Paper 050-2009

Information Maps That Rock: How to Design for Optimal Performance While Balancing Cost and Other Business Requirements

Paul Christenson, Blue Cross Blue Shield of Minnesota, St. Paul, MN
Susan Trelstad, Blue Cross Blue Shield of Minnesota, St. Paul, MN

ABSTRACT

Information Maps can be sourced from relational database tables or views, SAS datasets, SAS views, SAS OLAP cubes, or SAS stored processes, or a combination of the above. This "Stack" of source options has varying pros and cons for development and ongoing maintenance costs. Depending upon any required data manipulation and/or the need to build data hierarchies, filtering requirements, and considering the need to return data results quickly in an interactive environment, some choices for Information Map sources may be better than others. In this paper, we will share our lessons learned in source selection and Information Map design, some realistic performance testing guidelines, and recommendations for improving the user experience using SAS 9.1.3 Intelligence Platform.

INTRODUCTION

The blessing and the curse of using the SAS 9.1.3 Intelligence Platform is that there are multiple options for delivery of reporting. For example, Web Report Studio (WRS) is much more geared toward the interactive exploration of data than are traditional SAS reports that have detailed and firm presentation requirements. Once the determination has been made that Web Report Studio reports are what you want to use for a particular project, there's a whole new array of choices to be made. The initial choices involve selecting the data source for the Information Map and determining where to incorporate business logic. Weighing on those decisions are resources, initial cost of development, cost of long-term ownership, and various business requirements. Also, the speed of data retrieval becomes an additional inherent requirement. It is in the light of these factors that we use the following process to choose what combination of components we want to use for building an Information Map and set of Web Report Studio reports.

BUSINESS REQUIREMENT ANALYSIS

Before considering design options, there is a level of business requirement analysis to be done. Meet with the business users to gain an understanding of what business decisions a given report or metric will impact. Review report requirement specifications, if provided, with them. Once you know what information is required, you can decide which data source best meets those requirements. A simple approach to identifying what data is needed in the Information Map is to create a metrics matrix. Create one row in a spreadsheet for each identified report or metric. Add a column for each individual data element and identify it as an attribute (A) or measurement (M). You can then associate the data source, table and column to the data element. Sort the columns and rows to identify groupings of metrics that use the same tables. These become the basis for defining an Information Map. Table 1 gives an example of what a metrics matrix can look like.

Table 1 Metrics Matrix

	Database Schema												
DB Table Names	USER_TBL				PROD_TBL		TASK_TBL						
DB Column Names	FULL_NM	USR_TTL	TEAM_NM	DEPT_NM	PROD_DS	PROG_DS	TYPE_DS	TASK_KEY	RSN_DS	OUTCM_DS	START_DT	DUE_DT	CMP_DT
Information Map Folder	**Call Team**				**Prod**		**Tasks**				**Dates**		
Information Map Attribute Business Name	**Nurse**	**Role**	**Team**	**Department**	**Product**	**Program**	**Type**	**Count**	**Reason**	**Obtained**	**Start Date**	**Due Date**	**Comp Date**
Count of Calls Attempts	A		A			A	A	M	A				A
Count of Calls to be completed		A				A						A	
% of Calls successful	A		A			A	A	M	A	A			A
Count, % of calls as wrong numbers.	A		A		A		A	M	A	A	A		

Presentation requirements can have a significant impact on determining the data source for an Information Map. Is there a required layout that has to be duplicated? Will this be used by internal customers or external customers? Do we need to see all the descriptive columns or attributes in a cross-tab report even if there is no measurement for the cell? Will the finished report be scheduled or parameter driven? Will the report be available through SAS Information Delivery Portal (IDP), the Microsoft plug-in, or emailed as a PDF? The answers to these questions will strongly influence the performance requirements of the data source and how the Information Map is designed.

Another significant impact to data source selection and Information Map design is the need for row level security. For example, we may need to dynamically restrict the claims a user can see based on what account they are a part of. Since we manage over 60,000 accounts at BCBSM, this needs to be managed in the data source. At Blue Cross Blue Shield of Minnesota (BCBSM) we also have to manage the security around Protected Healthcare Information (PHI). Because of this, we have several layers of security to pass through and stringent rules around who can access what kind of data.

