Paper 113

A Health Care Claims Data Mart: Construction and Exploitation

Marge Scerbo, CHPDM, University of Maryland Baltimore County
Stuart B. Levine, SAS Institute Inc.

Abstract

Health care claims data are complicated animals. The eccentricities of this data include differences in the types of claims files and the services which populate those files, variations in the type of providers who service the population and the information needed about each type of provider, and finally, the actual populations of health care recipients and all of their individual specifications.

Being able to sort through all of this data (and there is usually a lot of it!) and answers queries from insurers, providers, analysts, etc., and other people within your organization can be a real challenge. In the past, the solution was to have people submit requests for one or more programmers to extract data and run analyses. The problem was twofold: programmers were overwhelmed with requests, and worse, the data and/or analyses requested were not the correct solution to the problem.

This paper will highlight how this challenge was met through the development of a Data Mart. This includes the steps taken in the design of the Mart, its base tables and MDDBs (MultiDimensional DataBase), the definition of SAS/EIS® objects that can exploit MDDBs, and other specific customizations that were developed to allow one organization, the UMBC Center for Health Program Development and Management (CHPDM), to address these data access issues.

Introduction

Like any health care organization, CHPDM has a number of people looking for different kinds of information at any given time. And, like other organizations, the usual method of getting that information was for a user to submit a request to a programmer to extract that data and format into a readable report. In the spring of 1997, CHPDM decided to simplify and automate this process by developing a Data Mart to contain this data and give these users access to this data. This would allow analysts to find their own answers without having to wait on the backlog of programming requests.

A team of people was put together with staff from UMBC, SAS Institute, and Price Waterhouse. Each of these groups came to the project with different strengths. The team worked well together from early in the project, quickly defining the roles of each group and each team member. This partnership between the three entities involved helped (and was essential) in making the project a success. And, the result was a Data Mart application that has been well-received by the users, both in the data they now can access that they could never see before and in the functionality that allows them to see the data in ways they never thought they could see.

Conceptual Design

The team of Price Waterhouse, SAS Institute and CHPDM managers and analysts assembled for several weeks to begin the design process. As with any data mart or data warehouse project, the most time consuming process is that of design. The first step in the process was for the team to get acquainted. Each member of the group had particular strengths and weaknesses. Each would add something slightly different to the thought process. And each would learn something from the other members. Since the design process covered a large range of topics and several months in time, leadership of each meeting rotated among the members of the team, depending on the topic and whose specialty this fell within. It would have been an interesting process to observe!

The first several meetings allowed the group to set priorities, review timetables and arrange schedules, lay out the steps in the process, and define the general outcomes desired. Of course, as the project moved forward and was better understood, each of these specifics changed. Timetables were amended, priorities reinterpreted, steps shuffled, and outcomes clarified. Throughout this initial process and every subsequent one, it became imperative to maintain management support and direction.

Certain issues were assessed as ‘givens’. The data for the data mart would be stored on a Sun Solaris system. Although all the data were stored in SAS data set format, the format of the Data Mart was yet to be determined. Confidentiality and security were imperative; Maryland Medicaid data contain very confidential material. The data mart would encompass one year of claims and eligibility data; this would be for fiscal year 1995. Each CHPDM staff member had an individual PC attached to a Novell network. No new hardware purchases were planned.

One of the first steps was for the non-CHPDM team members to become acquainted with the file that was to be the basis of the Data Mart. CHPDM documentation was studied. An intense database review was undertaken which included studies of the data validation process and business rule definition. The Maryland Medicaid claims and eligibility files had been verified, validated, cleansed, and sorted, and all data were stored in SAS data sets. Some business rules, which should include clear definitions of field creation and usage approved by manager or director, already existed. The team was able to make use of these previously accomplished steps.

