Paper 191

WEB ENABLING A LARGE DATA WAREHOUSE: IMPROVED METADATA FOR EXPERT ANALYSTS AND WIDER ACCESS TO MANAGEMENT INFORMATION

Clare Somerville, Counterpoint Consulting Ltd, Wellington, New Zealand
Colin Harris, Warehouse Solutions, Wellington, New Zealand

ABSTRACT

The New Zealand Department of Social Welfare (DSW) has a large, well-established and successful corporate data warehouse. This warehouse first went into production in 1996. It has been continually enhanced since that time, and now services more than one government department. The hardware has been upgraded, the data quantity and sources have increased, and more vital information is being provided from the data warehouse. Although the number of users has doubled to over fifty, until recently access was still limited to analysts with specialist SAS® skills.

This paper presents our experience of, and the lessons learnt from Web enabling a large and mature data warehouse: from determining the real need and benefits, to undertaking research and appropriate planning, through to implementation.

The decision to Web enable the warehouse was motivated by two business needs. The first requirement was to provide analysts with improved metadata: centralising the information from a variety of locations and disseminating it to analysts in a timely and relevant manner. The second requirement was to provide direct access to management information from the warehouse to non-expert users, particularly in a time of rapid political change.

THE WAREHOUSE BACKGROUND

The Department of Social Welfare data warehouse first came into production in January 1996. The department sees the data warehouse as a continuous on-going project.

Environment

The data warehouse is based on a UNIX platform: an HP9000/K580 with six 64-bit PA8200 processors. There is one terabyte of available disk space operating in RAID5, and four gigabytes of RAM. All disks are fibre channels. The platform operating system is HP-UX version 10.20.

The database software is Oracle®, version 7.3.2.3. SAS release 6.12 is used on the warehouse platform and on the warehouse user desktop. SAS is used for analysis and reporting, and to process, update and maintain the Oracle database. The desktop is currently Windows NT 4.0.

The data warehouse is accessed through the department's TCP/IP wide area network. This network has about 180 UNIX servers, three Web servers, and 6,000 users around the country. Oracle Web Server 3.02 is running on the warehouse.

The Department

At the time the work on this project started, the Department of Social Welfare was made up of a number of separate business units:

- Income Support (IS): assess and pay beneficiaries
- Social Policy Agency (SPA): provide the Minister of Social Welfare with policy advice
- Children, Young Persons and their Families Service (CYPFS): manage children at risk
- Community Funding Agency (CFA): co-ordinate and develop community based services
- Corporate Services Group: provide centralised common services.

Each business unit has its own general manager, some have a governing board, and all report to the Director General. Recent developments have seen Income Support merge with another government department (Employment Services) to form a new department named Work and Income New Zealand (WINZ).

Data sources

Each business unit has its own computer systems. By far the largest system is the 95 gigabyte WINZ system, SWIFTT, which pays around NZ$10 billion each year to beneficiaries.

The original intention with the warehouse was to replicate the department's operational systems data, to enable reporting and analysis. The replication would record all details of transactions – no data would be overwritten. The warehouse team has been working towards this goal since the inception of the platform, with constant pressure to include more data from new systems. These operational systems are not static: systems come and go, SWIFTT itself is being rewritten, and all systems are being continually updated.

The data warehouse has a number of different data sources on it. The first data source was the consolidation of the historical SWIFTT data, dumped each month out of the operational SWIFTT system. This data is now static, since the monthly dump process has stopped. There is around 150 gigabytes of historical data stored in SAS data sets, covering a period of five years.

Historical monthly dump data extracted out of the Trace system deals with benefit debt. This Trace dump process continues. We store the latest month's data and consolidate a couple of tables – around 10 gigabytes.

We store some Human Resource data for each of the business units, in SAS data set format. We replicate the Student Allowances system for WINZ and the Social Work system for CYPFs, and we store around five other minor data sources.

The largest system on the warehouse is the replication of the SWIFTT system into an Oracle database. This database currently replicates around 60 tables out of a possible 218. For a variety of reasons, not all tables will be included in the replica, but new tables continue to be incorporated. This Oracle database currently stands at around 200 gigabytes and includes data starting from June 1996.

Because of the large size of some of the tables in the Oracle database – up to about gigabytes – a series of subsets are extracted from the Oracle tables and stored in SAS data sets. These subsets hold the latest month's current data; they provide quick access and improved performance. Summarised data sets

and multidimensional data sets are created, which provide a time series for the Forecasting Unit in SPA, and they feed into the Web access.

Business Users

There are around 55 analysts known to the data warehouse. They come from different business units, and their access and security requirements vary depending on the business unit they belong to and the data required to do their jobs. Analysts from the Social Policy Agency, for example, use data belonging to different business units – they have no large operational systems of their own.

All users of the warehouse require SAS skills to access the data. The need for flexibility means that a menu driven approach is not appropriate.

Reporting requirements

Different business units have a different focus in their reporting requirements. SPA tends to look at data over extended periods of time, and often perform one-off analyses. WINZ analysts often use the most recent month's data, and have a high number of routine reports that they generate. These routine reports are then circulated both internally within the business unit, and externally to the other business units, libraries and other government departments.

