# AVASWI 시스템을 위한 저장소의 설계 및 구현

*유남현° **정강용 *김원중
*순천대학교 컴퓨터과학과 **순천제일대학 컴퓨터과학과
kwj@sunchon.ac.kr

# The Design and Implementation of Repository for AVASWI System

*Namhyun Yoo° **Gangyong Jung *Wonjung Kim
*Dept. of Computer Science, Sunchon National University
**Dept. of Computer Science, Suncheon First College

요 약

AVASWI 시스템이 기존의 아바타 시스템과 다른 점은 아바타를 표현하기 위하여 GIF나 Flash등을 이용하지 않고 W3C의 2D 그래픽 표준인 SVG를 이용하는 것이다. SVG는 XML의 서브셋으로서 SVG를 저장하기 위해서는 XML을 위한 저장소를 이용할 수 있으나, SVG는 기존의 다른 XML과 다르게 속성과 그에 대한 값이 매우 중요한 역할을 수행한다. 이에 본 논문에서는 SVG를 효과적으로 저장하기 위하여 기존의 Shared-Inlining방법을 확장한 Extended-Inlining방법을 설계, 구현하였다.

## 1. 서 론

AVASWI(AVAtar System on Wired and Wireless Internet using svg) 시스템은 정보통신부 산하 정보통신연구진흥원의 지원하에 2004년 7월부터 개발중인 멀티 플랫폼 기반의 아바타 시스템이다. AVASWI 시스템이 기존 시스템과 다른 점은 아바타를 표현하기 위한 파일로 GIF, Flash를 사용하지 않고 W3C(World Wide Web Consoritum)의 SVG를 사용한다는 것이다[1]. SVG의 장점은 텍스트로 구성된 이미지 파일이라는 점과 벡터 그래픽이라는 점이다. 이미지를 텍스트로 표현하기 때문에 포토샵과 같은 무거운 그래픽 어플리케이션을 사용하지 않고도 이미지 편집이 가능하며, 온라인상에서도 편집이 가능하다. 또한 벡터 그래픽이기 때문에 핸드폰용으로 만든 이미지를 PC 환경에서도 변경 없이 사용할 수 있다. 이러한 특징을 이용하여 유·무선 인터넷을 지원하는 아바타 시스템을 구현할 수 있다. 그러나 SVG는 기존의 XML과는 다르게 엘리먼트 내의 속성들이 중요한 역할을 수행한다. 이에 본 논문에서는 아바타를 표현하는 SVG 파일을 효과적으로 저장하기 위하여 기존의 Shared-Inlining 기법을 확장한 Extended-Inlining 기법을 적용한 저장소를 설계, 구현하였다.

## 2. 관련연구

1990년대까지 대부분의 정보 시스템에서는 정보를 저장하고 교환하기 위하여 관계형 데이터베이스 시스템을 많이 사용하였으며, 비디오 파일, 오디오 파일, 3D 동영상 등을 저장하는 특정 분야에서만 일부 객체지향형 데이터베이스 시스템을 이용하기도 하였다. 그러나, 2000년대의 정보시스템에서는 정보를 저장하고 교환하기 위한 수단으로 XML 파일을 이용하게 됨으로서 XML 파일을 효과적으로 저장하고 관리하기 위한 다양한 연구 방법들이 제안되었으며, 기존의 데이터베이스 벤더들도 XML 파일을 저장하기 위한 다양한 방법들을 적용하고 있으며, Tamino와 같은 XML 전용 데이터베이스 시스템을 이용하여 적용하는 경우도 있다.

XML 파일을 저장하기 위한 연구는 기존의 파일 저장 시스템을 이용하거나, 관계형 데이터베이스 시스템을 이용하는 방법, 객체지향형 데이터베이스 시스템을 이용하는 방법, XML 전용 저장 시스템을 이용하는 방법 등 크게 4가지로 구분할 수 있으며, 각 구분된 방법 안에서도 다양한 연구 방법들이 제안 및 테스트되어 지고 있다.

현재 오라클, DB2와 같은 대부분의 상용 데이터베이스 시스템에서는 XML 파일을 저장하기 위한 방법으로 크게 XML 문서를 엘리먼트(Element)와 어트리뷰트(Attribute)들로 분해한 후 관계형 모델에 매핑하여 사용하는 방법과 CLOB나 BLOB형태로 저장한 후 B-Tree 인덱스를 제공하는 방법을 이용하고 있다.

