Journal of Korea Multimedia Society Vol. 9, No. 8, August 2006(pp. 939-949)

PDTnet XML Schema 데이터를 위한 저장 기법 개발

이경혜[†], 이월영[††], 용환승[†††]

요 약

오늘날 산업의 발전에 따라 제품의 개발에서 생산 및 판매에 이르기까지의 모든 과정을 컴퓨터를 통해 일원적으로 관리할 수 있는 환경을 구축하는 것이 중요해지고 있다. 특히 PDTnet XML 스키마는 PDM (Product Data Management) 데이터를 교환하기 위한 국제 표준으로써, 이 스키마에 따라 작성된 XML 데이터를 효율적으로 처리하는 것이 필수적이다. 그러나 PDTnet XML 스키마는 기존의 XML 스키마와는 달리 IDREF/IDREFS 속성이 비정규적이기 때문에 이를 효율적으로 다루기 위해서는 어떠한 저장장치에 어떠한 방식으로 저장하느냐가 중요한 요인이라고 할 수 있다. 우리는 안정성 있는 성능을 보장 받고 있어 널리 사용되고 있는 관계형 데이터베이스를 이용하여, PDTnet XML 스키마의 고유 속성을 살려 다양한 타입의 질의를 지원할 수 있는 저장 기법을 개발하였다. 본 논문에서 개발하고 구현한 저장 기법은 PDTnet XML 스키마를 따르는 데이터 외에도, 구조가 불규칙적이거나, 참조 속성을 이용하여 데이터 중복을 최소화 시킬 필요가 있는 XML 데이터를 다루는 모든 응용에 적용되어 효율적인 데이터 관리를 가능케 할 수 있을 것으로 기대된다.

Development of Storage Techniques for PDTnet XML Schema Data

Kyoung-Hye Lee[†], Wol Young Lee[††], Hwan Seung Yong[†††]

ABSTRACT

With the development of industry, product data management system is becoming more and more important. An expanded view of product definition function that include a bill of material and routing database, current and historical engineering data and specifications and engineering changes order history. PDM (Product Data Management) systems hold and manage such material as product specifications, plans, geometric models, CAD drawings and images. Furthermore, PDM enables companies producing complex products to spread product data in to the entire launch process. Especially, PDTnet XML Schema is an international standard for exchanging of PDM data. But PDTnet XML Schema differs from existing XML Schema in the way that its property of IDREF/IDREFS is irregular. Therefore it is important factor that what do we use storage devices and storage techniques. We developed storage techniques and application supporting various query types and preserving PDTnet XML Schema using a relational database that guaranteeing the performance nowadays. In this paper, we will show that our storage techniques minimize repeated data and optimize query processing by using application comparison with storage techniques of existing XML Schema data.

Key words: XML Schema(XML스키마), reference(참조), IDREFS(IDREFS), Storage technique(저장기법)

※ 교신저자(Corresponding Author) : 이월영, 주소 : 서울시 은평구 신사동 1-19(122-879), 전화 : 02)380-2582, FAX : 02)380-2519, E-mail : wylee@scu.ac.kr(wylee@ewha. ac.kr)
접수일 : 2005년 9월 21일, 완료일 : 2006년 5월 2일
[†] 준회원, 삼성전자
(E-mail : you-rhee@hammail.net)
[††] 서울기독대학교 국제경영정보학과
[†††] 종신회원, 이화여자대학교 공과대학 컴퓨터학과
(E-mail : hsyong@ewha.ac.kr)
※ 본 연구는 한국과학재단의 연구 지원(KOSEF 20044014)에 의한 결과임.

1. 서 론

PDM 시스템은 제품 데이터와 워크플로우 관리를 위한 정보 시스템 개념을 총칭하는 것으로, 모든 제품 관련 정보를 체계적으로 관리할 수 있도록 하는 기능과 함께, 엔지니어들간의 정보 교환 생산성을 대폭 향상시킬 수 있는 기능을 보유하고 있다. 인터넷과 더불어 웹이 출현함으로써 대부분의 PDM 공급업체에서는 자사의 PDM 정보 시스템을 웹을 통하여 보다 많은 사람들이 쉽게 접근할 수 있는 기능의 모듈을 제공하고 있다. 한편 XML(eXtensible Markup Language)은 다양한 종류의 데이터를 인간이 생각하는 것과 같은 방법으로 손쉽게 표현할 수 있도록 하는 데이터 모델일 뿐 아니라 웹 상에서 데이터를 교환하기 위한 사실상의 표준이 되어 가고 있다. 이에 PDM 정보를 XML 형식으로 교환하려는 시도가 있었고 독일의 ProSTEP에서 PDM 데이터 교환을 위한 XML 기반의 국제 표준인 PDTnet(Product Data Technology and Communication in OEM and Supplier Network) XML 스키마를 발표하였다[8]. 그러나 PDTnet XML Schema는 기존의 XML Schema와는 달리 참조 속성이 매우 비정규적이다.

따라서 PDTnet XML Schema 데이터를 관계형 데이터베이스를 이용해 처리하기가 쉽지 않으며, XML 데이터를 관계형 데이터베이스에 저장하는 방법은 테이블의 조각화 현상이나 데이터 중복 등의 문제로 인하여 검색 시에 많은 조인을 유발하기 때문에 효율적으로 XML schema에 적용하기 어려운 실정이다. 그러나 관계형 데이터베이스는 가장 보편화되어 있는 저장 모델이고 저렴한 비용으로 웹 환경의 애플리케이션들을 쉽게 통합할 수 있도록 해주며, XML 문서를 구조화하여 관리할 수 있다는 장점을 지니고 있기 때문에 다른 저장 모델을 사용하여 XML schema 데이터를 관리하는 것보다 유리하다고 할 수 있다. 본 논문에서는 비정규적인 구조의 PDM 데이터를 관계형 데이터베이스를 이용하여 저장하되 테이블의 조각화 현상이나 데이터 중복을 최소화하여 검색할 때 조인의 횟수를 줄일 수 있도록 하였다. 또한 저장된 데이터를 빠르게 검색할 수 있는 응용 프로그램을 개발하였다.