Information generated from the data in the warehouse is widely used, but often those receiving this information do not know where the data originated. They know they have an Information Centre within WINZ who will perform the necessary magic to give them the numbers and information they require. In some cases, the Information Centre circulates reports such as their quarterly report to over 50 users around the department. This is all done by sending out hard printed copies, and email.

Changing times

The department is in a state of flux: Income Support and Employment Services have formed a new government department. Children, Young Persons and their Families is in the process of merging with the Community Funding Agency. The Department of Social Welfare is turning into a policy advice Ministry.

Other government departments are requesting access to the warehouse, for example the Ministry of Education perform forecasting on the Student Allowances System for WINZ; Department of Housing analysts now also access the warehouse.

New data sources are constantly being included on the warehouse, along with new access methods and interfaces.

In Summary

There are several business units and different government departments involved in the warehouse, either as suppliers or users of the data, or both. There are multiple data sources and replication systems, users with different access rights to data, and varying needs in terms of analysis and reporting. All these things are constantly changing.

Any project of this nature must involve the users of the data, and recognise these different needs.

WHY WEB ENABLE THE WAREHOUSE?

As a result of this platform, analysts have access to data they could not access before, or could only previously access at a high cost. They are able to do time series analysis much quicker with the high availability and accessibility of the data. It's now possible to track individuals and client cohorts through the system over any period of time; analysts can build up profiles of clients and investigate why people stay on benefits for long periods of time. The warehouse is used for customised service, helping to identify and clarify staffing workloads and shortages; it's used for identifying debtors and assists in geocoding.

The warehouse has a high profile. Treasury is a strong supporter of the platform: they see a high level of access to data from the systems on the warehouse compared to those that are not. The Director General also supports the platform and encourages other business units to ensure their data is made available on the warehouse.

So why change? Why not keep going the same way? Add in more data sources, give access to more data analysts from more business units, upgrade the hardware, and so on.

There are several reasons behind the project to Web enable the warehouse.

- It was always the intention to widen the access to the data. This would allow all relevant staff direct access to more up to date information at all times; allow them to explore the information in many different ways; and substantially reduce the considerable burden of ad hoc requests to business analysts, freeing their time for more sophisticated analysis.
- Many reports produced by WINZ are widely circulated in the form of printouts and spreadsheets. This process could be facilitated with the provision of Web access.
- There is no existing easy method of disseminating information about the data to analysts. Metadata resides in different areas, on different platforms and in different forms. The Web would provide one centralised place to store metadata and give analysts access to the information on the data that they require.

THE PLAN

The decision was made to Web enable the data warehouse. We identified the need to provide access to static reports, graphs, tables of data, documentation, and the ability to query the data to create ad hoc reports on the fly and drill down capabilities.

We began this project from a position of ignorance: we were familiar with the Web as users but had never designed a Web page before. We knew about data warehouses but not much about graphic design.

Initially we started out by experimenting with the different software tools and seeing what we could or couldn't do with them. Working in this way we found it pretty simple to set something up. But we then had to pause and rethink: was this something really worth having? What was the value of what we were producing? Put simply: was it any good? And the answer we reached was that it probably wasn't.

So we developed a project plan. In the plan we identified five phases of development for the project:

- Investigate
- Research
- Implement
- Launch/publicise
- Review

INVESTIGATE

The investigation phase included finding out about the DSW Web policy. We needed to find out who to approach within the department. We liaised with the different business units, and determined the SAS software status.

As the managers of the warehouse platform we are responsible for making sure that the data arrives on the platform, that it is validated, transformed, consolidated, and whatever else is required. But we are not responsible for analysing and reporting on the data. So there was a question of definition of roles here, and a reliance on those who did analyse the data. In the same way that the initial development and continuing extension of the warehouse involved the users, this project also would need to have the input of users from different business units. This input was required not only to help identify the appropriate content of the Web access, but also for testing and review.

Tools and Infrastructure

There was a DSW intranet infrastructure in place, so we didn't need to start from scratch: a nation-wide network existed, three dedicated web servers were in place, a DSW home page with some applications was already deployed, and all desktops had Netscape and MS Office installed.

We considered what was required to cover all aspects of the project, and determined what tools would be needed.

The content of the data warehouse site would include:

- Information from the data warehouse:
  use SAS/Intrnet® software
- Word and Excel documents:
  use Microsoft® Office web tools
- Web site structure and menus:
  Microsoft FrontPage® and Image Composer
- For fine tuning:
  HTML

For delivery to users, we were advised to set up our own Web server on the data warehouse platform that would be linked from the DSW site on the generic Web servers. We note that this is contrary to general advice, which is to keep the Web content on dedicated Web servers, and only process the dynamic SAS requests on the SAS application server. We intend to monitor usage and performance, and change this if required. Oracle Web Server is the DSW standard, and hence was used for this project also.

We identified five new pieces of software which we would have to become familiar with: MS FrontPage, HTML, Oracle Web Server, MS Image Composer, SAS/INTRNET, plus of course the Web infrastructure.

RESEARCH

Phase two was research. As it turned out, we underestimated what was involved in this stage – it required a lot more time and effort then we had anticipated.

Researching Web Design

What makes a good Web page? Often you go to sites and it annoys you or you leave it because you can't be bothered to wait for it to appear. You may not even know why you like or dislike a site. We had to know more.