### 2.1 관계형 데이터베이스를 이용한 XML 저장

관계형 데이터베이스를 이용하여 저장하는 방법으로는 Inlining, Edge, Binary, Universal, Monet 방법 등이 있으나 대부분 Inlining 방법을 많이 이용한다. Inlining 방법에는 Basic Inlining, Shared Inlining, Hybrid Inlining 방법 등이 있으나 대부분 Shared Inlining 방법을 많이 이용한다[2][3]. [표 1]은 한 XML 파일의 DTD이며, [그림 1]은 해당 DTD를 이용하여 XML 파일을 관계형 데이터베이스에 매핑한 예이다. 관계형 데이터베이스에 XML 파일을 매핑하기 위해서는 제안된 XML 파일에서 DTD나 XML Schema 구조를 분석한 후, 다음과 같은 규칙을 따른다.

① 한 개 이상의 in-degree를 가지는 노드에 대한 테이블을 각각 생성한다.
② 한 개의 in-degree를 가지는 노드에 대한 테이블은 따로 생성하지 않는다.
③ 0개의 in-degree를 가지는 노드의 경우에는 부모

* 본 논문은 정보통신부 정보통신연구진흥원에서 지원하고 있는 2004 정보통신기초연구지원사업의 연구결과입니다.

노드로부터 연결된 패스를 찾을 수 없기 때문에 한 개 이상의 in-degree를 가지는 노드의 경우와 같이 따로 테이블을 생성한다.

④ DTD 그래프의 간선(edge)중에서 '*'를 가지는 간선의 경우에 간선이 가리키는 노드가 한 개 이상이라는 것을 의미하므로, 간선이 가리키는 노드에 대한 테이블을 따로 생성한다.

⑤ DTD 그래프에서 하나의 테이블을 생성한 노드로부터 유도된 경로(Directed Path)에 포함되는 노드들은 모두 생성된 테이블의 애트리뷰트로 인라인 시킨다. 하지만, 유도된 경로에 '*'가 포함되어 있지 않는 경우에만 해당한다.

```
<?xml?>
<!ELEMENT Dept (Student)*>
<!ATTLIST Dept dept_id ID #REQUIRED>
<!ELEMENT Student (Name,Enroll*)>
<!ATTLIST Student student_id ID #Required>
<!ELEMENT Name #PCDATA>
<!ELEMENT Enroll #PCDATA>
```

[표 1] XML 파일의 DTD

```
<?xml version="1.0"?>
<!DOCTYPE Dept SYSTEM "Dept.dtd">
<Dept dept_id="detp1">
  <Student student_id="123">
    <Name>St1</Name>
    <Enroll>CS10</Enroll>
    <Enroll>CS20</Enroll>
  </Student>
  <Student student_id="124">
    <Name>St2</Name>
  </Student>
</Dept>
```

[표 2] [표 1]의 DTD를 사용하는 XML 파일

| ParentID | ID | Dept_id |
|---|---|---|
| 1 | 2 | "dept1" |
| The Dept table | | |

| ParentID | ID | TEXT |
|---|---|---|
| 3 | 5 | "CS10" |
| 3 | 6 | "CS20" |
| The Enroll table | | |

| ParentID | ID | Student_id | Name |
|---|---|---|---|
| 2 | 3 | "123" | "St1" |
| 2 | 4 | "124" | "St2" |
| The Student table | | | |

[표 3] [표 2]을 Inlining 방법으로 관계형 데이터베이스 매핑

2.2 객체지향형 데이터베이스를 이용한 XML 저장

객체지향형 데이터베이스에 데이터베이스를 저장하는 방법은 대부분 XML 파일을 분해하여 엘리먼트 단위로 저장하여 사용하거나, 파일 자체를 CLOB나 BLOB 형태로 저장한 후 그에 대한 검색 속도의 향상을 위하여 B-Tree형태의 인덱스를 구성하여 제공한다. [표 4]는 [표 2]의 XML 파일을 B-Tree인덱스로 구성하는 일부분이다.

| Offset | Record |
|---|---|
| 0 | Length=40, Dept, parent=nil, prev=nil, next=nil<br>first_child=40,last_child=140, attr(dept_id="dept1") |
| 40 | Length=40, Student, parent=0, prev=nil, next=140<br>first_child=80, last_child=120, Attr(student_id="123") |
| ⋮ | ⋮ |
| 180 | Length=20, Name, parent=140, prev=nil, next=nil<br>no children, no attribute, #PCDATA="St2" |

