Paper 184-26

Dynamic Ad-hoc Reporting via the Web: Real-world Comparison of MDDB Report Viewer and AppDev Studio®

Colin Harris, Ministry of Social Policy, Wellington, New Zealand

ABSTRACT

There is no question that user access to warehouse information is moving more and more to Web delivery mechanisms. This paper reviews and contrasts SAS Institute's two main Web OLAP offerings, MDDB Report Viewer (part of SAS/Intrnet® software) and the recently introduced AppDev Studio. Both promise fast implementation of OLAP reporting with minimal development and maintenance.

In the New Zealand Ministry of Social Policy, Web technology has opened up a vast amount of warehouse information to all staff – over 8,000 desktops. Previously this was only available to a few dozen specialists familiar with SAS software. This reporting mechanism has been in production for over 18 months, and has been enthusiastically received with peak usage of over 1,000 hits per day. It currently uses the MDDB Report Viewer. AppDev Studio is being extensively evaluated to replace or complement the current solution.

INTRODUCTION

This paper compares two SAS Institute Web OLAP tools, but does so within the context of the real business requirement of a large organisation's requirement for information delivery.

Background
The Ministry of Social Policy (MSP) operates a large data warehouse that services four government departments across the social sector, representing about 25% of New Zealand government. These are the Department of Work and Income (responsible for assisting job seekers, and paying all benefits and public pensions in New Zealand), the Department of Child, Youth and Family Services (works with children, young people and their families at risk), the Ministry of Social Policy (provides social policy advice to appropriate Ministers) and the Ministry of Education.

For an overview of the initial Web enablement project for this warehouse (including Metadata and Management Information being surfaced via the Web), refer to the SUGA 99 paper entitled "*Web enabling a large data warehouse"* or the SUGI 24 paper entitled "*Web enabling a large data warehouse*: *Improved Metadata for expert analysts and wider access to Management Information*".

Business Requirement for Web Reporting
The warehouse has been in operation since January 1996, and access to all information had been through 60 business analysts utilising their SAS System skills.

The new business requirements were:

- To allow more people to directly access information from the warehouse, rather than putting information requests via analysts. This would provide two main benefits:
  - Allow people to get immediate answers to satisfy their information needs.
  - Free business analysts to enable them to devote time to more sophisticated analysis work.
- Provide flexibility to answer many types of questions.
- To provide this capability in a cost effective way.

Environment
The warehouse is based on a large UNIX platform: an HP N4000 server with six PA8600 550 Mhz processors, 12 GB of memory, 2 TB of disk in an XP256 disk array and 4 TB of automated tape storage. The operating system is HP-UX 11.

Over 8,000 desktops running NT 4.0 are connected via the Ministry's TCP/IP wide area network.

The software being used is Oracle releases 7.3.4 and 8.1.7, and SAS releases 6.12 and 8.1. Data from multiple source systems is stored in an Oracle repository (about 450 GB) and SAS data sets and MDDB's (about 200 GB).

The SAS system is used on both the desktop and the server, and plays a critical role in the warehouse. It is used for all analysis and reporting work, and to process, update and maintain the Oracle repository.

Solution Overview
The best solution approach consisted of:

- OLAP (On-line analytical processing) technology to give the flexibility and speed required to meet users needs. OLAP capability allows you to 'slice-n-dice' your data in a straightforward way. This includes the ability to choose which dimensions to view information by (e.g. by age, by income, by time period) and which subset of information to view (e.g. all people over 50 years old). You can also 'drill-down' to get more detailed information.
- A Web delivery mechanism. The benefits of web solutions are generally well known, but worth repeating:
  - No desktop set-up effort required. Sometimes there may be minimal work, e.g. plug-ins or viewers may need to be installed. In our case there was a standard desktop image that consisted of a Netscape browser, Word, Excel and Acrobat, and therefore no changes were required on the desktop.
  - No desktop maintenance is required. Any change is made centrally, and therefore the desktop doesn't need to be modified. Also the latest change is immediately available to all users.
  - Familiarity. Most people are now comfortable with the browser interface.
  - Technology has now matured for Web applications, to give both the functionality and speed required.
- A fast and efficient data storage component that would cope with the large volumes of data in our envrionment.
- Preferably a SAS based solution, as SAS is used for most of the data warehouse work at the Ministry, and our team was familiar with SAS.
- Minimal development effort! With only 3 people in our core warehouse team, and a large number of data sources and users to support, we would not be able to invest a lot of development time in the project.

Solution Options
Initially we thought we would need to develop something ourselves at a low level using SAS/Intrnet (e.g. HTML pages to submit code through the application dispatcher). However we were lucky that SAS Institute introduced an early version of the MDDB Report Viewer (MRV) at the time we were exploring options (mid 1998), and this met most of our needs.

As of today there are now a number of options for providing Web OLAP capabilities, with most not requiring any code to be written. These include:

- The MDDB Report Viewer, which still continues to be enhanced with Version 8.
- AppDev Studio, which includes WebEIS® with pre-written Web OLAP capability, and WebAF® to develop more customised Web reporting interfaces.
- Xplore, a sample application provided with SAS/Intrnet which includes drill-down reporting.
- SAS Design Time controls, announced in July 2000 as experimental. These are add-in components for your WYSIWYG HTML editor (e.g. Frontpage), and include an HTML-based MDDB report and SAS graphic Java applet.
- With SAS now providing open OLAP access, even third party tools (such as Excel or Cognos) can be used to report on an underlying SAS MDDB structure.
- Create your own HTML or Java front-end, that accesses the underlying SAS data and functionality through SAS/Intrnet.

MDDB REPORT VIEWER

What is it?
The MDDB Report Viewer (MRV) is a pre-written Web OLAP tool. It operates through a Web browser and only uses SAS software from the application server (i.e. SAS is not required on the desktop). It is HTML based (except the graphics applet), and as its names suggests, it requires data to be stored in a SAS MDDB.

Versions are available that run with SAS 6.12 (MRV Version 1.2) and SAS 8.x (MRV Version 8.x). It comes on the installation media with Version 8, or can be downloaded for 6.12. It can be used as long as you have the relevant SAS products licensed (primarily SAS/Intrnet, SAS/EIS® and SAS/MDDB® Server).