THE SOURCE STACK

Once the metrics to be generated from an Information Map have been identified, and presentation and security requirements are known, a data source or sources can be selected. The diagram in Figure 1 shows the basic options for data sources that can be fed directly into an Information Map. All possible cross connections between views, tables, and sources are not drawn. The SAS Scalable Performance Data Server and other warehouse appliances are not in the scope of this paper.

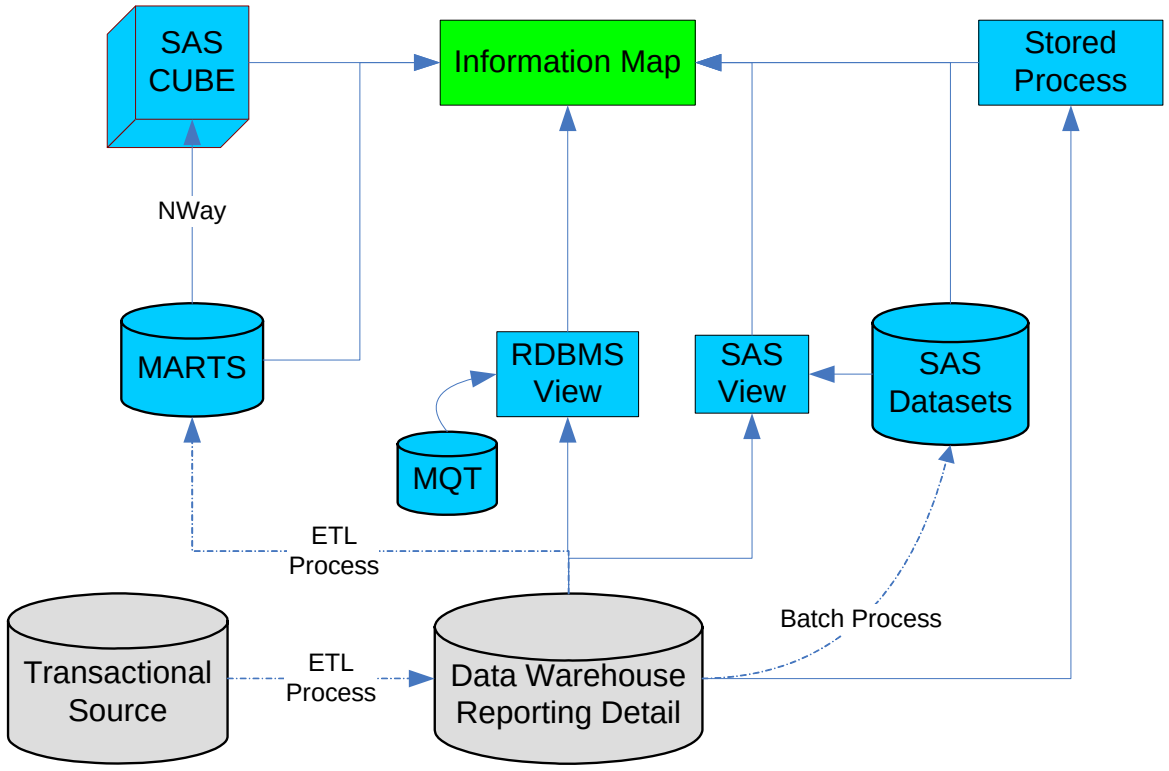

Figure 1 Information Map Source Stack

It is tempting and sometimes necessary to query the transaction source of record. This has the least number of levels to build reducing development time and eliminates data replication. However, transactional databases are usually designed with 3rd normal form tables optimized for transactional processing and are governed by Service Level Agreements (SLAs). Queries against transactional databases for analytical purposes may end up with multi-table joins and complex filters, causing poor performance and impacting the SLA. At BCBSM we are moving away from directly querying transactional sources and providing data warehouses for analysis and reporting. Before using a transactional source, work with the IT owner to verify that the SLA for the database is not impacted. Then extract the data needed into a SAS dataset during off-hours.

