XML Database를 위한 스키마 설계 방법론 개발

최문영, 주경수
순천향대학교 공과대학 컴퓨터학부

Developing a Design Methodolgy for XML Database

Choi Mun-Young, Joo kyung-Soo
Dept. of Computer Science and Engineering, College of Engineering
Soonchunhyang Univ. Asan 336-745. Korea

요 약

XML을 이용하면 이기종 컴퓨팅 환경으로 구성되어 있는 웹 상에서 정보를 공유할 수 있었고, 이제 XML은 정보가 아닌 프로세스를 공유할 수 있는 아주 단순하면서도 유연한 방법을 제공해 주고 있다. 이러한 XML 기술을 기반으로 하는 웹 서비스와 ebXML을 이용하면 어떤 어플리케이션은 물론 어떤 비즈니스 프로세스 역시 웹 상에서 통합될 수 있다.

기업용 애플리케이션에 XML을 이용하는 일이 점점 늘어남에 따라 많은 조직들이 XML 문서를 저장하고 관리하는 문제에 직면하고 있다. 문제는 이미 많은 기업들이 이들 XML 문서를 저장하는데 기존에 사용하던 관계형 데이터베이스를 계속 사용해도 되는 것으로 생각하고 있다는 것이다. 이것은 XML 데이터를 다루는데 많은 문제를 야기한다. 관계형 데이터베이스는 XML 같은 확장성 데이터를 다루도록 설계되지 않았다는 태성적 한계가 있기 때문이다. 그러므로 본 논문에서는 XML Database 스키마 3단계 설계방법론을 이용하여 이러한 문제점을 해결하려한다.

1. 서론

새로운 XML database의 개발에 있어서 어떻게 DBMS에서 데이터를 획득해야 되는지 이해하는 것은 중요하다. 사용자와 DBMS사이에 가장 많이 영향을 미치는 상호작용은 데이터베이스에 대한 스키마를 디자인하는 것에 있을 것이다. 스키마는 데이터베이스의 자료가 보이는 것처럼 기술된다. 스키마는 데이터베이스에 대한 청사진이며 데이터베이스의 내용을 기술한다.

스키마 디자인은 데이터베이스에 접근하는 애플리케이션을 개발하는 것을 이해하는 또 다른 이유이다. 데이터베이스 스키마를 위한 훌륭한 디자인도 데이터베이스를 사용하는데 필요한 설명이다.

이 논문에서는 데이터베이스를 디자인하는 방법을 표현한다. 데이터베이스 디자인 처리는 개념적 디자인, 논리적 디자인, 물리적 디자인의 세 가지 단계로 이루어져 있다.

2. 데이터베이스 디자인

데이터베이스 디자인은 멀티단계 처리이다 그리고 각 단계에서는 서로 다른 영역을 요구한다. 어떠한 영역에서는 기술, 구조, 직물 디자인, 그래픽 아트, 또는 요구사항을 이해하고 계획하는 것을 요구한다. 데이터베이스 개발에서 가장 중요한 상태는 도메인이다. 어떠한 디자인에서 선택된 모델링 언어는 중요한 형식과 디자인 차이의 정도에 따라 모델링 언어의 능력에 크게 영향을 미칠 수 있다.

논리적 설계 처리에서 데이터 모델 중에 수학적으로 정밀한 논리는 프레임워크로 선별된다. 비록 도메인의 몇몇 차이의 정도가 자주 상실되도, 결과적으로 볼 때 데이터베이스 모델링은 능률적이고 유용하다.

물리적 디자인은 수학적으로 문제가 없는 데이터 모델을 제작하는 것이다. OS와 DBMS 대다수 교환, 항목, 그리고 최적화는 이 단계에서 해결된다.
물리적인 디자인은 전체 설계에 영향을 미치지 않고 논리 데이터 모델에 포함된다. 그러나, 이 항목은 데이터베이스를 작용하는데 중요하다. 개념적 설계는 새로운 건물을 설계하기 위하여 흥미로운 대화를 여는 것과 유사하다. 논리적 설계는 계획 약간 청사진을 개발하고 다이어그램을 설계하는 것과 유사하고 물리적 설계 단계는 건물의 돌을 쌓는 방법과 유사하다.

디자인 3 단계의 언어로 XML을 사용한다.

