# 확장가능한 멀티플랫폼 아바타 시스템을 위한 저장소의 설계 및 구현

박나연*, 유남현**, 김원중**
순천제일대학*, 순천대학교 컴퓨터과학과**
e-mail:kwj@sunchon.ac.kr

# The Design and Implementation of Repository for Scalable Multi-Platform AVATA System

Na-Yeon Park*, Nam-hyun-Yoo**, Won-Jung Kim**
Suncheon First College*
Dept of Computer Science, Sunchon National University**

요 약

기존의 아바타 시스템에서는 아바타를 표현하기 위하여 GIF, VRML, Flash로 표현하였으나, GIF, VRML, Flash등의 파일들은 바이너리 파일 형태로서 사용자가 임의로 파일을 수정하기 위해서는 별도의 도구가 필요하다. 그러나, 확장 가능한 멀티플랫폼 아바타 시스템에 사용하는 기본 이미지 포맷은 텍스트 기반의 SVG로서 사용자가 웹 상에서도 별도 도구의 필요 없이 손쉽게 아바타 파일을 수정할 수 있다. 본 논문에서는 XML의 서브셋인 SVG 파일을 효과적으로 저장하기 위하여 확장 가능한 멀티플랫폼 아바타 시스템에 사용되는 SVG 파일을 분석하고, 그에 따른 최적의 성능을 발휘하는 SVG-Store를 설계 및 구현하였다.

## 1. 서론

확장 가능한 멀티플랫폼 아바타 시스템에서 사용하는 기본 이미지 포맷은 SVG(Scalable Vector Graphic)이다. SVG는 W3C(Worild Wide Consoritium)에서 2D 그래픽을 위한 텍스트 기반의 이미지 포맷으로서 2001년에 최초로 제안되었다. SVG는 XML(eXentisble Markup Language)의 서브셋으로서 XML 문서가 가지는 대부분의 특징을 가지고 있다. 그러나 SVG는 기존의 XML 문서와는 다르게 다양한 속성들과 속성 값들이 아바타를 표현하는데 중요한 역할을 수행하기 때문에 기존의 XML 파일들을 저장하기 위한 방법들을 적용하기에는 문제점이 있다. 본 논문에서는 SVG 파일이 최적의 성능을 발휘하도록 저장하기 위한 저장소를 설계 및 구현하였다.

## 2. SVG

SVG는 XML의 서브셋으로서 웹에서 2D 그래픽을 표현하기 위한 마크업 언어이다. 현재 웹에서는 JPG, GIF 등과 같은 이미지 파일 포맷들이 표준으로 사용되고 있으나 비트맵 기반의 바이너리 형태의 파일 형식이다. 그러나 SVG는 벡터 기반의 텍스트 형태의 그래픽 포맷이다. 1998년 W3C에서 PGML(Precision Graphics Markup Language)과 VML(Vector Markup Language)이라는 두 가지 표준안을 통합하여 만든 그래픽 이미지 포맷이 바로 SVG 그래픽 포맷이다. SVG의 목적은 동적이고 자유롭게 변환할 수 있으며 사용자와 상호 작용할 수 있는 그래픽을 플랫폼에 독립적으로 표현하는데 있다[1]. 또한 기존의 웹 브라우저에서는 SVG 파일을 보기 위해서는 별도의 SVG Viewer가 필요하다. 그러나 Mozlia(모질라)나 Firefox(파이어폭스)의 경우 자체적으로 SVG를 렌더링 하는 모듈이 내장되어 있기 때문에 별도의 플러그인 형태의 SVG Viewer 없이 웹 환경에서도 SVG 파일을 바로 볼 수 있다[2].

(1) SVG의 구조

SVG는 XML의 서브셋이기 때문에 XML 문서가 가지는 선언, 처리 명령어, 엘리멘트(Element, 요소),

* 본 논문은 정보통신부 정보통신연구진흥원에서 지원하고 있는 2004 정보통신기초연구지원사업의 연구결과입니다.