In order to ‘involve’ the entire department, interviews were arranged with the different entities in the department. The department was made up of several different analysis groups, each with their own set of skills and needs. Each entity had separate issues that they felt were imperative to their future studies and outcomes. Also, each group had a

different grasp of what data were available, how to manipulate this data, and what information could be extracted from this data. The team developed a list of questions to pose to each group; using a predefined set of questions made it easier to compile the information collected. Different members of the team attended the various interviews, although 1 or 2 members did attend all the interviews to add an overall view.

After the interviews were compiled, the team reviewed the responses, categorized answers, established priorities, and prepared a formal document. This contained those requests that could be met with the present data, analysts, hardware and time limitations. Defining and refining requirements vs. enhancements, needs vs. wants, and possible vs. impossible was a tough task, but a necessary one.

Once the final interview document was created, the programmers and analysts on the team created a prototype of the proposed system. The team from SAS Institute, Price Waterhouse and CHPDM discussed storage options and the mechanics of such. It was a general agreement that the final information to be displayed to the users was best stored as MDDBs. An MDDB is a MultiDimensional DataBase and was, at the time, a new storage mechanism offered by SAS Institute. A lengthy discussion of MDDBs and the SAS/EIS front end will be discussed later in the paper.

A JAD (Joint Application Design) session was held. This meeting called together all those who had taken part in the interviews. This full day session provided the opportunity to present to the group the outcome document from the interviews as well as the prototype of the system. At this point, users were able to see the different methods of displaying the data that were available. The users were shown maps, graphs and the multidimensional table. This table, a 'MultiDim', received the most positive responses and therefore was given the highest priority in creation. In this group setting, the team and the CHPDM staff were able to discuss the outcome and make changes as necessary. Also covered at the meeting were those requests that could not be fulfilled in this phase of the Data Mart and the reasons for these omissions. It proved to be a positive experience for the entire group.

Price Waterhouse took the lead in developing Critical Success Factors for the project. This involved several days of discussion, interpretation, debate, etc. with the entire team. These CSFs allowed the team to focus on the conditions that will deem the project a success from a business standpoint rather than a technical one. Examples would be:

- 'Reduce need for specialized computer programs to run analysis reports.'
- 'Reduce time from days to immediate access to data.'
- 'Allow ad-hoc reporting capabilities for general use.'

Database Design

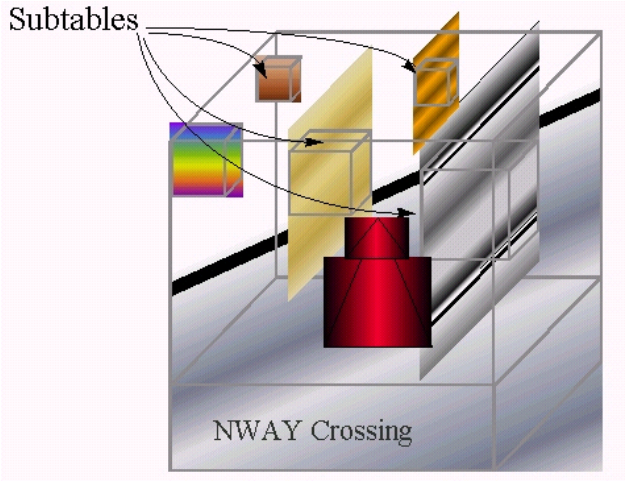
Once the conceptual design issues were at least in part finalized, the team was able to move on to the actual design of the data structures. This meant designing the MDDBs and their underlying base tables. To understand this process, it was important for each member of the team to have a grasp of what a multidimensional table is and how SAS® software creates this structure. SAS provided support from both the consulting organization and the development team.

Underlying each MDDB is a base or detail table, a SAS data set or data view. The design of these tables proved to be incredibly time-consuming. Although the claims data which was made available to the Data Mart had been cleansed, sorted, verified, and validated, no one claims file could stand on its own as a detail table. Medicaid data, as with most health care data, is often analyzed by recipient. Studies are run analyzing a group of recipients by race, gender, location, age, etc. In order to provide this ability, each record would need to contain information about the particular recipient receiving the service. After the team studied the various aspects of recipient values, up to 16 variables were 'attached' to each record in a detail table.