① 개념적 모델링 언어로 XML을 사용하고 관계형 또는 객체지향 데이터베이스를 생성할 수 있다.
② eXcelon과 같은 XML DBMS에 사용되는 논리 데이터

모델링 언어로 XML를 사용한다.

③ 데이터베이스를 생성하기 위하여 XML 스키마를 사용하는 것과 같이 물리적 데이터 모델링 언어의 스키마를 지정할 때 XML을 사용한다.

3. 개념적 모델링

많은 개념적 언어는 도메인의 본질에 초점을 맞추는데 노력한다. XML을 위해 개념적 모델링 언어는 XML의 계층보다 구조적인 탄력성을 많이 가지고 있어야 하고 트리의 컬렉션으로 XML 문서의 수학적 추상화를 보존해야 한다. 또한, 개념적 언어는 논리 XML 데이터 모델에 대한 언어와 다르지 않다. 이처럼, 개념적 모델링은 쉽게 할 수 있다. XML 언어 그래프를 정의하는 정밀한 수식체계로의 변환은 요구사항을 직면하게 된다. 그리고 개념적 모델 그래프는 이 장에서 정의한다.

개념적 모델링 그래프는 개념과 도메인의 관계성을 캡쳐하는 것으로 사용된다. 개념적 모델링 처리의 목표는 도메인 정보를 추출하는 데이터베이스 개발을 위해 있다. 개념과 관계성은 그래프의 반복을 세분하여 도메인의 본질을 충분히 캡쳐한다.

그림 1은 개념과 도메인에 대하여 간단한 개념적 스키마 그래프이다.

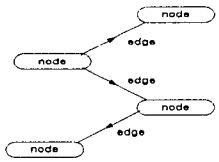

그림 1. 간단한 개념적 스키마

개념과 관계성은 그래프의 노드로 캡쳐되고 문자로 라벨이 붙는다. 구체적인 객체, 잘 알려진 개념, 그리고 분명히 정의된 관계는 그래프의 노드로 모두 캡쳐된다. 개념은 그래프의 모서리로 특성을 기술하는 것에 의해서 나눠진다. 그래프 개념적 모델은 개념과 도메인의 관계성이 그래프라는 형식적인 시스템이다. 도메인을 그래프로 표현하는 방법은 다음과 같다.

① 그래프는 개념, 레벨 모서리, 그리고 카디널리티의 컬렉션에 있다.

② 두 개의 개념과 모델, 개념의 특성 그리고 도메인의 2진 관계는 라벨 모서리에 연결한다.

③ 카디널리티 한 쌍은 완전한 정수 모서리로 연상된다. 카디널리티는 정수대신에 조건 "Many(다수)"로 표현한다. 다수 카디널리티는 0 또는 더 많은 출현을 언급하는 것이다. 그러나 그 카디널리티 "one-or-more"는 "+"에 의해서 나타내어진다. 만약 카디널리티가 없으면, 그것은 "1"로 가정한다.

3.1 개념적 모델링 처리 그래프

7단계 처리를 통한 개념적 모델 그래프를 개발할 수 있다. 이 처리에서 하나의 이점은 데이터베이스 시스템에 대한 최소 지식으로 실행될 수 있는 것이다. 그리고 결과로서 생긴 스키마는 전문 훈련 없는 도메인 전문가들에 의해 이해될 수 있다.

① 리스트 도메인 개념과 복잡한 관계성 개념은 실사회의 객체, 관계성, 또는 이벤트이다. 문장 구성상, 도메인의 설명에 대한 명사이다.

② 도메인에 관계성을 기술한 단문을 생성하는 것으로 도메인 개념을 결합한다.

③ 리스트로부터 도메인에 주요한 개념을 선택한다. 개념은 포함되어야 한다. 그것은 도메인 또는 그것에 기본 진동이 Step 2에 생성된 단문의 몇몇에 일어난다.

④ 개념 스키마에 노드로 주요한 개념을 그린다. 워드 주위의 타원형의 노드로 주요한 개념을 그리기 위하여 종이 또는 white-board의 공백 시트를 사용한다.

⑤ 그래프에서의 모서리로 단문을 그린다. 도메인 개념의 특성을 그릴 2가지 방법은 있다.

⑥ 모서리에 카디널리티 상태를 추가 그리고 도메인에 본질적인 그래프에 카디널리티 상태를 추가한다. 논리적 설계를 처리하는 동안에 카디널리티 상태를 추가 할 수 있다.

⑦ 도메인을 조사하여 스키마를 수정한다.