엔터티(Entity), 주소 등을 그대로 사용할 수 있다. 또한 Well-Formed 문서와 DTD(Document Type Definition)나 네임스페이스(NameSpace)가 있는 Valid한 문서를 지원한다. 그러나 엘리멘트의 경우 SVG에서 정의한 스펙 외에 다른 엘리멘트를 사용하는 경우 SVG Viewer 등이 해석하지 못하기 때문에 SVG를 저장하기 위하여 엘리멘트에 변형을 가하는 경우 다시 복원하기 위한 별도의 정보가 필요하게 된다. SVG를 구성하는 엘리멘트는 루트 엘리멘트인 <svg> 태그 안에 그래픽 요소인 <rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon>, <text>, <tspan>, <tref>, <path>, <pattern>, <lineargradient>, <radialgradient> 등과 , 컨테이너 요소인 <g>, <defs>, <symbol>, <use>, <desc>, <title>, <image>, <switch> 등으로 구성된다[1]. [표 1]은 SVG 파일의 예제이며, [표 2]는 SVG 파일을 구성하는 그래픽 요소를 정리한 것이다. [표 2]와 같이 SVG가 기존의 XML 문서와 다른 점은 속성(Attribute)들의 기능이 매우 다양하고 중요한 역할을 수행한다는 것이다.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 2 0 0 0 0 8 0 2 / / E N "
"http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/svg-20000802.dtd">
<svg width="100%" height="100%">
  <desc>Example 1 - One Rectangle</desc>
  <rect x="0.75cm" y="0.5cm" width="2.5cm"
height="3cm" style="fill:purple; stroke:gray;
stroke-width:0.1cm"/>
</svg>
```

[표 1] SVG 파일의 예

| 태 그 | 기 본 기 능 | 속 성 |
|---|---|---|
| <rect> | 사각형을 그린다. | x="좌표값" y="좌표값" width="크기"height="크기" rx="크기" ry="크기" |
| <circle> | 원을 그린다. | cx="좌표값" cy="좌표값" r="크기" |
| <ellipse> | 타원을 그린다. | cx="좌표값" cy="좌표값" rx="크기" ry="크기" |
| <line> | 직선을 그린다. | x1="좌표값" y1="좌표값" x2="좌표값" y2="좌표값" |
| <polygon> | 다각형을 그린다. | points="점들의 리스트" |
| <polyline> | 연결선을 그린다. | points="점들의 리스트" |
| <text> | 문자를 그린다. | x="좌표값" y="좌표값" textLength="길이" |
| <tspan> | <text>요소 내에서 문자열과 폰트속성과 위치를 지정한다. | x="좌표값" y="좌표값" dx="길이+" dy="길이+" rotate="숫자+" |
| <tref> | 텍스트를 참조하기 위해 사용한다. | x="좌표" xlink:href="<URI>" |

[표 2] SVG의 그래픽 요소와 관련된 엘리멘트

## 3. XML의 저장 방법

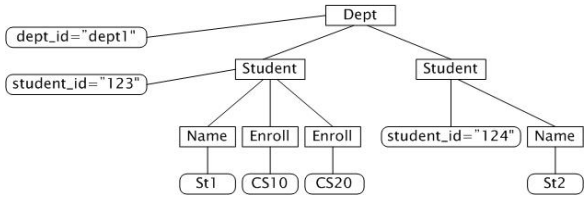
XML을 저장하기 위한 방법은 레거시 시스템을 이용하는 방법, 관계형 데이터베이스 시스템을 이용하는 방법, 객체지향형 데이터베이스 시스템을 이용하는 방법, XML 전용 관리 시스템을 이용하는 방법 등 크게 네 가지로 분류할 수 있다.