Features of the MRV allow users to:

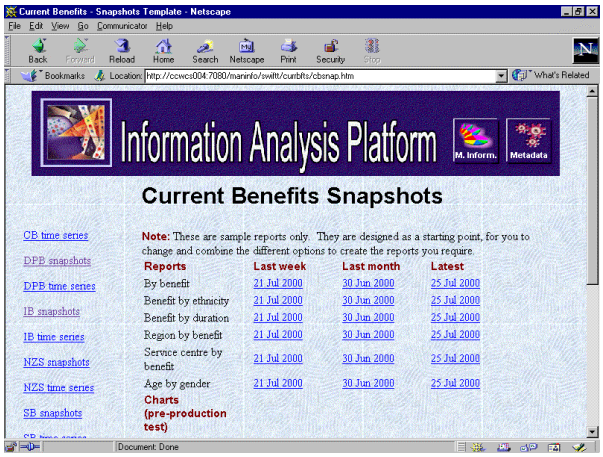
- Drill-down using pre-defined hierarchies (defined in an EIS Metabase) to expand within the current report, or create a new report with only the chosen category
- Apply filters, which are basically selecting from list boxes to build a `WHERE` clause
- Change down and/or across dimensions to build a completely different report, using variables or hierarchies or nested combinations
- Download the report to an Excel spreadsheet
- Rotate the report
- Reach-through to produce a detailed report of the underlying observations that make up a cell

The MRV can be used to generate text only reports, or graphics can be included. Graphics options include the old-style 2-D HTML based graphics (which are rather terrible) or the Java-based Graphics Applet.

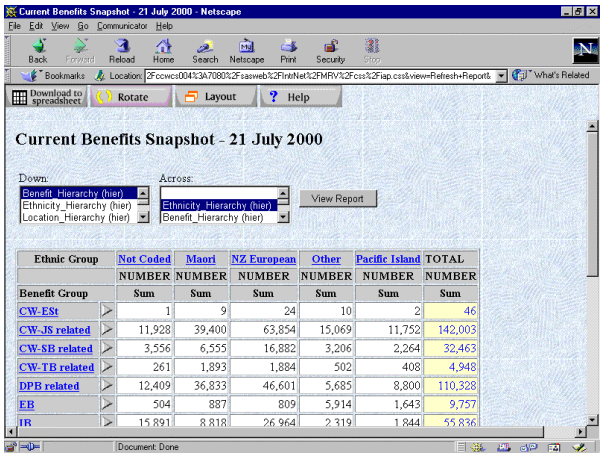
The Graphics Applet (still in pre-production status) is designed to work in conjunction with the MDDB Viewer, to give a very smart interactive graph to complement the drillable table being displayed. Drill-down can be performed using the text table or graph, and the table and graph will stay in synch. It also includes many interactive features for customisation of the graph, e.g. to change colours, style of chart or shape of bar. Although this applet will be extremely good when the wrinkles are ironed out, there are just too many bugs for it to be recommended for production use at present. It is also very slow to show the first graph in a browser session, more than 80 seconds, which is off-putting to users (compared to 2-3 seconds for most MRV text-only reports).

From the End-user Perspective: MSP use of the MRV
The standard and documented set-up for the MRV is for the user to go through three screens, selecting things like the MDDB to use, whether graphics are required, which dimensions and filter variables to use, and so on. We needed an easier and more direct approach for our wide range of users to cope with, and ended up with simple menu options that would display the desired MDDB report immediately. Details of how to develop this are in the set-up section further on in this paper.

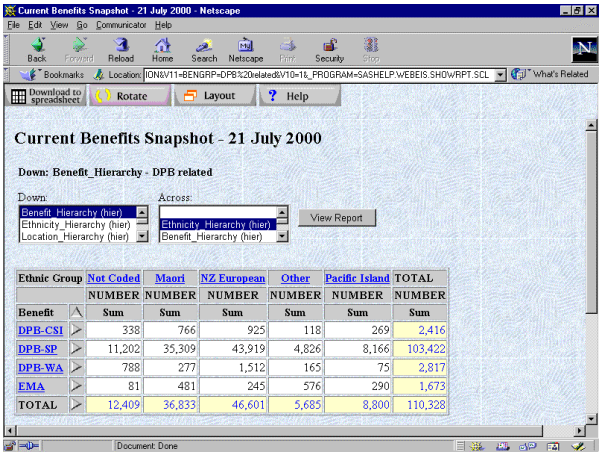
An example from our system will show how the MRV works.



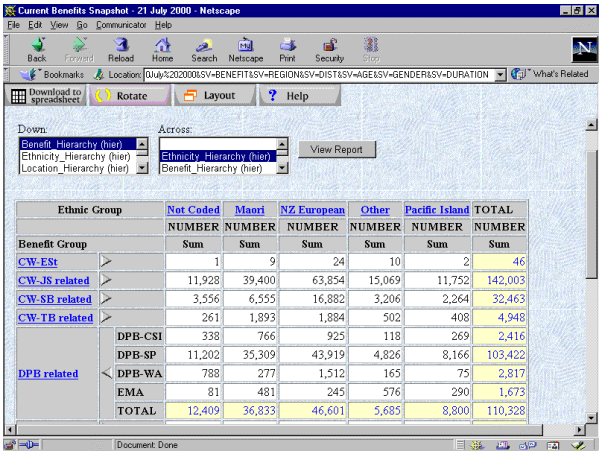
This screen is a standard HTML page, with each report link invoking the MRV. From the menu, choose the Current Benefit report of Benefit by Ethnicity for 21 July 2000:



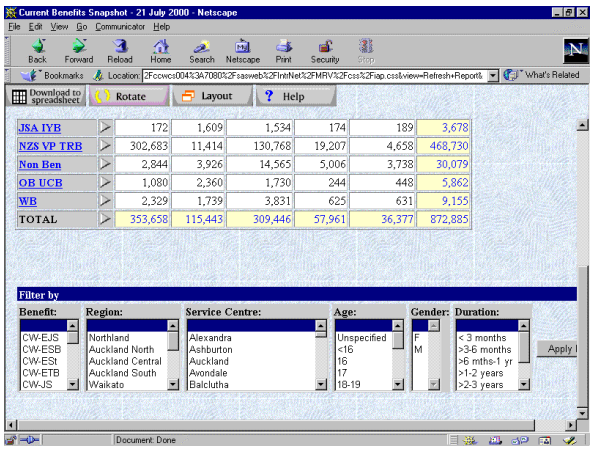
To get more detail on a particular area, for example DPB related, simply click on the underlined category (DPB related) and the report will drill-down to the level below. Only information on DPB benefits will then be displayed.



