The Journal of Korean
Institute of CALS/EC
Vol. 5, No. 2, Dec. 2000

# 웹 환경에서의 그래프 처리*

박성헌**, 박지헌***

# Graph Processing on the Web Environment

Park, Sunghun, Park, Jihun

## Abstract

There are many web-based applications which need graphs and charts to be generated from data stored in the database. This paper does a comparative study on graph processing techniques for web-based applications through a case study of building a stock information system. The result of this paper can be used for building effective web applications with graphs in areas of EC(electronic commerce), EIS(executive information system), and DM(data mining).

* 본 논문은 2000년도 정보통신부 우수대학원 지원 사업에 의하여 이루어졌습니다.
** 명지대학교 지식정보학부
*** 홍익대학교 컴퓨터공학과

# I. 서론

웹(Web) 기술은 현재 거의 모든 정보시스템을 변화시키고 있다. 인터넷 신문과 홈페이지에서 시작한 웹은 지금은 전통적인 Client /Server 환경[8]에서만 구축되던 주식정보시스템, 학사정보시스템, 인사정보시스템, EIS (Executive Information System) 등과 같은 응용정보시스템 구축에도 활발하게 이용되고 있다. 이와 같은 웹 기반의 응용정보시스템 구축에서 여러 가지 그래프와 차트들을 효과적으로 처리하는 것이 시스템 성공에 필수적이다.

웹 환경에서 그래프와 차트의 처리에는 여러 가지 방법이 있다. 첫째 방식으로 정보가 자주 변경되지 않는 경우 일/주/월별로 그래프 이미지 파일을 생성하여 홈페이지처럼 제공하는 방식이고[7], 둘째는 HTML에서 이미지 파일의 Width와 Height 속성을 이용하는 방법이며, 셋째는 CGI(Common Gateway Interface)[7] 프로그램에서 그래픽 라이브러리를 이용하여 동적으로 그래프를 생성하는 방법, 마지막으로 도메인에 맞는 적당한 그래프나 차트 프러그인(Plugin)[9]을 개발하여 필요한 데이터를 서버로부터 전달받아 그래프를 클라이언트에서 그리는 방법이다. 이들 방법은 각자 처리 속도(클라이언트, 서버, 통신), 그래프의 품질, 개발의 용이함, 확장성 등에서 서로 장단점을 가지고 있다. 웹 환경에서 처리해야할 그래프의 요구 사항에 따라 적절한 그래프 처리 방법이 선택되어야 하겠다.

본 논문은 단편적으로 제시된 여러 가지 웹 환경에서의 그래프 처리 방법을 주식정보시스템 구축의 사례를 통하여 체계적으로 비교할 수 있는 틀을 제공하고 적절한 처리 방식을 선택하는 기준을 제시하고자 한다. 지금까지 연구는 웹 환경에서의 그래프 처리 기술에 대한 각자의 기술 개발은 이루어졌지만 이들을 총괄적으로 비교하고 선택하는 기준에 대한 연구는 미비하였다고 본다.

# 2. 그래프 처리 방법

## 2.1 정적인 이미지 파일을 이용하는 방법

웹에서 제일 많이 이용되는 방식으로 필요한 데이터를 이용하여 그래프 파일을 생성하고 이를 HTML문서에서 <IMG> 태그를 이용하여 포함하는 방법이다. 웹 브라우저에서 지원하는 이미지 파일 양식이 Gif와 JPEG[10]이므로 정적인 이미지 파일의 생성에는 이와 같은 형식을 지원하는 Boutell의 GD 라이브러리[2]를 이용하거나 다른 그래픽 라이브러리를 이용하여 이미지를 만든 후 적당한 변환도구를 이용하여 Gif와 JPEG 형태로 변환하는 방법을 이용한다.

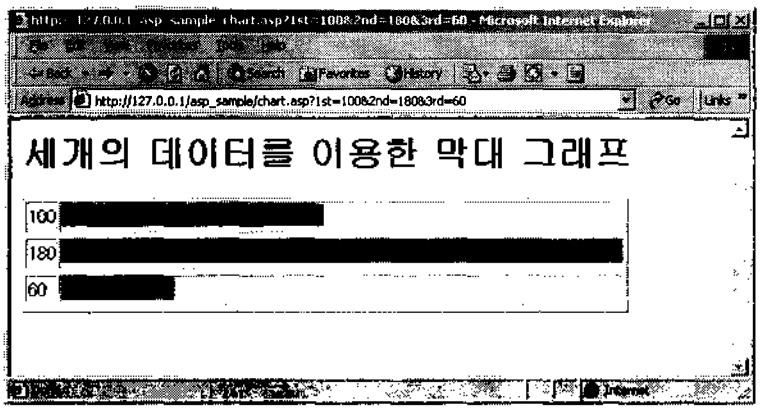
변경되지 않는 년/월/주별 웹사이트 방문 통계 그래프나 년/월/주별 주가 동향 그래프 등의 경우 년/월/주별로 하나의 정적인 그래프를 생성하여 반복하여 이용하는 것이 효과적인 방법이 된다. 그러나 시간에 따라 급변하는 정보나 사용자에 따라 요구하는 자료가 달라지는 경우는 동적으로 그래프를 생성하여 사용하여야 한다.