Data warehousing structures and strategies is an industry in itself and will be not be in scope for this paper. The data warehouse will be the default source of detail data that the Source Stack draws from. Database views, Materialized

Query Tables (MQT) and Data Marts that draw from the data warehouse will be discussed as sources for Information Maps.

SOURCE STACK VS COST

Different data sources have unique strengths and weakness when used in an Information Map. There are also development and administrative costs for each source. The following listing describes the data sources and their associated cost considerations in order of least expensive to most expensive, based on our current experience at BCBSM.

1. **SAS Views**
   - **Development Options:** SAS views can be used for simplifying join structures in Information Maps and enabling two pass queries against SAS datasets or a Relational database management system (RDBMS). They can also encapsulate SAS data step functionality. SAS views and SAS datasets can be joined together based on common relational rules.
   - **Administrative Costs**: A SAS view can be created easily by a SAS programmer for use in their application; it does not require database administrator (DBA) involvement. However, SAS views are not as flexible as a RDBMS view.
2. **RDBMS View**
   - **Development Options:** A fundamental structure in all databases, views can be used to create everything from simple isolation layers to complex multi-pass queries. Materialized Query Tables (MQTs) can be used with views to improve performance. The SQL logic contained within views can simplify logic within an Information Map. For example, a view can be used to create a hierarchical dimension from multiple tables and present it as a single source in the Information Map.
   - **Administrative Costs**: RDBMS views must meet the IT database owner's approval in order to deploy to the database. This may require extra validation steps but once deployed to the database, there is little added cost to view structures.
3. **SAS Stored Process**
   - **Development Options:** As the most flexible source for processing data, a stored process can use all SAS functionality and temporary data sets for multi-step transformations. The stored process fires after parameters are selected and its results sets are sent to WRS for further processing. Parameters are limited to static lists associated to the stored process metadata. Only one stored process can be associated to an Information Map.
   - **Administrative Costs**: A stored process requires added administration for temporary files and metadata, but does not require a separate process for building the data layer.
4. **SAS Datasets**
   - **Development Options:** SAS datasets use a similar structure to databases and follow SQL rules. They enable the use of all SAS functionality in their creation and the flexibility of processing is limited only to the developer's skill and frequency of refreshing the data. With proper design, a combination of SAS datasets can be reused to support multiple Information Maps.
   - **Administrative Costs**: SAS datasets require separate batch processes to load from the RDBMS and a scheduled job process. They also involve additional development time, unit and system testing.
5. **RDBMS MQTs**
   - **Development Options:** MQTs in DB2 or Materialized Query Views (MQVs) in Oracle are physical tables that are created in the database based on a views definition. When the view is accessed as part of a query, the optimizer determines if it can use part or all of the preprocessed data in the MQT rather than having to process the view logic. MQTs have limitations to the types of SQL they will support.
   - **Administrative Costs**: MQTs must be created by the DBA. They require extensive testing to verify that the database is using the MQT rather than the detail tables in the view. They also need to be refreshed separately, after the database has been refreshed.
6. **SAS OLAP Cubes**
   - **Development Options:** The OLAP engines based on pre-summarized data at multiple levels of a hierarchy, so drilling up and down a data hierarchy (for example, from year to quarter, from quarter to month, and back again) is very fast. There are two ways to store this summarized data. A Multidimensional On-Line Analytical Processing (MOLAP) cube stores the summarized data in a separate proprietary structure. A Relational On-

Line Analytical Processing (ROLAP) cube can use the proprietary structure and can access more detailed data in SAS Datasets, SPDS or RDBMS tables. The trade off between the two approaches is MOLAP requires less design time and has faster overall performance while ROLAP takes more design time but can access larger volumes of data. Cubes are generally created for a specific subject area and one Information Map.

- **Administrative Costs**: An OLAP cube requires configuration of a separate OLAP server and a well define refresh process for the data. ROLAP cubes have more options on how to refresh the data but also have more administrative complexity.