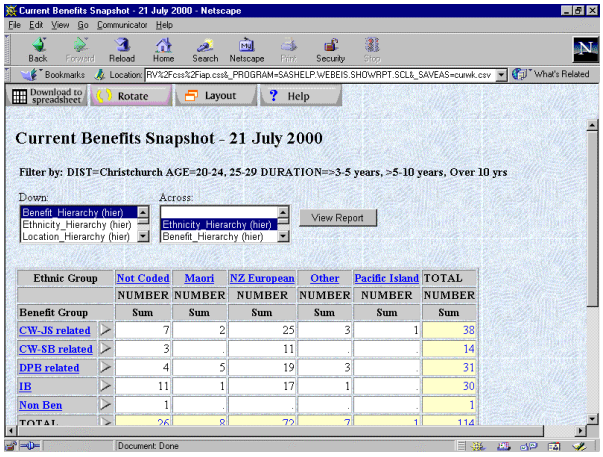
Or click on the arrow beside a category to expand that category to the lower level, but do so within the current table, to be able to compare the numbers with other categories:



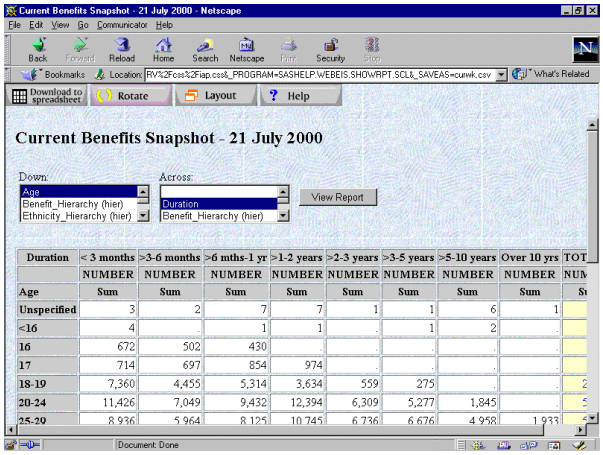
Going to the bottom of the report, you are able to choose filters to select the subset of information you are interested in:



Quite detailed requests can be made, for example let's choose to look only at people in Christchurch that are between 20 and 30 years old who have been on benefit for more than 5 years:



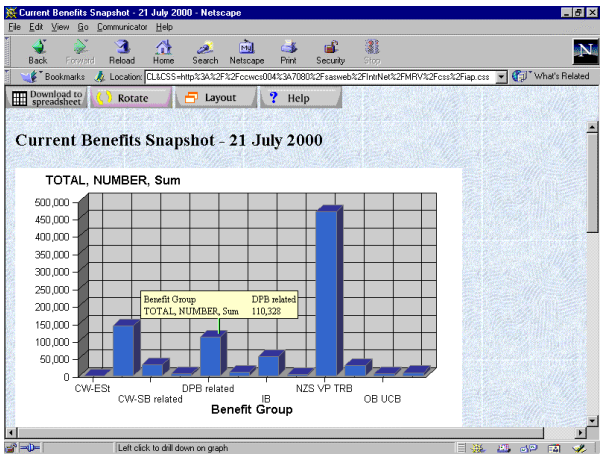
To change the row and column dimensions, simply choose from the down and across selection boxes, and press the View Report button. For example let's change to Age as the down dimension, and Duration as the across dimension:



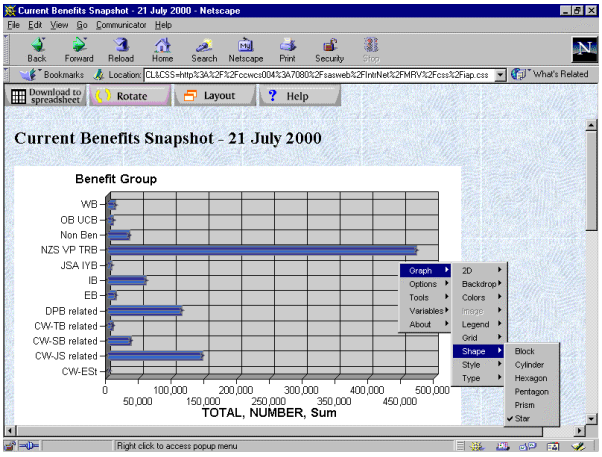
Because each report definition is simply parameters that are passed as part of the URL, customised screen definitions can be easily saved using the browser bookmark facility.

Adding the Graphics Applet gives a synchronised interactive graph and table. In the examples below the table is underneath the graph, and you need to scroll down to see it. You can reduce the size of the graph to fit both the table and graph on the same page, but then the graph size becomes too small.

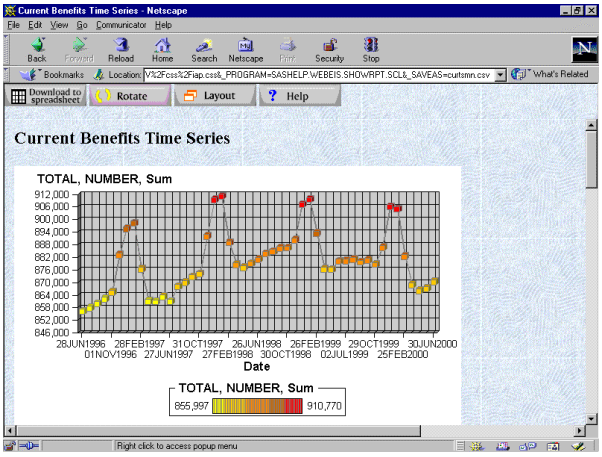
Graphics Applet bar chart. Hovering on a bar will display the exact values represented by the bar:



The graph can be customised on the desktop with features in the Java applet, e.g. changing colours, type of graph and bar shape:



And of course with time series data a graph shows far more than just a mass of numbers:



The SAS web site has a number of different working examples that you can go and try for yourself before committing the time to install in your environment.