It was ascertained that many of the claims files did not stand on their own as study elements. Many of the files had to be merged, summarized, sorted and rebuilt to satisfy analysis needs. For example, diagnostic studies lead the list of priorities. Diagnosis codes, ICD-9 codes, existed in the Inpatient, Outpatient, and Physician files. Therefore, these files had to be merged, counted and summarized by diagnosis code, and then the recipient level information added to each record. The final product were two SAS data sets, each with over 5 million records each. One data set held information on diagnostic counts per recipient and another held monetary amounts for each primary diagnosis. The process seems straightforward now but took many hours and days of design discussions (and disagreements!).

In the end, the Data Mart contained 11 detail tables with 11 MDDBs sitting on top of them. These 11 detail tables would provide the basis for studies by diagnosis codes, physician procedures, surgical procedures, home health studies, pharmaceutical therapeutic classes, and a large number of recipient based usage and costing across a year. As each detail table was created, the team moved forward to design MDDBs.

An MDDB creates storage for data in a summarized format that provides fast and easy access. The use of a multidimensional database gives the user multiple lines of access, 'dimensions', to the data and summarization by each of these dimensions. An MDDB can be viewed as a cube with multiple dimensions: a series of cubes creating one large cube. It would allow the users to interpret the data from any one of these dimensions or cubes. This design methodology stores aggregates at each dimensional crossing, called a cell. A 'cell' is a unique combination of each dimension's level (across and down) and will contain the summary value for that crossing. Below is an image depicting an MDDB:



The MDDBs' ability to examine large amounts of data with great speed was dependent on the structure of the MDDB. The speed is due in part to the technical design of the MDDB. The SAS/MDDB® Server product contains fast indexes to each subtable as well as to the detail level data. And speed is also accomplished by presenting users with summary level data, rather than all the records that created the summary. Rarely do end users need to look at all data for the entire population; usually views of summarized data which can be subset and/or drilled down are much more useful.

So MDDBs provided inherent speed by their internal structure. But, speed of access is also dependent on the PROC MDDB code used in the MDDB creation. An MDDB is built similar to a summary table in base SAS. The SAS programmer will name CLASS variables which serve as categorical fields and analysis (VAR) variables which are the fields to be used in calculations. CLASS variables usually have a discrete and definable number of values. These fields allow for identification of specific categories and provide a means to subset the data. Analysis variables are numeric fields that can be summarized, averaged, etc. When SAS executes PROC MDDB, an NWAY table is created. The NWAY table is a table that crosses all class variables. In the CHPDM Data Mart, there were between 15 and 20 class variables identified for each MDDB and between 1 and 20 analysis variables. Crossing that many class variables with one another created very large NWAY tables.

As the team learned by trial and error, the speed of access would also be greatly affected by the number of hierarchies that were defined. A hierarchy is a subtable that allows for quicker and easier access to the summarized data stored at that particular level. Hierarchies can also be called subtables or dimensions. Certain hierarchies can be drilldown; these hierarchies provide a method to look deeper into the information. For example, a drilldown path could be from county to zip code. The team therefore included in the design these drilldown hierarchies as well as non-display hierarchies. These subtables which are not 'drillable' provide a quick mechanism to access the data. If it were not for these subtables, each query into the data would require access of the NWAY table, a very large table! Deciding which subtables might be accessed by the users was actually a best-guess estimate of what the team felt might be useful. In order to provide a method to study the subtable use, a tracking system was designed within the front-end application and will be discussed later in the paper.

