Paper 049-2008

Going Beyond Simple Information Maps to Improve Access to Data Sources

Jerry High, Blue Cross and Blue Shield of Minnesota, Eagan, MN

ABSTRACT

SAS Information Maps provide users with easy access to rather simply-defined data sources. Although they can be directed to external relational databases, performance can be an issue due to the way in which queries are built and passed via the LIBNAME engine within the SAS/Access products. In addition, sometimes it is necessary to perform nested queries to databases in order to provide more meaningful information in reports. This paper will explore options to assure performance and go beyond simple table joins to provide easier access to end users.

INTRODUCTION

An Information Map is one component of the SAS Intelligence Platform designed to help bring reporting to a broader audience of users. By predefining the data sources, columns, join criteria and filters, more analysts can now build many of their reports without knowing a programming language or understanding the complicated data structures that house the various data sources. Rather than spending a lot of their time acquiring the data, they can now spend more time analyzing the data.

One challenge to Information Maps is assuring that the performance (time to return data) doesn't suffer as the complexity and size of the data sources grow. Normally there are many basic guidelines for building and using Information Maps. These include such things as predefining filters to limit the data returned and creating new calculated columns to bring consistency in reporting. For larger or more complicated data sources, additional work is needed to build an Information Map that assures that the proper information is returned and that it returns in a reasonable amount of time.

As an example, providing reasonable performance can be especially difficult when the data source is housed in an external relational database management system (RDBMS). Under this circumstance the SAS LIBNAME engine is used to generate SQL code block(s) that are submitted to the remote RDBMS. It is here that performance can suffer if the LIBNAME algorithm creates a SQL code block retrieving entire tables back to the SAS environment before merging and aggregating rather than performing this within the RDBMS.

When accessing an external RDBMS, it is possible that the structure doesn't lend itself well to simple aggregation. There are some cases where summarized data from two sets of tables needs to be merged to provide a meaningful calculation (see Figure 1). In standard SQL this can be accomplished with nested queries. With Information Maps, it is not possible to directly create a nested query.

```
proc sql;
  select product,
         (totclms/totcont) as clmspermbr
    from (select product, sum(claims) as totclms
            from product a, claims b
             where a.ckey_cd=b.ckey_cd
            group by product) e,
         (select product c, policies d
            where c.pkey_cd=d.pkey_cd
            group by product) f
    where e.product=f.product
    order by product
;
quit;
```

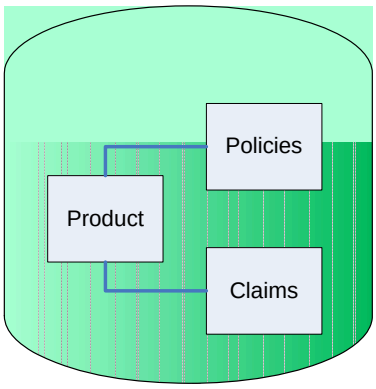

Figure 1

Under the previous two scenarios there are ways to work with the Information Maps to use a specific SQL code block or to call nested queries. This paper will explore a few of the options available to enhance the access path defined for Information Maps that can result in better performance and greater capabilities to bring data from more complicated structures.

INFORMATION MAPS

Information Maps contain no data. They only provide a read-only view to the data. They can have predefined filters or calculations that are applied before the data is returned to the user. They contain metadata about the data source including such things as column labels and extraction logic. Data sources normally include SAS datasets, SAS OLAP cubes or other external data sources available through the SAS/Access products.

Once created, Information Maps are typically used in the Information Delivery Portal (IDP) and Web Report Studio (WRS). They can also be accessed through Enterprise Guide (EG), the Add-in for Microsoft Office (AMO), and custom applications developed through AppDev Studio, although when accessed through these clients, Information Maps do come with some limitations.

When invoked, Information Maps return data as if it were contained in a single data table. If accessed through a GUI interface like WRS, the user is allowed to select the columns desired, filter the data and present data in a simple table or chart. Through the base product, one can also reference data through an Information Map. The SAS procedure used (e.g. PROC PRINT) takes the results of the query generated by the Information Map and processes it. To the user, it appears as if it were a single SAS dataset.

