Paper 30-27

Physicians Web Portal to Enterprise Information

Ahmed Rahman, Mayo Clinic, Rochester, MN
Priscilla Van Grevenhof, Mayo Clinic, Rochester, MN
Rosa Cabanela, Mayo Clinic, Rochester, MN

ABSTRACT

Physicians and administrators have little hope of responding appropriately to the challenges of the health care market without data to support decision making. Desired, for Mayo Clinic's Primary Care Practice, was to have the ability to access and integrate data from many platforms in many formats from the Enterprise and bring this information to the desktop in a robust interactive display. The solution was delivery of the data to the web through an interface using Java™ with access to Online Analytical Processing (OLAP) tools for summarization, graphical display and reporting. Communicating major trends, assisting in planning and management, visually displaying alerts in summary data and surfacing individual patient information are all possible through an easy-to-use web application. To really understand what the summarized data represents, the physicians must be able to drill down, download and explore their own detail data. SAS/MDDB®, SAS/IntrNet®, AppDev Studio™ (webAF™ & webEIS™) and SAS® MDDB Report Design-Time Control are used for development on Microsoft® Windows NT® platform.

INTRODUCTION

The focus of this paper is to describe the tools used and discuss application development experiences of the Mayo Clinic with SAS web tools. The advantages and disadvantages specific to our application development environment will be reviewed. A brief explanation of how we integrated the Windows® network security with the web application in the context of Mayo Clinic's Intranet is also provided.

METHODS

OVERVIEW

A pilot project was developed to test the capabilities of the development environment, the acceptance of the web tools, the ability to deliver timely information and the methodology of using a multi-dimensional database (MDDB) to define the data. The Family Medicine Practice at four separate locations was chosen for the demonstration project. Two practices in Rochester, MN and practices in two smaller towns, totaling 50 physicians plus administrative personnel, were the first clients using the system. A cross-functional team examined a variety of development issues such as data sources, data definitions, levels of security, data analysis types, and style of display.

DEVELOPMENT OF PROTOTYPE

The first prototype was developed in webEIS 1.2 to display summarized data stored in Multi Dimensional Database. Part of the development process was to rapidly build a prototype and present physicians' practice-related data. The physician could drill down into a specific set of data, see alerts, customize the report look and eventually get detail data. With each client meeting, additional requirements of quick display, more customization and security became priority.

DATA PREPARATION

The data collection process integrates data from many platforms in many formats from the Enterprise [Figure 1]. Summarizing mountains of data from a two-dimensional conventional data set to an MDDB is very easy and straightforward. A SAS/MDDB license is required in order to be able to take advantage of this development path. An example of PROC MDDB is shown in figure 2 in the appendixes.

METABASE REGISTRATION

The MDDB can be registered on any SAS server including the local SAS session on any platform that SAS supports. In order to display an MDDB in a webEIS application, registration in a metabase repository is required. Steps to register an MDDB are follows:

1) In a SAS 8.x session, click on "Solution->EIS / OLAP Application Builder" from the menu bar [Figure 4].
2) The "EIS Main Menu" window pops up. Double click on the "Metabase" icon to bring the "Metabase" window
3) Select a metabase to register an MDDB or using "File -> New…" from the menu bar, create a new metabase if necessary.
4) Click on "ADD" button to add an MDDB into selected metabase repository.

Note that an MDDB always has a base data set and this data set is the data set that is used to create MDDB. If base data is to be referenced from the MDDB, the application needs the library name of the base defined in the SAS server scope where the MDDB is served.

Three types of SAS web development are described below:

1. DEVELOPMENT WITH WEBEIS

BUILDING AN MDDB REPORT

The webEIS development environment is very easy and simple to use but yet very powerful. Data stored in the MDDB can be displayed in a meaningful graphical and/or tabular format in webEIS. Three simple steps in building powerful OLAP display include:

1) Open a webEIS window. In a new document, give a name to the section.
2) With the mouse, drag and drop a graph, table or other component from the tool bar into the display area.
3) Connect to a SAS server and select the MDDB from a repository to display data.