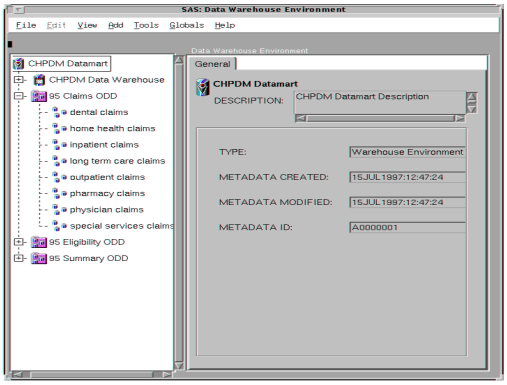
The actual database design process was a very iterative experience for the team. Often after a long design session, the team would let the dust clear (and their minds), and meet later to review what had been proposed and what would work and not work. This step was intense, difficult, stressful, but also gave the team a great feeling of accomplishment to complete.

Construction

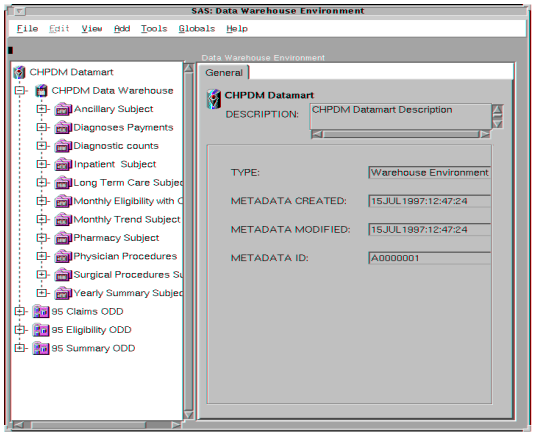
At some point in the process, the team decided to make use of another new product from SAS Institute, SAS/Warehouse Administrator™. This product would provide a means of organizing all the data that would underlie the data mart. The Warehouse Administrator would allow the team to create metadata, the basis for any data mart or data warehouse. Since it was such a new product, it was a learning experience for the entire team. SAS Institute provided a specialist for a full week to teach members of the team how to use the software.

After defining libraries where the data sets could be found and the MDDBs and detail tables would be stored, the next step in the Warehouse Administrator project was to define all the ODDs in the system. An ODD is an Operational Data Definition. This could be a SAS data set, data view or database table accessed by SAS. The Warehouse Administrator provides a simple mechanism to point to one of these files to 'register' the ODD. The definition of the ODD becomes a portion of the metadata underlying a Data Mart or data warehouse.

The team decided to define three groups of ODDs: Claims, Eligibility and Summary. Each of these groups contained between 1 and 10 separate data sets that presently existed in the system. Several new data sets were defined which would underlie the entire system. Most important was a recipient file that would contain all the identifying fields for each individual. At the same time, the true Medicaid identification numbers were replaced with specially encoded ids, this to add an important level of security and confidentiality. Below is an example of an ODD from the Data Mart:



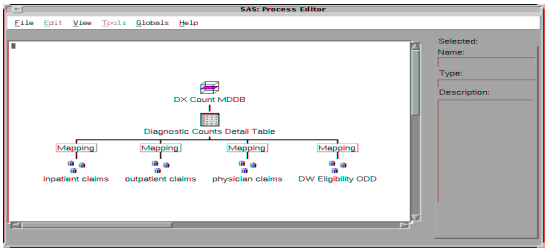
After the team defined the ODDs, the metadata could now be used to build the detail tables and MDDBs. Each detail table/MDDB became a separate subject area. By creating a subject area, the detail table can be defined by pointing to the various ODDs that provide all or portions of the data. Additionally, the manipulation and management of this data can be added as separate processes, either Mapping or Loading. This allows the program or programs that are used in the creation of the detail table to be stored in a specified source catalog. Each entry in the source catalog can be accessed through the load step. The Warehouse Administration therefore served as its own documenter. A source code entry could be clearly mapped to a particular subject area/detail table. Below is a display of the 11 subject areas developed by the Data Mart team.