The MRV has been in production use at the Ministry for over two years, and receives positive feedback from users. They like the ease of use, flexibility and speed of response (especially bearing in mind the large volumes of data we are dealing with). This has now become employees main source of standard information, as analysts receiving requests for this type of information will direct the person to the web site.

From the Developers Perspective: MRV set-up

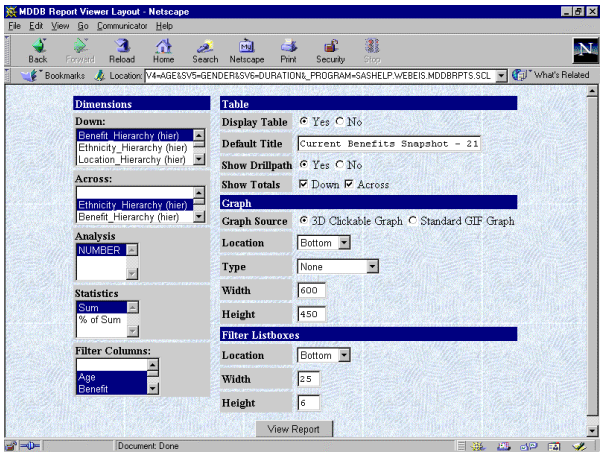
The SAS/Intrnet application dispatcher needs to be set-up and running. Initially, we didn't have SAS/Intrnet installed, so we used a SAS Institute consultant for two days to give us a fast start, which was well worthwhile. Although not complex, it is time consuming to understand all the various components and set-up requirements for SAS/Intrnet. He also set-up our first MDDB Report Viewer example. That for SAS 6.12, and we have subsequently implemented SAS 8.1 ourselves.

With SAS 8.1 there are a few extra issues to deal with, that were all resolved through the usage notes. These included:

- Initially it was running very slowly and then crashing with a "Segmentation violation". Setting –minstack 130000 when invoking SAS in start.pl resolves it
- The option –rsasuser needs to be removed when invoking SAS in start.pl
- The "unsafe" option should be removed from the PROC APPSRV statement in appstart_<portnumber>.sas

From the data perspective, MDDBs need to be created, either through SAS/EIS interactively or using PROC MDDB code which is similar to PROC SUMMARY. The MDDBs need to be registered in the EIS metabase, which is simply pointing at the MDDB and saying register it (Version 8 requires the extra step of creating the repository manager). The only bit of typing is if you need to define hierarchies (required for drill-down, so you probably will), and this takes a couple of minutes to define the three hierarchies we use. However we have 30 MDDBs, so the process will take an hour if all need to be re-created. We have looked for a batch method for defining these without success.

Following the set-up documentation results in an application where end-users have to work through three screens before viewing any report - choosing which MDDB to use, what row and column dimensions to use, whether a graph is required, what the filter variables are and so on. An example of the second screen for one of our data subjects:



This is simple for the developer to set-up, but not very quick or easy for the end-user. We decided it was too complex for the wide range of users our system would be catering for. We wanted the end-users to make a menu selection from a Web page and to be presented immediately with a high-level report.

This was fairly straightforward to accomplish – create a few lines of HTML specifying the parameters for the Viewer on a hyperlink option in the HTML page. However it was not so simple to find out how to do it, and what the parameters were, as this documentation did not exist with the pre-production MRV version! (good documentation does now exist on the SAS web site). Example HTML code for producing a report is:

```
… start of HTML page …
<LI>
<A href="http://server:port/ows-bin/
broker?_PROGRAM=SASHELP.WEBEIS.SHOWRPT.SCL&
_SERVICE=default&DEBUG=0&METABASE=libname.
metabase&MDDB=libname.mddb&D=Benefit_
Hierarchy&AC=&A=number&S=SUM&SV=benefit&SV=
region&SV-dist&SV=age&SV=gender&SV=duration
&ST=1&DT=Current+Benefits+Snapshot+-
+%%macrodate&DP=1&DC=1&ACB=X&GSC=1&GL=1&
GRT=NONE&GW=600&GH=450&SSL=4&SW=25&SH=6&CSS=
http%3A%2F%2Fserver%3Aport%2Fsasweb%2FintrNe
t%2FMRV%2Fcss%2Fstylesheet&
VIEW=Refresh+Report&">Summary Counts
</A>
</LI>
… rest of HTML page …
```

This may look rather cryptic, but are simply parameters such as:

D=variable	down variable or hierarchy
AC=variable	across variable or hierarchy
A=variable	analysis variable or hierarchy
S=statistic	statistics to be shown in the report
DT=title	title for report
SV=variable	variables for filter list boxes
DC=yes/no flag	show down totals or not
ACB=yes/no flag	show across totals or not

As we were still quite new to URL conventions we had many other things to learn, such as a : is represented as %3A and a / as %2F.

Testing and debugging could get somewhat frustrating. We created the web pages on a PC in Frontpage, moved the resulting HTML files to the UNIX server, and then invoked a desktop browser to test if they worked. Some errors got nicely trapped and reported on, e.g. MDDB does not exist. However others just left a blank browser window or the generic "syntax error", and were difficult to track down. An example would be having a filter variable specified that did not exist in the MDDB or EIS metabase. To be fair to SAS Institute, the developers probably assumed the parameters would only be used through the provided menu interface, and therefore some of the errors we were getting wouldn't occur. The SAS/Intrnet &DEBUG settings were very useful for assisting with some types of errors.

Cascading style sheets are also supported. They provide a lot of flexibility to smarten up the report, including background images, cell colours and text properties (font, colour, alignment, size and so on).

A number of our reports are updated daily. Another challenge was to automatically update the dates for the report which are hardcoded in HTML. The simple but effective solution was to have date place holders in the HTML page, and to use a batch editor to replace the place holders with correct dates each time an update occurred. A SAS program was used to generate the batch edit command file.

