드가 여러 번 나타날 수 있는 '+'나 '*'로 표현될 경우에, 노드는 VARRAY 타입으로 표현된다.

(4) 각 객체들은 트리-구조상에서 하위의 개체들을 필드로 가질 수 있다.

위의 규칙에 따라 생성된 객체 관계 데이터베이스의 스키마는, 관계형 데이터베이스와는 달리, 최상위의 노드와 맵핑된 하나의 테이블 안에 하위 노드들이 맵핑된 객체들이 사용됨으로써 XML 문서 구조의 유지가 가능하다.

3. XML 문서의 저장 예

3장에서는 간단한 XML DTD를 예로 들어 각 데이터베이스의 스키마 생성과 저장과정을 설명하도록 하겠다. 그림 1은 XML DTD의 예이다. 이 DTD는 간단한 상품의 정보를 표현할 수 있는 DTD이다. 이 DTD를 2장에서 제시한 규칙에 따라 생성한 각각의 데이터베이스 스키마는 다음과 같다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<!ELEMENT ProductCatalog (ProductName, DefaultLanguage,
Partner*, ProductDescription*, CountryOfOrigin?,
Manufacturer?, ProductPrice*, Specification*)>
<!ATTLIST ProductCatalog
    ProductID ID #REQUIRED>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT DefaultLanguage (#PCDATA)>
<!ELEMENT Partner (Name)>
<!ATTLIST Partner
    PartnerID ID #IMPLIED
    RelationShip (Seller | Manufacturer | Imtermediary)
#IMPLIED
    Role CDATA #IMPLIED>
<!ELEMENT Name (FirstName,LastName)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT ProductDescription (#PCDATA)>
<!ATTLIST ProductDescription
    DescriptionPurpose CDATA #IMPLIED>
<!ELEMENT CountryOfOrigin (#PCDATA)>
<!ELEMENT Manufacturer (#PCDATA)>
<!ATTLIST Manufacturer
    PartnerRef IDREF #IMPLIED>
<!ELEMENT ProductPrice (Amount, Buyer?)>
<!ELEMENT Amount (#PCDATA)>
<!ELEMENT Buyer (#PCDATA)>
<!ELEMENT Specification (#PCDATA)>
<!ATTLIST Specification
    TypeCode (Size | Weight | Ingredient | Color | Shape |
Packing | Performance | Content | Others) #IMPLIED
    UOM CDATA #IMPLIED
    SpecificationName CDATA #REQUIRED>
```

<그림 1> XML DTD 예

관계형 데이터베이스 스키마

예로 제시된 DTD에는 최상위 노드인 ProductCatalog가 하위 노드로 총 8개의 노드를 포함하고 있다. 이 중 4개의 노드인 ProductName, DefaultLanguage, CountryOfOrigin, Manufacturer는 마지막 노드이고 다중 속성(*, +)이 없기 때문에 상위 노드의 필드를 이루게 된다. 또한 ProductCatalog는 Attribute ProductID를 기본키로 갖게 된다.

남은 element인 Partner, ProductDescription, ProductPrice, Specification은 테이블로 생성하고 ProductCatalog_Table과의 연관을 고려해야 한다. 이는 각 엘리먼트는 *과 +같은 다중 속성이 있기 때문이다.

Partner 엘리먼트는 PartnerID, RelationShip, Role 등의 attribute를 포함하고 있고, Name 엘리먼트를 포함하고 있다. Partner에 표현되는 정보는 상품과의 관계를 나타내는 RelationShip과 Role, 그리고 Partner자체의 정보인 Name으로 나뉘어져 있다. 이는 각 상품과의 관계에 따라 Name이 여러번 저장될 수 있음을 뜻한다. 물론 본 예에서는 이름만 표현되지만 원본 DTD 상에서는 Partner의 정보가 더 많아진다. 따라서 이런 정보의 중복저장을 피하기 위해 테이블을 2개로 나눈다. 표 1은 위의 과정에 의해 생성된 테이블과 테이블의 필드를 보여준다. 표에서 보듯이 예제 XML DTD는 관계형 데이터베이스에 총 6개의 테이블을 생성했음을 볼 수 있다.