본 논문의 구성은 서론에 이어 2장에서는 관련 연구를, 3장에서는 두 가지 저장 기법에 대해 설명하고 4장에서는 각 저장 기법에 따르는 성능을 평가하고 5장에서 결론을 맺는다.

2. 관련연구

다음의 그림 1과 그림 2는 기존의 XML 스키마와 PDTnet XML 스키마의 참조속성의 차이를 예제를 통해 비교한 것이다. 기존의 XML 스키마는 하나의 참조 속성이 한 종류의 엘리먼트만을 참조하여서 이를 관계형 데이터베이스에 저장하고 질의하기에 어려움이 없었다. 그러나 PDTnet XML 스키마에서는 하나의 참조 속성이 여러 종류의 엘리먼트들을 참조할 수 있어서 이를 관계형 데이터베이스에 저장하고 질의하기가 어렵다.

한편 많은 연구가들은 XML 문서를 다룰 때 관계형 데이터베이스의 안정성 있는 성능을 이용하고 이미 존재하고 있는 데이터와 자연스럽게 연계하고자 연구를 활발히 진행해 왔다[7]. 이러한 저장 방식으로는 애트리뷰트 인라인 저장 방식[2], STORED 저장 방식[3], 혼합 인라인 방식[4], XML-DBMS 기법

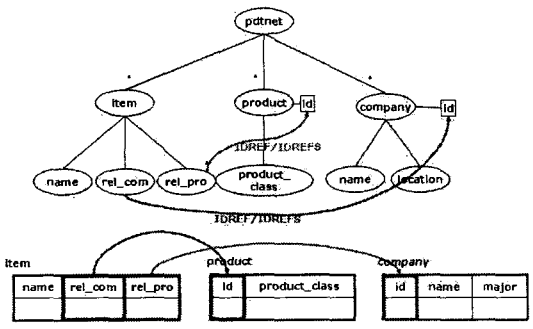


그림 1. 기존의 XML 스키마의 참조속성 예제

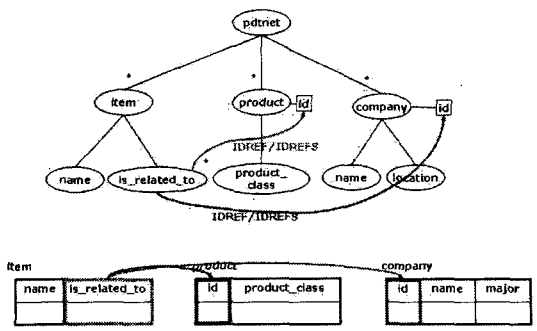


그림 2. PDTnet XML 스키마의 참조속성 예제

[1] 등이 있는데[9-11], 이 중에서 XML-DBMS 저장 기법은 특별히 DTD에서 추출될 수 있는 XML 문서의 구조를 관계형 스키마에 매핑하여 저장하는 방식으로, 저장 공간을 크게 낭비하지 않으며 관련된 데이터들을 같은 릴레이션에 저장하여 질의 시간을 감소시킬 수 있다[12].

그러나 XML-DBMS 저장 기법은 기존의 연구들과 마찬가지로 집합 값 애트리뷰트나 재귀적으로 반복되는 엘리먼트를 다른 릴레이션에 분리하여 저장하고 있다. 이러한 방식은 PDTnet XML 스키마의 IDREF/IDREFS와 같이 참조하는 엘리먼트가 여러 종류일 경우, 참조의 가능성이 있는 데이터베이스 내의 모든 테이블을 참조 엘리먼트의 횟수대로 스캔해야 한다. 따라서 PDTnet XML Schema의 참조 속성을 기존의 저장 기법으로 해결하는 것은 매우 효율적이지 못하며, 저장 공간의 낭비를 초래할 뿐 아니라 질의 처리 속도를 저하시키는 원인이 된다.

본 논문에서는 PDTnet XML schema 데이터의 저장을 위하여 테이블 조각화 현상을 최소화하는 저장 기법을 개발하고 저장된 데이터를 효과적으로 질의 처리할 수 있는 응용 프로그램을 개발하였다.

3. Key-Join 저장 기법과 String-Inlining 저장 기법

지금까지 연구된 관계형 데이터베이스를 기반으로 각 저장 기법들을 비교 분석해 본 결과, 이들은 PDTnet XML Schema 데이터에서 많은 부분을 차지하고 있는 IDREF/IDREFS나, 같은 이름의 자식 엘리먼트 수가 일정하지 않은 경우(unbounded)들을 효율적으로 다루고 있지 못하다. 따라서 관계형 데이터베이스를 기반으로 하되, 생성되는 테이블수가 적으며 데이터의 반복을 최소화 할 수 있는 새로운 저장 기법을 개발하여 Key-Join 기법과 String-Inlining 기법이라 명명하였다.

3.1 XML 문서의 데이터 모델

각 저장 기법을 비교하고 성능평가를 하기 위하여, PDTnet XML Schema 중 item 엘리먼트와 연관된 일부 엘리먼트들로 간단한 xsd 파일과 xml 예제 파일을 만들었다.