레거시 시스템을 이용하는 방법은 텍스트 파일 방식, 역 리스트 방식, XML 압축 방식 등으로 나눌 수 있다. XML 파일을 변환 과정 없이 그대로 저장할 수 있다는 장점이 있지만 파일 시스템이 가지는 단점인 동시성 제어, 보안, 파손 회복 등에서 성능이 떨어진다. 관계형 데이터베이스 시스템을 이용하는 방법은 DTD 종속적인 방법과 DTD 독립적인 방법들로 구분된다. DTD 독립적인 방법으로는 Edge 방법, Binary 방법, Universal 방법, Monet 방법 등이 있으나 용량이 작고 간단한 XML 문서에서만 적용이 가능하다는 단점이 있으며 대부분 DTD 종속적인 방법을 많이 이용한다. 객체관계형 데이터베이스 시스템을 이용하는 방법은 XML 파일을 엘리멘트 단위로 구분하여 저장하거나 파일 자체를 CLOB나 BLOB 형태로 저장하며, 그에 따른 검색 속도의 향상을 높이기 위하여 B-Tree 인덱스를 구성하여 제공한다. XML 전용 관리 시스템은 Lore, Tamino와 같은 시스템으로 XML 문서를 효율적으로 저장하기 위하여 제안되었으나 기존의 구조적 데이터와 통합 처리시에 많은 문제점을 내포하고 있으며, 다중 사용자 지원, 대용량 데이터 처리와 같은 부분에 대한 많은 테스트가 필요하다[3].

(1) Inlining 저장 기법

Inlining 저장 기법은 관계형 데이터베이스 시스템을 이용하는 방법 중 DTD 종속적인 방법으로서 XML 문서를 관계형 데이터베이스에 저장하는 경우 가장 많이 이용되는 방식이다. Inlining 저장 기법은 XML 문서의 DTD나 XML Schema를 분석하여 분석된 정보를 기반으로 관계형 데이터베이스에 맞게 스키마를 생성하고 파싱한 XML 문서를 관계형 데이터베이스에 저장하는 방식이다. Inlining기법으로는 Shared Inlining기법, Hybrid Inlining 기법이 있다. Shared Inlining기법은 DTD 그래프에서 한 개 이상의 in-degree를 가지는 노드에 대한 테이블을 각각 생성하고, 한 개의 in-degree를 가지는 노드에 대한 테이블은 따로 생성하지 않는다. 또한 0개의 in-degree를 가지는 노드의 경우에는 부모 노드로부

터 연결된 패스를 찾을 수 없기 때문에 한 개 이상의 in-degree를 가지는 노드의 경우와 같이 따로 테이블을 생성한다. [그림 1]은 XML 파일의 DTD그래프이며, [표 3]은 [그림 1]을 Inlining기법으로 관계형 데이터베이스에 매핑한 예이다.




[그림 1] XML파일의 DTD 그래프

| 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] [그림 1]을 Inlining 방법으로 관계형 데이터베이스 매핑

Hybrid Inlining기법은 두 개 이상의 in-degree를 가지는 노드가 '*'를 거치지 않고 도달 가능한 경우 해당 노드를 인라인 시킨다.

## 4. SVG-Store의 설계 및 구현

본 논문에서 설계, 구현하고자 하는 SVG-Store는 확장가능한 멀티플랫폼 아바타 시스템의 저장소이다. SVG-Store는 아바타를 구성하는 SVG 파일을 효과적으로 저장하고, 검색 및 업데이트 연산의 성능을 향상시키기 위한 저장소이다. SVG-Store를 설계 및 구현하기 위해서는 SVG-Store의 저장되는 SVG파일의 특징을 정의해야 한다.

(1) SVG-Store에 저장되는 SVG 파일의 특징

본 논문에서 설계, 구현한 SVG-Store를 사용하는 확장가능한 멀티플랫폼 아바타 시스템의 사용되는 SVG 파일들의 기본 특성은 다음과 같다.