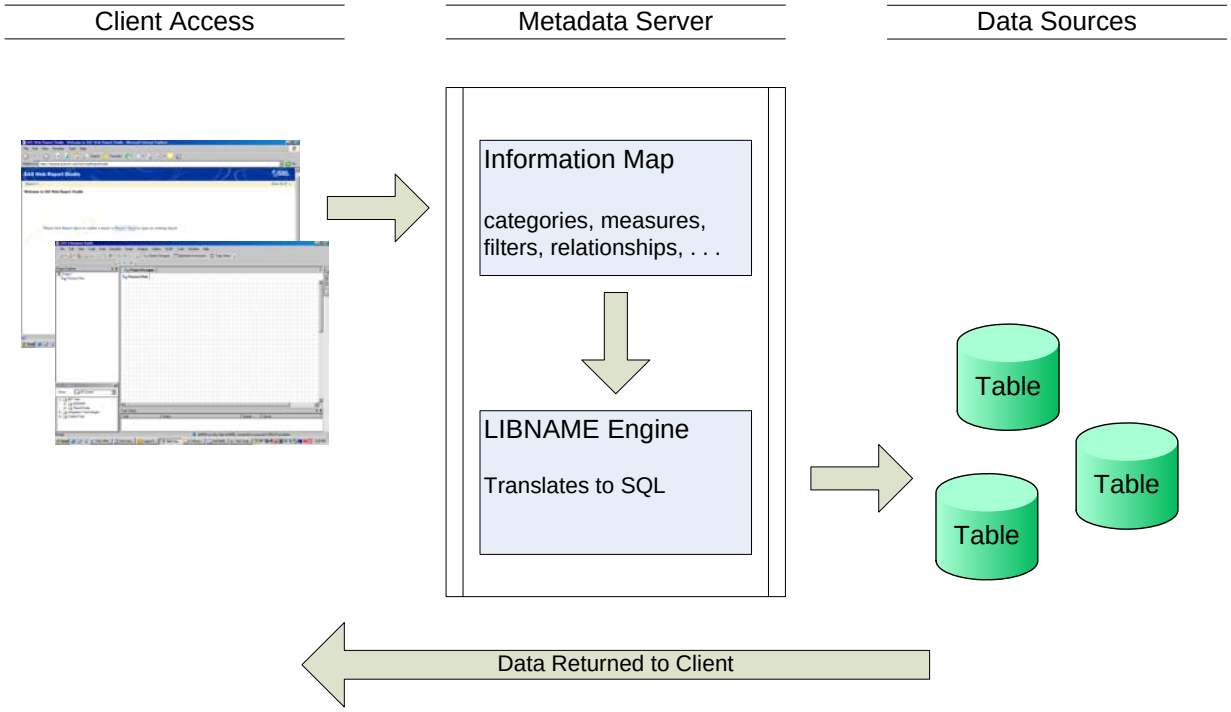

Figure 2

It doesn't matter if the data source is from a SAS dataset or an external RDBMS; the Information Map uses the SAS LIBNAME engine to access tables (see Figure 2). Again, the Information Map contains no data, just the attributes about the source data and how to build the extraction logic through the LIBNAME engine.

UNDERLYING PERFORMANCE ISSUE

SAS refers to external databases as foreign data sources. When accessing foreign RDBMS data sources, there are two basic paths: explicit and implicit access. When using the explicit method, the user typically writes the desired SQL

within the PROC SQL procedure. The PROC includes both the credentialing information and the exact SQL desired. Here SAS just performs a straight pass-through of the code to the RDBMS and the database optimizer then takes over. The results of the query are then fed back to SAS which in turn converts the results into a SAS work/permanent dataset. Under this method, much of the workload of filtering and aggregating is forced upon the RDBMS (see Figure 3).

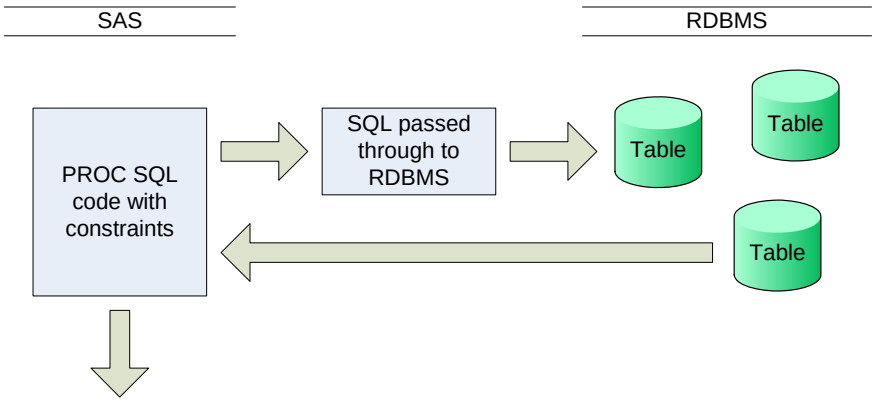

Figure 3

When using the implicit method, the credentialing information is put into a LIBNAME statement. The SQL that is within the PROC SQL procedure is then sent through the SAS Libname Engine to determine the SQL to send to the foreign data source. The Libname Engine provides tremendous power in that the user can reference a foreign database as if it were a SAS dataset. These tables can then be accessed from other SAS data steps and procedures. One can join SAS datasets with tables from other databases via the PROC SQL procedure. The down side is that SAS now takes over these requests and the Libname Engine determines the proper SQL to send to the RDBMS (see Figure 4). The user doesn't have control of this and the performance can be hit or miss depending on the complexity of the data request. Even in the PROC SQL, there is a chance that the specific SQL will not be the SQL sent to the RDMS. In extreme cases, a single SQL with multiple table joins can be translated into multiple SQL requests bringing full tables back to the SAS server where the results are joined together.