There was minimal set-up required for the Graphics Applet; place the JAR file in the correct directory and set a couple of parameters. To invoke the Graphics Applet was a matter of setting another parameter in the HTML when invoking the MRV, or for a user to press the *SETUP* button and choose to turn graphics on.

Performance

We have found the performance of the text-based reports to be very good, even better than was expected. Most of our reports are small, one to two pages, and will display in 2-3 seconds. Our largest standard reports are six to seven pages long, and take around 30 seconds to display. Some users have taken the system beyond it's expected boundaries by creating reports of over 1000 lines long (creating nested levels of results, e.g. age group within benefit type within location, to feed into Excel for further analysis), with these queries taking several minutes.

Most of the delay is due to network time and then the browser preparing the table for display, rather than data preparation on the SAS application server. From the browser you can see the server completes it's task almost immediately, and then the delay is while the generated HTML is transferred to the browser and formatted for display on the screen. The size of the final report (i.e. how many cells) determines the response time, rather than the volume of data being accessed. The HTML generated defines each cell with about 3 lines of parameters, with font, colour, size and so on specified. Hence the HTML file can easily get quite large.

The MDDB technology is so fast that we do not need to add any summary levels (i.e. sub-tables) for most of our MDDB's, those that contain less than 300,000 observations. It is impressive to see a report being displayed in 2-3 seconds, of say 6 numbers, and knowing that 300,000 cells have been summarised on-the-fly to produce the resulting numbers. It will depend on the speed of your hardware, and the trade-offs you want to make (i.e. disk space used versus CPU usage and slower response time), as to when you need to add sub-tables in your environment.

Our largest tables (time series over 4 years with many dimensions) are around 3 million observations, and these definitely require sub-tables to achieve acceptable performance.

Issues With MDDB Report Viewer

As you would expect with pre-production software, there were a number of things we felt should be changed, and a good number of bugs found also. SAS Institute were responsive to the problems raised, and all major items were corrected in the production release (Release 1.2 in early 1999). Each subsequent release has improved it further.

However there are still a number of outstanding issues we have with the MDDB Report Viewer, with the main ones being:

- Only one title is allowed. You really need more titles and the ability for footnotes also.
- Column headings need better customisation abilities.
- For time series data, the ability to lock the time dimension is important. Currently users can choose another variable for that dimension, which results in the whole time series being added together to create meaningless numbers, for example adding up monthly year-to-date figures over a year. We understand this can be achieved by overriding one of the methods, but haven't yet received appropriate information to do it.
- A nasty problem to track down was that some special characters (e.g. "/" and ",") in the data can cause HTML syntax errors. These don't show initially, only when the special character values are used for drilldown.
- A query limiter would be nice, with a warning when the query is too large to process in a reasonable time frame (user definable).
- If you drilldown then 'Download to spreadsheet', the spreadsheet shows errors.
- It would be nice to suppress the Layout button – this may be able to be done with a style sheet or overriding one of the methods.
- It would be good to freeze the title with the toolbar.
- Need to identify where a drill-down row or column has only one child, to save an unnecessary drilldown or expand operation.
- Need to add the ability to repeat the variables on the right hand side of a report, or even better to freeze the row and column headings.
- If more than one row is expanded, you get a "the report has invalid syntax" message when collapsing.
- The Back button should reset filter selections but doesn't – they are still highlighted and can be inadvertently reapplied.

We understand a number of these have been fixed for the next release (8.2), and others are on the drawing board for the releases following that.

Issues with the Graphics Applet
The Graphics Applet will be a very nice facility once two areas are addressed. The first issue is the number of bugs and problems in using the current version.

The second is the very long time it takes to show the first graph in a particular browser session: over 80 seconds as opposed to 2-3 seconds for an average MDDB Report Viewer table to display. This is because it is a Java applet, which enables it to have the great interactivity, but needs to be downloaded to the PC for each new browser session. Unfortunately there is not currently a way to store the Java classes on the PC to improve start-up performance as there is with AppDev studio.

We have included this graphics capability in a few places on our menus (with a pre-production label) . For showing basic graphs it is streets ahead in looks to the alternative SAS graphics available.

The main outstanding issues we have with the Graphics Applet are:

- The first graph shows correctly, as does modifying the layout and colours. However it does not allow additional variables to be specified for group, sub-group, depth etc. The options are there, but the only variable offered is the one already used in the graph, and selecting it again results in a non-sensical graph.
- Axis labelling overwriting.
- Not all tick marks are labelled for a categorical axis, therefore it is hard to know, for example, what's the bar between the 'Pakistan' and 'Portugal' bars?!
- Print and Save don't work.
- There isn't a way to save customised settings. You may spend 20 minutes changing bar style, colour schemes etc, but when you return to that graph all settings are back to the default ones.
- Similar to above, but more frustrating, is that after customising the graph you simply apply a filter and the graph settings return to the default values.
- For some nested axes (e.g. year and month) the order is incorrect.
- Pie chart text gets overwritten, e.g. slice labels.
- Not able to choose axis ranges.

Again we expect a number of these to be resolved in SAS 8.2, so hopefully this list will be shorter when presented at SUGI!

APPDEV STUDIO

What is it?
AppDev Studio is a self-contained drag-n-drop development environment for creating Java applications/applets (and HTML logic) for information delivery. You concentrate on building the user interface, and it effectively manages the back-end processes of connection to the SAS application server and accessing appropriate data. It consists of two components:

- WebEIS: For quickly building Java Web OLAP applications, with OLAP table viewers and graphs, text, images and hyperlinks.
- WebAF: For development of screens at a lower level, choosing from over 100 components. More effort, time and knowledge is required than with WebEIS, but you can build a more customised application, and produce a much wider range of application types. WebEIS is built using WebAF.

From our perspective WebEIS was more relevant, and that is what we looked at. We evaluated Version 1.2, and that is what this paper is based on. A new version (2.0) is about to be released with SAS 8.2. AppDev Studio supports both SAS 6.12 and Version 8.