예를 들어, 혼성체 필터에 대한 도메인을 만들고 그래프를 그려보면 다음과 같다.

1. 도메인 개념과 복잡한 관계성의 리스트
   - 실험장치, 실험장치 ID, 날짜, 주석, 사용자, 사용자 이름, 사용자 특권, 사용자 번호, 실험실, 탐침, 탐침 ID, 혼성, 장소, 장소 ID, 필터, 필터 ID, 날짜, 클론, 클론 ID
2. 도메인에 관계성을 기술한 단문을 생성하는 것으로 도메인 개념을 결합한다.
   - 실험장치는 사용자가 있다.
   - 실험장치는 생성된 날짜가 있다.
   - 실험장치는 ID를 가지고 있다.
   - 실험장치는 주석을 가지고 있다.
   - 사용자는 실험실을 가지고 있다.
   - 사용자는 이름을 가지고 있다.
   - 사용자는 특권을 가지고 있다.
   - 사용자는 사용자번호를 가지고 있다.
   - 실험장치는 탐침을 가지고 있다.
   - 실험장치는 필터를 가지고 있다.
   - 탐침은 ID를 가지고 있다.
   - 탐침은 혼성을 가지고 있다.
   - 실험장치는 혼성을 가지고 있다.
   - 필터는 ID를 가지고 있다.
   - 필터는 생성된 날짜를 가지고 있다.
   - 필터는 장소를 가지고 있다.
   - 장소는 ID를 가지고 있다.
   - 장소는 클론을 가지고 있다.
   - 클론은 ID를 가지고 있다.
   - 혼성은 클론을 가지고 있다.
3. 리스트로부터 도메인에 주요한 개념을 선택한다.
   - 실험장치, 사용자, 탐침, 혼성, 장소, 필터, 클론
4. 개념 스키마에 노드로써 주요한 개념을 그린다. 그림 2 보여진다.
5. 그래프에서의 모서리로써 단문을 그린다. 그림 3 보여진다. 각각의 문장은 하나의 모서리가 된다.
6. 모서리에 카디널리티 상태를 추가한다. 그림 4 보여진다.
7. 도메인을 조사하여 스키마를 수정한다. 추가적인 정보는 각각의 도메인에 대한 ID, 날짜, 이름, 또는 사용자번호 등등 추가할 수 있다.

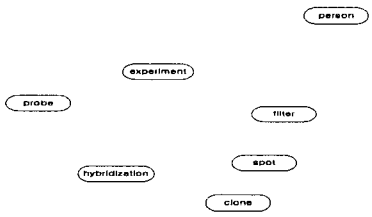

그림 2. 혼성체 필터 개념 그래프

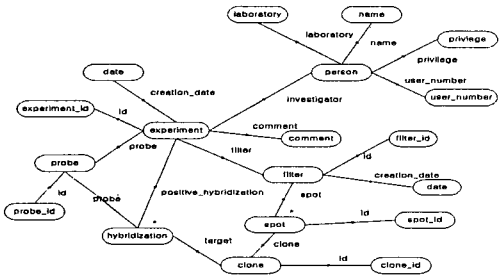

그림 3. 혼성체 필터 개념 스키마 그래프

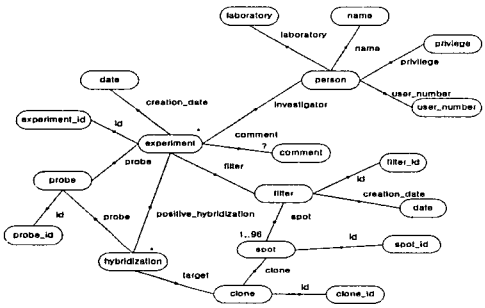

그림 4. 카디널리티 상태를 가진 필터 혼성체 스키마

3.2 XML 개념적 모델

XML을 개념적 모델링 언어로 사용하는 것을 이 절에서 기술한다. XML은 상속 없는 계층을 구조화된 도메인 집합의 정보를 캡처하기 위한 개념적 모델링 언어로 사용될 수 있다. 예제 1은 유용한 모델로 다음과 같이 적당한 규정으로 개발될 수 있다.

① XML 개념 스키마는 XML 엘리먼트의 컬렉션으로 이루어져 있다.

② 빈 속성 값은 제한이 없는 속성 값을 표현한다.