<표 1> 관계형 데이터베이스 스키마의 예

Table Name	Fields
ProductCatalog_Table	ProductID ProductName DefaultCurrency CountryOfOrigin Manufacturer
Partner_Table	PartnerID ProductID RelationShip Role
Party_Table	PartyID PartnerID FirstName LastName
ProductDescription_Table	ProductID DescriptionPurpose Description
ProductPrice_Table	ProductID Amount Buyer
Specification_Table	ProductID TypeCode UOM SpecificationName Specification

객체 관계 데이터베이스 스키마

2장에서 제시된 규칙에 따라 최상위 노드인 ProductCatalog는 테이블로 생성된다. 그리고 하위 노드중 마지막 노드인 ProductName, DefaultLanguage, CountryOfOrigin, Manufacturer, Specification은 생성된 테이블의 필드를 이루게 된다. 하지만, Specification과 Manufacturer는 Attribute를 포함하고 있기 때문에 객체로 표현된다. 특히, Specification은 다중 속성을 가지고 있기 때문에 VARRAY로 테이블의 필드를 이룬다. 마지막 노드가 아닌 Partner, ProductDescription, ProductPrice 엘리먼트는 중간 노드로써 객체로 선언되고, 다중 속성을 갖고 있기 때문에 VARRAY로 상위 노드인 ProductCatalog의 필드를 이룬다. Partner 노드의 하위 노드인 Name은 중간노드로 객체로 선언되어 상위 객체인 Partner의 필드를 이룬다.

FirstName, LastName, Amount, Buyer등의 마지막 노드는 각각 상위 노드인 Name 엘리먼트와 ProductPrice 엘리먼트의 필드를 이룬다.

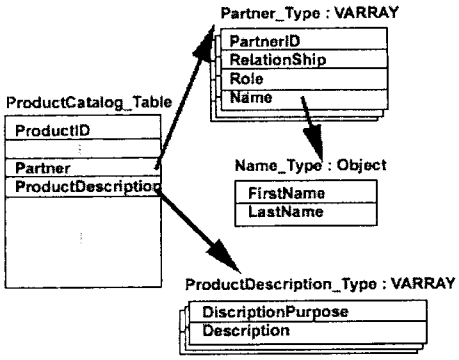
ProductID, PartnerID, RelationShip, Role, DescriptionPurpose, PartnerRef, TypeCode, UOM, SpecificationName등의 Attribute들은 각각을 포함하고 있는 엘리먼트의 필드를 이룬다. 표 2는 위의 과정에 의해 생성된 테이블과 테이블의 필드를 보여준다.

<표 2> 객체 관계 데이터베이스 스키마의 예

Table Name	Fields
ProductCatalog_Table	ProductID ProductName DefaultCurrency CountryOfOrigin Manufacturer of Object Type Partner VARRAY of Object Type ProductDescription VARRAY of Object Type ProductPrice VARRAY of Object Type Specification VARRAY of Object Type

표 2에 의해 객체 관계 데이터베이스에선 단 한 개의 테이블로 예제 XML DTD를 표현할 수 있었다. 그림 2는 생성된 객체 관계 데이터베이스 스키마의 구조의 예를 나타낸다. 테이블의 필드를 이루고 있는 각각의 객체들은 객체의 필드로 다른 객체를 포함하고 있음을 보인다. 따라서 데이터베이스 스키마에서 XML 문서의 구조를 유지 할 수 있다.