From the End-User Perspective.
WebEIS offers the same capabilities as the MDDB Report Viewer, and a few more:

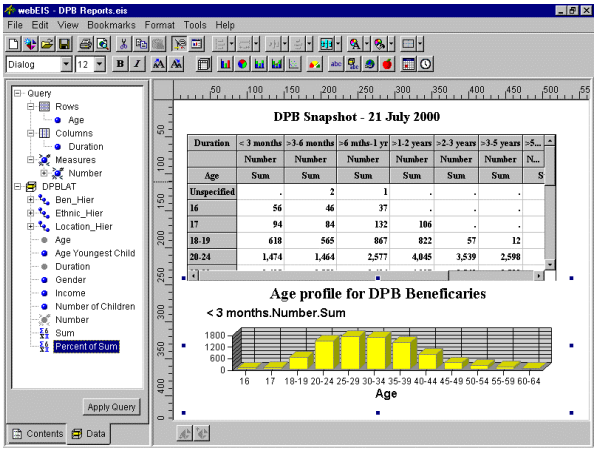
- Drill down or expand using pre-defined hierarchies
- Apply filters
- Change down and across dimensions
- Download to Excel
- Rotate the report
- Reach-through to underlying data

Extra capabilities include:

- Better customisation capabilities
- More report layout options
- Ability to change row and column sizes interactively
- Better integrated graphics
- Ability to sort a category column
- Ability to calculate new columns on-the-fly

It is also a much smarter looking interface, but it works in a totally different way to the MDDB Report Viewer, and is more difficult to use.

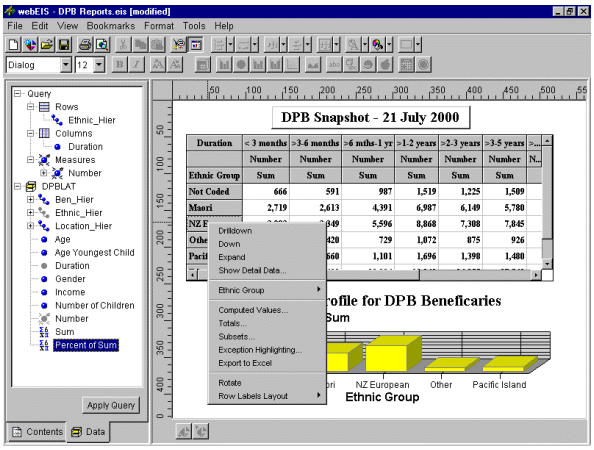
Let's look at a similar example to what we saw with the MRV. Note that these example screens are from the development environment: the user would see a much simpler interface in a browser, without the two lines of development icons.



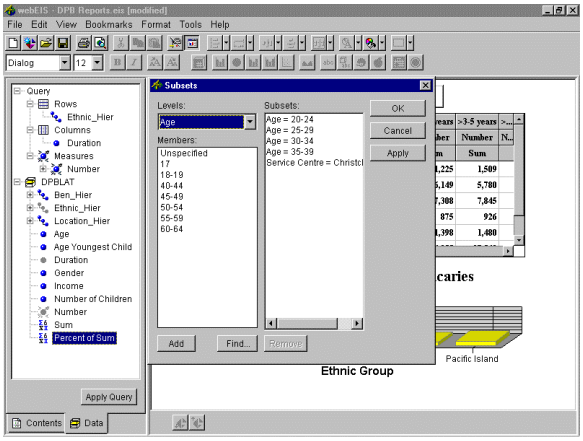
As well as the report, the user is presented with a Data Layout Window on the left hand side. The top *Query* section shows the current selections for the report: in this case the rows are *Age,* the columns are *Duration,* and the variable used in the cells is *Number* – the default statistic is sum.

The lower section, *DPBLAT* (the name of the MDDB data source), shows the other variables available for selection. To change the rows from *Age* to *Ethnicity, Age* is selected from the *Rows* category and dragged back and dropped on the data source. Then *Ethnic_Hierarchy* (for the ethnicity hierarchy) is selected and dropped on the *Rows* category. If different statistics were required, they would be selected from the bottom of the list and dropped onto the *Measures* category. When ready the *Apply Query* button is pushed.

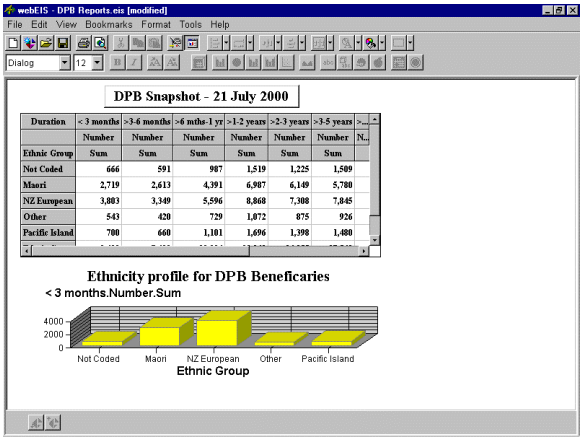
To drill-down you can double click on the category (although there is no indication you can drill-down, unlike the underline in the MRV), or use right mouse click to bring up a menu that gives access to lots of options, including drill-down. Options include drill-down, expand, rotate or export to Excel – the same functions as in the MRV:



Getting a subset of information is more cumbersome also. Select Subsets from the pop-up menu, choose the variable required to select on, and then select the values required from the list provided, as shown for age:



To gain more screen space for the report, or if the users won't be changing rows and columns, the Data Layout panel on the left can be switched off:



AppDev Studio Install and Set-up

Installation of AppDev Studio is a lot more automated than MRV. You load the CD and the installation procedure asks a couple of questions and then loads the software, including additions to the SAS System on the desktop (only used for development purposes) and the required Java Plug-in.

The SAS System on the server also needs to be updated to deal with AppDev Studio data requests. This is also straightforward – for UNIX servers simply copy a TAR file to the server and use sasmanager to apply the changes.

Testing and initial development can quite happily be performed totally in the PC environment. It is sensible to cut down large data sets to smaller representative ones to improve performance and save disk space for testing.

The installation procedure should set up a number of entries under the Windows start menu, such as WebEIS, WebAF and Start SAS spawner (which is the background SAS process that serves the data). When we installed 1.1 the spawner option was missing, so we had to create an icon to start the spawner manually. Our 1.2 install successfully created these options and more.

You also need to create an autoexec for the spawner, to set up access to libraries containing the required MDDB's, data sets and formats.

From the Developer Perspective

Following installation, WebEIS can be invoked and all further actions taken from within that environment. The help system is reasonably comprehensive, and the "Quick start" section a nice touch.

The first thing to do is to create a new document, and a new section within that. WebEIS maintains its own contents area (like Windows Explorer), and a complete application can be delivered to end-users as one item, with users choosing what reports they want to work with from the contents.

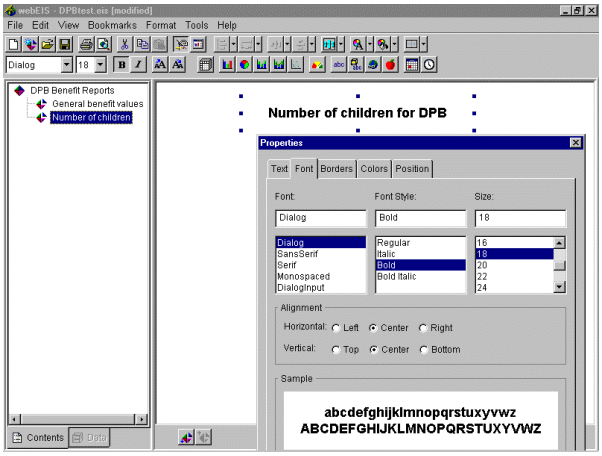
A connection definition to a data server needs to be created (from the Tools menu item), with appropriate parameters set. Set-up for the PC desktop as the data server is virtually automatic, as long as the spawner has been started. A nice facility is "Test the connection", which checks that WebEIS is talking to the backend data server. There is also a very helpful connection troubleshooting guide available on the SAS web site.

We did have difficulty connecting to our UNIX data server, and were unable to get WebEIS talking to the UNIX spawner. As we were only testing at this stage and not having large numbers of people using the data server, we used an alternative approach of invoking SAS each time we connected to the server, as advised by SAS Technical Support. This was done by adding the following statement into the SAS command field in the connection definition.

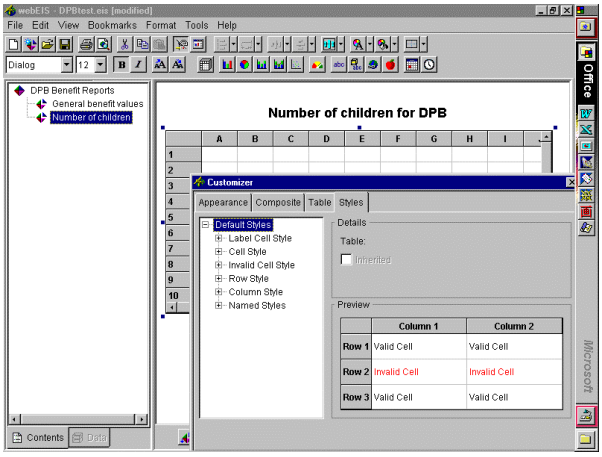
Sas – nodmr –comamid tcp –noterminal –cleanup
–config myconfig.sas –autoexec myautoexec.sas

Once connected to the data server you are prompted to select a data set or MDDB to report on.

Then development can begin. In Design Mode you simply drag-n-drop various components from the tool bar at the top to where you require them on the screen. For example (as shown in the next screen shot) drag the text component to the top of the screen to create a title, re-size it to the desired size, right-mouse click to bring up a menu and customise the text. Enter the text required, define the font, size, colour, borders and so on:



Then drag the table-viewer component onto the screen, re-size and set the required parameters, and the same again for a bar chart:



This is all that is required to end up with the example application showed the in “From the End-User Perspective” section. Switch to Preview Mode, and the report can be tested as the user would test it.

You should also be able to use Preview in Browser to see exactly what the user would once the application has been deployed on a Web server, however we haven’t been able to get that working in our environment yet.

To deliver the application to users the Package Wizard is used to create a single package as a JAR (Java Archive file), for use on any platform. Added in 1.2 is the ability to package it as a ZIP file or CAB (Microsoft Cabinet file format) file for PC servers. This packaging only selects the files and classes that are required for this document, and compresses the resulting file to keep the result as compact as possible. The JAR file is moved onto the Web server and is pointed at as part of an HTML page, and the job is complete.

Even though JAR files are kept as small as possible, they are still of considerable size for an applet – typically 1 to 3 megabytes in size. Generally Java applets are loaded completely from the Web server each time they are first used in a new browser session. This is not a problem with small applets or if they are used in a local intranet over a very fast LAN. For an internet environment or an intranet with slower connections, the applet start-up time can be unacceptable, for example over 5 minutes!

SAS Institute have devised a way to remedy this, called SASNetCopy. The applet itself is typically quite small - 10k to 20k - but the underlying Java classes libraries are the bulk of the data size. These don’t change often, and can be stored on the desktop. This will have a tremendous impact on initial applet start-up performance, and allow internet and distributed intranets to perform nearly as well as fast in-house LAN environments. There are a number of approaches to accomplish this: both for the initial download of classes, and ensuring that when the Java class library is updated when changes are made. A good paper on this topic called “SASNetCopy Overview” is available on the AppDev Studio Developers web site.

What has been discussed is using WebEIS at the fundamental level. More sophisticated customisation can be made in a number of ways, including going under the covers to override methods, change parameters in the HTML file calling the applet, or combining WebEIS and WebAF capability.

There is a sample application provided with 1.2 that embeds a WebEIS report inside a WebAF application, which provides push-buttons and selection boxes like the MRV, but with the smarter look and features of WebEIS. This would make a WebEIS application much easier to use for inexperienced users, and is an approach we will be looking into in more detail.

Performance

As discussed previously, a WebEIS application can take a long time to invoke if the Java classes are stored on the desktop.

Once invoked WebEIS is impressive with it’s speed. With small report sizes WebEIS was faster than the MRV, and dramatically faster for large reports. A few examples are provided in the table below.