예제 1. 빈 XML로 표현된 혼성체 필터 예제

```
<experiment id=" " creation_date=" " comment=" ">
  <investigator>
    <person name=" " priviege=" " user_number=" ">
      <laboratory/>
    </person>
  </investigator>
  <probe id=" ">
  <filter id=" " creation_date=" ">
    <spot id=" ">
      <clone id=" "/>
    </spot>
  </filter>
  <positive_hybridization>
    <hybridization>
      <probe/>
      <target>
        <clone/>
      </target>
    </hybridization>
  </positive_hybridization>
</experiment>
```

4. 논리적 모델링

논리적 설계는 DBMS의 데이터 모델에 대한 스키마에 개념 스키마를 그리는 처리이다. 도메인이 데이터베이스 디자이너에 의해서 이해될 때, DBMS의 데이터 모델은 유일한 모델링 언어로 자주 사용된다. 그러나, 어떤 개념적 모델이 사용될 때, 논리적 설계 단계 부분의 방법이 논리 스키마에 개념 스키마를 번역하는 것을 기본으로 도메인을 정확하게 캡처하고 논리 데이터 모델의 구조물을 효과적으로 사용한다. 논리 데이터 모델은 개념적인 모델보다 기능성과 이상에 더 많이 제한적이다.

4.1 객체 모델

개념 스키마 그래프에서 객체 모델까지 변환하기 위하여 다음 처리가 사용된다.

① 카디널리티에 대한 더 많은 특정한 것으로 인하여 개념적인 모델을 개정한다. 만약 그것들이 알려지면 숫자의 범위 또는 세트를 사용한다. 그리고 0 또는 이상인 다수 상태를 구별한다.(별표로 표시한다. "*"). 임의의 특성들은 "0", "0..1", "?", "opt", 또는 dashed-edge 선으로 표시될 수 있다.

② 특성을 가지고 있는 각각의 개념은 클래스 다이어그램에서 클래스가 된다.

③ 개념의 각 특성에 대하여, 만약 그것의 카디널리티가 1보다 더 많다면 또는 추가적인 도메인 정보가 연결이 생성되는 것을 제안한다면 연결된다. 만약 그렇지 않으면, 만약 그것의 카디널리티가(다중도) "1" 또는 옵션이면 특성은 속성이 된다.

④ 특성을 가지고 있지 않은 개념은 속성 또는 연결이 된다. 이 정리가 되지 않은 개념은 새로운 클래스, 내장 클래스, 또는 부분형이 될 수도 있다.

⑤ 다양한 구성의 개념에 대하여 일반화는 유용할 것이다.

객체 모델에 그래프 개념 스키마를 변환하기 위하여 중간 다이어그램을 생성하는 것은 매우 도움이 될 수 있다. 확장된 카디널리티의 추가는 도메인 객체 다이어그램과 집합 트리에 유용한 2개의 추가적인 다이어그램이다. 도메인 객체 다이어그램이 개념 스키마에 개념을 변환시키는 것은 도메인의 중심이다.

도메인 객체 다이어그램은 단순히 다이어그램으로부터 없어진 중요한 개념과 개념 스키마 그래프이다. 그것은 도메인에 객체 사이의 관계성을 나타낸다. 하나의 객체가 도메인의 존재를 위하여 또 다른 하나에 의존할 때, 관계성은 집합 다이어그램에서 변환된다. 집합은 도메인에 객체와 다른 객체사이의 관계성이다. 객체의 설명은 다른 객체에 바탕을 둔다. 집합 다이어그램은 개념적인 다이어그램에서의 속성 역할을 다한

도메인 객체를 기술한다.

필터 혼성체 예제에 대한 개념 스키마 그래프는 그림 3에 보여진다. 더 많은 특정 카디널리티 상태는 그림 4에 보여지는 것처럼 규정될 것이다. 개념 스키마로부터, 그 주변의 개념은 그림 5의 도메인 객체 다이어그램을 생성하기 위하여 삭제된다. 집단화 관계성으로부터, 그림 6에 집단화 트리로 생성된다.