3.1.1 데이터 모델

다음의 그림 3은 PDTnet XML Schema의 일부이다. Pdtnet_schema가 최상위 엘리먼트이며 Item, Document 엘리먼트를 자식 엘리먼트로 갖는다. Item 엘리먼트는 Item_version, 이 중 Document_assignment 엘리먼트의 하부 엘리먼트인 Assigned_document 엘리먼트는 IDREFS 타입으로, Document, Document_file, Document_representation, Document_version 엘리먼트들 가운데 한 가지 엘리먼트를 선택적으로 참조할 수 있다. 아래의 그림 4는 그림 3의 XML 스키마에 대한 예제이다. Item 엘리먼트의 자식 엘리먼트인 Item_version에는 Document_assignment 엘리먼트가 두 번 나타난다. 두 Document_assignment의 Assigned_document 엘리먼트는 각각 Document 엘리먼트와 Document_version 엘리먼트를 참조한다.

3.1.2 DOM으로 모델링 된 XML 문서

그림 5은 그림 4의 예제를 DOM으로 모델링 한 모습이다. 굵은 점선의 화살표는 각 IDREF/IDREFS가 참조하고 있는 엘리먼트를 가리키고 있다.

3.2 Key-Join 기법

Key-Join 기법은 하위 엘리먼트들을 갖는 엘리먼트들을 클래스로 변환하고, 하위 엘리먼트들은 해당 클래스의 애트리뷰트에 매핑하며 부모와 자식 엘리먼트들은 데이터베이스에서 기본키(primary key)와 외래키(foreign key)로 관계를 나타낸다. 이 기법은 부모 테이블이 unbounded 속성의 자식 엘리먼트들에 관한 정보는 갖고 있지 않으며 IDREF/IDREFS 속성의 자식 엘리먼트들의 참조 id 값만을 갖고 있다. unbounded 속성의 자식 엘리먼트들은 부모 테이블에 자신의 정보가 없는 대신, 자신의 테이블에 부모의 id를 갖고 있어서 부모 테이블과 자식 테이블의 연결이 기본키와 외래키로 이루어진다.

3.2.1 Key-Join 기법을 이용한 데이터베이스 저장 모습

Key-Join 기법을 이용하면, 부모 테이블이 unbounded 속성의 자식 엘리먼트 및 그 하부 정보를 갖고 있지 않고 unbounded 속성의 자식 엘리먼트 테이블이 부모 테이블의 id를 외래키로 갖고 있어 이

를 통해 두 테이블 간의 연결이 이루어진다. 그림 6의 Pdtnet_schema 테이블은 Pdtnet_schema 엘리먼트의 애트리뷰트인 Id, Version_id 컬럼으로 구성되어 있다. Pdtnet_schema 엘리먼트의 자식 엘리먼트들 중 unbounded 속성인 Document, Item 엘리먼트는 Document과 Item 테이블에 부모 엘리먼트의 id값을 fid 컬럼을 할당하여 저장한다.

3.3 String-Inlining 기법

String-Inlining 기법은 부모 테이블이 unbounded 속성의 자식 엘리먼트들의 id 값들과 IDREF/IDREFS 속성의 자식 엘리먼트들의 참조 id 값들을 모두 갖고 있다. 자식 엘리먼트들의 id 값들은 ';'으로 구분된 하나의 문자열로 묶여 부모 테이블의 한 필드에 저장된다. 따라서 String-Inlining 방식은 부모 테

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="http://www.w3.org/2001/xml.xsd" />
    <xs:element name="Pdtnet_schema" type="Pdtnet_schema" />
    <xs:complexType name="Pdtnet_schema">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="Document" type="Document" />
          <xs:element name="Item" type="Item" />
        </xs:choice>
        <xs:attribute name="id" type="xs:ID" use="required" />
        <xs:attribute name="version_id" type="xs:string" use="required" />
        <xs:attribute ref="xml:lang" use="optional" />
    </xs:complexType>
    <xs:complexType name="Document">
        <xs:sequence>
          <xs:element name="Description" type="String_select" minOccurs="0" />
          <xs:element name="Name" type="String_select" />
          <xs:element name="Document_id" type="xs:string" />
          <xs:element name="Document_version" type="Document_version"
           maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
    </xs:complexType>
    <xs:complexType name="Document_assignment">
        <xs:sequence>
          <xs:element name="Assigned_document" type="xs:IDREFS" />
          <xs:element name="Role" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
    </xs:complexType>
    <xs:complexType name="Document_version">
        <xs:sequence>
          <xs:element name="Description" type="String_select" minOccurs="0" />
          <xs:element name="Id" type="xs:string" />
          <xs:element name="Document_representation" type="Document_representation"
           minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
    </xs:complexType>
    <xs:complexType name="Item">
        <xs:sequence>
          <xs:element name="Id" type="xs:string" />
          <xs:element name="Name" type="String_select" />
          <xs:element name="Description" type="String_select" minOccurs="0" />
          <xs:element name="Item_version" type="Item_version" maxOccurs="unbounded" />
          <xs:element name="Document_assignment" type="Document_assignment"
           minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
    </xs:complexType>
    <xs:complexType name="Item_version">
        <xs:sequence>
          <xs:element name="Id" type="xs:string" />
          <xs:element name="Description" type="String_select" minOccurs="0" />
          <xs:element name="Design_discipline_item_definition"
           type="Design_discipline_item_definition" minOccurs="0"
           maxOccurs="unbounded"/>
          <xs:element name="Document_assignment" type="Document_assignment"
           minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
    </xs:complexType>
</xs:schema>
```

그림 3. PDTnet XML schema의 일부

```
<Pdtnet_schema version_id="1.8" id="ps:01" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Pdtxml schema.xsd">
    <Item id="ii:1">
        <Id>i1</Id>
        <Name>supplied_Item</Name>
        <Item_version id="iiv:1">
            <Id>Item_version id1</Id>
            <Document_assignment id="ida:1">
                <Assigned_document>id:2 id:3</Assigned_document>
                <Role>Latest document description</Role>
            </Document_assignment>
            <Document_assignment id="ida:2">
                <Assigned_document>idv:1 idv:2</Assigned_document>
                <Role>Latest version</Role>
            </Document_assignment>
        </Item_version>
    </Item>
    <Document id="id:1">
        <Name>pdtnet</Name>
        <Document_id>Document id 1</Document_id>
    </Document>
    <Document id="id:2">
        <Name>igi</Name>
        <Document_id>Document id 2</Document_id>
        <Document_version id="idv:1">
            <Id>document v1</Id>
        </Document_version>
        <Document_version id="idv:2">
            <Id>document v2</Id>
        </Document_version>
    </Document>
    <Document id="id:3">
        <Name>Ewha</Name>
        <Document_id>Document id 3</Document_id>
    </Document>
</Pdtnet_schema>
```

그림 4. Item.xml 데이터

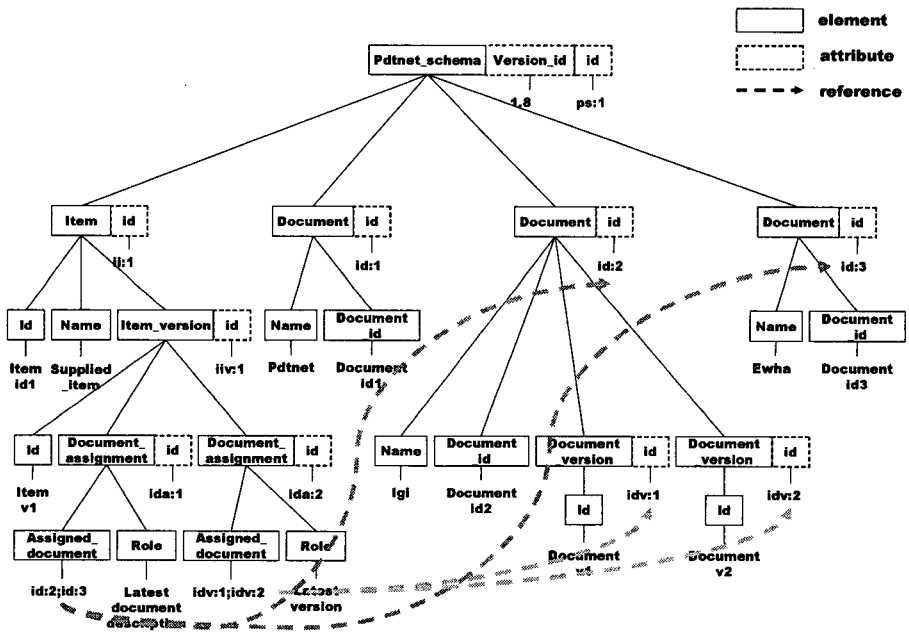
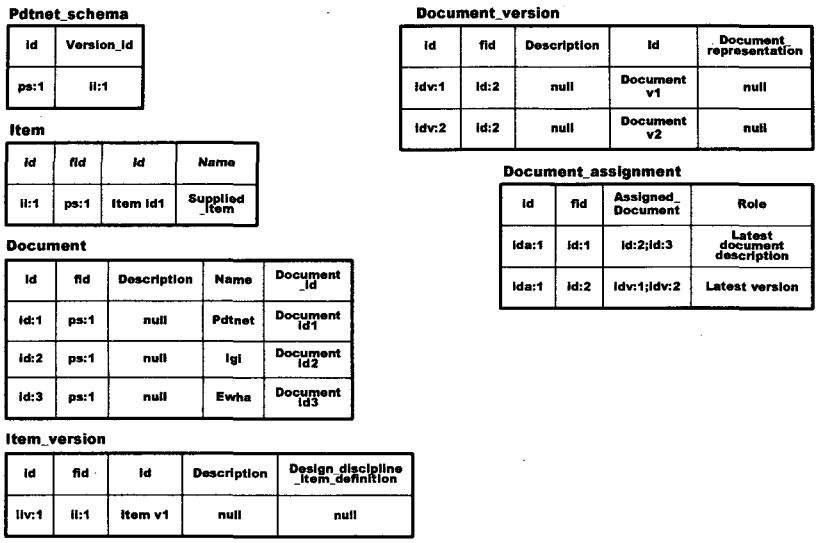


그림 5. DOM으로 모델링 된 XML 문서

이블과 자식 테이블이 다-대-다 관계를 형성할 수 있는 방식이므로 XML 문서의 내용을 간단하면서도 알아보기 쉽게 표현할 수 있다.

3.3.1 데이터 모델

그림 3과 그림 4의 XML 스키마와 XML 예제 파일을 사용하여 String-Inlining 기법으로 데이터베이스



Pdtnet_schema

Id	Version_Id
ps:1	ii:1

Item

Id	fid	Id	Name
ii:1	ps:1	Item id1	Supplied _Item

Document

Id	fid	Description	Name	Document _id
id:1	ps:1	null	Pdtnet	Document id1
id:2	ps:1	null	Igi	Document id2
id:3	ps:1	null	Ewha	Document id3

Item_version

Id	fid	Id	Description	Design_discipline _Item_definition
iiv:1	ii:1	Item v1	null	null

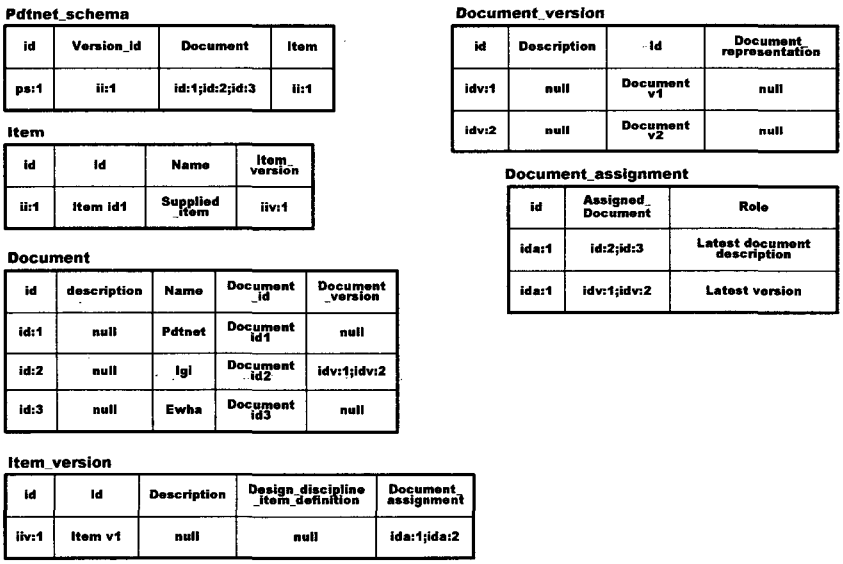
Document_version

Id	fid	Description	Id	Document representation
idv:1	id:2	null	Document v1	null
idv:2	id:2	null	Document v2	null

Document_assignment

Id	fid	Assigned_ Document	Role
ida:1	id:1	id:2;id:3	Latest document description