CONNECTING TO A SAS SERVER

Displaying an MDDB requires connecting to a SAS server and can be accessed using different methods such as JCONNECT, IOM etc. To make things simple, let's connect to local machine using JCONNECT method. When AppDev studio is installed, by default a connection gets registered in both webAF and webEIS environment and usually the Name of this default connection would be "[your network Id or machine name]'s PC".

1) Before using the connection, start a SAS Connect Spawner on the PC. The spawner continuously listens for requests from webEIS or webAF applications. Upon receiving a request, a SAS session is contacted or invoked if necessary and the request is passed to the SAS session for processing. The spawner icon can be found in the AppDev Studio program group icons.
2) In the webEIS environment, in the navigation tree, right click on the section that has been defined at the time of creating a new document and click on "Data…" from the drop down menu

[Figure 3]. Select the default server connection from a popup window. After a successful connection to the server, a list of registered items populates the list box. Select a desired MDDB and click "OK". At this point, refresh the environment by clicking on the refresh icon located at the bottom left corner of the display area.

3) Note that after refreshing, data may not be reflected in the graph or table components. This is because rows, columns or measures may not have been properly set. Rows, columns and measures can be set by dragging and dropping from a graphical list of rows, columns and measures respectively in the "Data" window.

In the webEIS 2.0, an application can be distributed either as an applet or as a Java Server Page (JSP). Earlier version of webEIS supports only applet distribution. Creating an applet or JSP out of a webEIS application is as simple as clicking on File menu -> "Save as Applet…" or "Save as JSP…". We developed our prototype as applets in webEIS 1.2.

SECURITY WITH WEBEIS

One of the requirements of our application is to allow individual physicians to see only their respective data. This can be accomplished in two ways, either by creating individual smaller MDDB for each physician or subset a large MDDB by physician id where physician id would be a class variable. We chose to go with the later one. We used SAS/EIS® Access Control List (ACL) definitions to control which user can see what portion of the MDDB.

RESULTS: WEBEIS APPLET

The two major sections of our application are patient visit statistics and analysis of a physician's practice. The demographics and characteristics of all the patients for whom the physician is the primary care provider are displayed over a year grouped monthly. Statistics at the department level are available to both physicians and administrators.

All the visit statistics are shown in an interactive webEIS Java applet with a bar, pie, scatter or overlay chart of client's choice through the web interface. Each chart segment can be dynamically drilled down into a variety of user defined hierarchies. The application allows physicians to compare their practices with the average practice of the rest of the department. At any state in a defined hierarchy, data can be seen in a detailed form and downloaded into a Microsoft Excel or .CSV file. Problems can be identified through user-defined alerts. Clients can literally do any kind of analysis on their data in any way that they want. All these are at the user's fingertip in one standardized web interface [Figure 5].

Demonstrations of the prototype developed in webEIS 1.2 as applet met with an overwhelmingly positive response from administrators and department leaders. As we started to roll out the application to test, the team looked for workable solutions for the following deficiencies.

1) WebEIS applet is heavily dependent on client environment such as browser with proper Java Runtime Environment (JRE), client machine's availability of Random Access Memory (RAM) through Java Virtual Machine (JVM) and network speed. SASNetCopy technology partially supplements network speed as it installs almost all of the runtime classes on client machine. This is problematic in our Electronic Environment Device (EED) as users are not permitted to do installs.
2) SAS/EIS ACL is not user based rather it is a group based access control. For each user we need a group. To allow user to change their password would require building another system with web interface.

To overcome the above problems, we tried implementing the application in MDDB Report Design-Time Control.

2. DESIGN-TIME CONTROLS

BUILDING AN MDDB REPORT

SAS Design-Time Controls (DTC) need to be installed on the development machine. DTC host such as Microsoft FrontPage® 2000 is used for this example.