① 사용자는 기본적으로 시스템에서 제공하는 아바타를 사용할 수 있으며, 해당 아바타 시스템의 변형을 시도하지 않는 경우 사용자별로 할당되는 저장공간에 배치되지 않는다.

② 아바타를 표현하는 SVG 파일은 텍스트 파일 형태이기 때문에 기존의 바이너리 형태의 GIF 파일이나 Flash와 다르게 아바타를 특정 부분별로 분할 저장이 가능하며, 사용자에게 SVG파일을 전송하기 전에 별도의 조인 작업을 거치지 않고 SVG 파일 내에 정해진 순서대로 사용자에게 전송되면서 아바타를 표현할 수 있다.

③ 기본적으로 제공되는 아바타 시스템의 기본 파일 사이즈는 30KB에서 50KB 사이이며, 아바타 시스템을 구성하는 left leg, right leg, right arm, left arm, body, left ear, right ear, face, hair, cap, clothes, right blusher, left blusher, mouth, nose, right eyebrow, right eye, left eyebrow, left eye들의 기본 사이즈는 2KB에서 8KB로 구성된다.

④ 아바타를 수정하는 경우 각 부분별로 수정이 가능하며, 사용자가 아바타를 수정하는 경우 사용자별로 할당되는 별도의 저장 공간에 수정된 아바타 시스템을 저장한다.

⑤ 기본적으로 제공되는 아바타 이외에 안경, 귀걸이, 모자, 옷 등과같이 기본 아바타를 장식할 수 있는 액세서리들을 사용자가 원하는 경우 덧붙일 수 있다. 액세서리를 포함시켜 장식하는 경우에 액세서리들이 호출될 때 필요한 순서는 사용자별 저장 공간에 별도로 저장 되어진다. 또한 액세서리 파일들은 사용자별로 별도의 수정이 가능하지 않으며, 완벽한 구조의 SVG파일로 저장되는 것이 아니라 <g> 엘리멘트에 의해 그룹화 되어 저장되어진다.

⑥ 모바일 기반의 사용자 단말기로 전송되는 경우 별도의 변환 컨버터에 의하여 사용자에게 전송되어지는 시점에 변환되어 전송되어진다.

⑦ 아바타 시스템의 애니메이션 기능은 별도의 Animation 관련 태그들을 이용하여 구현하지 않고, DOM을 이용하여 애니메이션 기능을 지원한다. Animation 관련 태그들을 이용하지 않은 이유는 시스템에서 기본적으로 제공하는 아바타들의 경우 애니메이션 기능을 지원하지 않기 때문이다. 이는 사용자가 다양하게 애니메이션 기능을 구현할 수 있도록 하기 위해서이다.

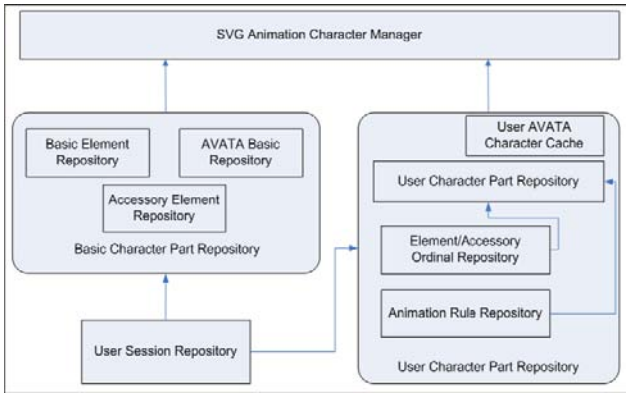
(2) SVG-Store의 구현

SVG-Store는 SVG파일을 저장하기 위한 저장소로서 Basic Character Part Repository, User Character Part Repository, User Session Repository로 구성된다. [그림 2]는 SVG-Store의 구조로서 세부적으로 다음과 같이 구성된다

① Basic Element Repository

Basic Element Repository는 아바타를 구성하는

다리, 팔, 몸통, 귀, 얼굴, 머리카락, 눈썹, 볼터치, 입, 코 등으로 구성된다. 각각의 구성 요소들은 기본적으로 <g> 엘리멘트에 의해 그룹핑 되어 있으며, 각각의 부분별로 별도의 XML Schema에 CLOB 형태로 저장되며, 그에 따른 XML 구조 및 검색은 B-Tree 인덱스를 기반으로 수행된다.




[그림 2] SVG-Store의 구조

② AVATA Basic Repository