The next step in the process was to create the MDDBs. Since the detail tables were now created, this was not too difficult. Or–should not have been too difficult. At first, the team used the Warehouse Administrator code generator for the MDDB procedure. Following the process, a Summary Group was defined. From that definition, the MDDB was generated. Unfortunately, the Warehouse Administrator code generator attempted to cross every possible class field and combination thereof, with every other class field and combination thereof. The first attempt with an MDDB containing 16 class fields generated 500,000 lines of code. The compilation of this code was cancelled after several hours. Additional code generators were tested, but the results were similar. In the end, one team member simply typed in manually the PROC MDDB statement, which included about 100 hierarchy statements. This in fact turned out to be the most efficient method.

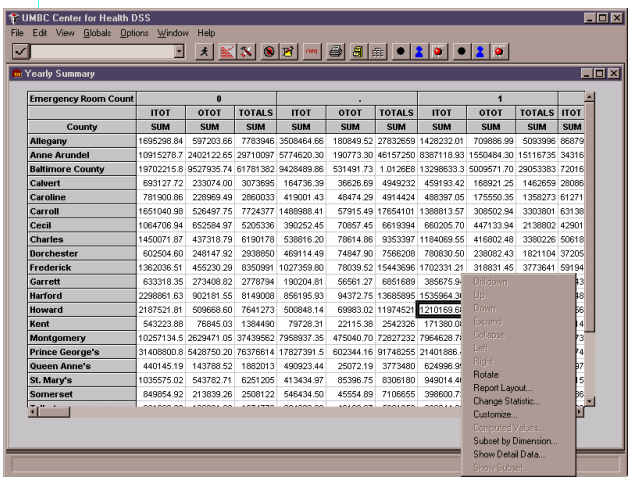
When executed, PROC MDDB code first creates a large NWAY table. This caused problems in memory usage until SAS Update 045, along with a new Solaris version, was placed on the Unix machine. The Warehouse Administrator allowed the team to make use of the Unix scheduling system; it was clear that MDDBs should be created at off-peak hours. The creation of a large MDDB could slow the system to a crawl.

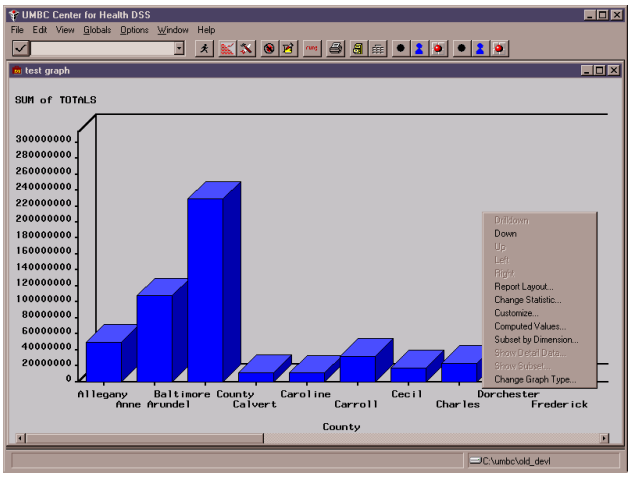
The Warehouse Administrator did provide a clear graphical outline of what files/ODDs created the detail file that in turn created the MDDB. These displays made it visually clear as to what metadata formed the backbone of a subject area. Below is a copy of one of these displays which shows which files created the Diagnostic Counts MDDB.