ida:1	id:2	idv:1;idv:2	Latest version

그림 6. Item.xml 데이터를 Key-Join 기법을 이용하여 RDB에 저장한 모습



Pdtnet_schema

id	Version_Id	Document	Item
ps:1	ii:1	id:1;id:2;id:3	ii:1

Item

id	Id	Name	Item_ version
ii:1	Item id1	Supplied _Item	iiv:1

Document

id	description	Name	Document _id	Document _version
id:1	null	Pdtnet	Document id1	null
id:2	null	Igi	Document id2	idv:1;idv:2
id:3	null	Ewha	Document id3	null

Item_version

id	Id	Description	Design_discipline _item_definition	Document_ assignment
iiv:1	Item v1	null	null	ida:1;ida:2

Document_version

id	Description	Id	Document representation
idv:1	null	Document v1	null
idv:2	null	Document v2	null

Document_assignment

id	Assigned_ Document	Role
ida:1	id:2;id:3	Latest document description
ida:1	idv:1;idv:2	Latest version

그림 7. Item.xml 데이터를 String-Inlining 기법을 이용하여 RDB에 저장한 모습

에 저장하였을 때의 모습이다. 그림 7의 Pdtnet_schema 테이블은 Pdtnet_schema 엘리먼트의 애트리뷰트인 Id, Version_id 컬럼과 자식 엘리먼트들 중 unbounded 엘리먼트 속성인 Document, Item 컬럼으로 구성되어 있다. 자식 엘리먼트들 중 unbounded 엘리먼트들의 엘리먼트 값은 발생된 엘리먼트의 id값을 ';'으로 구분하여 하나의 문자열로 한 필드에 입력한다.

4. 성능 평가

성능 평가는 크게 두 부분으로 나뉘는데, 참조 엘리먼트의 종류 및 개수에 따른 질의 시간을 XML-DBMS, String-Inlining의 두 가지 기법으로 측정한 것과, String-Inlining과 Key-Join 기법간의 다양한 성능 비교로 이루어진다.

4.1 IDREF/IDREFS 처리 방식에 따른 질의 시간

4.1.1 참조하는 엘리먼트의 종류가 하나인 경우

String-Inlining 기법은 IDREF/IDREFS의 참조 데이터를 부모 테이블에 그대로 저장하는 반면, XML-DBMS 기법은 기존의 다른 기법들과 마찬가지로 다른 테이블에 분리하여 저장한다. 이에 본 성능평가에서는 IDREF/IDREFS 속성이 한 종류의 엘리먼트만을 참조할 때, 참조되는 엘리먼트의 수가 증가함에 따라, XML-DBMS 기법과 String-Inlining 기법의 질의 처리 속도가 어떻게 변하는지를 비교하였다.

참조되는 엘리먼트의 수를 1개에서 100개까지 증가 시키면서 질의 처리 속도를 측정해본 결과, 참조 엘리먼트의 수가 증가 할수록 String-Inlining 기법이 XML-DBMS 기법보다 훨씬 빠른 것으로 나타났다. 이는 XML-DBMS 기법이 참조 속성의 데이터를 다른 릴레이션에 분리하여 저장하기 때문에, 부모 테이블과 참조 속성 테이블의 조인이 참조 엘리먼트의 개수만큼 추가되어 일어나기 때문이다.

4.1.2 참조하는 엘리먼트의 종류가 하나 이상인 경우

PDTnet XML Schema의 IDREF/IDREFS는 기존의 XML Schema와는 달리 하나의 참조 속성이 여러 엘리먼트를 참조할 수 있다는 특성을 가진다. 기존의 저장 기법들은 IDREF/IDREFS의 집합 값 애트리뷰트를 분리된 릴레이션에 저장하는 반면, Key-Join, String-Inlining 기법은 하나의 필드에 string으로 저장한다. 다음은 하나의 엘리먼트나 애

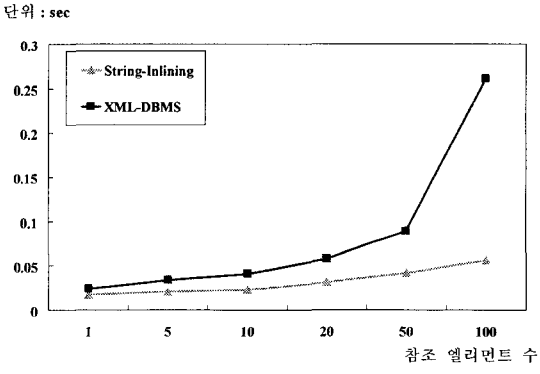


그림 8. 참조하는 엘리먼트의 종류가 하나인 경우

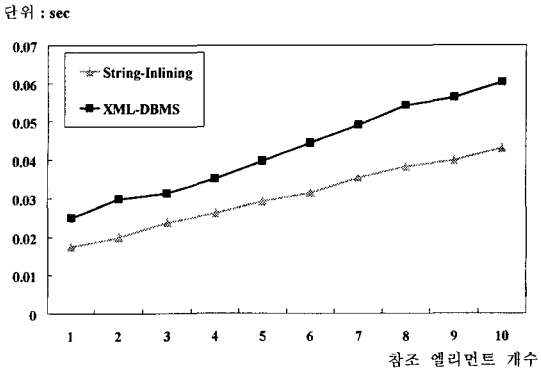


그림 9. 참조하는 엘리먼트의 종류가 하나 이상인 경우