7. **RDBMS Data Mart**

   - **Development Options:** RDBMS data marts are the best practice structures for supporting multiple users and applications with business critical data. These generally take the form of Star Schema designs that aggregate detail data from the data warehouse. Other "big flat table" or reporting tables can be useful for other types of analysis. Designing data marts have higher development costs due to the multiple skills sets needed and longer lead times needed for data design and building of extract, transform, load (ETL) processes.
   - **Administrative Costs**: A RDBMS data mart has the highest administrative cost since it requires a separate RDBMS application and disk space. Also required is additional staffing for DB administration and for ETL processing.

Making design decisions based solely on selecting the lowest cost source option isn't practical, since re-work and additional cost may be incurred once initial performance behavior is established. Development cost, administrative overhead, reusability, and performance are all factors that have to be balanced to optimize the total cost of ownership for a BI application.

SOURCE STACK VS REUSEABILITY

It is always a good design goal to be able to reuse sources for multiple purposes. This will ultimately cut down on your development and administrative costs. Objects in the source stack (see Figure 1) have different qualities that make them more or less reusable. The following list describes data sources in order of those that serve one Information Map to those that can support multiple Information Maps.

1. **SAS Stored Process:** An Information Map can only be associated to one stored process. The stored process can produce one to many tables for the Information Map but the result tables must always have the same structure or the Information Map will produce errors.
2. **RDBMS MQTs:** These tables are specialized to improve the performance of a specific type of SQL. Our experience at BCBSM has shown the optimizer to be picky in what SQL it will accept as passable to the MQT. A single MQT is not very flexible, but a family of MQTs could be reused across multiple projects.
3. **SAS OLAP Cubes:** Buy the nature of their design; cubes target a specific subject area. Since MOLAP cubes summarize data at multiple levels in a separate structure they have a size limitation on the number of dimensions and measurements that can be stored in the cube. ROLAP can have an underlying data mart and reuse can be based on the data mart design.
4. **SAS Views:** SAS views can be joined efficiently to other SAS views as long as they are sourcing from SAS datasets. If the SAS view is using explicit SQL to a RDBMS, they cannot be joined efficiently.
5. **SAS Datasets:** SAS datasets can be used across multiple projects with similar properties as a RDBMS. One limitation is that only SAS code can be used to access them. They also should be refreshed from the data warehouse on a scheduled basis.
6. **RDBMS Data Mart:** This data design strategy is widely implemented and can be used to solve many types of reporting issues when data aggregation is need in the context of a specific subject area. Conformed dimensions can also be used to create a network of fact tables that share the same dimensions.
7. **RDBMS View:** As the most flexible component in the source stack, RDBMS views can be used as a stand alone "glass" view, aggregation logic, or in conjunction with MQTs.

SOURCE STACK VS PERFORMANCE

Web applications are generally expected to return results fast; often companies will have security measures in place to protect against denial-of-service attacks (DoS attack) by imposing a time limit on web pages. BCBSM security has a two minute timeout on results returning from the web server before the connection is truncated. BCBSM also has corporate standard, that all web pages must return in 25 seconds or less and be less than 100kb in size. The bottom line is this: users don't like to wait! We have ranked data sources, from Figure 1, in order of increasing ability to improve performance.

1. **SAS Views:** A SAS View has minimal improvement on the performance of a query. A view can be used to force the order that indexes are selected. It is not recommended to join multiple SAS views that use explicit SQL to a RDBMS since each view would have its own session SQL pass-through would be disabled.
2. **SAS Stored Process**: All of SAS performance improvement tricks can be used in a stored process. However, the performance of the stored process is dependent upon the volume of data that is being processed and the I/O bandwidth of the server.
3. **RDBMS View**: Database views can be joined together without issues but SAS views and RDBMS views cannot be joined together without significant performance issues. MQTs are the primary avenue for improving the performance of views.
4. **SAS Datasets**: SAS datasets can be indexed, but the SAS query optimizer is not star schema aware. Any number of aggregation strategies can be used with datasets. Depending on the data storage connected to the Workspace Server there can be performances issues due to I/O constraints if sourcing large volumes of data with multiple users.
5. **RDBMS MQTs**: MQTs are created in the database to increase the performance of an existing view. Multiple MQTs can be created based statistics of usage over time to maximize database performance.
6. **RDBMS Data Mart**: A star schema with well defined hierarchical dimensions can improve performance with the correct database optimizer settings. For Example, DB2 has a star join optimizer that combines the filtered results of the dimension tables before joining to the central fact.
7. **SAS OLAP Cubes**: Because all combinations of dimensions at multiple levels are summarized, OLAP cubes are among the fastest data sources available. Unlike the RDBMS structure, the slower performance occurs when querying lower levels of detail. If you try to put too many measurements or dimensions in a cube they can become too large and have associated performance degradation.

MEASURING PERFORMANCE

So how do you test the performance of an Information Map? You could have hundreds of different combinations of calculations, table joins, and filters that would each produce a unique SQL statement. One approach is to determine the least number of queries that exercise all the components in the Information Map. The following steps can be used to define the queries.

1. Create a table that identifies all the calculations in the Information Map versus the data source tables that the calculation is based on. See Table 2.

Table 2 Calculations

	Tables or data sources				
	A	**B**	**C**	**D**	**F**
Calc 1				C	
Calc 2				C	
Calc 3					C
Calc 4					C
Calc 5					C

2. Create a second table of all the filters, both prompted and fixed, in the Information Map versus what tables they are from. See Table 3

Table 3 Filters

	Tables or data sources				
	A	**B**	**C**	**D**	**F**
Filter 1	X				
Filter 2	X				
Filter 3		X			
Filter 4		X			
Filter 5			X		
Filter 6			X	X	

3. Create a third table identifies queries that exercise realistic combinations of filters, tables and calculations. Be sure to create one query that forces the joining of all tables within the Information Map.

Table 4 Queries

	Tables or data sources				
	A	**B**	**C**	**D**	**F**
Query 1	F1	F3		C1	C4
Query 2		F4		F7	C3
Query 3	F2		F5	F8	C5,C6
Query 4			F6	C2	C3
All Join	F1	F3	F5	F7	C4

Based on the queries defined and on an estimated number of users who will be using the Information Map, you can create load test scenarios that will stress the system in a business-realistic way.

LOAD TESTING

As a general guideline, for every 100 named web users, 20 may be on-line and 2 could be running a report at the same time. If a report has a time-critical factor, such as an operations report that 20 people will be generating first thing in the morning, there will obviously be more concurrent users. The guidelines in Table 5 are used for evaluating levels of load testing for Information Map performance at BCBSM. Web performance is a critical issue and BCBSM since all internal and external websites sit behind a reverse proxy server that will terminate any connection that runs longer than 2 minutes without returning something from the server.

Table 5 Load Testing Guidelines

Named Users for Information Map	Concurrent users to test	Result time frame
1 – 50 Users	1 User	5-25 seconds
50 – 199 Users	Ramp up 1 – 10 Concurrent users	10 concurrent users less than 1 Min 30 Seconds average response
200+ Users	1 – 20 Users Concurrent users	20 concurrent users less than 1 Min 30 Seconds average response

There are multiple load testing tools on the market; BCBSM use the IBM Rational Tool Suite. If you do not have a load test tool in house you may want to consider capturing the SQL that is being generated by WRS and build a simple concurrent query generation tool in SAS using batch mode processing.

To estimate performance in a Test (Quality Assurance) environment, data volume in the environment should be approximately 50% of production database.

WEB REPORT STUDIO CONSIDERATIONS

WRS has options that can help or hurt the performance of a report. Here is a list of guidelines we use at BCBSM:

Report Layout

- Use Group Breaks to eliminate scrolling. WRS will only query and process for one element of the group break field at a time, so performance is an added benefit.
- Cross Tabs are memory intensive; use list reports, if possible. List reports will paginate the output one page of results at a time, greatly reducing system overhead.
- Limit size to a maximum two pages of generated output. Use group breaks in conjunction with the cross-tab, if necessary, or in conjunction with a list report. If the total output (including all group breaks) is over 20 pages, consider scheduling the report. If the total output is over 100 pages, do not use WRS.

Scheduling

- If the report is to be used by separate groups (department, sales region), and it does not require a prompted response from the user, you can schedule the WRS report and use the distribution option to deliver a PDF to a subscription channel.

Prompts

- Do not store pick-list values that will change frequently within the Information Map. You will need to re-deploy the Information Map every time the list needs updating.
- SQL pass-through to a RDBMS works for dynamic pick-lists if you are running Hotfix 10 for SAS Web Report Studio 3.1 and set the webreportstudio.strip.based.filtering.strategy to 0 in LocalProperties.xml
- Avoid using database driven lists greater than 3000. Dynamic lists do not paginate; use a user-entered filter.
- Do not use the main/fact table as a source for a dynamic pick list filter. The system needs to use a "select distinct" and if it is doing that on a multi-million row table, users will wait too long for the pick-list to generate.

Finally, if none of the above options provide adequate performance to support a parameterized report, use a parameter-driven stored process reports running in background mode, with an Alert Portlet to let the user know when the report is ready.

CONCLUSION

SAS has a rich set of options for building robust Business Intelligence applications. They do require a level of analysis and design that other BI tools do not, simply because other BI Tools have less options. The use of different data sources to meet business requirements can vary based on the environment, architectural standards, and developer preferences. At BCBSM our goal is to balance the different areas of cost, reusability, and performance to minimize overall cost and maximize performance. Table 6 and Figure 2 show a relative comparison of the different data sources based on current experience at BCBSM.

Table 6 Comparison Summary

Data Sources	Development Cost	Administrative Cost	Reusability	Performance
SAS Views	Low	Low	Medium	Low
SAS Stored Process	Medium	Medium	Low	Medium
SAS Datasets	Medium	Medium	Medium	Medium
SAS OLAP Cubes	High	High	Medium	High
RDBMS Views	Low	Low	High	Medium
RDBMS MQTs	Medium	Medium	Low	High
RDBMS Data Marts	High	High	High	High