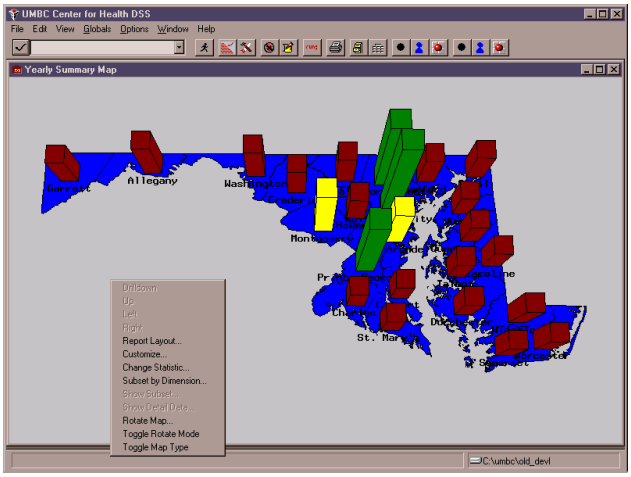


Exploiting the MDDBs

Giving users access to the data in the Data Mart presented a different set of challenges. The first step in the process was to work with the users to select which EIS objects best suited their needs. In Version 6.12, there are 5 objects that can be used to display information stored in an MDDB. Three of these are being used within this application, the Multi-Dimensional Report (MDR), the 3D Business Graph, and the Map. Below are examples of each of these reporting objects:







Before defining these EIS reporting objects, the MDDBs had to be registered. This process identifies the MDDBs to EIS and makes them available for the objects. Registering data structures within EIS is a task that must be performed whether using MDDBs or standard SAS data sets. The difference is that registering MDDBs is a much simpler task. Since the MDDB may already contain the drill hierarchies, therefore they do not have to be defined within the registration. If one were to register a data set in EIS for use in a Multi-Dim Report where drilldown was needed, each drill hierarchy would have to be created individually. But with an MDDB, registration is a few simple mouse clicks.

Besides these 'usual' tasks of defining EIS objects to display the MDDBs to the users, a method had to be developed to provide access to the EIS report objects, some unique customizations to those reports, and additional functionality.

Users had to be able to choose from among the eleven MDDBs that are currently part of the Data Mart. Rather than create a new window for each selection, a menu was developed from generic classes and windows. This menu was to be a data-driven system, so addition of new reports to the system would be as easy as defining the new report. Adding access to that report would be accomplished by adding a new observation to the data set used to populate the menu. The image below is the primary menu screen:

To view any of the reporting objects, the user selects one of the icons on the menu screen, scrolling to find their selection if necessary. The initial selection changes the WHERE clause on the menu data set, then re-displays the menu showing a sub-menu, allowing the user to choose between the Multi-Dim Report, Graph, or Map:

The next challenge was to allow users to subset the data in the MDDBs based on specific criteria. Some objects by default included a subset mechanism. There are methods in the EMDDB_M class such as _ADD_SUBSET_ and _UPDATE_ALL_SUBSETS_ that allow subset criteria to be passed to an MDDB prior to displaying a report. However, these methods assume that the subsetting is being done on class variables. The requirement was to provide the ability to subset the MDDBs on non-dimension variables (variables that are not on the CLASS statement of the MDDB procedure).

Additionally, much of the data were stored in codes. For example diagnoses are stored as ICD-9 diagnosis codes. Users needed the ability to search within in the codes and the descriptions of these codes.

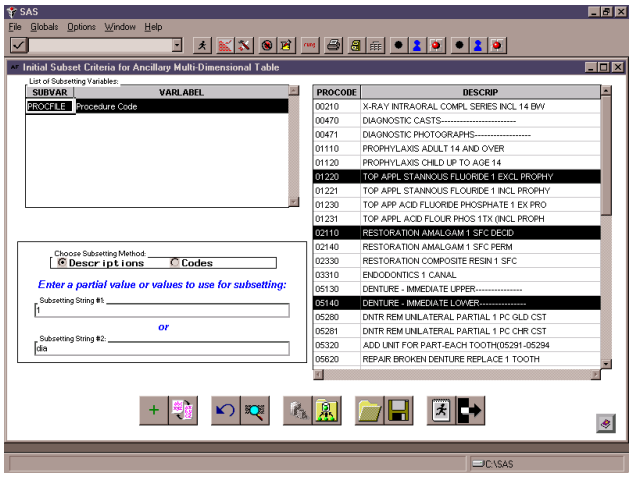
Before discussing how giving users subsetting capabilities was done, some information on the class mentioned above. The EMDDB_M class is a non-visual class giving

the user the ability to navigate through the defined data hierarchies of an MDDB, drill down, reach-through, subsetting, and report modifications "on the fly". This class, which is a model class, is used by the viewer classes (Multi-Dimensional Report, 3-D Business Graph, etc.).

Instructions can be sent to any of these classes by calling methods that are defined for the class. Examples are the two methods mentioned above, which affect the model class (EMDDB_M) when they are called. The viewer classes also have methods that can affect the display of the class or report,

To meet this technical challenge, a custom FRAME entry was developed to give the users a way to define their subset criteria. Because each MDDB could require different variables to use for subsetting, this entry would also have to be data-driven. The user had to be able to select an appropriate to subset on, then enter or select values of the selected variable. This subset could be a single value or a list of values of the selected variable. The ability to subset by multiple fields was also a requirement.

Additionally, users needed the ability to display the subset criteria, the 'where clause', which was created and to save and retrieve these where clauses to be used at another time. To accomplish this task, the following FRAME entry was developed:



This screen provides the user the ability to generate a where clause. However, an MDDB cannot be subset on non-dimension variables. In actuality, the permanent MDDB is not being subset. The where clause built by the user is passed into a macro variable, which is in turn used in an MDDB procedure call. This PROC MDDB code is the exact same code used to create the primary MDDBs in the Data Mart with a where clause placed on the input data set.

First, this version of the procedure includes the where clause built by the user on the subsetting screen. Second, the new MDDB does not replace the existing one; rather, it creates it as a temporary MDDB. Methods have been written to determine if the where clause exists and if so, to replace the defined MDDB in the EIS report object with this temporary one. A warning about replacing one MDDB with another in an EIS object: The two MDDBs MUST have the same structure (identical CLASS and VAR statements). If not, the method to replace the MDDB will fail and the report will not display.

Once this temporary MDDB is generated, the appropriate EIS reporting object is called. A number of methods are overridden for these objects, some to add features and others to modify them. Besides the method to replace the MDDB defined within the report with the temporary one, examples of the method overrides include:

- Code to capture and track subtable usage from the MDDB to see if users are accessing the NWAY tables.
- Code to make an Excel spreadsheet generated from a Multi-Dimensional Report look more like the table.
- Code to change how the _REACHTHRU_ method operates.

Another technical challenge we had to solve involved the Map object. A requirement for the application was to provide the ability to view a map of the state of Maryland showing data by county, then to drill down by county to see the data by ZIP code. This fairly standard (for this application) drill hierarchy exists in the other objects.

The challenge was to determine how to draw the county maps showing each ZIP code. The data to show values was already stored in the MDDBs, but the data to define the ZIP code borders is not part of any of the MAPS data sets provided with the SAS System®. With a little work by staff from both UMBC and SAS Institute, we were able to get the data sets needed to draw these maps.

Phase II – New Technology for Improved Performance and Functionality

All of the work described to this point was to accomplish Phase 1 of the application. However, it was also to lay the groundwork for the future. By the time this paper is published, Phase 2 of the system will have been implemented and in production. This enhanced version of the application includes vastly improved performance of the MDDBs, much more data and ways to access it, and more functionality. As with the first part of this project, it was a joint effort between CHPDM/UMBC and SAS Institute staff.

CHPDM management was fully supportive of Phase 2 and understood the importance of the design phase. Business rules that were difficult to define for Phase 1 were developed to not only deal with the Data Mart but all analytical processes taking place within the department. With these business rules in place, all results produced by the department are now based on the same algorithms.

The most important and global enhancement to the application was the implementation of the HOLAP (Hybrid OLAP) extensions to the SAS/EIS product. These methods and classes transfer the majority of the processing from the PC (client) to the Sun platform (server), taking advantage of the much faster and more powerful machine to do most of the work. Implementing

HOLAP also greatly reduced the disk space and design time needed for the MDDBs. Without HOLAP, trying to optimize performance required a lot of thought to determine all of the non-display reporting subtables and a lot of disk space to store those tables. For example, one of the new MDDBs for this phase originally contained over 1300 subtables in an attempt to guess what users might need! Because HOLAP creates a new MDDB for each data request, these non-display tables are no longer necessary. Guessing what users might do and trying to anticipate NWAY table hits became unnecessary because of how HOLAP works.

If a user makes a request for data in a client/server environment based on a combination of dimensions not anticipated by the developer/programmer, the only way EIS can satisfy the request (without HOLAP) is to download the entire NWAY table of the MDDB (which could contain millions of cells!) to the local machine, then summarize it to display what the user wanted. HOLAP eliminates this problem by reading the requested dimensions, extracting only the data needed for the request from the MDDB, building and downloading a new and much smaller MDDB, and displaying this new MDDB on the screen. And, all of this processing is performed on the server.

HOLAP also greatly improves the performance of the Subset by Dimension feature associated with multidimensional objects. Due to the very slow performance of this without HOLAP, it was removed from the Multi-Dimensional Report popmenu completely in Phase 1. To perform a simple subset could take in excess of 20 minutes! Where the EIS process would extract and download all values of all dimensions, HOLAP passes information, including the subsetting criteria, to the server for processing. This has reduced the time for subsetting to well under a minute, often taking just a few seconds.

Some of the new functionality that was implemented in the second phase included the ability to save information such as Medicaid recipient lists to be used as subsetting criteria for generating temporary MDDBs and modeling some of the data to allow 'numerator/denominator' processing.

This processing allows the user to build a population of recipients within a certain group (all people within a certain age group with a certain diagnosis, for example), then apply that population against the total population to determine how many recipients per thousand match that diagnosis. At the time of this writing, two methods are being considered and prototyped to implement this process, one of which involves implementing one of the other features of HOLAP, the ability to build a single MDDB from one or more other MDDBs or other data sources.

The new Data Marts will include data from multiple years of Medicaid claims (the Data Mart developed in the initial phase of the project only contains one year of claims information). Also, these new Marts will be designed with the challenges encountered during the development of the pilot Mart in mind. Finally, as data will be made available to CHPDM quarterly, the MDDBs will be updated quarterly. This will give the users access to the most up-to-date data that exists.

Other new features to be added in the next phase of this project include some new customizations for the EIS reporting objects and access to individual modules of the SAS/ASSIST® product to give users additional analysis tools. Also, at the time of this writing, a new module is being considered for the application. CHPDM and SAS are investigating the feasibility of a SAS/GIS® module, allowing the analysts at UMBC to see the data based on the geography and demographics of the state of Maryland.

Conclusion

This project continues to be a success in many ways. First of all, it proved that three groups of people with different backgrounds and strengths can work together with a single goal in mind and accomplish what they set out to accomplish. Second, that with a little patience, hard work, and teamwork, almost any technical challenge can be overcome. And finally and most importantly, the users were given access to their data using a tool set that allows them to satisfy their business needs quickly and efficiently.

Acknowledgements

To contact the authors:

Marge Scerbo
CHPDM
University of Maryland, Baltimore County
1000 Hilltop Circle, SS Room 309
Catonsville, MD 21250
Phone: 410-455-6807
Email: scerbo@chpdm.umbc.edu

Stuart B. Levine
SAS Institute Inc.
1700 Rockville Pike
Suite 600
Rockville, MD 20852
Phone: 301-881-8840 ext. 3363
Email: sasszl@wnt.sas.com