### 2.2 HTML 태그의 속성을 이용하는 방법

HTML[7,10]의 <IMG> 태그에는 Width와 Height를 속성을 지정해 줄 수 있다. 이들 속성을 지정해 주지 않으면 SRC에서 지정한 이미지 파일의 크기를 그대로 이용하나 Width와 Height에 값이 지정되면 새로운 값에 맞추어 웹 브라우저는 이미지 파일의 크기를 축소하거나 확대하게 된다. 많은 이미지들을 효과적으로 화면에 배열하기 위해서는 HTML의 <TABLE> 태그를 이용한다. <그림 1>은 <IMG> 속성을 이용하여 작성한 간단한 막대 그래프를 보여 준다.

<그림 1>은 다음의 Microsoft사의 ASP (Active Server Page)[4] 코드의 수행 결과로 생성되었다. 이는 세 개의 데이터를 읽고 이들 값의 범위를 계산한 후 적당한 넓이의 픽셀 값을 계산하는 간단한 코드이다. 막대 그래프는 빨간 하나의 픽셀로 만들어진 "spacer_red.gif"의 width 속성과 height 속성을 새로 지정해 주어 만들어진다. 이 코드는 ASP 대신 Perl, C/C++, Java Servlet, PHP 등의 다른 CGI(Common Gateway Interface)용 언어로도 구현될 수 있으며 특정 도구에 종속되어 있지 않다.

```
<HTML><H1>세개의 데이터를 이용한 막대 그래프</H1>
<%
DIM A(3)
A(1) = CDbl(ReQuest.QueryString("1st"))
A(2) = CDbl(ReQuest.QueryString("2nd"))
A(3) = CDbl(ReQuest.QueryString("3rd"))
max = -10000000
min = +10000000
For i= 1 to 3
        if A(i) < min then min = A(i)
        if A(i) > max then max = A(i)
Next
width = (max - min)
%>
<table border=1>
<% For i =1 to 3 %>
<tr><td><% = A(i) %></td>
<td><IMG SRC="spacer_red.gif"
width="<% = Int(( A(i)- min)/width*400.0 +
    100) %>" height="20">
</td><tr>
<% next %></table></HTML>
```




<그림 1> HTML 태그를 이용한 막대 그래프

이 방식은 서버에 큰 부하를 주지 않고 여러 가지 색깔의 Gif 파일들만 이용하여 다양한 2차원 막대 그래프를 효과적으로 처리할 수 있도록 해준다. 그러나 막대 그래프 이외 다른 종류의 그래프를 처리하기에는 제약이 너무 많다.

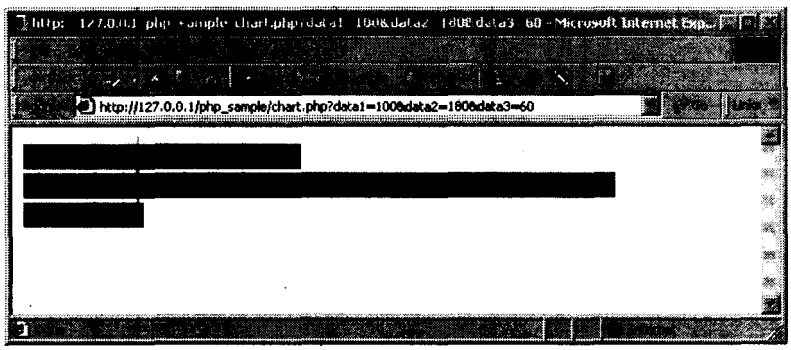
### 2.3 그래픽 라이브러리를 이용한 CGI(Common Gateway Interface)

#### 2.3.1 이차원 그래프

제일 보편적으로 사용되는 방법으로 CGI 프로그램 내부에서 그래픽 라이브러리를 이용하여 실시간으로 필요한 그래프를 생성하는 방법이다. <그림 2>는 Boutell의 GD 라이브러리[2]를 이용하여 PHP[11]로 작성된 CGI에서 Gif 파일로 생성된 막대 그래프를 보여준다.