Graphics were another area for research. What size of graphic will provide an adequate response time for a Web page? How long will people wait? What are the different types of graphics and how can you reduce their size, or make them appear gradually without waiting until they are perfect? What graphic file type should you use, and when? Should you use animation -- how?

The Web itself would provide us with the information we needed on Web design (see the Reference section for a list of useful sites). In researching Web design, it was soon apparent that the more we learnt, the more we needed to know.

Users

We identified two sets of users within the department who would want to access the warehouse Web pages:

- Analysts looking for information about the data
- Wider audience wanting to access management information.

We had an advantage in that we were designing an intranet site – we would not be setting out to entice and invite unknown users from around the world. We knew our users and why they would visit the site. As an intranet site, we also knew that users would go there with a purpose and would not want their time wasted. They would want to find what they were looking for and get on with their job.

We estimated the level of Web familiarity of our users as being not very high. Not everyone in the department has access to the WWW, and not a lot of use is currently made of the intranet as most business units are just now in the process of developing their sites.

Basics

There are some basic things to think about or include on your Web pages. Pages should be spell checked, grammar checked, and no slang used. Users need to know what to expect on a Web site, especially if they are not experienced Web users. There are issues of confidentiality to consider – particularly on a platform that contains personal details of customers.

Web pages are not like the printed page of a book. Web pages are more independent, and so they need more information on them. They need a title, author, revision date, and at least one link on each page.

Avoid "Being Developed" signs: a good Web site is always developing, it should not be static. And too many "Under Construction" signs can put a user off for good. Allow for growth.

Content

The contents of the Web pages should be useful, they should add value. It should be easy to find what you're looking for, and shouldn't waste users' time. The contents should be accurate, current, and complete, with all links in working order. The home page should provide an overview of the site.

Layout/Appearance

The page layouts should be consistent. A user first sees the page as blocks of text. If these are all aligned in different ways then this can be disorienting. Left align your text, and avoid the need to scroll across pages – 30 to 40 characters per line is ideal.

In our culture we read top to bottom, from left to right. On the Web page, it is the top ten centimetres that are visible. The Web page should reflect this. The most important facts should go at the top, in particular in the first four lines.

A clear system of menus is required, an identifiable structure. The optimum number of menu options is about five to seven, but beware of too many menus – it can simply take too long to get where you want.

Consideration needs to be given to the length of the pages: users should not have to scroll through long documents – two or three pages is the maximum. Long pages are easier to manage and to print. Short pages should be used for the home page and menus; they are easier to browse and read online.

Pages should look nice. Good use of white space is recommended, just as for any other document. Don't overuse bold text as this jumps out at you on the page, and avoid using all upper case letters.

Graphics

There are different types of graphic files and it pays to know something about them. Graphic Interface Files (GIF) are limited to 256 colours. They are good for diagrams, navigational graphics and transparency. JPEG files are better for photos and can be compressed. There is always a balance to be maintained between a great graphic, and how long the user has to wait before it arrives on the screen. Modify the image so that it uses no more colours than necessary. Maintain a balance between the quality of the graphic and the size of the file – it should load within ten seconds. Interlaced GIFs and progressive JPEGs are ways of ensuring that the graphic starts to paint itself on the screen roughly, before the full image arrives.

Graphics should fit on the page being viewed. They should be interesting, pleasing, add style, be consistent and meaningful. Icons can be useful in jogging the memory. Backgrounds should be light pastel shades. Avoid too many different styles of types, colours and links.

Animation will catch the eye. For that reason it can be distracting and should be used sparingly. If used, animation should be meaningful; it should add to the content and not disrupt.

Navigation

Navigation refers to navigation around the site, but also to navigation within a particular document. Users should be able to enter a site, and find the information they want in the fewest number of steps. They should always know where they are. Provide a link to the home page on every page. Link to the top and bottom of the document. Provide a meaningful next and previous section – with the name of the section not just a "Next" or "Previous" button. And duplicate your navigational items at the top and bottom of pages.

Avoid dead ends, links that don't go anywhere, and the option to click here.

Software Tools

We had to research and obtain the requisite software. We didn't know how to put it all together: how do you move Web pages onto the UNIX platform? How do you incorporate SAS output into FrontPage? How do you link the warehouse Web pages into the Corporate Web?

We had to research how to load static SAS output onto the Web, and how to incorporate dynamic pages for ad-hoc requests and OLAP-style investigation.

There were Word documents, Excel spreadsheets, SAS data sets, and the contents of the data dictionary Oracle CASE repository. All these things had to be included in the Web pages.

From the software perspective we wished to make sure we gained a basic understanding of the tools, and so we created a couple of 'proof of concept' examples to be satisfied it could deliver what we required. Then we needed to ensure that the components would work well together. This should enable us to enter the implementation phase confidently – knowing everything works, and that it's just a matter of putting in the time to create the results.

SAS/Intrnet tools

For static reports, we found the SAS Web publishing tools very easy to use, being able to produce good results only minutes after installing them. Although the tools are good, and provide a fair range of customisation, there are still times when even more flexibility is desired. Initially we thought that a nice find was the ability to insert one HTML file into another HTML page. This was extremely useful to insert SAS generated output into a skeleton page created by FrontPage, therefore having consistent headers, footers, table of contents, images and so on. Unfortunately other limitations prevent us from using this feature at present.

What concerned us was the development of the dynamic drill-down reports we required. We knew what we wanted – something akin to SAS/EIS OLAP reporting – but via the Web. We had seen some examples, but those required a lot of coding to build the HTML pages (or Java programs), and the underlying SAS code. Fortunately, during our period of research, the MDDB viewer was made available by SAS, which fitted our needs well.

Microsoft Office Web tools

These also worked well, and in a straightforward way: it's easy to choose the "Save as HTML" option. Because of the size of some of the files we were dealing with, and the fact that they are constantly changing, we did not take this approach.

Microsoft FrontPage

We found FrontPage to be a very well featured and useful product, but frustrating to use. It generally provided more than we required, and impressive results could be obtained quickly. However we continually battled with inconsistent performance and reliability – even after installing different versions on different machines. It often took two or three attempts to perform an action, regularly locked completely and had to be restarted. Image Composer is a very nice tool for creating and manipulating images.

Care needs to be taken when opening an image from FrontPage Editor in Image Composer, that you do in fact save the whole image, not just part of it.

Options within Navigator in FrontPage Explorer were too limited to make it useful. But something like this is required: even with careful file naming standards, very soon there is a large collection of files to keep track of. A well planned directory structure and file naming conventions are essential.

In hindsight, and after discussion with others, we would take the time out to learn HTML and would not use FrontPage.

IMPLEMENT

Armed with our new found knowledge, we set about making some basic design decisions. We searched the Web for an agreeable background and experimented with its size and colour. We created a logo for our Web site, which would convey some abstract representation of the ideas behind data warehousing.

We wanted to clearly brand the site and the two main branches: the metadata and management information options. We designed and created a warehouse image map (with Image Composer) to be used as the top banner of each page. Within that image map the Home Page, Metadata Home Page, and Management Information Home Page options are clearly identifiable.

The next step was to create a skeleton page to be used as the basis for all other pages, to ensure a consistent look and feel throughout the site. We tried a variety of effects, images and colours, but in the end settled on a simplified design using just the basics, to avoid distractions and wasted time. This included the information required, the branding banner, and useful navigation and help facilities.

After that we mapped out the menu structures. This was also more time consuming than expected because of the possible combinations.

METADATA BRANCH

The Need for Metadata

There are several definitions of metadata. It has been used to refer to tables of summary data; Oracle refers to it as the layer between the data and an application, for example in Oracle Discoverer. But the most common definition within the data warehousing concept is “data about data”.

If we run a `PROC CONTENTS` we get information about the data set we ran the contents on: the data set label, the variables and their length, format, sort order etc. The SAS dictionary facilities takes this one step further with a number of tables and views containing information on SAS data sets and libraries. For many people, that’s where metadata begins and ends.

Metadata within the data warehouse is much more than this. A data warehouse without metadata has been described as being like a tourist in a city, but having no information about the city, and the city administrators having no information on how big the city is or how fast it’s growing.

There are different reasons why metadata within the warehouse is so vital. One reason is that the warehouse can be a large and complex environment. For example, in DSW there’s one terabyte of disk space with multiple data sources. One data source may have data in several locations. SWIFTT data alone has 150 GB of historical data, plus a 200 GB Oracle database, then there’s subsets of data, summarised and multidimensional data sets. Assuming you knew that this was the source you required, how would you do something as simple as counting the number of people currently receiving benefits?

Another reason for the need for metadata within the warehouse is that there are two distinct sets of users of the warehouse. There are the administrators and developers, but there are also business users and analysts. Both groups of people need information or data about the data in the warehouse.

Just as there are different types of users, there are also different types of metadata for their use: business metadata and technical metadata. Business metadata provides the link between the database and the business users. It tends to be less technical, and provides what amounts to a road map for access to the data. For example, the users maintain a Business Rules document that would tell an analyst how to count beneficiaries. It gives a brief description, the business rules to follow, the code rules, and sample code.

There is some overlap between business and technical metadata: table names, indexes etc. But technical metadata needs to include what programs are run, the job dependencies, how the data is transformed, execution times, security, and much more.

Central Metadata Repository

Before the Web development, analysts’ access to metadata was limited. Some SAS data sets had been set up which kept them informed about what tables had been updated, and when. There were documents like the Business Rules document that resided on a shared drive, but access to this drive would change with the creation of new departments. Warehouse developers maintained many files that contained historical information about the data, but these files were not available to users.

In short, it was metamuck: files in multiple products and repositories. There were Word documents, Excel spreadsheets, the contents of SAS data sets, output from SAS jobs, and the contents of the Oracle CASE data dictionary, all in different locations. What was needed was one central metadata repository. Although we were aware of the need for a central repository, the reality of achieving it is far from easy. One reason for this is that there are no common standards or guidelines concerning metadata. There are a number of metadata repository tools, but they don’t have much in common. Different tools from different vendors must be able to freely and easily access, update and share metadata.

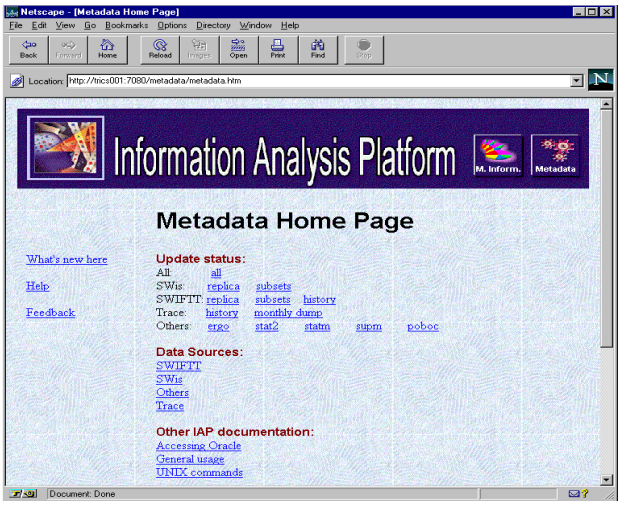
Before the Web development we turned to the SAS Warehouse Administrator. One of the things we hoped this software could deliver for us was to fill this need for a central metadata repository.

Some work was done with Warehouse Administrator. We found that it could not easily be made to fit a large and already well-established data warehouse of this kind. In many cases, code would need to be written to enable it to cope with things as simple as a data source input file that changed its name each day. And Warehouse Administrator is designed to take multiple sources into one subject area, rather than taking one data source to populate multiple tables. There were also issues of surfacing the data to the users, without the ability to drill through to the data.

Warehouse Administrator was primarily seen as a tool for developers; trying to make it fit the role of delivering metadata to users did not seem to work well. Instead, we concentrated on surfacing the metadata to users through the Web interface.

The Metadata Pages

The department's data warehouse is known as the Information Analysis Platform, or IAP. The Metadata Home Page is designed to put the information which is likely to be most frequently accessed by analysts at the top of the page: the update status of the various sources.



Metadata Home Page

This page has links to the home page for each of the main data sources. Other general documentation includes instructions on how to use the warehouse and get the most out of it, a page of useful UNIX commands, hints on how to access Oracle using SAS for the best performance etc.

Update Status

One link from the Metadata Home Page leads to the update status of all the data sources.

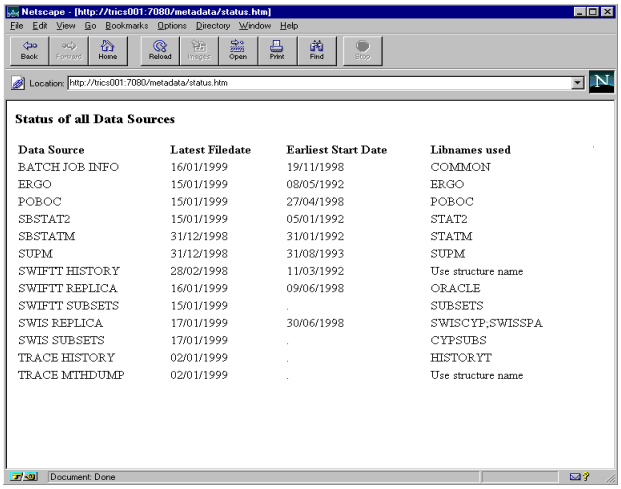
Following the successful completion of each update, SAS updates an HTML file with the update status, using the data set to HTML macro (ds2htm) from the SAS Web Publishing tools:

```
%ds2htm(
  runmode=B,
  htmlfile=/metadata/status/replica.htm,
  openmode=replace,
  data=update status data set,
  border=n,
  talign=left,
  obsnum=n,
  clface=timesnewroman,
  clhalign=left,
  cltag=no formatting,

  id=table name variable,
  iface=timesnewroman,
  ihalign=left,

  vhalign=left,
  var=filedate startdt,
  labels=y) ;
```

This code refers to an update data set which users can also reference programmatically.



Update Status of all Data Sources

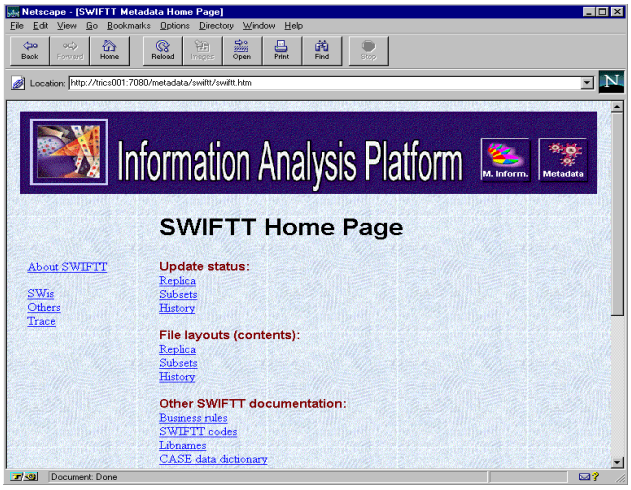
A conscious decision was made in the original design of the Web pages to brand the pages with the warehouse logo as much as possible. We wanted the HTML generated by SAS for the update status to be displayed with the warehouse banner at the top of the page, within a standard warehouse page.

The mechanism for achieving this elegantly is called Server Side Includes (SSIs), and is different for each Web server product. In Oracle Web Server it is called LiveHTML. This worked well for pages with less than 32 lines, but refused to work for larger pages. Different versions of Oracle Web Server were installed and tested, but to date no solution has been found.

An alternative is to use SAS processing to concatenate the components of an HTML page together – the skeleton followed by the generated code. This gets messy, and so hasn't been attempted. We wait for an Oracle solution. In the interim, the HTML output generated by SAS is displayed without a warehouse banner and other standard settings.

SWIFTT Home Page

The main data sources all have home pages.

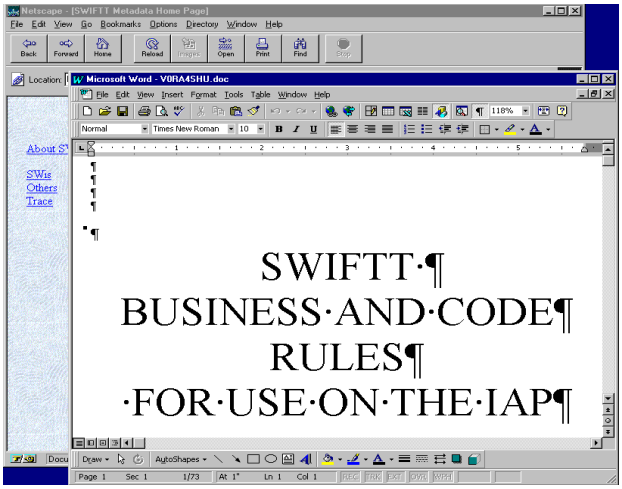


SWIFTT Data Home Page

In addition to the status, this page includes the contents of the different files. A `PROC CONTENTS` is run, the contents output slightly modified, and an HTML file is updated by SAS.

Various documents and files can be accessed for this data source: business rules; codes; variable details; libnames; indexes; sort orders; sizes of tables (number of observations and gigabytes used to store the data to give an indication of likely processing times); change history; and the data dictionary from the Oracle CASE data dictionary.

These files are in different formats: Word documents, Excel spreadsheets, Acrobat files, csv files etc. Large Word documents presented their own set of problems. Once converted into HTML format, they lost much of the Word formatting, like the facility to navigate through the document. To display them more effectively on the Web, they should be split into separate documents; and MS FrontPage has difficulties with very large HTML files. The question arose of how much work we should put into reformatting documents, particularly when they could quite likely be changed and updated. There was also the question of wanting to display them within the warehouse banner pages.



Word Document Displayed

Because of these problems, we decided to simply display Word documents within Word, Excel spreadsheets within Excel and so on, and forego the warehouse banner.

This in turn raised other issues. Although all the users had a standard desktop image and all relevant software products, the settings within Netscape for the display for other file formats was anything but standard. Instructions had to be issued for users to adapt settings in order to gain access to the different file formats.

MANAGEMENT INFORMATION BRANCH

The development of the Management Information pages was time-consuming, as fundamental work was required for user needs analysis, and the creation the of the underlying summary data layer required.

Objectives

The major objectives of supplying Management Information through the Intranet were:

- Provide all relevant staff with direct access to appropriate information
- For this information to be as up to date as possible, generally as at the end of the previous business day
- To allow users to explore the information in various ways, to gain new insights in understanding
- Free up the time of the business analysts by allowing staff to produce many results themselves. This would enable analysts to devote more time to more sophisticated analysis work.

Users

Previously when staff required a report or some information that was not available from the regularly produced standard reports, they would submit a request to the central business information group. This group had good access to a wide range of up-to-date data in the DSW data warehouse and skills in using SAS System tools.

Our requirement was for a wide range of people with varying IT skills to access the information via the Web site, from CEOs through to front-line staff. We needed it to be simple to use, but also offer enough functionality to explore the data further. It was important that good results could be achieved without massive amounts of development time, as there were only two people in our data warehouse team.

We narrowed our initial needs to two areas: firstly some static reports and graphs to give high-level summary information, and secondly the ability for users to do ad-hoc reporting and analysis – particularly slice-n-dice OLAP-style work.

Because of the large potential user base (up to 6,000), it would not be practical (or economical) to implement a SAS-based solution on every desktop. The advances in Web technology allow these objectives to be met in a cost-effective way.

Development Stages

As with many projects of this type (Management Information, Reporting systems and so on) we knew it was essential to undertake appropriate analysis to determine users' needs, and the required supporting data structures, before getting too far down the track. This was even more important with the scale of the DSW data warehouse – from initial testing we calculated it would take 35 days of continuous processing on the large UNIX box to create the historical time series for the largest subject area.

We also wanted quick results, which of course meant an iterative development approach was most appropriate, as it is for most information delivery systems. The iterative approach also means you are obtaining early feedback from users to enable the system to be fine-tuned to meet the real business needs.

Much of the analysis work was carried out in workshops with representatives from key user groups, where requirements were discussed, compromises made and decisions agreed.

The broad development stages were:

1. Determine overall scope.

The wide range of information needs was categorised into 14 subject areas, and these were prioritised. Each subject area required 'snapshots' at a point in time (e.g. yesterday, last week, last month and last year), and historical time series (e.g. three years of monthly data with many variables, and three years of daily data with only key variables).

Responsibilities were also set – who would refine the data requirements, who would write the extraction and summarisation logic, who would determine a representative group of test users and so on. It is important that each user group is assigned a set of responsible tasks: as well as taking some workload off the development team, it also gives the users more 'buy-in' to the project.

A Project Definition Document was produced that documented the agreed objectives, responsibilities and so on. This was a living document that grew to contain additional information as it was made available, for example data definitions for each subject area.

2. Focus on initial subject area for feasibility, 'a quick win', and to gain valuable feedback for other subject areas

The detailed analysis of a subject area took even longer than expected. Firstly there were literally dozens of variables appropriate for each subject area, many of which could be summarised at different levels. It was not feasible to provide all information -- there would be trillions of combinations requiring terabytes of disk -- so a compromise position had to be agreed. The Web interface would hopefully satisfy the majority of common information needs, but the remainder would still need to be met by the analysts.

It was also made more difficult as we were not starting with a clean slate. Each business unit was already producing reports from the data warehouse, but using different data groupings, e.g. for age bands or income groups for customers. We wanted at least the default groupings to be consistent, so we created a central format library containing the agreed definitions, and a controlled update mechanism for changes.

With the initial definition agreed for the first subject area, we quickly developed the logic to create a single data set from the data warehouse, summarised that to the required level, and fed the results into an MDDB. It was valuable to get the results of this early, to give a good indication of feasibility before grinding through the definitions of all other subject areas. Was the use of disk space viable for the time series required? Was the data preparation time practical? No use having daily updates to data that takes 30 hours for instance! It is not really possible to estimate these factors, as with ten or fifteen classification variables you are not able to predict the number of valid crossings containing data..

With an MDDB structure now in place, we put together a simple Web interface to act as a demonstrator to show what could be achieved, and to be used to gather feedback. We were very pleased with the performance of the MDDB – response was virtually instantaneous, with the only delay being transferring the results to the Web browser.

3. Roll-out pilot Web site to test user group.

The Web site was made available to a small test group of users who had been involved with the project. They raised a number of questions about usage, and provided feedback. The Web site was enhanced again, basic usage instructions prepared, and made available to a wider test group – this time business people that had no prior involvement with the project.

4. Repeat definition and implementation process for other subject areas.

Data Preparation

In our case we already had a data warehouse as a stable platform to draw the required data from. If a data warehouse does not exist, it is thoroughly recommended to implement one as a separate part of the information delivery project.

Many data warehouses would already contain pre-summarised data that can be directly used to feed the Web reporting requirement. Our data warehouse did not contain that summarisation layer, and hence the need for the analysis and data definitions discussed earlier.

As a spin-off from this project, the data warehouse now contained jointly agreed summarised sets of data. As well as feeding the Web interface, these data sets can be used directly by analysts for their reporting and analysis needs – saving them development and processing time, and ensuring they are using a correct set of summarised data.

The processing logic to produce the required set of detailed data was quite involved, due to the nature of our data. Standard base SAS data steps and procedures were used, but processing data for one day took about an hour CPU for our largest subject area.

Once the detailed set of data was created, about a million observations, the summarisation steps were very rapid – both to develop and run. `PROC SUMMARY` created the summarised set of data in less than a minute, and `PROC MDDB` loaded that into an MDDB in around 30 seconds. `PROC MDDB` code is similar to `PROC SUMMARY`, with an example being:

```
proc mddb data=summary_input_data_set
    out=mddb;
  class benefit serv income age gender;
  class region district/ascformatted;
  var number;
```

These needed to be defined in the EIS metabase for the MDDB Report Viewer to operate, all being automatic except for defining the hierarchies – which only took a minute or two.

Management Information Web Interface Tools

The requirement for static reporting was straightforward, with the Web Publishing Tools meeting the need with minimal development effort.

Ad hoc reporting posed more of a problem. We had been monitoring SAS/Intrnet developments since it's introduction, and had expected that we would have to code something ourselves at a low level (e.g. develop HTML pages to submit SAS code through the application dispatcher). Luckily for us SAS introduced the MDDB Report Viewer, which met our needs pretty well.

The MDDB Report Viewer allows users to perform OLAP-style reporting and graphics, including drill-down and slice-n-dice capabilities, through a Web browser without running SAS software on the desktop.

The first couple of releases were rather basic, but 1.2 is quite reasonable. It was still pre-production status when this paper was written, but hopefully will be production by SUGI. Documentation is still sketchy at present, and it took some digging and guessing to get what we wanted. There are quite a list of things we'd like to see changed, and of course a few bugs. Some of the issues are listed in the Report Viewer Details section below.

The major advantages are that it is very quick to set-up, provides a good deal of functionality immediately, and performs quickly. The down side is that you don't currently get much customisation control, which is frustrating.

The new Graphics Applet (also pre-production) 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. Unfortunately we are still having problems getting this operational in our environment, but are very keen to implement it once the problems are resolved. We are running Netscape 4.5, and although the sample graphs work correctly, the applet fails with our data and viewer set-up.

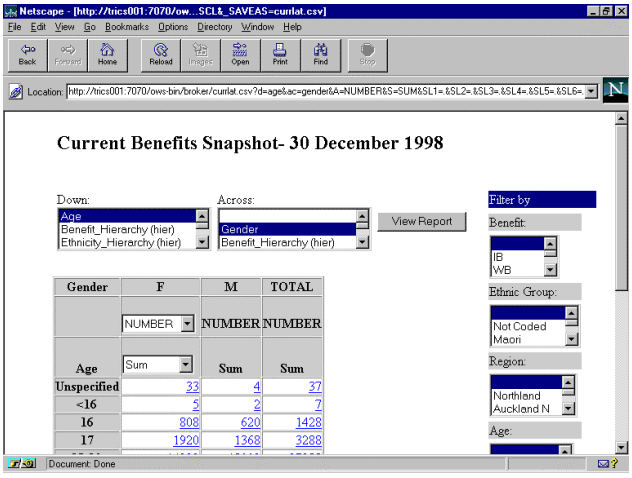
MDDB Report Viewer Details

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.

This was not simple enough for our needs. We wanted an end-user to make a menu selection from a Web page and to be presented immediately with a high-level report (for example number of customers by certain categories). Many of our users may not go any further than that.

However users can then choose further capability, for example:

- Drill-down using pre-defined hierarchies (defined in 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 with the press of one button
- Rotate the report
- Reach-through to produce a detailed report of the underlying observations that make up a cell, using chosen variables from the underlying summary SAS data set that created the MDDB.



Screen shot of MDDB Report Viewer

The main outstanding issues we have with the MDDB Report Viewer are:

- Only one title is allowed. You really need more titles and the ability for footnotes also.
- There seems to be no way to format cell values, and percentages are shown with many decimal places.
- Column headings need better defaults, and the ability to customise them. For example a selection listbox for available variables and statistics appears, even when there is only one of each.
- 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.
- The drill path shown under the title uses variable names, not variable labels.

MDDB Report Viewer 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.

As noted earlier the standard set-up results in 3 screens being presented to the user. We required just one. This was very straightforward to develop – simply creating a few lines of HTML specifying the parameters for the Viewer. However it was not so simple to find out how to do it, and what the parameters were, as this documentation did not exist. The HTML code for producing the example shown earlier 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=age&AC=gender&A=number&S=SUM&
ST=1&DT=Current+Benefits+Snapshot&
&SV=benefit&SV=ethgrp&SV=region&SV=age&
DP=1&DC=X&ACB=X&SG=2&GSC=1&GL=1&
GRT=VBAR&GW=600&GH=450&SSL=1&SW=15&SH=3&
VIEW=View+Report&
BGTYPE=image&BG=background.jpg">
Current Benefits – Age by Gender Report
</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

If you are very nice to your SAS Institute contact they will be able to procure a list of all MDDB Report Viewer parameters for you.

CONCLUSION

Making data warehouse information available via the Web can be very beneficial. It's cost-effective, and significant results can be achieved over a short period of time.

We have found the Web approach valuable for making metadata available to expert analysts, as well as a mechanism for supplying Management Information to a large number of users.

The software tools now available are relatively straightforward to learn, and results can be obtained quite rapidly. It is very tempting to 'just get on with it', and start producing web pages.

We would recommend you to take a step back, put some time into investigating best practice for Web development from the myriad of sources available, and put together a structured plan for achieving a quality result.

In particular, things that should be considered are:

- Why do it at all? What is the return for the investment of time, money and effort?
- Always bear in mind the customer – the Web page end-user: does it provide them with worthwhile information in a timely manner?

References

SUGI 23 Proceedings of the 23rd Annual *SAS® Users Group International Conference, Nashville, Tennessee* March 22-25 1998

SAS/IntrNet® Software Delivering Web Solutions, SAS Institute Inc, Cary, NC27513.

SAS/IntrNet online documentation

An Introduction to SAS/EIS Software and the SAS/MDDB Server, Course Notes, SAS Institute (UK) Ltd

MS FrontPage online help

An Introduction to SAS/IntrNet® Software, Course Notes, SAS Institute (NZ) Ltd

Web Database Primer Plus 996 Piroc Mohseni. Waite Group Press, 200 Tamal Plaza, Corte Madera, Ca 94925.

Keeping Bytes Down Mark Gibbs. Network World, April/May 1998

Meta Meta Data Data Ralph Kimball DBMS March 1998

Meta Data Architecting for Data Warehousing S. Sachdeva DM Review April 1998

The Meta Data Interchange Specification Initiative Meta Data Coalition

Managing Meta Data David Marco Data Warehouse Survival Kit

The Expanding Role of Enterprise Meta Data R. Tanler et al Data Warehouse Survival Kit

Meta Data Maturity D. Hackney Data Management Review, March 1996

http://www.info.med.yale.edu.caim/StyleManual_Top.HTML

http://applenet.apple.com/hi/web/intro.html

http://www.ameritech.com.1080/corporate/testtown/library/standard/web_guidelines/introduc3/05/98nl

http://www.man.ac.uk.MVC//SIMA/Isaacs/guide.html

http://www.cl.cam.ac.uk/users/gdrll/style-guide.html

http://www.sum.com/styleguide

http://union.ncsa.uiuc.edu:80?HyperNews/get/www/guide.html

http://www.w3.org/hypertext/www/Provider?style/Overview.html

http://www.microsoft.com/intranet

http://www.webreference.com

http://www.stars.com

http://www.webdeveloper.com

Author Contacts

Clare Somerville
Consultant
Counterpoint Consulting
6 Raroa Crescent
Kelburn
Wellington
New Zealand
Phone 64 4 25 230 7979
Fax: 64 4 475 9150
email clare.somerville@dsw.govt.nz
email clares@counterpoint.co.nz

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