Avata Basic Repository는 아바타 시스템에서 기본적으로 제공되는 SVG 파일의 Repository이다. 확장가능한 멀티플랫폼 아바타 시스템의 경우 사용자가 속해 있는 그룹별로 기본 아바타를 제공하는 기능이 있기 때문에 매우 다양한 기본 아바타들이 존재하며, 이런 다양한 기본 아바타들을 저장하는 공간이다.

③ Accessory Element Repository

Accessory Element Repository는 사용자가 아바타를 치장하기 위해 사용되는 액세서리 들이 저장되는 Repository이다. 액세서리는 안경, 귀걸이, 목걸이, 옷 등이 있으며, 종류별로 XML Schema가 구성되어 있으며, XML Schema 내부에서 구분하기 위하여 OID를 할당한다.

④ User Character Part Repository

아바타를 수정하는 사용자들의 경우 대부분이 다시 아바타를 수정하기 때문에 검색 연산과 업데이트 연산 능력이 뛰어난 기능을 제공할 수 있는 Shared Inlining 기법을 이용하여 리파지토리를 구성하였다.

⑤ Element/Accessory Ordinal Repository

SVG 파일은 특별한 URI를 선언하는 경우를 제외하고는 SVG 파일 내의 엘리멘트들은 순서대로 사용자 브라우저에 표현된다. 이러한 특징은 '다리' 부분보다 '몸통' 부분을 먼저 보여주는 경우 '몸통'에 '다리'가 올라있는 형태의 아바타가 구성될 수 있다. 이러한 문제를 해결하기 위하여 엘리멘트나 액세서리들의 SVG 파일 내에 순서 정보를 기록하고 있는 저장소가 Element/Accessory Ordinal Repository이다.

⑥ User AVATA Character Cache

사용자가 아바타를 수정하지 않고 단순하게 아바타를 표현하기 위한 SVG 파일을 요청하는 경우 User AVATA Character Cache에 저장되어 있는 사용자들의 아바타를 전송함으로서 저장소의 로드 부하를 줄이고 사용자에게 응답하는 속도의 성능 향상을 가져 왔다.

⑦ Animation Rule Repository

아바타의 애니메이션 기능은 온라인 상에서 설정이 가능하다. 사용자가 멀티플랫폼 기반 애니메이션 아바타 시스템에 접근하여 사용자 아바타에 애니메이션 기능을 부여한 경우, 해당 애니메이션 룰들이 단순한 Animation Rule Repository에 SVG 형태로 저장된다. 애니메이션 기능의 경우 모든 엘리멘트들이 사용자에게 전송된 후 제일 마지막에 전송되기 때문에 별도의 순서 정보를 저장하는 저장소는 필요하지 않다.

## 5. 결론

본 논문에서는 확장 가능한 멀티플랫폼 아바타 시스템의 아바타들을 저장하기 위하여 확장 가능한 멀티플랫폼 아바타 시스템의 특성과 SVG 파일의 특성을 분석하여 SVG 파일의 검색 및 수정 연산의 기능을 향상시킨 SVG-Store를 설계, 구현하였다. SVG-Store는 SVG 파일의 업데이트 연산 속도를 향상시키기 위하여 관계형 데이터베이스에 본 논문에서 개발한 Shared Inlining 기법을 이용하여 매핑하였으며, 업데이트 연산이 거의 수행되지 않은 기본 아바타들과 액세서리 부분들을 표현하는 SVG 파일은 객체형 데이터베이스에 매핑하여 검색 성능을 향상시켰다.

## 참고문헌

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

[2] http://www.firefox.org

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

[4] D. Florescu, D. Kossmann, "Storing and Querying XML Data using an RDBMS," IEEE Data Engineering Bulletin 22(3), 1999.