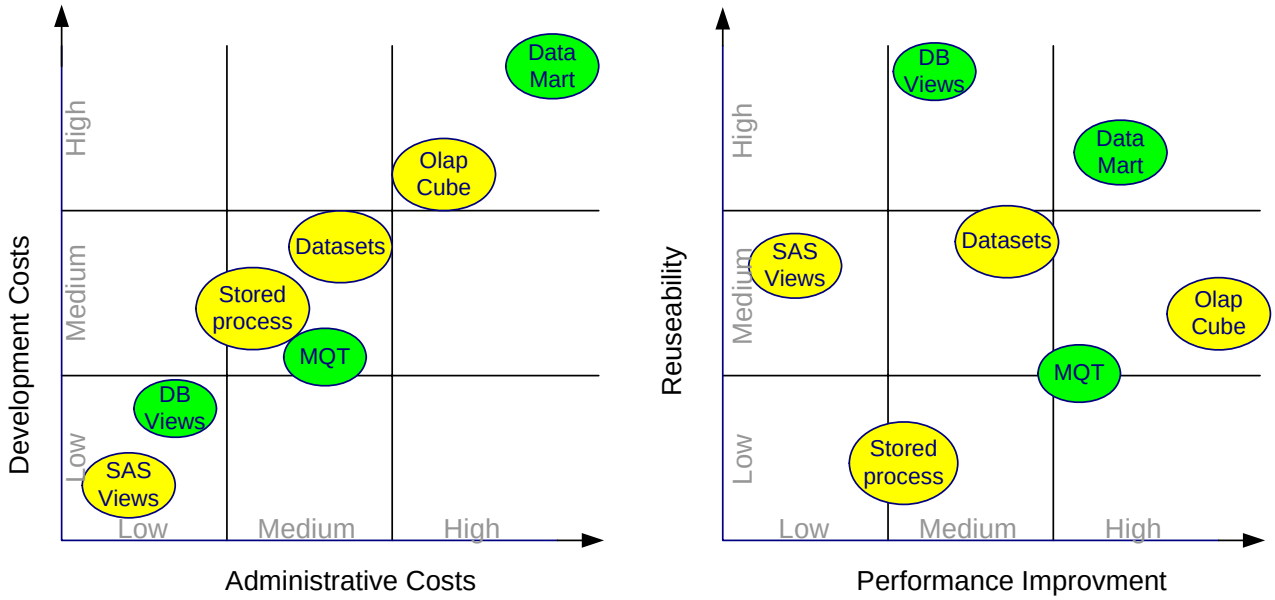

Figure 2

RECOMMENDATIONS

SAS Views: SAS views are best used for prototyping RDBMS views or for creating two-pass SQL queries to datasets. When creating aggregated dataset, use SAS Views to force the order in which SAS indexes are selected.

RDBMS View: Use RDBMS views for accessing detail data or data that is refreshed daily. Use views for creating column name isolation layer from the database tables. Use views to create dimensional hierarchies if the data needed is distributed across multiple tables. This approach works well if you need to create row level security in the Information Map.

SAS Stored Process: Use SAS stored processes for Information Maps with very specific scope and with complex, on the fly, calculations that cannot be calculated until after the user has selected parameters. If working with multiple Information Maps, it is better to create SAS data sets or a RDBMS data mart to source the Information Maps than to create multiple stored processes.

SAS Datasets: Do not use SAS datasets to store historical detail data. Use them as an integration/aggregation data layer to simplify complex calculations and improve performance by pre-summarizing data. If table naming conventions are maintained, an Information Map can be redirected from the dataset back to a RDBMS data mart without changing code.

RDBMS MQTs: Use MQTs for tables and views that are frequently used at a higher aggregation level but still require access to detail. Take advantage of the DBA's performance analysis tools for defining what kind of MQTs to create.

SAS OLAP Cubes: An OLAP cube is a good fit for reporting requirements that involve multiple levels of reporting aggregations and data that has well defined dimensional hierarchies. For example, time is a good dimension for a cube (from 1 year to 4 quarters; from 4 quarters to 12 months; from 12 months to 30 days). On the other hand, a customer account dimension that goes from 3 divisions to 20,000 accounts would make a poor OLAP cube drill path.

RDBMS Data Mart: A data mart is the best (and most costly) approach for dealing with large volumes of data that have a frequent (daily or sooner) refresh rate. For Information Maps that need to support a large number of users, a data mart will also improve performance.

These recommendations fall in line with our use of RDBMS views for daily reports with certain views supporting multiple Information Maps. BCBSM also uses stored processes when data calculations are not easily accomplished in SQL and SAS datasets are used as an aggregation level to support monthly reports.

Because of the many data source options available in SAS and in the RDBMS, there is one that will bring your project into balance.

REFERENCES

High,Jerry. “Going Beyond Simple Information Maps to Improve Access to Data Sources” *Proceedings of the 2008 SAS Global Forum*. March 2008. http://www2.**sas**.com/proceedings/forum2008/**049-2008**.pdf

Zenick, Ben and Miles, Brian. “Beyond the Basics: Advanced OLAP Techniques” *Proceedings of the Thirty First SAS Users Group International Conference.* March 2006 http://www2.**sas**.com/proceedings/sugi31/**219-31**.pdf

Melynk, Roman DB2 Information Development , IBM Canada Ltd. “DB2 Basics: An Introduction to materialized query tables” *DeveloperWorks* September 2005 http://www.ibm.com/developerworks/db2/library/techarticle/dm-0509melnyk

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name:	Paul Christenson	Susan Trelstad
Enterprise:	Blue Cross Blue Shield of Minnesota	Blue Cross Blue Shield of Minnesota
Address:	3535 Blue Cross Road, Rte:M321	3535 Blue Cross Road, Rte:M321
City, State ZIP:	Eagan, MN 55122-1154	Eagan, MN 55122-1154
Work Phone:	(651) 662-4422	(651) 662-7070
Fax:	(651) 662-4422	(651) 662-7070
E-mail:	paul_e_christenson@bluecrossmn.com	susan_trelstad@bluecrossmn.com