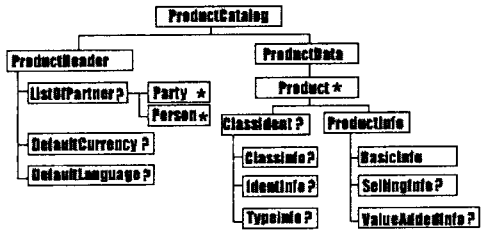
이상으로 XML 문서를 각각의 데이터베이스 스키마로 생성하는 과정을 보였다. 관계형 데이터베이스의 스키마 생성시 각 테이블간의 연관을 고려하여 6개의 테이블이 생성된 반면, 객체 관계 데이터베이스 스키마는 별도의 고려 없이 4개의 규칙에 따라 1개의 테이블로 구성이 가능했음을 보여준다. 또한 그림 2에서 보여주듯 스키마는 XML 문서의 구조를 포함하고 있음을 알 수 있다.



<그림 2> 객체 관계 데이터베이스 스키마 예

4. 구현 및 분석

그림 3은 우리가 지난 연구에서 제안했던 XML기반 통합상품표현 모델이다. 이번 장에서는 이 상품표현 모델을 적용하여 관계형 데이터베이스와 객체 관계 데이터베이스에 XML 문서를 저장하고 검색하는 시스템을 구현한다

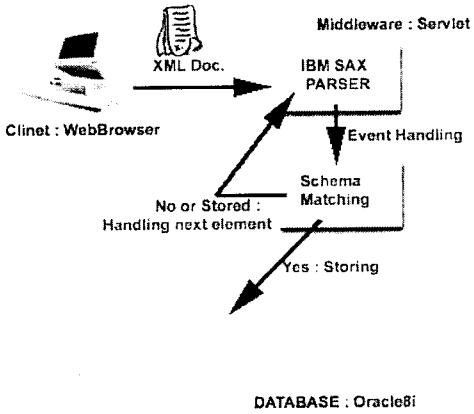


<그림 3> XML 기반 통합 상품 표현 모델

시스템을 위한 환경으로, 시스템 서버에는 Sun OS 5.7의 운영체제가 사용되었고, 상품 정보 저장을 위한 데이터베이스에는 Oracle8i[6]가 사용되었다. 또한, 웹 서버에는 Tomcat3.2.1 서버가 사용되었다.

제안된 상품표현모델에 의해 작성된 DTD에 의해 관계형 데이터베이스와 객체 관계 데이터베이스의 스키마를 생성하고, 생성된 스키마에 따라 저장 및 검색 시스템을 구현한다. 시스템은 Client와 Middleware, DataBase의 세 부분으로 구성되어 있다. Client는 사용자로부터 XML 문서를 입력받을 WebBrowser이고, WebBrower로부터 XML 문서를 전달받아 파싱하고 정보를 데이터베이스에 저장하기 위해 Java의 Servlet[7]을 사용하였다. DataBase는 Oracle8i를 사용하였다.

그림 4는 본 시스템의 구성도를 나타낸다.



<그림 4> 시스템의 구성 및 흐름

XML 문서의 저장시, 최초 사용자에 의해 WebBrowser에 입력된 XML 문서의 URL은 Servlet으로 전달된다. Servlet은 전달받은 주소의 XML 문서를 SAX파서[8]에 의해 파싱한다. 파싱을 위해 IBM에서 제공하는 SAX 파서를 사용하였다. SAX 파서의 특성상 문서가 파싱 과정 중에 XML 문서 내의 각각의 엘리먼트는 처리 가능한 상태가 된다. 이때 각각의 엘리먼트에 대해 데이터 베이스 스키마와 비교가 이루어지고, 각 단계에서 처리된 엘리먼트가 데이터베이스의 스키마와 일치하여 저장 조건과 만족된다면, 처리된 정보들을 인자로 데이터베이스에 저장하는 메소드를 수행한다. 저장 메소드에는 전달된 정보들로 저장 쿼리를 수행한다. 처리된 엘리먼트의 저장이 끝난 후 SAX 파서는 다음 엘리먼트의 처리를 시작한다. 파싱은 문서의 마지막 엘리먼트가 처리될 때까지 반복되며 문서의 파싱의 종료와 함께 시스템은 종료된다.

3장에서 구현한 두 개의 시스템에 실제 웹상의 상품 정보를 대상으로 문서의 저장시간과 정보의 검색시간을 비교해 보았다. 테스트 데이터는 웹 사이트 amazon.com에서 Computer 카테고리의 Java 키워드에 의해 검색된 도서들 중에서 40개를 순차적으로 선택하여 이를 제안된 모델에 따라 XML 문서로 제작하였다.

비교를 위해 각각의 시스템에 동일한 XML 문서를 저장하였다. 비교기준은 각 데이터베이스에 생성된 테이블의 개수와 정보 저장에 필요한 저장시간, 그리고 동일한 키워드에 의해 정보가 검색되는 시간이다. 표 3은 이러한 관점에서 두 데이터베이스를 비교한 결과이다.

<표 3> RDB와 ORDB의 비교

	# of Tables	Storage Time(Sec)	Retrieval Time(Sec)
RDB	21	33.225	0.0135
ORDB	4	10.575	0.0128
RDB/ORDB	525%	314%	105%

객체 관계 데이터베이스는 한 테이블의 컬럼으로 객체 또는 테이블이 올 수 있다. 객체는 관계형 데이터베이스의 하나의 테이블의 역할을 충분히 수행할 수 있다. 이는 객체 관계 데이터베이스의 특징으로, 이 특징에 의해 테이블수가 87.5%로 현격하게 줄어들어 525%의 효과를 보였다. 시스템에서 정보저장을 위해 데이터베이스에 수행시키는 저장 쿼리는 일반적으로 각 테이블 당 한 번의 수행을 필요로 한다. 위의 객체 관계 데이터베이스의 특징에 의해 현격하게 줄어든 테이블의 수는 역시 쿼리 수행의 수를 감소시켜 68.2%의 저장시간 감소로 314%의 효과를 가져왔다. 정보의 검색시간은 쿼리시 Join연산을 하지 않는 객체 관계 데이터베이스의 특징상 많은 감소를 가져올 것으로 예상했으나 내부적으로 Join연산을 수행하는 Oracle8i의 특성상 5.19%로 105%의 효과를 보여, 별다른 감소효과를 보지 못했다.

5. 결론

본 논문에서는 관계형 데이터베이스와 객체 관계 데이터베이스에 XML 문서를 저장시에 효율성을 분석하기 위해 XML DTD로부터 데이터베이스 스키마를 생성하는 과정의 테이블 수 및 구현의 편의성 등의 효율을 비교했고, XML 문서를 각 스키마에 따라 저장하는 시스템을 구현하고, 구현된 시스템에 실제 데이터를 적용하여 각 데이터베이스에 따른 문서 저장시간과 검색시간의 효율을 분석하였다.

분석의 결과로 객체 관계 데이터베이스가 관계형 데이터베이스보다 수월하게 시스템을 구축 할 수 있었고, 저장 및 검색시간 등의 단축을 가져옴으로써 XML 문서 저장에 더 효율적이란 결론을 내렸다.

[참고문헌]

[1]조현성 외 7인, "XML 기반 전자상거래 프레임워크 기술", 정보과학회지, 19권, 1호, p38, 2001. 1.

[2]"Extensible Markup Language (XML)"

http://www.w3.org/XML/, 2000.
[3]김경래, 하상호, 서건수, "XML 기반 통합 상품 표현 모델", 정보처리학회학술대회, 2001. 4.
[4]Steve Muench, Oracle XML Applications, Reading, O'REILLY, 2000. 11
[5]김경래, 하상호, "객체-관계 데이터베이스에의 XML 문서의 저장 시스템", 멀티미디어학회학술대회, 2000, 11.
[6]Kevin Loney, George Koch, Oracle8i:The Complete Reference, Reading, McGraw-Hill, 2000.5.
[7]James Goodwill, Developing JAVA Servlets, Reading, SAMS, 2001
[8]Hiroshi Maruyama, Kent Tamura, Naohiko Uramoto, XML and Java: Developing Web Applications, Reading , Addison-Wesley, 1999. 5.