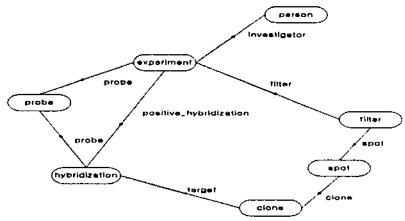

그림 5. 필터 혼성체 실험을 위한 도메인 객체 다이어그램

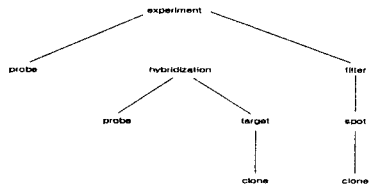

그림 6. 필터 혼성체 실험을 위한 집단화 트리

개념적인 다이어그램, 카디널리티 정보, 도메인 객체 다이어그램, 그리고 집단화 트리를 그리는 것은 많은 결정의 객체 모델에 대하여 만들어질 필요가 있다. 이 결정은 Unified Modeling Language (UML)같은 객체를 위한 모델링 언어일수 있다. 필터 혼성체를 위한 UML 다이어그램은 그림 7에 주어진다.

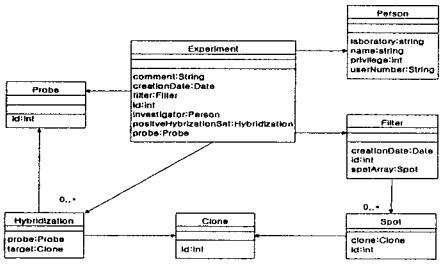

그림 7. 필터 혼성체 실험을 위한 UML 다이어그램

UML은 객체지향 디자인들의 다이어그램을 그린 표준 방법이다. 그 기법이 각 접근의 장점을 결합시킨 방법론에 의해서 초기의 객체지향 설계 방법론으로부터 나왔고 Object Modeling Group(OMG)에 의해서 표준화되었다. 이 논문에서는 UML의 클래스 다이어그램을 사용한다. 다이어그램은 클래스명, 속성, 그리고 동작으로 구성된다. 다이어그램도 또한 클래스의 특징, 연결에 대한 카디널리티 상태, 그리고 일반화 사이의 관계성을 나타낸다. 연결은 단순한 화살표 그림으로 표시된다. 그리고 일반화는 열린 삼각 화살표 그림으로 표시된다.

4.2 XML 논리 스키마

일반적인 스키마 언어는 논리적 설계 단계에서 XML을 모델링할 수 없다. Document Type Definition(DTD)과 XML Schema Definition Language(XSDL)를 위해 언어는 물리적 설계에서 더욱 적합하다. 그러나, 문서형식 또는 문서 스키마 정의가 다음과 같이 설명된 처리로부터 생성될 수 있다. 논리 XML 스키마의 생성은 이전 장의 처리 모델을 만드는 객체로부터 다이어그램 사용을 원할 수 있다. XML 논리 스키마는 그래프 개념 스키마에서 다음 단계를 통해 생성된다.

① 각각의 방식은 같은 이름으로 엘리먼트 형이 된다.

② 모서리 없는 개념을 특정하게 나타낼 때, 속성 또는 보조 엘리먼트 타입이다. 만약 특성이 도메인 존재에 대한 개념을 결정하고 카디널리티 1:1 또는 Many:1을 가지고 있으면 속성이 된고 그렇지 않으면 보조 엘리먼트가 된다.

예제 2. XML 엘리먼트를 사용하는 표현의 혼성체 필터 예제

```
<?xml version="1.0"?>
<schema>
   <element name="experiment">
      <element name="investigator">
         <element name="person">
            <element name="laboratory"/>
            <attribute name="id"/>
            <attribute name="creation_date" type="date"/>
            <attribute name="comment" minOccurs="0"/>
         </element>
      </element>
      <element name="probe">
         <attribute name="id" type=string"/>
      </element>
      <element name="filter">
         <element name=""spot: minOccurs="1" maxOccurs="96">
            <element name="clone">
               <attribute name="id"/>
            </element>
            <attribute name="id"/>
         </element>
         <attribute name="id"/>
         <attribute name="creation_date" type="date"/>
      </element>
      <element name="positive_hybridization">
         <element name=""hybridization" maxOccurs="unbounded">
            <element name=""probe"/>
            <element name="target">
               <element name="clone"/>
            </element>
         </element>
      </element>
   </element>
</schema>
```

스키마가 형식을 포함한 문서의 컬렉션을 기술하기 위하여 "empty" XML 문서를 사용한다. 스키마 예제 2는 XSDL의 문법에 기초를 두었다. 예제 3은 XSL이 템플릿에 처리 명령을 많이 추가하지만 형 정보와 빈 문서를 증가시키기 위하여 XML Namespaces를 사용한 두 가지 예제의 혼합 접근이다.