트리뷰트가 참조 할 수 있는 엘리먼트 종류의 개수에 따른 저장 기법들의 질의 시간을 측정한 것이다. 참조되는 엘리먼트 종류의 개수를 1에서 10까지 증가시켰으며, 각 엘리먼트는 한번씩 참조된다.

위의 결과에서 알 수 있듯, String-Inlining 기법이 XML-DBMS 기법보다 항상 더 빠른 질의 시간을 보이고 있으며, 참조 엘리먼트 종류의 수가 증가할수록 차이는 더 커진다.

한 엘리먼트나 애트리뷰트가 참조할 수 있는 엘리먼트가 복수 개일 경우, 기존의 관계형 데이터베이스 접근법에서는 참조 엘리먼트의 종류만큼 별도의 테이블을 만들거나, 질의 시 모든 참조 가능 엘리먼트에 대해서도 한꺼번에 질의해왔다. 그러나 PDTnet XML Schema v1.8은 참조 가능 엘리먼트의 종류가 약 50개 이상인 IDREF/IDREFS가 다수 존재하므로 기존의 방법을 사용하는 것은 저장공간의 낭비는 물론, 질의 시 많은 오버헤드가 생기게 된다.

4.2 데이터베이스 크기 측정 및 비교

최적의 저장 기법을 결정하는 것은 저장된 데이터를 얼마나 효율적으로 관리할 수 있는가에 따라 평가될 수 있다. 본 논문에서는 데이터베이스에 저장된 문서 형태에 따른 데이터베이스 크기와 질의 시간을 평가 기준으로 삼았다. XML document의 크기, 각 매핑 스키마의 데이터 크기와 인덱스 크기를 모두 측정하였으며, 데이터베이스 크기는 unbounded 속성의 엘리먼트의 발생 횟수를 1회, 6회, 12회, 18회, 24회로 증가시키면서 측정하였다.

표 1. unbounded 속성의 엘리먼트의 발생 횟수에 따른 데이터베이스 크기

		XML file	Key-Join	String-Inlining
occurrence 1	Base data	770	328	344
	Indices	-	736	368
	Total	770	1064	712
occurrence 6	Base data	852	400	408
	Indices	-	736	368
	Total	852	1136	776
occurrence 12	Base data	909	600	592
	Indices	-	736	368
	Total	909	1336	960
occurrence 18	Base data	970	648	648
	Indices	-	736	368
	Total	970	1384	1016
occurrence 24	Base data	1030	680	744
	Indices	-	736	368
	Total	1030	1416	1112

아래 그래프는 인덱스 크기를 제외한 순수 데이터 크기만을 비교한 것이다.

Key-Join 기법과 String-Inlining 기법은 모두 데이터베이스 크기의 최소화를 고려하여 설계한 것이므로 예측대로 두 schema의 기본 데이터에 대한 저장 크기는 거의 차이가 없었다. 다만 인덱스의 크기는 Key-Join 기법의 외래키(Foreign key) 컬럼에 검색속도의 향상을 위하여 인덱스를 생성하였기 때문에 String-Inlining 기법에 비해 두 배에 가까운 크기를 보이고 있다. Key-Join 기법은 외래키(Foreign key) 필드가 추가되어 각 행에 id값이 하나씩 저장되며, String-Inlining 기법은 반복되는 id값이 하나의 필드에 ';'로 구분되어 일렬로 저장된다는 차이가 있다. 그러나 이 차이가 데이터베이스의 크기에는 거의 영향을 주지 않기 때문에 실제 두 스키마의 저장 크기 차이는 없다고 봐도 무방하다.

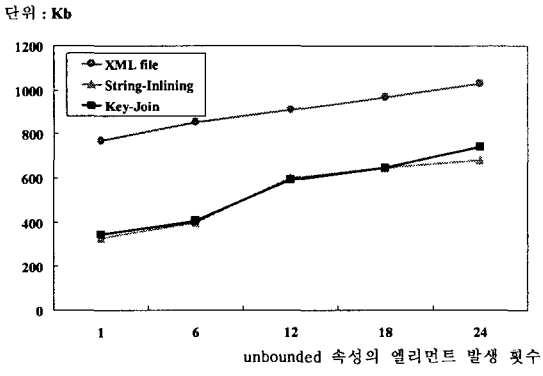


그림 10. unbounded 속성의 엘리먼트 발생 횟수에 따른 XML file과의 데이터베이스 크기 비교

4.3 질의 비용 측정 및 비교

4.3.1 질의 처리기

데이터베이스에 저장된 PDM 데이터를 검색하고 이를 XML 파일로 생성하기 위해 PDTnet Query handler를 구현하였다. 사용자는 질의 처리기에 데이터베이스 접속 정보와 질의문을 입력할 수 있으며, 필요에 따라 해당 테이블의 데이터만 가져오거나 선택한 테이블과 연관된 하위 테이블들 참조하는 테이블의 데이터들까지 모두 가져올 수 있다. 또한 질의 처리기를 통하여 가져온 데이터를 XML 문서로 자동 변환 할 수도 있다. 본 논문에서는 우리가 구현한 PDTnet Query handler를 통해 질의 비용을 측정하였다.

4.3.2 단일 발생 엘리먼트에 대한 질의 비용 측정 및 비교

본 성능 평가에 사용될 질의 유형은 다음과 같다.

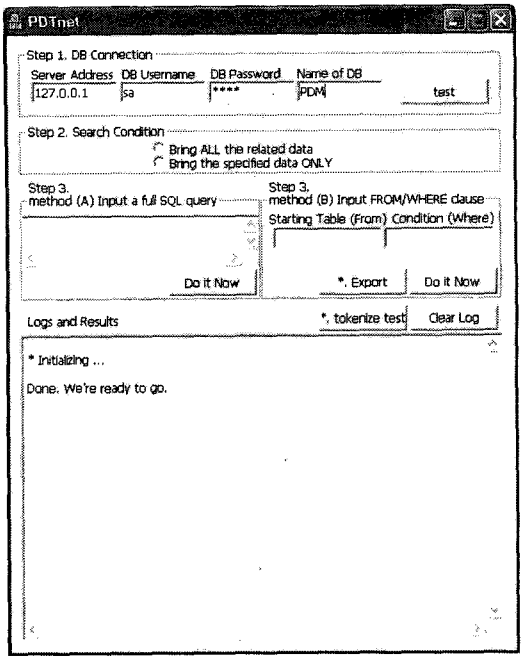


그림 11. PDTnet Query handler의 초기화면

아래의 질의 유형은 P. Griffiths Selinger가 [5]에서 제시한 관계형 데이터베이스의 질의 유형 분류를 일부 사용한 것이다.

본 논문에서는 위의 질의유형에 대해 예제 질의를 적용하여 성능평가를 수행하였다. 단위 시간은 ms이며, 모두 5회에 걸친 결과 값의 평균을 기록하였다.

여섯 개의 질의 유형에 대한 Key-Join 기법과 String-Inlining 기법의 질의 시간을 비교해보면, 전반적으로 비슷하나, 삽입와 조인 시에 String-Inlining 기법이 더 좋은 성능을 보임을 알 수 있다. Key-Join 기법에서는 외래키(Foreign key)가 있어서 삽입 시 참조 무결성을 체크하는 시간이 걸리므로

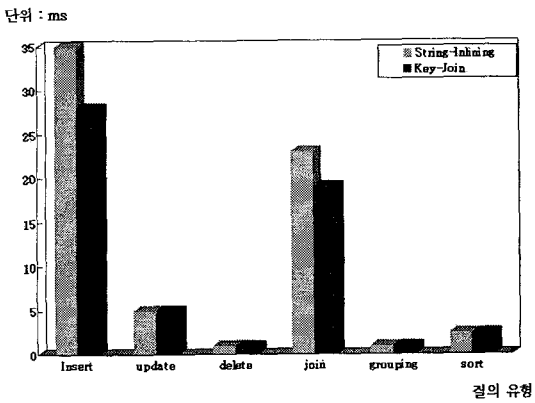


그림 12. 단일 발생 엘리먼트에 대한 질의 비용

String-Inlining 기법 보다는 느린 질의 시간을 보인다. 조인 시간에서도 String-Inlining 기법이 더 나은 질의 시간을 보이는데, 이는 String-Inlining 기법이 pid를 통해 데이터를 직접 찾기 때문이다.

4.3.3 복수 발생 엘리먼트에 대한 질의 비용 측정 및 비교

'단일 발생 횟수의 엘리먼트에 대한 질의 비용'에서 삽입과 조인 질의에 String-Inlining 기법이 Key-Join 기법보다 더 나은 성능을 보였다. 그러나 실제 PDM 데이터는 집합 값 애트리뷰트나 반복되는 엘리먼트들, 참조하는 엘리먼트가 다수개인 IDREF/IDREFS가 발생될 경우가 많기 때문에 이들의 발생 횟수에 따른 성능을 측정 할 필요가 있다.

표 3은 성능평가를 위해 사용한 질의 유형이다. Q1 질의는 item의 id 값이 주어졌을 때 item 테이블의 해당 데이터만을 출력하는 것이며, Q2 질의는 id 값에 해당하는 item 테이블의 데이터와 하위 엘리먼트 테이블의 데이터를 모두 출력하는 것이다.

표 2. 성능 평가를 위한 질의 유형

질의 유형	예제 질의
Insert	Q1. 23개의 테이블 모두에 들어갈 애트리뷰트들을 저장하여라.
Update	Q2. Address의 Facsimile_number 값이 87415인 것을 43045로 수정하여라.
Delete	Q3. Item의 id가 'item id2'인 것을 삭제하여라.
Join	Q4. Item의 pid가 'ii:3'인 Item_version의 description을 찾아라.
Grouping Query	Q5. Date_time의 date가 '2005-01-01'인 것을 세어서 출력하라.
Sort	Q6. Date_time의 date가 '2005-01-11'인 것을 pid 순서대로 정렬하여 출력하라.

표 3. 성능 평가를 위한 질의 유형

질의 유형	예제 질의
Traversal-based Query	Q1. Item의 pid가 ii:5인 item 테이블의 데이터를 출력하여라.
	Q2. Item의 pid가 ii:6인 모든 사항을 출력하여라.

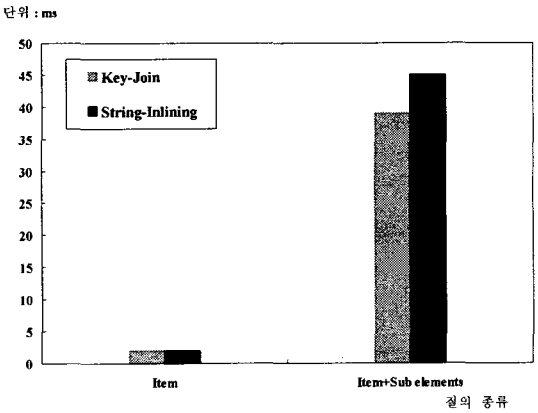


그림 13. 복수 발생 엘리먼트에 대한 질의 비용

item의 id값을 통해 item 테이블만의 내용을 가져오는 시간과 해당 id에 대해 item의 하위 엘리먼트들까지 모두 가져오는 시간을 측정해본 결과 Key-Join 기법이 15% 정도 더 나은 성능을 보였다. 이는 String-Inlining 기법이 읽어 들인 부분의 데이터를 분석하여 그 하위 엘리먼트를 읽는 질의를 생성해나가는 방식이기 때문에 Key-Join 기법보다 높은 질의 횟수를 가지기 때문이다. 그러나 Key-Join 기법은 처음에 프로그램으로부터 주어진 질의를 DBMS 내에서 최적화 할 수 있으므로 더 좋은 성능을 보였다.

5. 결 론

우리는 본 논문에서 PDTnet XML Schema의 특징을 잘 고려하여 이를 관계형 데이터베이스에 저장하는 두 가지 기법을 제안하였다. 또한 전용 질의 처리기를 개발함으로써, 데이터를 효율적으로 검색할 수 있으며 질의도 편리하게 할 수 있도록 하였다. 우리의 저장 기법은 PDTnet XML Schema의 불규칙적이고 반복적인 구조를 처리하는데 있어, 기존의 저장 기법들에 비해 데이터의 반복을 최소화 했을 뿐 아니라, 질의 시에 조인으로 처리하는 기존의 방식에 비해 훨씬 더 효율적이기 때문에 질의 속도를 향상시킬 수 있도록 하였다. 따라서 우리의 저장 기법과 애플리케이션은 기존의 저장기법에 비해 PDTnet XML Schema 데이터를 관계형 데이터베이스에서 훨씬 효과적으로 사용할 수 있으며 다른 데이터와도 자연스럽게 연계하여 재사용할 수 있다.

참 고 문 헌

[1] R. Zwol, P. Apers, and A. Wilschut, "Modeling and querying semistructured data with MOA," *in Proceedings of the Workshop On Query Processing for Semistructured Data and Non-standard Data Formats*, Jerusalem, Israel, 1999.

[2] D. Florescu and D. Kossmann, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," *Technical Report, INRIA*, No. 3680, 1999.

[3] D. suciu, A. deutsch, and M. Fernandex, "Storing Semi-structured Data with STORED," *in Proceedings of ACM SIGMOD international conference on Management of data*, pp. 431-442, 1999.

[4] J. Shanmuhasundaram, K. Tyfte, G. He, C. He, C. Zhang, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Documents : Limitations and Opportunities," *In Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 302-314, 1999.

[5] P. G. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, "Access Path Selection in a Relational Database Management System," *in Proceedings of the ACM SIGMOD international conference on Management of data*, pp23-34, 1979.

[6] P. Smith, D. Reinertsen, *Developing Products*

in Half the Time, Van Nostrand Reinhold, New York, USA, 1998.

[7] S. Abiteboul, P. Buneman, and D. Sucui, *Data on the Web - From Relations to Semistructured Data and XML*, Morgan Kaufmann, New York, USA, 1999.

[8] http://www.prostep.de/en/services/projekte/pdtnet/.

[9] http://www-306.ibm.com/software/data/db2/extenders/xmlext/.

[10] http://www.oracle.com/technology/oramag/oracle/ 04-sep/o54xml.html.

[11] http://www.rpbourret.com/xmldbms/readme.htm.

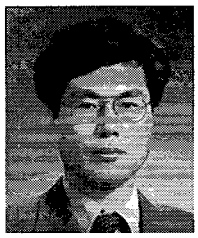
[12] 김지심, "XML 문서의 저장기법에 대한 성능평가," 이화여자대학교, 2001.

이 경 혜

2004. 2 이화여자대학교 기독교학과 학사
2006. 2 이화여자대학교 컴퓨터학과 공학석사
현재 삼성전자

이 월 영

1988. 2 이화여자대학교 전자계산학과 학사
2000. 2 이화여자대학교 과학기술대학원 컴퓨터학과 공학석사
2005. 2 이화여자대학교 과학기술대학원 컴퓨터학과 공학박사
2005. 9~현재 서울기독대학교 국제경영정보학과 교수
관심분야: XML 데이타베이스, 정보검색, 데이타 마이닝



용 환 승

1983. 2 서울대학교 컴퓨터공학과 학사
1985. 2 서울대학교 컴퓨터공학과 공학석사
1994. 2 서울대학교 컴퓨터공학과 공학박사
1995~현재 이화여자대학교 컴퓨터학과 교수
관심분야: 객체관계 데이타베이스, 데이타 마이닝,유비쿼터스 데이타베이스