이 막대 그래프를 생성하는 PHP CGI는 2.2절에서 사용한 알고리듬을 그대로 사용하고 있다. 다만 <IMG>태그의 속성을 바꿔주는 방법 대신에 GD 라이브러리에서 제공하는 여러 가지 함수들을 직접 사용하여 이미지 파일을 만들고 있다. GD 라이브러리가 제공하는 함수들은 2차원 그래프를 작성하는 데 필요한 선, 원, 사각형, 색깔 지정 등의 거의 모든 함수들을 제공하고 있다. 자세한 내용은 GD 라이브러리의 메뉴얼[2]을 참고하기 바란다.

```
<?php
$A[0] = $data1; $A[1] = $data2; $A[2] = $data3;
$max = -10000000;
$min = +10000000;
for ($i=0; $i < 3; $i++ ) {
        if ($A[$i] < $min) $min = $A[$i];
        if ($A[$i] > $max) $max = $A[$i]; }
$width = ($max - $min);
/* Header( "Content-type: image/jpeg");  */
echo ("Content-type: image/gif");
/* create image */
$image = imagecreate(500,100);
$red = ImageColorAllocate($image,255,0,0);
$white = ImageColorAllocate($image,255,255,255);
/*  create white background*/
ImageFilledRectangle($image,0,0,500,100,$white);
for ($i=0; $i < 3; $i++) {
        $wid = (int) (( $A[$i]-
$min)/$width*400 + 100);
        ImageFilledRectangle($image,
        0, $i*25 ,$wid,($i+1)*25-5,$red); }
/*  render image */
/* ImageJPEG($image); */
ImageGIF($image);
/*cleanup memory */
ImageDestroy($image);
?>
```