예제 3. 빈 XML을 표현한 필터 혼성체 예제

```
<?xml version="1.0"?>
<schema xmlns:xs="http://www.xweave.com/xmlns/xmldb/xsl">
   <experiment>
      <investigator>
         <person>
            <xs:attribute name="id"/>
            <xs:attribute name="creation_date" type="date"/>
```

```
            <xs:attribute name="comment" minOccurs="0"/>
            <laboratory/>
        </person>
    <probe>
        <xs:attribute name="id"/>
    </probe>
    <filter>
        <xs:attribute name="id"/>
        <xs:attribute name="creation_date" type="date"/>
        <spot xs:minOccurs="1" xs:maxOccurs="96">
            <xs:attribute name="id"/>
            <clone>
                <xs:attribute name="id"/>
            </clone>
        </spot>
    </filter>
    <positive_hybridization>
        <hybridization xs:maxOccurs="unbounded">
            <probe/>
            <target>
                <clone/>
            </target>
        </hybridization>
    </positive_hybridization>
</experiment>
</schema>
```

5. 물리적 모델링

물리적 설계는 사용되는 DBMS로 연상되는 구현 언어의 구조물로 일반적인 논리 데이터 모델의 스키마를 능률적으로 변환시키는 처리이다.

물리적 모델링은 디자인의 실용적인 표시 양상을 보존한다. 다양한 인수는 자료의 크기, 기밀 보호, 그럴듯한 접근 패턴, 자화율의 변화, 가능성의 재사용 그리고 필수 접근 속도를 포함하여 물리적 설계에게 영향을 미친다. 부가적인 물리적 모델링 데이터베이스에 발생되는 문제에 추가하여 데이터베이스 그리고 데이터 교환을 위해서 사용되는 XML을 위해 물리적인 스키마 생성이 추가적인 문제로 발생한다.

5.1 XML 물리적 스키마

물리적 설계를 만드는데 필요한 하나의 선택을 사용하기 위하여 XML 문서의 부분과 어떤 메커니즘으로서 스키마를 명백히 규정하는지를 결정한다. XML DBMS에서 스키마를 규정할 때, 결정은 선택된 DBMS에 의해서 결정된다.

스키마를 규정하는데 DTD를 사용한다. DTD는 데이터베이스에 대하여 일반적인 형 상태의 단계를 제공할 수 없기 때문에, XML Schema는 개발되었다.

스키마를 디자인하기 위하여 또 다른 메커니즘을 사용하여야 한다. 스키마를 디자인한 후에, 데이터베이스에 대한 XSDL에 물리적인 스키마 정의를 생성할 수 있다. DTD를 사용하여 예제 4에 표현하고 XSDL 스키마는 예제 5에서 표현했다.

예제 4. 혼합체 필터 XML DTD 표현 예제

```
<!ELEMENT experiment (investigator, probe, filter,
positive_hybridization*)>
<!ATTLIST experiment
    id CDATA #REQUIRED
    creation_date CDATA #REQUIRED
    comment CDATA #IMPLIED>
<!ELEMENT investigator (person)>
<!ELEMENT person (laboratory)>
<!ATTLIST person
    name CDATA #REQUIRED
    privilege CDATA #REQUIRED
    user_number CDATA #REQUIRED>
<!ELEMENT laboratory ANY>
<!ELEMENT probe EMPTY>
<!ATTLIST probe
    id CDATA #REQUIRED><!ELEMENT filter (spot*)>
<!ATTLIST filter
    id CDATA #REQUIRED
    creation_date CDATA #REQUIRED>
<!ELEMENT spot (clone)>
<!ATTLIST spot
    id CDATA #REQUIRED>
<!ELEMENT clone EMPTY>
<!ATTLIST clone
    id CDATA #REQUIRED>
<!ELEMENT positive_hybridization (hybridization)>
<!ELEMENT hybridization (probe, target)>
<!ELEMENT target (clone)>
```