Action	MRV	WebEIS
5 line report: 300,000 obs		
- initial load	5 secs	2 secs
- rotate	6 secs	1 sec
- drilldown	5 secs	2 secs
150 line report: 300,000 obs	29 secs	4 secs
10 line report: 1.2 million obs	30 secs	3 secs

Issues with WebEIS

We found some problems with WebEIS, but not as many as with the MRV. A number of the listed MRV issues don’t arise in WebEIS because it operates in a different way.

General problems include:

- Not able to choose Preview in Browser (however this is expected to be a site-specific issue).
- Printing a report creates messy output, especially when there are more columns than can fit across the page. At least all columns are printed, unlike the MRV which just cuts off all columns that don’t fit on one screen width. WebEIS also gives the option of printing to a PDF file, or emailing the current section being displayed.

The text viewer component seems to be very robust, with only a couple of issues found. The issues below were found in 1.1. We haven’t experienced these problems with 1.2 yet, so hopefully they have been resolved.

- Occasionally rows or columns are missing from the report. This would be of concern if it still exists in 1.2
- If scroll bars exist on both the MDDB text viewer and on the overall report, the text viewer has occasionally become corrupted.

We were rather disappointed with the graphics component. It is a form of the Graphics Applet, and although it is better than the version we are using with the MRV, we would still rate it as pre-production due to the number of issues we encountered. Many of the same issues exist, and include:

- The worst is that the default chart only represented values from the first column in the table being shown, e.g. the first age group. We would expect it to represent all people being reported on, and allow you to select a particular column if you wished. Other columns could be selected for display, but not the total, even if a total column existed.
- Not being able to specify sensible variables for group and sub-group categories. These options were often greyed out, which at least is better than MRV graphics where you could select a variable that didn't make sense.
- Axis labelling overwriting.
- Pie chart slice labels being overwritten.
- Not all tick marks labelled for a categorical axis. This isn't too bad if using it interactively, as moving onto the bar will show it's value. Printed copies are a problem.
- For some nested axes (e.g. year and month) the order is incorrect.
- Not able to choose axis ranges.
- Legends aren't very good, with some not showing anything. Others have a scrollable legend area that isn't very intuitive, and it's difficult to discover how it works.

COMPARISON OF MDDB REPORT VIEWER AND APPDEV STUDIO (WEBEIS)

User Perspective

The MDDB Report Viewer is more basic looking, but is simple and effective to operate. It is also fast to invoke, and gives good performance on all but large reports. For our user population it is a very appropriate tool, as we need to cater for a wide range of users, and it has some more advanced capabilities also.

WebEIS is a much swisher application, looks better, has more interactivity and offers more features. For the right target user group (anyone with a reasonable amount of PC knowledge, or if instruction can be given), AppDev Studio would probably be the superior solution. AppDev Studio also has a better growth path for an application, as further specific capabilities can be added via WebAF as desired. However it is more complex to use, and we feel it would be too difficult for a number of our users. We are considering deploying it as an advanced option, or for use by the business and data analysts (who are more PC literate).

Another advantage of AppDev Studio is the greater ability to create a customised application. WebEIS alone has better flexibility for the user interface than the MRV, and added to that is the comprehensive capability of WebAF.

Now that the effort of building the underlying AppDev Studio environment has passed, we would expect to see a number of other components added into WebEIS.

The graphics in WebEIS are better than with the MRV, but not by much.

Developer Perspective

Even though the MRV is not too difficult to work with, it is rather messy to implement, test and change when used as we do.

AppDev Studio is an integrated and comprehensive tool that is very easy to develop and test. Because it is data driven and choices are made by pointing and clicking it is not really possible to enter incorrect information, e.g. an MDDB or variable that doesn't exist (we haven't tested deleting an MDDB or variable to see what messages it returns).

There is minimal control over layout with the MRV, broadly selecting locations (e.g. filter boxes above, to the side or below the table viewer) and the sizes of the graph and table area. WebEIS is far more flexible, with text, images and hyperlinks being able to be added anywhere on the screen, as well as the core MDDB viewer and graphics components.

CONCLUSION

Web OLAP technology is a great approach for delivering easy-to-use fast and flexible information, in a cost-effective way.

Both the MDDB Report Viewer and AppDev Studio products work well for text-based reporting, albeit with a few issues to resolve. Graphics capabilities in both tools is disappointing, and really should be deemed pre-production status.

The MDDB Report Viewer is simpler to use for the end-user and should be considered where the target audience includes people that aren't very computer literate. The graphics portion can be considered for showing basic 3D graphs (rather than using the interactive capabilities), but it takes considerable time to invoke.

WebEIS from AppDev Studio is a more sophisticated and smarter looking tool, with better customisation possible and greater functionality available for the end-user. It could provide a more complex environment than untrained users would be comfortable with.

From a development point of view both tools live up to their promise of quickly delivering a useful Web OLAP interface with minimal development and maintenance effort. AppDev Studio is a comprehensive development environment with a wide range of features, whereas the MDDB Report Viewer is really a fixed application with the ability to change some parameters.

It is worth considering other options as well (for example Xplore and the new Design Time Controls) to determine which will meet your specific needs in the best way.

References

Web enabling a large data warehouse: *Improved Metadata for expert analysts and wider access to Management Information*: SUGI 24 Proceedings of the 24th Annual *SAS Users Group International Conference, 1999.*

Web enabling a large data warehouse: This is an updated version of the above paper, and presented at SUGA 99 and SEUGI 2000.

MDDB Report Viewer documentation on the SAS web site.

WebEIS online help.

Various papers from AppDev Studio Developers web site: www.sas.com/AppDev_Studio.

Acknowledgements

To Clare Somerville of Counterpoint Consulting, Wellington, New Zealand, for her assistance and encouragement in preparing this paper.

Author Contact

Colin Harris
Consultant
Warehouse Solutions
114 Homebush Road
Khandallah
Wellington
New Zealand
Phone 64 4 479 3193
Fax 64 4 916 3916
Email colin.harris005@mosp.govt.nz
Email cjharris@ihug.co.nz