1) Open a FrontPage 2000 window.
2) In a new page, from the menu bar, click on "Insert -> Advanced -> Design Time Control…" [Figure 6] and the "Insert Design-Time Control" window pops up.
3) There should be a list of DTC to choose from. If the list is empty, click on customize button located at the bottom of the same window to bring up the "Customize Design-Time Control" window and select all the listed SAS DTC and click on the "OK" button.
4) At this point the "Insert Design-Time Control" window should have a list of SAS DTC.
5) Select SAS MDDB Report and click on "OK" button. After successfully inserting the DTC into the new page, DTC either shows up as a bold string of text or as an icon. Double click on the text or icon to bring up "Design-Time Control Properties" window.
6) Select the "Connection" tab and check if all the connection parameters are set properly. For help with connection, please refer to SAS DTC help for SAS/IntrNet configuration.
7) Select the "Data" tab. In the "Data" tab, select name of the Repository and name of the MDDB. If no repository is listed, either the connection parameters are not set properly or SAS/IntrNet server needs some configurations.
8) Select "Dimension" Tab to set MDDB dimension parameters. At this point SAS MDDB Report DTC has good enough information to build an MDDB Report. Report can be previewed in FrontPage 2000 or can be saved in web server and is ready to be served.

SERVER SIDE CUSTOMIZATION

The MDDB Report DTC can be customized at run time on the server by sub-classing MDDB Report Viewer class. For example, over-riding _outputHtmlAfterBody method of sashelp.webeis.webeis class will cause Report Viewer to display your own custom output along with the rest of the output of the report. If subclass is needed, name of the subclass must be mentioned in the "Design-Time Control Properties" window in build mode under "Display" Tab. A simple example of over-riding _outputHtmlAfterBody method of sashelp.webeis.webeis class is shown in figure 8 in the appendixes. It is written in SAS Component Language (SCL). More help on customizations may be found at SAS web site.

RESULTS: MDDB REPORT DTC

Being able to customize MDDB Report Viewer gave us a great deal of flexibility over an applet-based report built with webEIS. Security was the primary issue that still remained. The Objective was to integrate Windows network security with the application's security. AppDev studio release 2.0 gave the additional functionality of saving as JSP or Servlet. A report using an MDDB with a webAF interface would now provide the ultimate in flexibility of interface design. The ability to integrate Windows network security with our application was now possible.

One thing worth mentioning here; we also developed some other applications using MDDB Report DTC that are currently in production. We do not have any plans of re-building them in webAF, as those applications do not require a high degree of customization.

3. DEVELOPMENT WITH WEBAF

BUILDING AN MDDB REPORT

1) Open a webAF window.
2) Start a new JSP Project.

3) Copy and paste the code from figure 9 in the appendixes into index.jsp and the index.jsp is created by default when a new JSP project is created.
4) From the menu bar select “Build->Build Project”.
5) From the menu bar select “Tools->Serivices->Start Java Web Server”.
6) Start SAS Connect Spawner if it is not running already.
7) From the menu bar select “Build->Execute in browser” to execute index.jsp in your favorite browser.

SECURITY WITH WEBAF

A Java class called “User” is developed to handle security tasks. One of its methods takes a user id and password in the parameter and with the Java Native Interface (JNI), method makes a call to Windows “logonUser “ API to get a security handle that represent the user for process impersonation. As soon as “logonUser” succeeds, method destroys the handle representing user because process impersonation or changing the security context of the thread in which the JSP or Servlet is running is not required for the purpose, all it is needed is to know, whether the user is a valid domain user or not. If “logonUser” call succeeds, method looks up in a local database of application users to verify if the user has access to the current application. If database lookup succeeds with verification, a user context is built and is placed into HttpSession object for later verifications. Example code of user object to verify user credentials is shown in figure 7 in the appendixes. It is written in Java (JSP).

RESULTS: WEBAF JSP

As of writing this paper, this new development has not been assessed by our usability lab yet. However, our understanding is that, JSP is dependent on the server machine’s environment rather than client machine. JSP does not require browser to have any sort of Java Runtime Environment. All the graphs are shown as Graphical Interchange Format (GIF) images. JSP has a faster display than MDDB report built as Java applet in webEIS. Even though in webEIS 2.0, MDDB reports can also be built as JSP, being able to directly work with source code in webAF 2.0 gives the ultimate flexibility of custom interface design.

In order to run JSP, setup of a Java app server such as Tomcat on the web server side is required. To scale up any application developed in both webAF and webEIS that uses SAS as a back end server may require Integrated Object Model (IOM) and MiddleWare server set up.

From a security standpoint, being able to integrate network security with the application, eliminated users’ need to remember an extra set of user ids and passwords for the application.

CONCLUSION

The MDDB Reports built with DTC or with webEIS as Java applet or with webAF as JSP are all easy to use and very fast ways of building and deploying powerful OLAP applications on the web. Choice of development path may be dependent on the specifications of the target application. In today’s fast moving technological arena, a rapid application development environment always gives us competitive advantages.

The Physician Patient Management Project assisted us exploring different SAS web tools, which are necessary for us in understanding different technologies and their proper applications.

TRADEMARKS

ACKNOWLEDGMENTS

The authors would like to express their appreciation to the following people for their assistance in this paper:

James Naessens, Mayo Clinic, Rochester, MN

Donnis Lassig, Mayo Clinic, Rochester, MN

CONTACT INFORMATION

Authors can be contacted by mail or email:

Ahmed Rahman
Mayo Clinic
200 First Street SW
Rochester MN 55987
Email: rahman.ahmed@mayo.edu

Priscilla Van Grevenhof
Mayo Clinic
200 First Street SW
Rochester MN 55987
Email: vangreve@mayo.edu

Rosa Cabanela
Mayo Clinic
200 First Street SW
Rochester MN 55987
Email: cabanela.rosa@mayo.edu

APPENDIXES

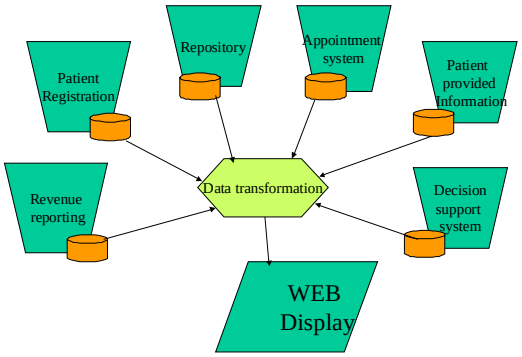

Figure 1

```
proc mddb data=sashelp.prdsale
out=sasuser.mddbdemo label='MDDB
DEMO';
class country region division prodtype
product year quarter month;
var actual  predict / sum;
hierarchy year quarter month /
name="time" display=YES;
hierarchy country region division /
name="geography" display=YES; run;
```

Figure 2

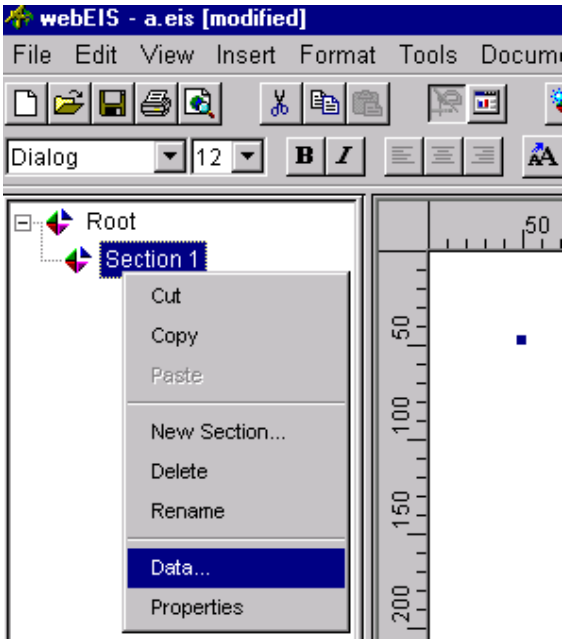

Figure 3

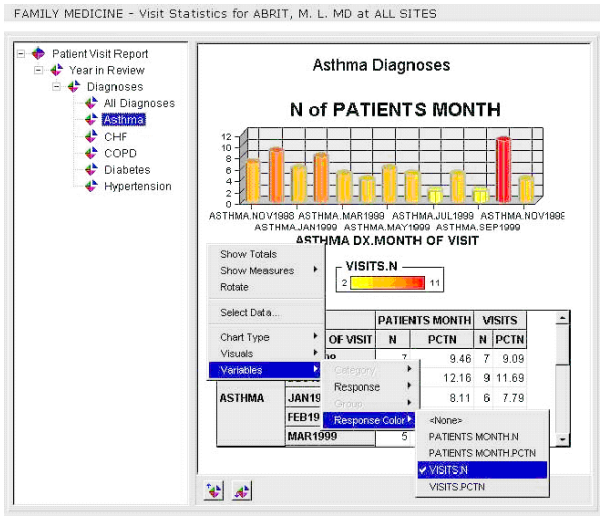

Figure 5

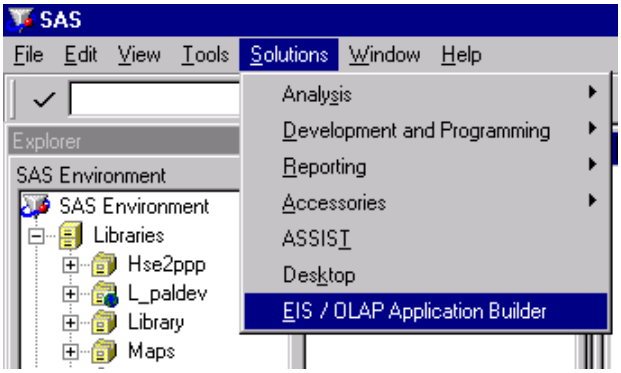

Figure 4

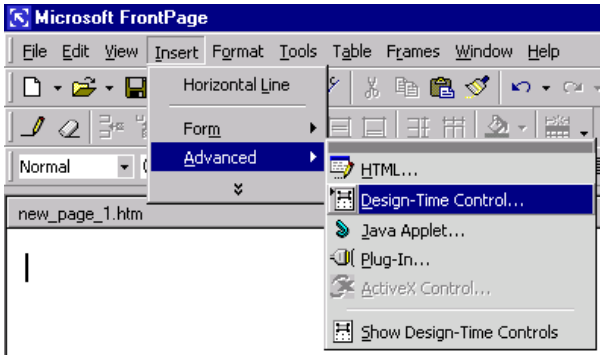

Figure 6

```
<%@ page import = "hsr.hse.pal.*" %>
<%@ page errorPage= "errorFraLeftBottom.jsp" %>
<jsp:useBean id="user" class="hsr.hse.pal.User" />

<%
user = (User)session.getAttribute("user");
     if (session.isNew() | user==null)
            throw new Exception();
     else if ( !user.hasAccessComp( "ABC" ) )
            throw new Exception ("User does not have access to requested document.");
%>
```

Figure 7

```
class MyReport extends sashelp.webeis.webeis;

_outputHtmlAfterBody: method / (state='o', signature='n');
     dcl num fid;
     dcl num rc;
     dcl htmout o_htm = _new_ htmout();

     fid = fopen('_webout','o');
     rc = fput (fid, '<H3>MY OWN CUSTOM OUTPUT</H3>');
            rc = fwrite(fid);
            rc = fclose(fid);
endmethod;
```

Figure 8

```
<%@taglib uri="http://www.sas.com/taglib/sasads" prefix="sasads"%>
<sasads:Connection id="localhost" serverArchitecture="PC" persistedName="localhost"
initialStatement="" scope="session"  />
<sasads:MDModel id="MDModel1" database="SASHELP.PRDMDDB" metabase="SASHELP"
 connection="localhost" scope="session" >
     <sasads:MDRowAxis >
            Geographic
     </sasads:MDRowAxis>
     <sasads:MDColumnAxis >
            Time
     </sasads:MDColumnAxis>
     <sasads:MDMeasure measure="ACTUAL" selectedStatistics="SUM" />
</sasads:MDModel>
<sasads:MDCommandProcessor id="MDCommandProcessor1" scope="session" />
<sasads:MDBar id="MDBar1" model="MDModel1" commandProcessor="MDCommandProcessor1"
imageLocation="/assets/" scope="session" />
<sasads:MDTable id="MDTable1" model="MDModel1" commandProcessor="MDCommandProcessor1"
maxRows="25" maxColumns="10" scope="session" />

Note that for the simplicity reason this above code is given here. This same code can
be generated by drag and drop operation of each of these components from the MDDB
JSP/Servlet tool bar and set parameters to proper value. In MDModel some of the
attribute values may be case sensitive.
```

Figure 9