예제 5. 혼합체 필터 XML Schema 표현 예제

```
<schema>
<complexType name="cloneType">
    <attribute name="id" type="string"/>
</complexType>
<complexType name="probeType">
    <attribute name="id" type="string"/>
</complexType>
<element name="experiment">
<complexType>
    <sequence>
        <element name="investigator">
            <complexType>
                <sequence>
                <element name="person">
                    <complexType>
                        <sequence>
                        <element name="laboratory">
                            <complexType>
                                <any/>
                            </complexType>
                        </element>
                        </sequence>
                        <attribute name="id" type="string"/>
                        <attribute name="creation_date" type="date"/>
                        <attribute name="comment" type="string"
minOccurs="0"/>
                    </complexType>
                </element>
                </sequence>
            </complexType>
        </element>
        <element name="probe" type="probeType"/>
        <element name="filter">
            <complexType>
                <sequence>
                <element name="spot" minOccurs="1" maxOccurs="96">
                    <complexType>
                        <sequence>
                            <element name="clone" type="cloneType">
                            <attribute name="id" type="string"/>
                        </sequence>
                    </complexType>
                </element>
                </sequence>
                <attribute name="id" type="string"/>
                <attribute name="creation_date" type="date"/>
            </complexType>
        </element>
        <element name="positive_hybridization">
            <complexType>
                <sequence>
                    <element name="hybridization"
maxOccurs="unbounded">
                    <complexType>
                        <sequence>
                        <element name="probe" type="probeType">
                        <element name="target">
                            <complexType>
                                <element name="clone" type="cloneType">
                            </complexType>
                        </element>
                        </sequence>
                    </complexType>
                </element>
                </sequence>
            </complexType>
        </element>
    </sequence>
</complexType>
</element>
</schema>
```

자료 전송을 위하여 XML을 사용하는 것의 단점은 문서에서의 태그가 공간을 차지하는 것이다. 문서에서 자료를 가지고 있을 때 고정, 규칙적인 구조, 태그가

중복된다. 예제 6에서, 문서의 태그는 자료가 15%으로 문자공간이 85%을 나타낸다.

자료의 크기를 줄이는 것은 애플리케이션에의 전송 성능을 개량한다. 데이터베이스에 기억이 필요한 공간을 감하거나, 문서를 검색하는 시간을 감하거나 또는 애플리케이션과 서버에 필요한 데이터베이스로부터 질의 결과나 자료를 분석하는 사용자 애플리케이션에 반응을 증가시키고, 메모리를 바꾼다.

예제 6. 규칙적인 구조의 XML 파일

```
<?xml version="1.0"?>
<colors>
   <color name="Black" red="0" green="0" blue="0"/>
   <color name="Silver" red="C" green="C" blue="C"/>
   <color name="Gray" red="8" green="8" blue="8"/>
   <color name="White" red="F" green="F" blue="F"/>
   <color name="Maroon" red="8" green="0" blue="0"/>
   <color name="Olive" red="8" green="8" blue="0"/>
   <color name="Yellow" red="F" green="F" blue="0"/>
   <color name="Navy" red="0" green="0" blue="8"/>
   <color name="Blue" red="0" green="0" blue="F"/>
   <color name="Teal" red="0" green="8" blue="8"/>
   <color name="Aqua" red="0" green="F" blue="F"/>
</color>
```

6. 결론

인터넷은 새로운 정보 처리 방식을 요구한다. 기업과 기업 간의 비즈니스 네트워크의 관문을 개방하여 정보는 이제 날씨같은 것이 되어 가고 있다. 네트워크 안에서 이들은 스스로 표현하고, 취소하고, 변경하고, 제거한다. 컨텍스트와 데이터가 동시에 존재하고 이들을 실시간으로 변경하고 확장할 수 있는 XML의 출현은 정보를 보다 풍부하게 그리고 보다 동적으로 만들었다. 이런 인터넷의 새로운 데이터들을 처리하기 위해서 이 논문에서는 XML Database를 설계하는 개념적, 논리적, 물리적 방법을 제안했다.

XML 데이터베이스는 비즈니스 정보 시스템에의 적응능력을 가능하게 한다. 이들 XML 데이터베이스 시스템들은 비지니스 프로세스와 데이터베이스 계층을 모두 확장하는 XML의 핵심 요소를 지원한다. 선두 시스템들은 기업이 보다 쉽고 저렴한 비용으로 변화되는 비즈니스 요구를 수용할 수 있도록 하여 준다

[참고문헌]

[1] Extensible Markup Language(XML)1.0, http://www.w3.org/TR/1998/REC-xml-19980210

[2] Liam Quin, Open Source XML Database Toolkit, WILEY, 2000

[3] 'XML and Databases', Ronald Bourret, Technical University of Darmstadt, September, 1999, HTTP://www.informatik.tudarmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm