Paper 046-2009

Identifying Complications and Overcoming Problems Using SAS® Enterprise BI Server

Clare Nicklin, The Open University, Milton Keynes, UK

ABSTRACT

The Open University in the United Kingdom has been involved in analysing, designing, and implementing a data warehouse system. There is nothing unusual about such a project, but the university had no exposure to the SAS®9 suite of tools and has come across many problems and issues during the project lifecycle. This paper, firstly, details the system's software requirements where a combination of SAS software (including SAS® Data Integration Studio, SAS® OLAP Cube Studio, SAS® Information Map Studio and SAS® Web Report Studio) was used. Secondly, a brief outline of the project itself will be described to put the development into context. Thirdly, a number of issues and problems are highlighted to describe how the development of the system was stalled for a number of months. Finally, the implemented solutions are highlighted including how the specified tools were used to overcome several problems. This paper is intended to be of use to both developers and managers to aid in the conception, construction, and problem management of developing OLAP based data structures with The SAS System.

KEYWORDS

SAS Data Integration Studio, SAS OLAP Cube Studio, SAS Information Map Studio and SAS Web Report Studio, OLAP, drillthrough.

INTRODUCTION

Technical problems are often experienced during the design and implementation of any computer based system. It is important to identify them quickly and analyse them correctly so that they can be managed and do not cause long delays to the continuing development. This paper describes the complications that were identified for one such project, which built a series of OLAP cubes from a data warehouse and delivered them to the user community using SAS Enterprise BI Server software. It then describes the technical solutions employed to overcome those problems.

SOFTWARE USED

- SAS Data Integration Studio was used to load all data from flat files into physical data stores, using various ETL techniques. A star schema data warehouse was built, followed by a single, large data mart. The tool naturally builds all the metadata repositories as it progresses.
- SAS OLAP Cube Studio was used to create three business specific cubes from the single data mart. Calculated measures were defined and implemented to create all the statistics necessary for the requested reports. Aggregation tuning was used to improve performance after problems were encountered in the speed of report generation.
- SAS Information Map Studio was used to provide business specific terminology for the cube contents, apply formats and switch drillthrough on and off. Two maps for each cube then fed the metadata into SAS Web Report Studio, one set of maps with drillthrough switched on for authorised users and one set with it switched off.
- SAS Web Report Studio was used to design and create all the reports to be published to the business users. Those allowed to log in directly (normally through the portal) can create their own reports and edit some others, given the correct authorisation level through the User Manager of SAS® Management Console.
- SAS Information Delivery Portal is currently used to allow the business users to access the reports from the 'Public Kiosk'. These reports therefore:
  - launch SAS® Web Report Viewer.
  - are available to everyone.
  - remain read-only.
  - access the information maps where drillthrough is disabled.

SYSTEM OVERVIEW

The data warehouse seemed to be a simple structure. The requirement of the pilot project was to detail staff numbers at the university, giving precise headcounts for any point in time and at any level of hierarchy such as department or age. Demographic and employment information was modelled into cubes with the ultimate aim of slicing and dicing that data to gain in depth knowledge of staff movement and profiles. Therefore, the main statistics to be generated consisted of a series of counts. The data was originally modelled so that one record would represent a single unit and 'n' counts would be used to generate the statistics. However, this process had

to be changed due to inconsistent values when data was drilled through and expanded/collapsed on the reporting screens. The structure of the data mart was therefore adjusted to create additional variables for counting and all the subsequent cubes, maps, and reports also had to be updated. The following explains some of the issues which pre-empted the changes to the data mart (and subsequently the cube design) and some of the knock-on effects of those changes to the ensuing reporting processes such as drillthrough.

DEVELOPMENT DIFFICULTIES/TECHNICAL PROBLEMS/DELAYS

- Semi-additive measures – counting staff numbers across a time hierarchy which did not accumulate like normal amounts was a problem. When results were represented at the lowest level (the data was held at month level in the mart), they were displayed normally. However, when the results were then collapsed, for example to display numbers by quarter or year, they were summed indiscriminately and displayed discrepancies of many tens of thousands:

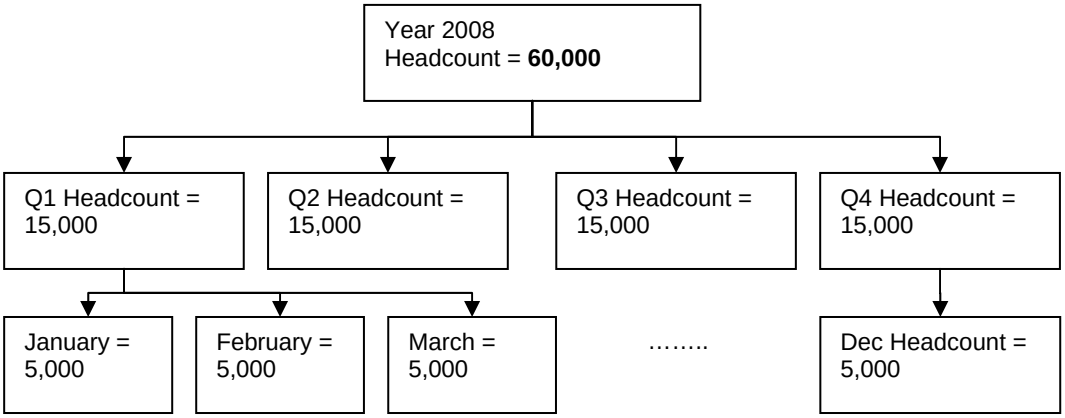
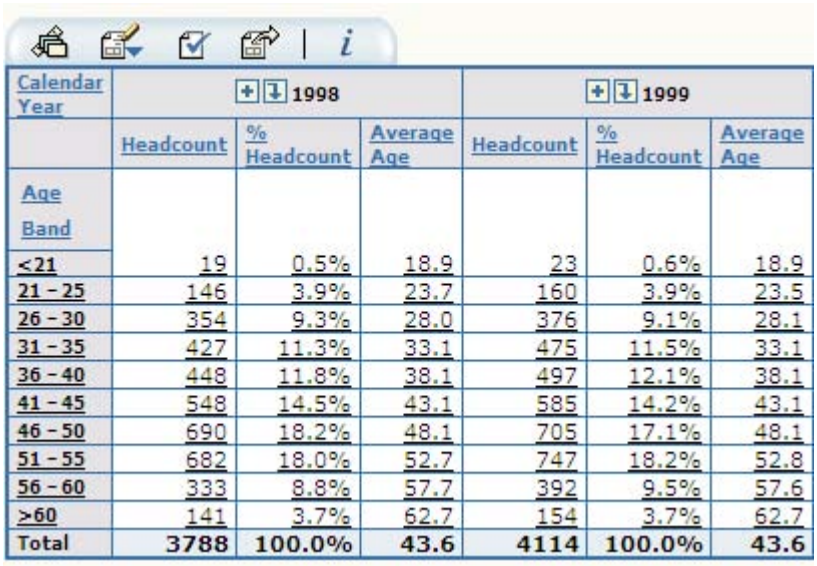


Figure 1 - Non-accumulating Headcount Values

Headcount numbers are not traditional, accumulating values over time. This feature was researched unsuccessfully for a short time. It was not until it was identified with the terminology 'semi-additive values' that solutions could be researched on the internet and the SAS support site.

- Performance – once reports were finished they took too long to load. The more calculated measures that existed and the more hierarchies that were included, the slower the screens became when they were loading. Therefore, as the design and contents of several reports progressed, the performance diminished.
- Report Aesthetics – pinpointing report structures was sometimes difficult especially as users would often expect to display reports with the same power that Base SAS exhibits, such as:
  - Total only percentages and averages – the desire to display only a single line to represent percentages or averages for the total instead of having to display an entire column. In the following figure, it can be seen that an Average Age column provides an age for every age band on every row of the table. The requirement of the report was to show a single average age on the total line without the additional, unnecessary figures throughout the report. The two illustrations below respectively show the format that was unavoidable in Web Report Studio and the kind of format that was initially required.



Calendar Year	1998			1999		
	Headcount	% Headcount	Average Age	Headcount	% Headcount	Average Age
Age Band						
<21	19	0.5%	18.9	23	0.6%	18.9
21 - 25	146	3.9%	23.7	160	3.9%	23.5
26 - 30	354	9.3%	28.0	376	9.1%	28.1
31 - 35	427	11.3%	33.1	475	11.5%	33.1
36 - 40	448	11.8%	38.1	497	12.1%	38.1
41 - 45	548	14.5%	43.1	585	14.2%	43.1
46 - 50	690	18.2%	48.1	705	17.1%	48.1
51 - 55	682	18.0%	52.7	747	18.2%	52.8
56 - 60	333	8.8%	57.7	392	9.5%	57.6
>60	141	3.7%	62.7	154	3.7%	62.7
Total	3788	100.0%	43.6	4114	100.0%	43.6

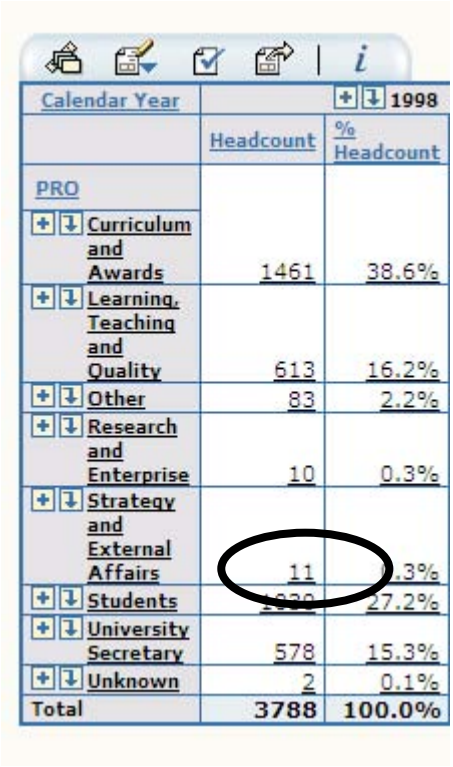
Figure 2 - Average Age in Web Report Studio

	Unknown	<21	21-25	26-30	31-35	36-40	41-45	46-50	51-55	56-60	>60	Total	Avg age	Median age
2005	0	9	73	297	463	505	619	666	756	727	266	4381	46	47
2007	0	5	95	308	521	565	664	699	744	765	386	4752	46	47
2009	0	2	115	337	517	644	677	740	757	742	496	5027	46	47

Figure 3 - Average Age in SAS

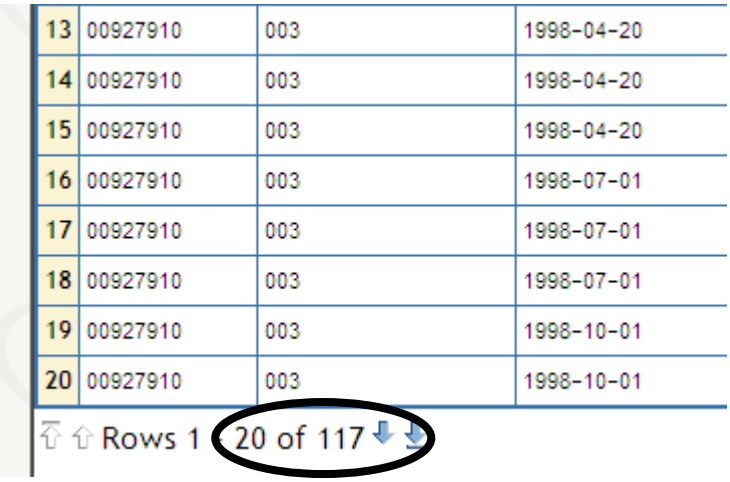
- Headcount percentages – the data is based on a simple headcount of staff and a percentage of that headcount was required for whichever dimension was selected to be displayed in the report. There is no way of dynamically finding out which hierarchy is in use in a report to dynamically create calculated measure percentages. In MDX this would have been achieved by using an Axis() function but this function is not yet supported in the current implementation of MDX in the software.

- Drillthrough – unexpected results were returned where the drillthrough functionality could not deliver what was required:
  - Data is held in a data mart at month level. One record for each staff member for every month of their employment is represented which simply flags whether that employee was employed during that month and therefore contributes to a headcount. This data structure displays multiple observations for any one staff id i.e. twelve records for every full year of employment. When drilling through from the cube, displayed in Web Report Studio, to the detail data at **month** level in a hierarchy, the number of records retrieved from the drillthrough are more or less correct as staff id numbers are distinct. However, when the drillthrough is selected at year level, all the monthly records are still retrieved and the level that the time hierarchy is expanded or collapsed to is not automatically accounted for. For example, this means that when a number of 11 staff is selected from a report at year level, 117 records were displayed instead. There was no way of ensuring the drillthrough count was distinct based on staff id. This was a side-effect of the eventual design of the cubes. When a number of some hundred was selected in a report, thousands of records were returned and attempted to be displayed instead. This caused further problems.



Calendar Year	1998	
	Headcount	% Headcount
PRO		
Curriculum and Awards	1461	38.6%
Learning, Teaching and Quality	613	16.2%
Other	83	2.2%
Research and Enterprise	10	0.3%
Strategy and External Affairs	11	.3%
Students	[illegible]	27.2%
University Secretary	578	15.3%
Unknown	2	0.1%
Total	3788	100.0%

Figure 4 - Headcount Report for Drillthrough in Web Report Studio

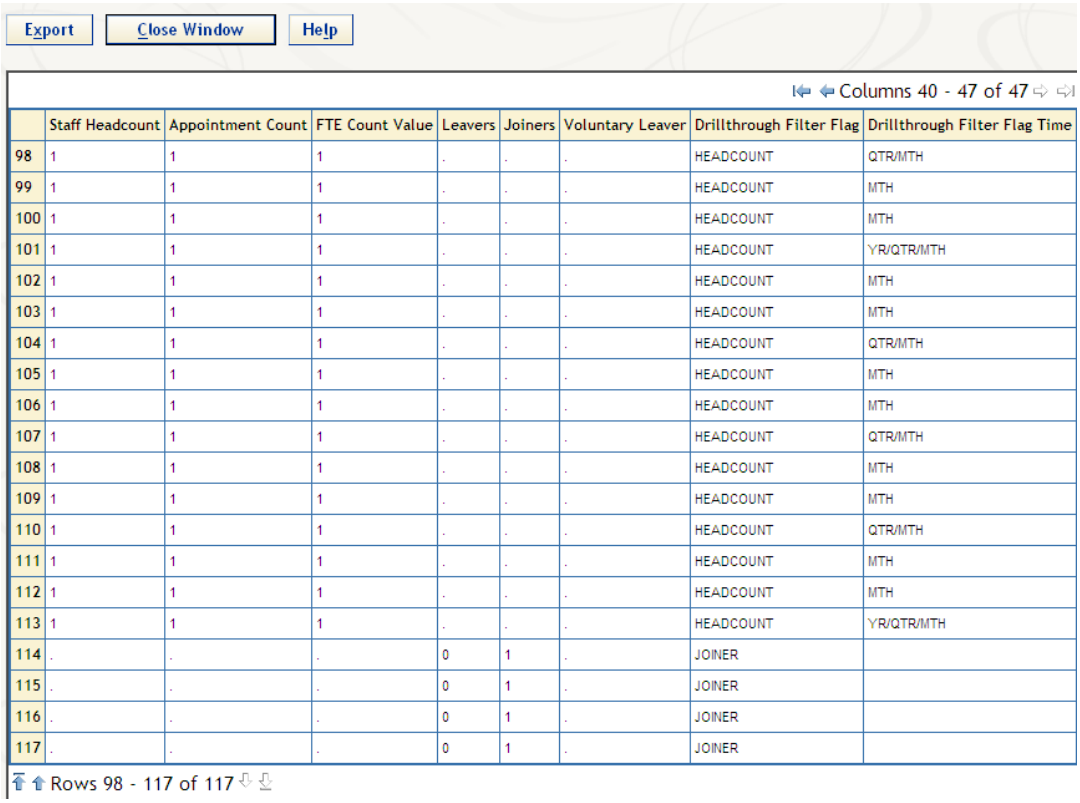


13	00927910	003	1998-04-20
14	00927910	003	1998-04-20
15	00927910	003	1998-04-20
16	00927910	003	1998-07-01
17	00927910	003	1998-07-01
18	00927910	003	1998-07-01
19	00927910	003	1998-10-01
20	00927910	003	1998-10-01

Rows 1 - 20 of 117

Figure 5 - Results of Drillthrough, 117 records

- Unexpected results would also arise as there are different types of information in the data i.e. not just headcount but also a record for when a staff member joined or left the university. These additional records were also retuned in the drillthrough results. Even though the analysis variable selected for drillthrough was ‘Headcount’, and ‘Joiners’ and ‘Leavers’ are in fact separate columns in the data and separate analysis variables in the cubes, the drillthrough still retrieved the records. The software did not account for which analysis variable had been selected. This added again to the additional records being displayed over and above the report’s number that was originally expected. In this example, four additional ‘JOINER’ records were retrieved:



Export | Close Window | Help

Columns 40 - 47 of 47

	Staff Headcount	Appointment Count	FTE Count Value	Leavers	Joiners	Voluntary Leaver	Drillthrough Filter Flag	Drillthrough Filter Flag Time
98	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
99	1	1	1	.	.	.	HEADCOUNT	MTH
100	1	1	1	.	.	.	HEADCOUNT	MTH
101	1	1	1	.	.	.	HEADCOUNT	YR/QTR/MTH
102	1	1	1	.	.	.	HEADCOUNT	MTH
103	1	1	1	.	.	.	HEADCOUNT	MTH
104	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
105	1	1	1	.	.	.	HEADCOUNT	MTH
106	1	1	1	.	.	.	HEADCOUNT	MTH
107	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
108	1	1	1	.	.	.	HEADCOUNT	MTH
109	1	1	1	.	.	.	HEADCOUNT	MTH
110	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
111	1	1	1	.	.	.	HEADCOUNT	MTH
112	1	1	1	.	.	.	HEADCOUNT	MTH
113	1	1	1	.	.	.	HEADCOUNT	YR/QTR/MTH
114	.	.	.	0	1	.	JOINER	
115	.	.	.	0	1	.	JOINER	
116	.	.	.	0	1	.	JOINER	
117	.	.	.	0	1	.	JOINER	

Rows 98 - 117 of 117

Figure 6 - Results of Drillthrough, Retrieval of Additional Analysis Variables

- When many thousands of records were inadvertently returned, the application would sometimes fall over and emit Java errors:

```
Cause:
An Exception has been thrown while attempting to get Show Detail results.
at com.sas.commands.olap.ShowDetailDataCommand.
getShowDetailDataModel(ShowDetailDataCommand.java:266)
```

  The ensuing Java errors seemed to be due to the extra number of records being picked up by the drillthrough. The environment was stable until approximately 14,000 records (and 50 variables) were returned.
- Another requirement for drillthrough, which was requested to prevent sensitive data being viewed by unauthorised users, was some kind of selective drillthrough. This would involve the ability to either turn drillthrough off for specific columns which would access sensitive data or remove the sensitive data from drillthrough results, based on specific users and their permissions. This was identified as not being possible.

SOLUTIONS

- Semi-additive measures – to solve this issue with the headcount figures, use was made of the Complex Calculation area of the Calculated Measures OLAP feature. We had to learn some MDX to use a tail function to implement semi-additive measures techniques detailed by a SAS consultant. The tail function returns a subset from the end of a set, so if a timeline is determined, the last set of the time period below the current one can be returned.

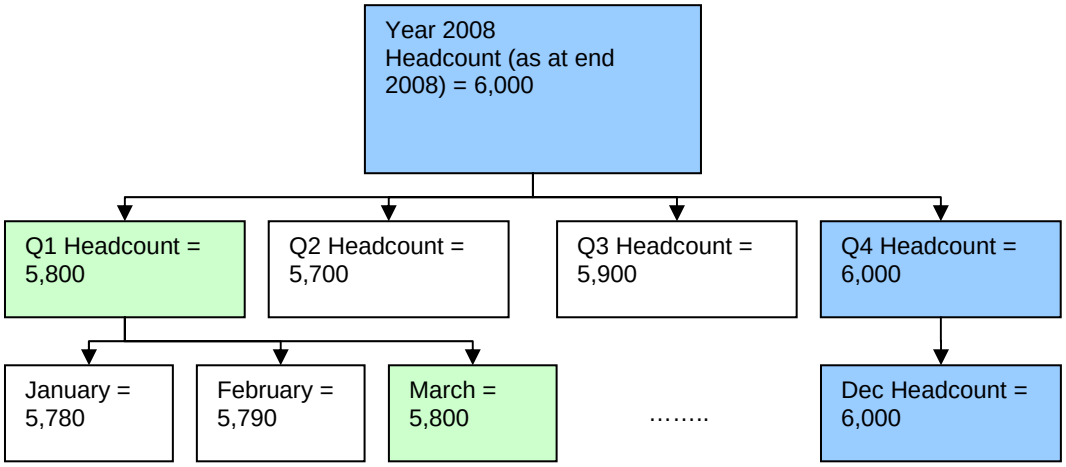


Figure 7 - Representation of Headcount Data Over Time

Therefore, the first action is to find all the descendants of the current time member. Then the second action is to use the tail function to get the last member of those descendants. Using the illustration above, this means that to get the headcount for the year 2008, query the descendants (in this case quarters) and get the number for the last quarter of the year. In turn, to get the headcount for Q4 of 2008, retrieve the three months making up the descendants for Q4 and take the number for last member i.e. December. The function is as follows:

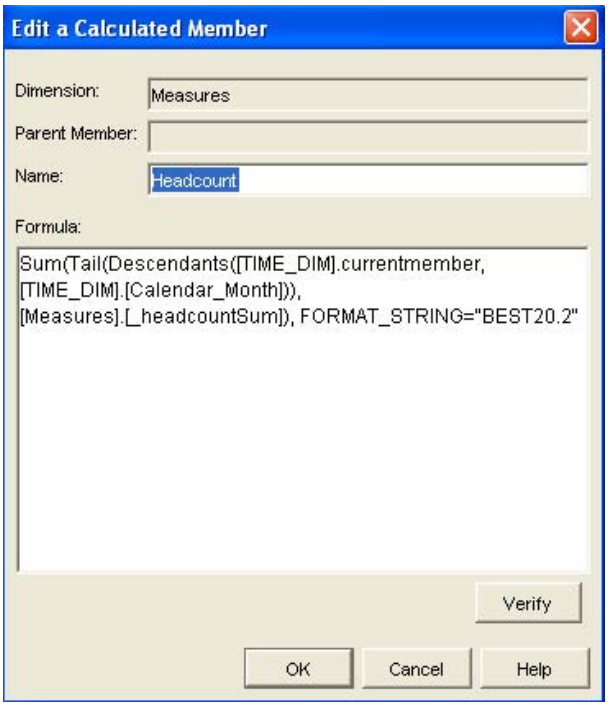


Figure 8 - MDX Tail Function in OLAP Cube Studio

Subsequently, further calculated measures could be defined (such as percentages) using this calculation as a basis and using a correct 'solve order' to ensure the headcount is processed first. These issues and techniques were found to be well documented once the terminology of semi-additive measures had been brought to our attention.

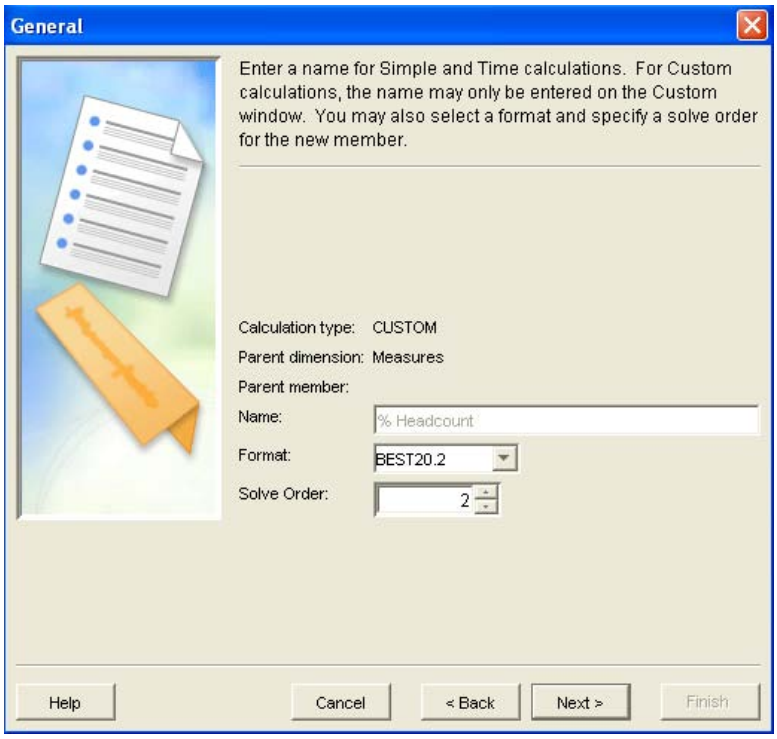


Figure 9 - Defining a Calculated Member's Solve Order in OLAP Cube Studio

- Performance – this concern was addressed by performing a number of tasks:
  - Firstly, it is believed that calculated measures perform their dynamic calculations on all hierarchies included in a report screen, whether they are actually active in a table/graph's definition or not. If hierarchies are selected in the 'Data selected from:' area of Web Report Studio, even though they are set to hidden (and not included in any current graphic or tabular definition), they are still available for selection in the setup properties of a graph/report. As such, it seems that the calculated measures for those hierarchies are still being calculated dynamically as the screen is displayed, as the appearance of the screen takes much longer than if the hierarchies are removed completely. If the hierarchies are not selected at all, they are not available for selection in the definition of a graph/table, but the screen seems to be displayed in a much shorter time.

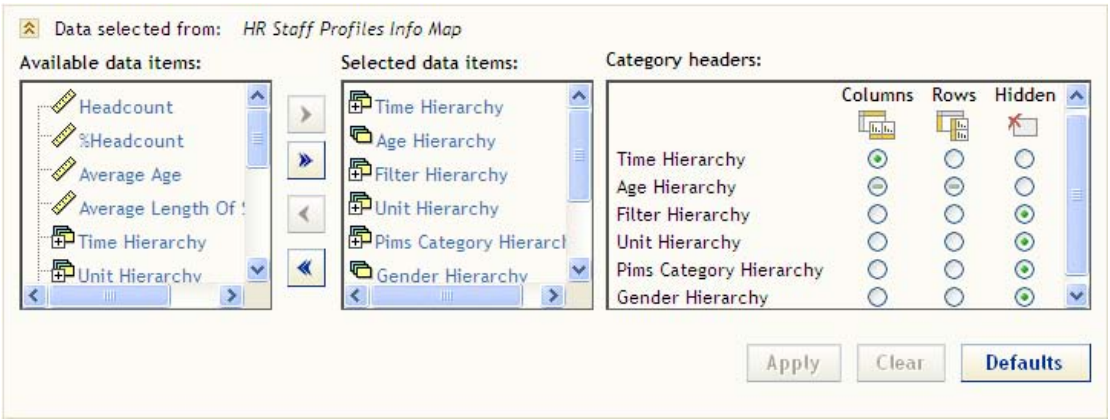


Figure 10 - Selected Hierarchies in Web Report Studio

For this reason, any additional hierarchies not immediately involved in the default displays of the reports were removed from the 'Data selected from' area of the reporting tool. Additional help was defined for users so that they could reintroduce the hierarchies at any time. This seemed to decrease the time taken to display report results.

  - Secondly, the Performance Aggregation Tuning tool of OLAP Cube Studio was used to improve the efficiency of the cube for retrieving results. When a cube is created, aggregations can be specified with the aim of improving query performance. Understanding which queries are run against the cube the most frequently allow the tuning of those specific aggregations and therefore improve query response times. The cubes were tuned using the manual tuning function, where the most common queries were pre-empted and applied. Eventually, when the systems are established in their live environment, the

advanced tuning techniques will be applied using ARM analysis. This creates a log of activities, added to every time a cube is queried. The ARM log is then used to deduce which aggregations will have the most positive impact on query performance. The above two methods combined halved the report display speed.

- Thirdly, the application of various hot fixes downloaded from the SAS support site proved invaluable. The hot fixes had not, at that point, been identified as vital. However, during testing, the university upgraded to the latest version of our internet browser software. The SAS applications ceased to work correctly and it was realised that a series of hot fixes would need to be applied to all our browser based software (specifically Web Report Studio/Viewer, Information Delivery Portal, and Web OLAP Viewer). As well as fixing all the browser related problems, the application of these fixes speeded up the display of the report results to such a degree that all performance issues were immediately resolved.

- Report Aesthetics – some report design issues could not be resolved such as wanting percentages or averages only on a total line. However, a work around was implemented to avoid creating a separate, calculated percentage measure for each potential dimension of a cube (about eight measures). The requirement was to create a percentage headcount measure which would calculate the headcount as a percentage of the total for the dimension or hierarchy in use within a report. As several hierarchies were available in many reports, which could be added, removed and substituted at any time, any percentage would have to be dynamic somehow i.e. it would need to know in the MDX code which dimension was being used in order to find the parent value. Without the ability to create a generic MDX query using a function such as Axis() to dynamically retrieve which dimension is being used in a report (therefore deducing the current member's parent value), SAS recommended using a series of nested if statements:

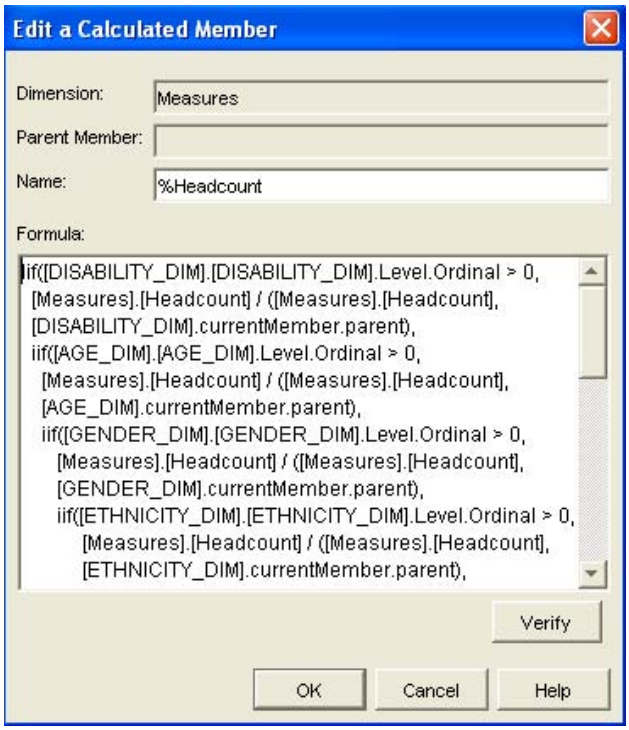


Figure 11 - MDX Code to Create Complex Calculations in OLAP Cube Studio

Every dimension listed as available in the report is checked to see if it is currently being included in a table/graph and the percentage is then calculated dynamically.

- Drillthrough – the problem of unexpected drillthrough results seemed to be a combination of the data structure/design of the cube and the functioning of the software.
  - When drillthrough was selected for a year, the affected staff ids were found for the selected slice of the cube and then every record for those affected staff ids was returned i.e. all monthly records and all other analysis variables (not just headcount, also records modelling other types of counts). There was no neat way of creating a distinct count and restricting the drillthrough results to only reflect the expected number of records. However, it was realised that if a report was carefully filtered to the correct time period and the correct analysis variable before the drillthrough was used, then the correct number of records could still be returned. Additional functionality was therefore created to enable this filtering. Two extra variables were created in the data mart - one to filter the time line (Drillthrough Filter Flag Time) and one to filter the analysis variable (Drillthrough Filter Flag):

Export | Close Window | Help

Columns 40 - 47 of 47

	Staff Headcount	Appointment Count	FTE Count Value	Leavers	Joiners	Voluntary Leaver	Drillthrough Filter Flag	Drillthrough Filter Flag Time
98	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
99	1	1	1	.	.	.	HEADCOUNT	MTH
100	1	1	1	.	.	.	HEADCOUNT	MTH
101	1	1	1	.	.	.	HEADCOUNT	YR/QTR/MTH
102	1	1	1	.	.	.	HEADCOUNT	MTH
103	1	1	1	.	.	.	HEADCOUNT	MTH
104	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
105	1	1	1	.	.	.	HEADCOUNT	MTH
106	1	1	1	.	.	.	HEADCOUNT	MTH
107	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
108	1	1	1	.	.	.	HEADCOUNT	MTH
109	1	1	1	.	.	.	HEADCOUNT	MTH
110	1	1	1	.	.	.	HEADCOUNT	QTR/MTH
111	1	1	1	.	.	.	HEADCOUNT	MTH
112	1	1	1	.	.	.	HEADCOUNT	MTH
113	1	1	1	.	.	.	HEADCOUNT	YR/QTR/MTH
114	.	.	.	0	1	.	JOINER	
115	.	.	.	0	1	.	JOINER	
116	.	.	.	0	1	.	JOINER	
117	.	.	.	0	1	.	JOINER	

Rows 98 - 117 of 117

Figure 12 - Additional Drillthrough Variables

These variables then made up the two levels of an extra hierarchy, defined in all the cubes specifically to feed into the reporting software and be used as a 'filter hierarchy'. It was then available for use by the standard filter option of the Web Report Studio software seen below.

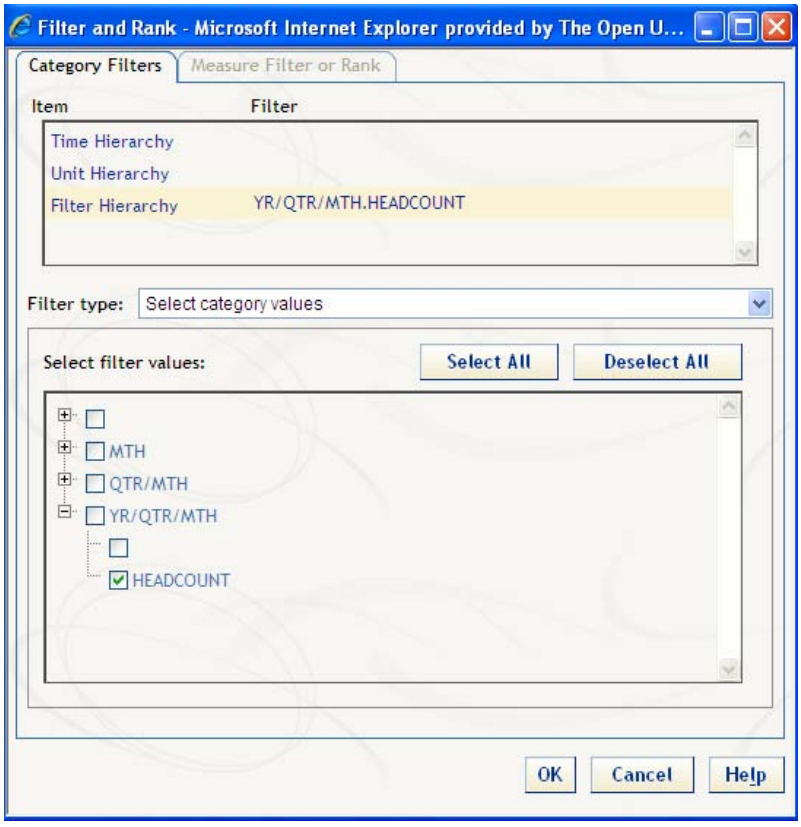
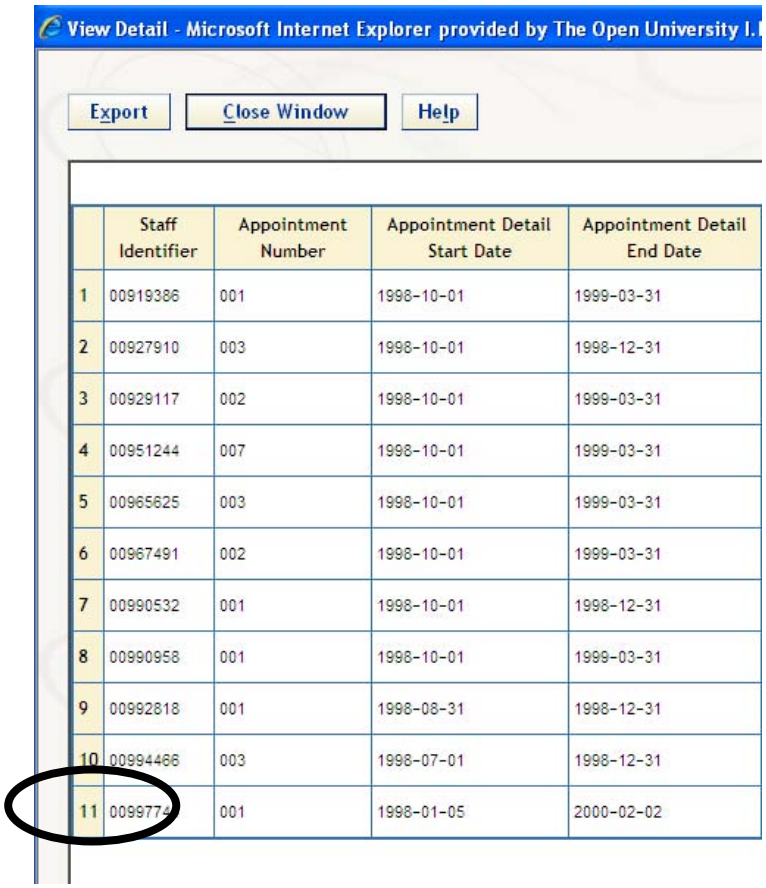


Figure 13 - Filter Functionality of Web Report Studio

The downside to this feature is that it cannot be automated. It can only be included in the training of the users. They must be educated about the issue and advised that they must apply the filter in the correct way for their specific drillthrough requirement to function. The illustration above shows how the filter is applied before drilling through to get headcount detail for a selected year. To drill through for a quarter, both YR/QTR/MTH and QTR/MTH boxes would be checked for headcount and so on. The illustration below shows the results of the drillthrough demonstrated in the previous section i.e. the 11 records that were clicked on in a headcount report. Once the filter was applied the drillthrough results were correct.



	Staff Identifier	Appointment Number	Appointment Detail Start Date	Appointment Detail End Date
1	00919386	001	1998-10-01	1999-03-31
2	00927910	003	1998-10-01	1998-12-31
3	00929117	002	1998-10-01	1999-03-31
4	00951244	007	1998-10-01	1999-03-31
5	00965625	003	1998-10-01	1999-03-31
6	00967491	002	1998-10-01	1999-03-31
7	00990532	001	1998-10-01	1998-12-31
8	00990958	001	1998-10-01	1999-03-31
9	00992818	001	1998-08-31	1998-12-31
10	00994486	003	1998-07-01	1998-12-31
11	0099774	001	1998-01-05	2000-02-02

Figure 14 - Correct Drillthrough Results

- Once it was identified that the number of records being returned in the drillthrough functionality affected the application and sometimes caused a Java error, the number could be limited to ensure stability. During the definition of the cubes in OLAP Cube Studio, when defining the data set to be used for drillthrough, an ‘obs=14000’ data set option was included in the table options area.

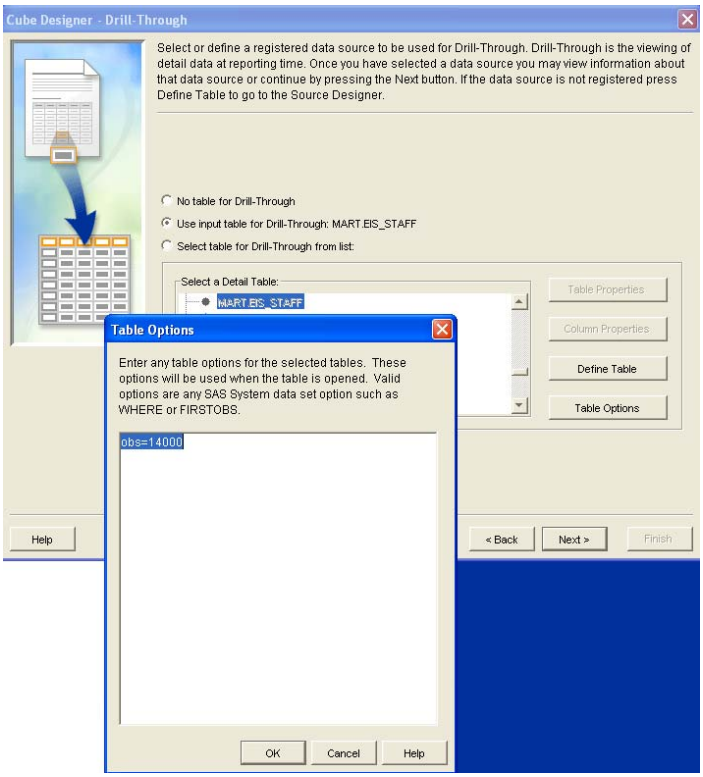


Figure 15 - Restricting Drillthrough Results in OLAP Cube Studio

This solution assumed that there would be enough records to provide answers to the reason for the drillthrough. It guaranteed that the application did not display errors and in fact, after the aforementioned hot fix was applied the error did not return. The observation limit has been left in place even though the problem should not reoccur as long as the filtering functionality is used.

- Selective drillthrough does not currently seem possible. However, to provide some level of drillthrough control, two sets of information maps were defined – one with drillthrough turned on, and one set with it turned off. This is done through selecting the Properties of the required information map in Information Map Studio and checking the box illustrated below:

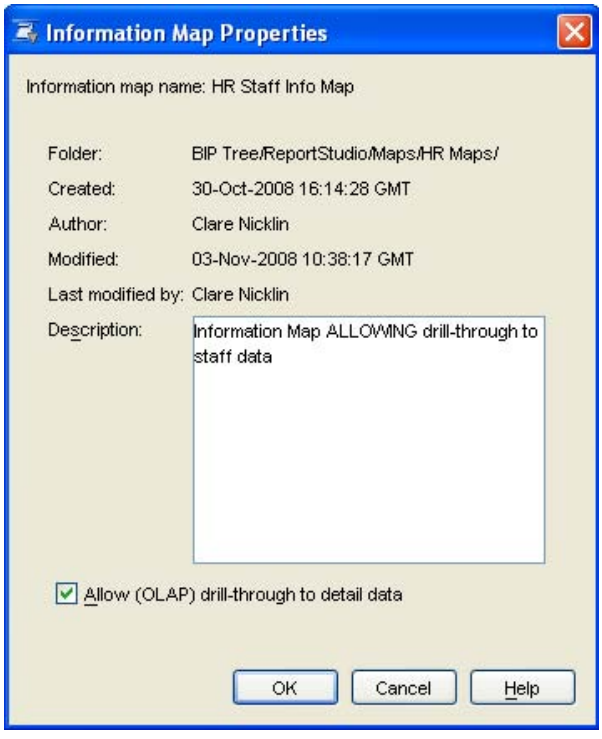


Figure 16 - Information Map Properties in Information Map Studio

The Information Delivery Portal's Public Kiosk is the way in which the reports are mostly accessed. This access route is open to all regardless of permissions and therefore the reports are linked to the non-drillthrough information maps. Any users who are allowed to access sensitive data have their own logons. As such they can log on to the Information Delivery Portal, or to Web Report Studio directly, which allows them to access the reports based on the information maps with drillthrough switched on.

CONCLUSION

Many of the issues identified during the development process have been overcome but there are still some outstanding. It is hoped that some issues will be addressed, and some improved upon, with new releases of the software. For example, selective drillthrough will become a possibility in SAS 9.2 where permissions allocated depending on user profiles will include drillthrough authorisation.
This data warehouse system was intended to prove the benefits of the software tools and the ability to apply the necessary business concepts before progressing with a plan to build a larger, more complicated data warehouse. Despite the complications of the original project and the time taken to solve the resulting problems, this larger warehouse is now in development. It is modelling data based on the university's student population and the intention is to use all the lessons learned in the previous implementation to foresee design issues, make better use of the software tools and encourage faster development.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Clare Nicklin
The Open University
Walton Hall
Milton Keynes
MK7 6BJ
United Kingdom
+44 (0) 1908 274066
c.a.nicklin@open.ac.uk