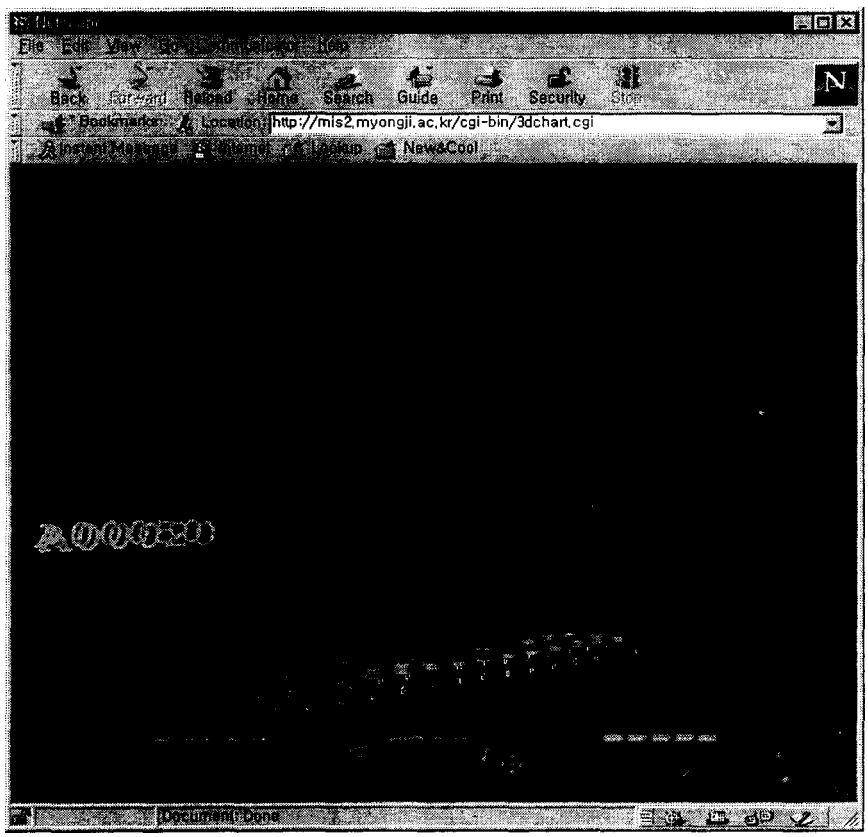

<그림 2> GD 라이브러리로 생성된 막대 그래프

2.3.2 삼차원 그래프

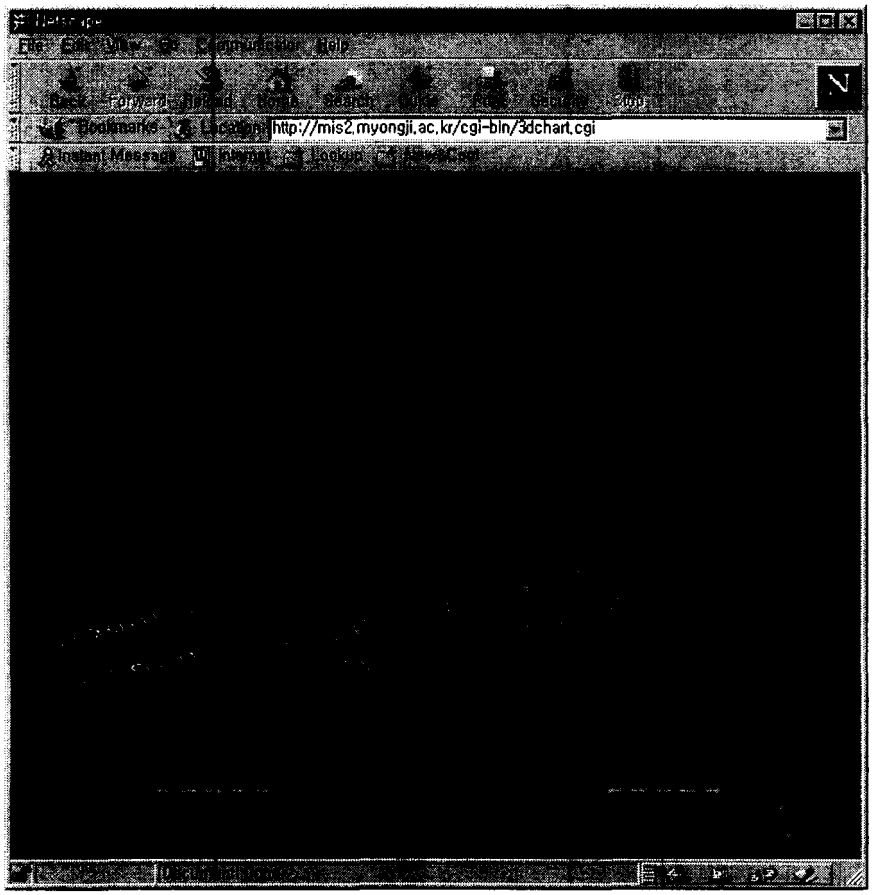
이차원 그래픽 라이브러리를 이용하는 대신 VRML[5]을 이용하여 삼차원 그래프를 생성할 수 있다. <그림 3>은 VRML을 이용한 삼차원 주식정보 그래프를 보여준다. 주식 투자자들에게는 천 개 이상 되는 투자 종목 중에서 원하는 몇 개 종목을 선택하여 여러 가지 투자 종목을 동시에 효과적으로 비교해 볼 수 있는 기능이 절실히 필요하다. 이차원 그래프로 여러 가지 종목을 동시에 표시하기가 어렵지만 그림 3과 같은 삼차원 그래프를 이용하면 다양한 정보를 동시에 표현이 용이할 수 있다.

이와 같은 3차원 VRML 파일은 SGI사에서 개발한 CosmoPlayer Plugin 등과 같은 뷰어(Viewer)를 웹 브라우저에 추가 설치함으로써 기존의 HTML 브라우저가 새로운 타입의 데이터 포맷을 처리할 수 있도록 해 준다. 이와 같은 VRML 뷰어는 하단에 표시되어 있는 여러 가지 버튼을 이용하여 삼차원 그래프를 다양하게 조사해 볼 수 있는 기능을 제공한다.

삼차원 그래프에 필요한 데이터는 주식 DB에서 Oraperl API[3, 12]를 이용하여 검색하였다. 검색된 결과를 삼차원 그래프로 출력하기 위해서는 VRML 문법을 지원하는 그래




<그림 3> 삼차원 주식 그래프 CGI의 수행 결과




<그림 4> 회전한 삼차원 그래프

프의 정의로 변환되어야 한다. 그리고 웹 클라이언트에서 VRML 그래프를 처리할 수 있도록 해당되는 MIME Type에 대한 정의가 다음과 같이 그래프 출력 바로 전에 선행되어야 한다.

```
#!/usr/local/bin/perl
print "Content-type:x-world/x-vrml\n\n";
print "#VRML V1.0 ascii\n\n";
# 삼차원 그래프를 VRML 형식으로 생성하는 코드들...
```

여기서 중요한 것은 Parsing된 데이터가 다시 클라이언트로 전송되어지는 방식인데, 먼저 웹 브라우저가 전송되는 데이터의 타입이 무엇인지 파악하여야 그에 따른 뷰어 프로그램을 실행시키게 된다는 점이다. 위에 나타난 Content-type:x-world/x-vrml\n\n이 그것인데, 이 부분은 데이터의 전송결과를 웹 브라우저 내의 VRML 뷰어(viewer) 프로그램을 실행시키게 된다.

이와같이 CGI(Common Gateway Interface)

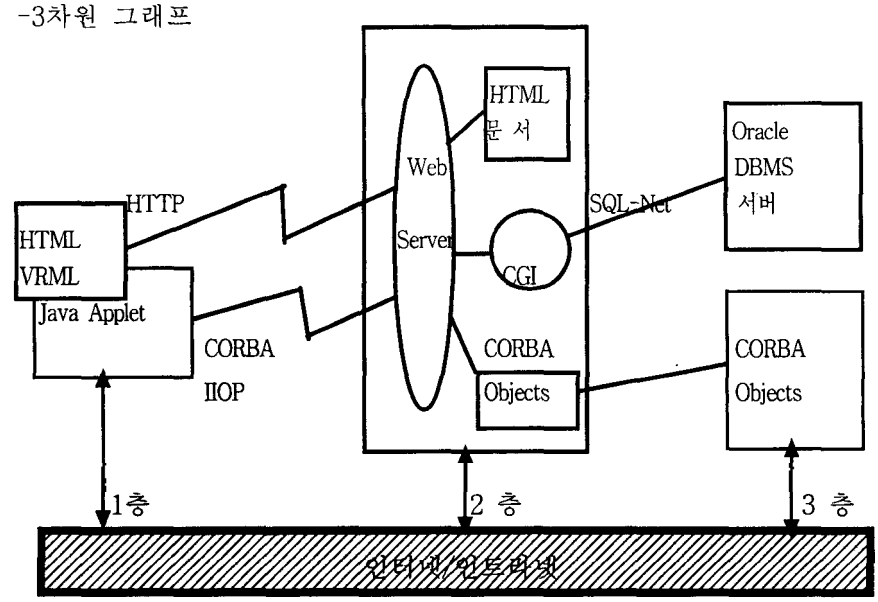
를 이용하여 필요한 이차원 혹은 삼차원 그래프를 동적으로 생성하는 방법은 원하는 다양한 그래프를 마음대로 생성할 수 있다는 장점이 있어나 개발 노력이 많이 소요되고 특히 모든 계산 및 처리가 서버에서 발생하므로 그래프에 대한 사용자 요구가 많아지면 서버에 걸리는 부하가 급속도로 증가하게 된다.

이러한 부하를 줄이기 위해 특별한 목적의 프러그인이 개발되기도 하며, 이러한 추세는 최근에 각광받고 있는 Web3D[1]로 이해할 수 있다. Web3D는 전송되는 삼차원 주식 정보를 보여주기 위한 삼차원 자료 대신 주식정보 자체를 전송함으로써 전송 부하를 줄임과 동시에 VRML 보다 양질의 화면 정보를 제공할 수 있다.

### 2.4 자체 프러그인을 이용한 그래프 처리방법

마지막 방법의 그래프 처리는 <그림 5>와 같이 분산객체[9]를 지원하는 World Wide Web으로 구성하는 방법이다.

2차원 주식 그래프는 테이블로 표시하는 주식시세 정보와 함께 Visibroker for Java을 이용하여 CORBA IIOP상에서 작동하는 Java Applet으로 구현하였다. Java Applet에서



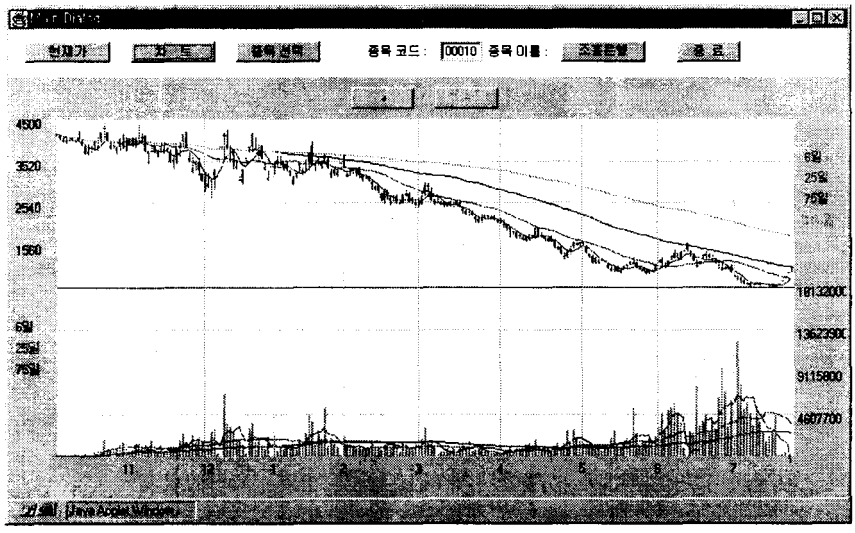

<그림 5> 주식정보 시스템의 3층 구조(Tier)

CORBA를 통해 요청하는 해당 기업의 주식 시세 검색을 처리하는 주식 서버가 CORBA 객체로 구현되어 있다. 주식 서버 CORBA 객체는 IIOP를 이용하여 다른 객체(예를 들어 투자 분석 객체, 통계 분석 객체 등)들과 손쉽게 연동할 수 있다.

<그림 6>의 주식 그래프는 2차원 봉 차트로 주식 시세 변동을 표현한다. 그래프의 위쪽은 주가를 아래쪽은 거래량을 나타낸다. 일봉 차트 외에 6일 평균, 25일 평균, 75일 평균, 150일 평균 등의 그래프를 제공한다.

이와 같은 도메인에 맡는 프러그인(Plugin)을 개발하기 위해서는 자바(Java)나 ActiveX 등의 몇 가지 대안이 있다. 클라이언트에서 작동하는 그래프 프러그인과 그래프 서버간의 통신에서도 여러 가지 대안이 있다. 즉 기본적인 TCP/IP의 소켓, SUN사에서 제공하는 Java RMI(Remote Method Invocation), OMG에서 정의한 CORBA, Microsoft사에서 지원하는 DCOM(Distributed Component Object Model) 등이 있다. 본 논문에서는 OMG의 CORBA(Common Object Request Broker Architecture)를 지원하는 Inprise사의 Visib-roker를 이용하여 구현하였다.

CORBA는 Client/Server 호출 구조를 갖는다. 클라이언트는 CORBA를 통해 원격지의 구현 객체와 통신을 한다. 구현객체는 서버 프로세스 내에 생성된 객체이자 구현된 메쏘드를 갖고 있는 객체이다. 이에 반해 클라이언트는 서버 프로세스 내의 구현 객체의 메쏘드를 호출하고, 그 결과를 전달받은 후 가공하는 역할을 담당한다. 이때 클라이언트와 구현 객체는 IDL 언어를 통해 해당 인터페이스를 정의한다. 정의된 인터페이스는 CORBA에서 제공하는 IDL 컴파일러를 통해 클라이언트와 구현 객체간의 통신을 위해 필요한 스텁(Stub)과 스켈리톤(Skeleton)을 비롯해 ORB 런타임을 이용하기 위해 필요한 각종 코드를 자동으로 생성해낸다. 생성해낸 파일은 IDL의 기능에 따라 자바나 C++, 코볼 같은 다양




<그림 6> 2차원 봉 그래프

한 언어로 작성할 수 있다. 본 논문에서는 자바로 작성된 파일을 생성해낸다.

스텁과 스켈리톤은 클라이언트와 구현객체간의 통신에 필요한 모든 기능을 제공한다. 이 기능에는 서로 다른 플랫폼과 구현 언어간의 일관성을 유지하기 위해 모든 파라미터를 패키징화(마샬링)하고 다시 언패키징(언마샬링)하는 부분이 포함되어 있다. 따라서 클라이언트의 호출은 스텁을 통해 CORBA 런타임에 전달되고, 다시 이 호출은 CORBA의 표준 프로토콜인 IIOP와 TCP/IP 연결을 통해 서버 구현 객체 측의 CORBA 런타임에게 전달된다. 구현 객체 측의 CORBA 런타임은 다시 이 호출을 해당 스켈리톤 코드를 통해 언패키징한 후 해당 구현 객체의 메쏘드를 호출하고 그 결과를 다시 클라이언트에게 반환한다.

CORBA를 이용한 Java 주식 그래프는 다음 3개의 모듈로 구성되며 각각 모듈에 대한 개괄적인 설명은 다음과 같다.

2.4.1 IDL관련 모듈

주식 그래프 클라이언트와 주식 서버가 서로 이해하는 함수를 정의하고 이를 IDL 컴파이러를 이용하여 Java Template 파일을 만든다. 다음 두 개의 함수가 정의된다.

curAccount(in string key) : 클라이언트로부터 받은 key 값으로 DB에서 검색한 해당 종목의 데이터를 String형태로 리턴한다.

curCode() : 클라이언트에게 주식의 기업코드와 기업명, 기업정보의 기업코드를 String 형태로 전달한다.

2.4.2 서버 모듈

주식 서버는 주식 데이터를 저장하고 있는 Oracle 데이터 베이스에 연결하고 IDL 모듈에서 정의한 두 개의 함수를 실제 구현에 필요한 자바 코딩을 다음과 같이 하였다.

다음 DB연결 모듈은 JDBC 드라이버를 사용해 오라클과 연결을 유지한다.

```
....
Class.forName
("oracle.jdbc.driver.OracleDriver");
Connection conn =
DriverManager.getConnection
      ("jdbc:oracle:thin:@mis2.myongji.ac.kr:
1521:mis2",  "company", "companydb");
   ...
```

IDL 구현 모듈에서 정의한 두 개의 함수의 대체적인 코드는 다음과 같다.

curAccount()는 기업코드를 클라이언트에서 받아 이 코드에 해당하는 주가의 매일의 시가, 고가, 저가, 종가, 거래량을 문자열로 변환하여 리턴해 준다.

```
curAccount(in string key)
ResultSet rset = stmt.executeQuery
  ("select * from Stock_Item where item_no =
'"+key+ "' order by day desc ");
while (rset.next () && i !=241 ) {
 temp      +=      rset.getString("Day")+sep+
rset.getString("Begin")+sep+
 rset.getString("High")+sep+
rset.getString("Low")+sep+
 rset.getString("End")+sep+
rset.getString("Volume");
        }
return temp;
```

curCode()는 모든 회사의 주식 코드를 저장하고 있다 클라이언트의 요청에 따라 필요한 정보를 전해준다. 이 함수의 개괄적인 코드는 다음과 같다.

```
curCode() {
DataInputStream in = new DataInputStream( new
FileInputStream("data/code.txt" ));
while ( (line=in.readLine()) !=null ) {
        StringTokenizer st = new
        StringTokenizer(line, "    ");
        code[0][codeNum]= st.nextToken();
        code[1][codeNum]= st.nextToken();
        if (st.hasMoreTokens())
                code[2][codeNum]          =
st.nextToken();
        }
for(int i=0; i< codeNum-1; i++)
        temp += code[0][i]+sep+
        code[1][i]+sep+ code[2][i]+",";
        return temp;
...
```

### 2.4.3 클라이언트 모듈

모듈은 CORBA 설정 모듈, 주식 데이터 로딩 모듈, 그래프 처리 모듈들로 구성된다. 다음은 이와 같은 작업에 필요한 개괄적인 코드를 보여준다.

```
/* CORBA 연결 */
org.omg.CORBA.ORB orb= org.omg.
CORBA.ORB.init(this,null);
idl=StockIdlHelper.bind(orb,"nanari");
....
/* 주식 데이터 로딩 모듈 */
String temp = applet.idl.curAccount(key);
while (st1.hasMoreTokens()) {
        StringTokenizer st2=new String
Tokenizer(st1.nextToken(), sep);
        date = st2.nextToken();
        start = st2.nextToken();
        top = st2.nextToken();
        bottom = st2.nextToken();
        end = st2.nextToken();
        yesterday = ""+ 111;
        volume = st2.nextToken();
        Stock stt = new Stock (date, start, top,
bottom, end, yesterday, volume);
        stockData.addElement(stt);
stockLength += 1;
}
...
/* 주식데이터를 그래프로 그리는 모듈 */
/* 저장된 주식 데이터를 분석해 시가와 종가의 크
기를 서로 비교해 시가가 크면 빨간 사각형으로 종
가가 크면 파란 사각형으로 같으면 선으로 그린다.
*/
...

if(juGa[0][i]-juGa[3][i]>1) {        // 시가가 크면
        g.setColor(Color.blue);
        g.fillRect(747-i*3*(scopeStatus+1),   160
-juGa[0][i], 2+scopeStatus,
juGa[0][i]-juGa[3][i]);
        g.drawLine(747-i*3*(scopeStatus+1)
+1+scopeStatus/2, 160 -juGa[1][i],

747-i*3*(scopeStatus+1)+1+scopeStatus/2,
160-juGa[2][i]);    }
else if(juGa[0][i] -juGa[3][i]<-1 ) { //종가가 크면
        g.setColor(Color.red);
        g.fillRect(747-i*3*(scopeStatus+1),   160
-juGa[3][i], 2+scopeStatus, juGa[3][i]-juGa[0][i]);
g.drawLine(747-i*3*(scopeStatus+1)+1+scopeStatu
s/2, 160 -juGa[1][i],

747-i*3*(scopeStatus+1)+1+scopeStatus/2,
160-juGa[2][i]);
        } else {                        // 같을때
        g.setColor(Color.red);
        g.drawLine(747-i*3*(scopeStatus+1), 160
-juGa[0][i],
        747-i*3*(scopeStatus+1)+2+scopeStatus,
160-juGa[0][i]);

g.drawLine(747-i*3*(scopeStatus+1)+1+scopeStatu
s/2, 160 -juGa[1][i],

747-i*3*(scopeStatus+1)+1+scopeStatus/2,
        160-juGa[2][i]);
        }
...
```