[표 4] [표 2]를 위한 B-Tree 인덱스

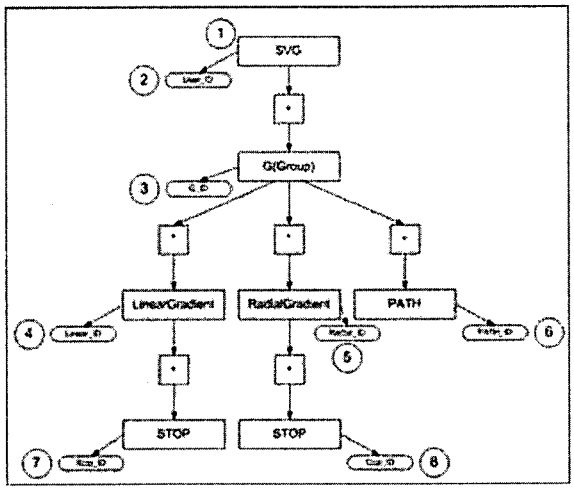
## 3. AVASWI 시스템을 위한 저장소의 설계 및 구현

### 3.1 AVASWI 시스템

AVASWI 시스템은 정보통신부 산하 정보통신연구진흥원의 지원하에 2004년 7월부터 개발중인 멀티플랫폼 기반의 아바타 시스템으로서 SVG(Scalable Vetctor Graphic)가 가지는 장점을 활용한 신개념의 아바타 시스템이다. 향후 유·무선 기반의 포탈 사이트에서 적용할 수 있으며, 또한 유·무선 인터넷을 동시에 지원할 수 있는 콘텐츠 개발에 활용하기 위한 기반 기술을 축적하고 있다. AVASWI 시스템에서는 기존의 아바타를 제작하기 위하여 GIF, VRML, Flash와 같은 바이너리 기반의 이미지 파일을 이용하지 않고 W3C에서 제안한 텍스트로 구성된 2D 기반의 벡터 그래픽 파일 포맷인 SVG를 이용한다.

### 3.2 아바타를 구성하는 SVG 파일의 구조

AVASWI의 아바타를 구성하기 위하여 사용되는 SVG는 XML이기 때문에 XML 파일을 저장하기 위한 여러 가지 방법들을 적용할 수 있다. 그러나, 기존의 레거시 시스템과 호환성을 유지하면서 성능을 발휘하는 것은 관계형 데이터베이스를 이용하는 방법이다.




[그림 1] 아바타 파일의 DTD 구조

[그림 1]은 아바타를 표현하는 SVG 파일의 DTD로서 다음과 같은 특성을 갖는다.

<svg> 엘리먼트를 구성하는 기본적인 속성으로는

'width', 'height', 'viewbox', 'enable-background'로 구성된다. 기본적인 속성들은 SVG 파일이 표현되는 영역과 위치를 나타내며, 사용자의 접근할 때 사용하는 브라우저의 특성에 따라 유동적으로 변경된다. <svg> 엘리먼트를 DBMS에 저장하기 위해서는 별도의 'userid'라는 사용자 속성이 필요하다. 'userid' 속성은 사용자 아이디별로 캐릭터 파일의 구성을 위한 구분자로 활용되며, SVG 파일이 실제로 재구성되어 사용자에게 전송되는 시점에서는 제거된다. <svg> 엘리먼트의 하위 엘리먼트로 아바타의 각 부분을 구분하는 <g> 엘리먼트가 있다.

<g> 엘리먼트를 구성하는 기본적인 속성은 존재하지 않는다. <g> 엘리먼트의 특징이 SVG의 그래픽 속성 엘리먼트들을 그룹핑하여 하나의 부분 그래픽 개체를 표현하는 역할을 수행하기 때문이다. SVG파일 내에서 <g> 파일의 위치는 실제로 화면에 표현될 때의 순서를 의미하기 때문에 <g> 엘리먼트의 호출되는 순서가 저장되어 있는 별도의 정보가 필요하다. <g> 엘리먼트에는 아바타의 해당되는 부분을 의미하는 'g_id' 속성과 순서 정보가 필요하다.

<g> 엘리먼트에 사용되는 'g_id' 역시 사용자에게 전송되는 시점에서는 제거된다. <g> 엘리먼트에 포함되는 엘리먼트들은 <path>, <linearGradient>, <radialGradient>이다. 이중에서 <path> 엘리먼트는 <g> 엘리먼트 안에 한 번 이상 나와야 하며, <linearGradient>와 <radialGradient>는 존재하지 않을 수 있다. 또한 <linearGradient>와 <radialGradient>는 동시에 <g> 엘리먼트 안에 존재하지 않는다.

<path> 엘리먼트는 아바타 캐릭터의 외곽 부분을 드로잉 하는데 이용된다. <path> 엘리먼트에 포함되는 기본 속성으로는 'fill', 'd', 'stroke', 'stroke-width' 등이 있으며, 'd'에는 M, Z, L, H, V, C, S, Q, T, A등과 같은 서브 속성 값이 있다. 'fill' 속성에는 <linearGradient>와 <RadialGradient>의 'linear_id'와 'radial_id'값이 대입된다. <path> 엘리먼트에는 사용자 태그로 'path_id' 속성이 필요하다. 다른 사용자 속성과 마찬가지로 클라이언트로 전송시 제거되어 전송된다.

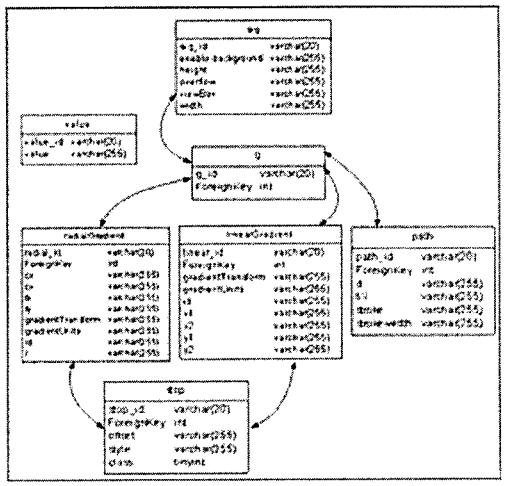
<linearGradient>와 <RadialGradient> 엘리먼트들은 <path> 엘리먼트가 외곽 부분을 그려 놓은 부분을 채우는데 사용한다. <linearGradient>는 기본 속성으로 'gradientUnits', 'x1', 'y1', 'x2', 'y2', 'gradientTransform'등이 있으며, <radialGradient> 기본 속성으로는 'cx', 'cy', 'r', 'fx', 'fy', 'gradientTransform', 'gradientUnits' 등이 있다. <linearGradient>와 <radialGradient>는 하위 엘리먼트로 <stop> 엘리먼트들을 가지고 있으며, 사용자 속성으로는 각각 'linear_id', 'radial_id'가 있다.

<stop> 엘리먼트는 <linearGradient>와 <radialGradient>를 표현하기 위한 핵심 하위 엘리먼트로서 실제로 <stop> 엘리먼트에 의해 그라디언트 효과가 표현된다. 기본 속성으로는 'offset'과 'style'이 사용되며, 사용자 속성으로는 'stop_id'가 있다.

### 3.3 관계형 데이터베이스로의 매핑

아바타를 표현하는 SVG 파일을 관계형 데이터베이스에 매핑하기 위하여 Shared-Inlining 방법을 확장한 Extended-Inlining 방법으로 매핑하였다.

Extended-Ilining 방법은 기존의 Shared-Inlining 방법에서 엘리먼트의 값 부분을 별도의 테이블에 저장하는 방식이다. 앞에서 언급한 것과 같이 AVASWI에서 사용되는 SVG파일의 경우 엘리먼트의 값은 거의 존재하지 않으며, 대부분 엘리먼트 내의 속성과 그에 대한 값으로만 이루어졌기 때문이다. Extended-Inlining 방법을 이용하게 되면 아바타 파일을 재구성시 기존 방법보다 더 빠른 성능을 제공한다. [그림 2]는 AVASWI에서의 아바타를 저장하는 저장소의 스키마를 나타낸 것이다.



[그림 2] 아바타 파일의 DTD를
관계형 데이터베이스에 매핑한 스키마

## 4. 결론

AVASWI 시스템을 위한 저장소를 구현하기 위하여 다양한 XML 파일들의 저장 방법을 적용하였다. 그러나 기존의 정보 시스템의 대부분이 관계형 데이터베이스 시스템을 사용하고 있기 때문에 아바타를 저장하기 위한 별도의 데이터베이스를 도입하는 것은 스키마의 일관성 및 무결성을 유지하기 위한 별도의 비용이 소요되는 문제가 발생한다. 이에 본 논문에서는 기존의 Shared-Inlining 방법을 확장한 Extended-Inlining 방법을 이용하여 AVASWI 시스템의 저장소를 설계, 구현하였다.

## 참 고 문 헌

[1] http://www.w3c.org/Graphics/SVG

[2] 민준기,박명제,안재용,정진완, "다양한 저장소에서의 효율적인 XML 저장기법에 대한 연구", 데이터베이스연구, 제19권 제1호, 2003.

[3] J. Shanmungasundaram, K. Tufte, C. Zhang, H. Gang, D. J. DeWitt, and J. F. Naughton, "Relational databases for querying XML documents:Limitations and opportunities," In Proceedings of VLDB Conf., 1999.

[4] http://www.oracle.com/