<표 1> 그래프 처리 기법의 비교

| | 정적인 이미지 | HTML 태그 | CGI(Common Gateway Interface) | 분산 프러그인 |
|---|---|---|---|---|
| 이미지 품질 | 일/주/월별 통계 정보에 적당 | 막대 그래프에 제한됨 | 이차원 삼차원등 다양함 | 이차원 삼차원등 다양함 |
| 시스템 부하 | 배치(Batch)작업으로 효과적 | Gif파일의 확대는 브라우저에 의해 처리됨 | 자료의 검색 및 그래프의 생성이 서버에서 이루어짐(서버에 집중되는 부하) | 분산 처리를 통해 자료의 검색은 그래프 서버 그래프 생성은 클라이언트에서 이루어짐 |
| 개발 노력 | 그래픽 프로그래밍 + DB 검색 | 크기 변환정도로 제일 간단 | 그래픽 프로그래밍 + DB 검색 | 통신 + 그래픽 프로그래밍 + DB 검색 |
| 사용자 상호작용 정도 | 배치(Batch)방식 | CGI를 이용한 제한된 상호 작용 | CGI를 이용한 제한된 상호작용 | Client/Server 수준의 상호 작용 |

2.4.4 자체 프러그인의 장단점

본 논문에서 구현한 분산객체 기술인 CORBA와 Java를 이용한 주식 그래프 처리는 기존에 많이 사용되는 소켓(Socket), Client/Server 2 층(Tier) 구조를 지원하는 JDBC에 비하여 비슷한 처리 속도를 제공하면서 3 층(Tier) 구조 Client/Server 시스템의 장점을 제공하고 개발자에게 통신 프로그램의 고급 기능을 제공하여 프로그램 개발을 돕고 있다.

그러나 제일 큰 단점은 CGI나 HTML의 태그의 활용법에 비해 통신프로그래밍에 따르는 난이도가 큰 단점이 된다고 본다. 장점으로는 클라이언트 서버간의 적당한 처리 부하를 배분할 수 있고 클라이언트에 저장된 자료를 이용하여 사용자의 요구에 따라 서버에서 새로운 데이터를 요청하지 않고 다른 종류의 그래프를 생성할 수 있는 것 등이 있다.

## 3. 결론

지금까지 크게 네 가지 방식의 그래프 처리 방식을 소개하였다. 이들 방식은 웹 환경에서 그래프를 구현하는 도메인의 요구 상황에 따라 적절한 기법들이 선택되어야 할 것이다. 각각의 기법들의 장단점을 다음과 같은 표로 정리해 볼 수 있겠다.

본 논문은 웹 환경에서의 그래프 처리에 대해 주식 정보 시스템 구축 사례 연구를 통해 다양한 그래프 처리 방법을 검토해 보았다. 다른 웹 환경에서의 응용시스템 개발에서의 그래프 구현에는 <표 1>의 장단점을 고려하여 적당한 방식을 결정할 수 있을 것이다.

## 참고 문헌

1. 편집부, 인터넷 속의 3차원 CG: Web3D의 부활, Graphics Live, 2000년 8월
2. T. Boutell, gd library, http://www.boutell.com/gd/, 2000
3. G. Gunther Birznieks, S. Sol, CGI for Commerce, M&T Books, 1997
4. A. Fedorov et al, Professional Active Server Page 2.0, WROX Press, 1998
5. J. Hartman, J. Wernecke, VRML 2.0 Handbook, Addison Wesley, 1996
6. R. Kalakota, A. B. Whinston, Electronic Commerce, Addison Wesley, 1997
7. Liu, J. Peek, R. Jones, B. Buus & A. Nye, Managing Internet Information Services, OReilly & Associates Inc., 1994
8. R. Orfali, D. Harkey, J. Edwards, The Essential Client/Server Survival Guide, 2nd Ed., John Wiley & Sons, 1996
9. R. Orfali, D. Harkey, Client/Server Programming wih Java and CORBA, 2nd Ed., John Wiley & Sons, 1998
10. J. Ozer, Publishing Digital Video 2nd Ed., AP Professional, 1997
11. PHP Group, PHP Manual, http://www.php.net/, 2000
12. L. Wall, T. Christiansen, R. Schwartz, Programming Perl 2nd Ed., OReilly & Associates Inc., 1996

## 저자소개

**박성헌**
서울대 공대 금속학과 졸업
한국과학기술원 경영과학 졸업
Purdue University 전산학과 석사
Purdue University 경영정보학 박사
울산공과대학교 산업공학과 전임강사
성균관대학교 경영학과 부교수
현재 명지대학교 지식정보학부 부교수
관심 분야 : 사용자 인터페이스, 정보통신 응용, 전자상거래

**박지헌**
서울대 공대 기계설계학과 졸업
한국과학기술원 전산학과 졸업
University of Texas at Austin 전산학과 석사
University of Texas at Austin 전산학과 박사
한국전자통신연구원 연구원
부산외국어대학 컴퓨터 공학과 조교수
현재 홍익대학교 컴퓨터 공학과 조교수
관심 분야 : 컴퓨터 그래픽스