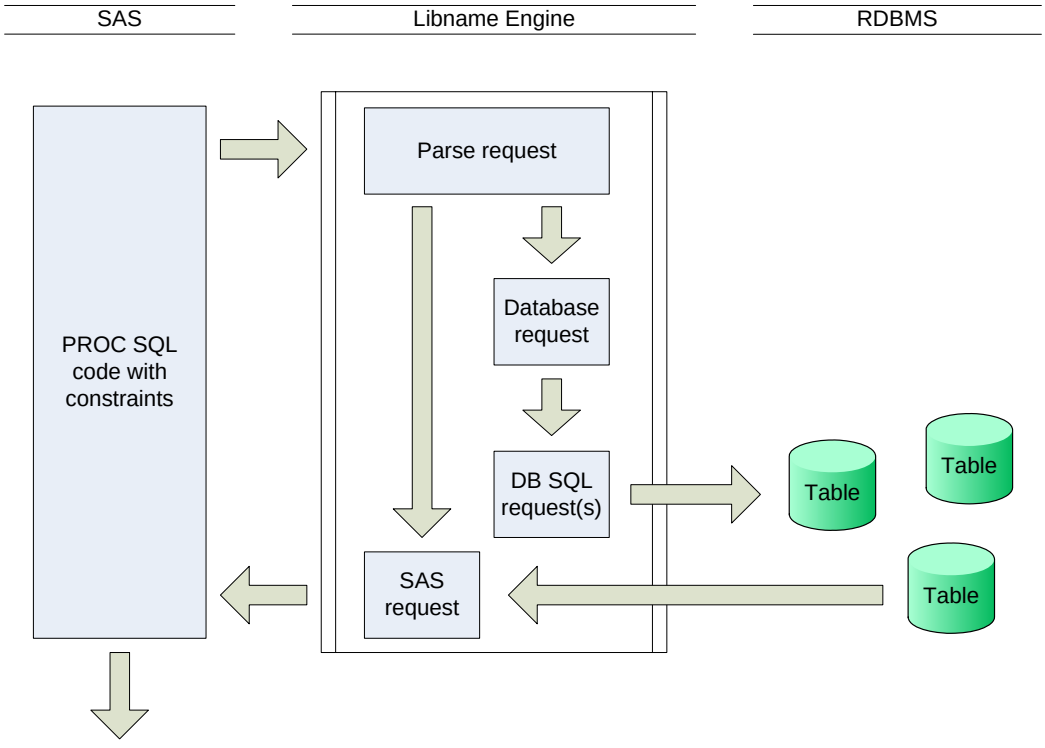

Figure 4

As a result, predicting performance against RDBMSs can be challenging. A two-table join resulting in a small result set on the database may be turned into two single queries, bringing back one or two large tables to the SAS server before it is then joined by SAS to produce the same small result. Because of the system fetch times (time to transfer data from the source), translating the large volumes of data into a SAS format and then processing the join on the SAS server, processing times could be much longer than expected. The results to the user are the same whether the processing occurs on the RDBMS server or on the SAS server. What can vary is the total time it takes to produce the results. This can be an issue when accessing data via the web (using IDP and WRS), where users have come to expect responses in under 30 seconds. The challenge is to force the bulk of the processing to the RDBMS, where, typically, there is more computing power.

PERFORMANCE OPTIONS

When building Information Maps to retrieve data from an external RDBMS, there are many options to improve the performance of queries generated by the Map. These range from basic options like using filters to more advanced options such as using stored procedures. Some of these options are easy to do; others require system or database access not normally given out to analysts.

BASIC OPTIONS

The first option to improve performance is to limit the data retrieved. Only bring in the data one wishes to analyze. Filters can be used in the Information Map to restrict the data retrieved. Filters can be predefined (fixed), prompted or user-generated at the time the report is requested. One should keep in mind that different clients (e.g. WRS or EG) allow different filter types (see Figure 5). And ultimately, the user can decide not to use any of the defined filters resulting in a long running query.

Filter Support for Information Maps and Stored Processes			
Client	**fixed**	**prompted**	**user defined**
Web Report Studio	√	√	√
Web Report Viewer (within IDP)	√	√	√
Information Map Viewer	√		
Enterprise Guide	√		
Add-in for Microsoft Office	√		

Figure 5

Other options that may be available to improve performance include replication and co-location of the data. If the data is in a SAS format, no translation from the foreign data source is needed. This can be accomplished through data replication of the foreign data source onto the SAS server. Depending on the company, uncontrolled data replication may be discouraged. If replication is allowed and SAS' Scalable Performance Data Server (SPDS) is available, additional performance gains can be obtained through its use. The SPDS has other features that can improve data access performance beyond the scope of this paper.

Another option to improve performance is to install the RDBMS on the same server as SAS to reduce the fetch time between systems. Again, many companies will keep those functions on separate servers due to software licensing costs, resource management or other architectural considerations.

Depending on the nature of the analysis, some data can be pre-summarized into tables that have fewer rows, thus reducing the time to retrieve. This can be done as a SAS dataset or requested of the DBA group to have additional tables created within the RDBMS. Many databases also support aggregation techniques, such as Materialized Query Tables (MQT's) in DB2, which can redirect the incoming SQL to an aggregate table. This is helpful when the SQL request has a *group by* clause matching the aggregate table. Unfortunately, this may not be useful if the Libname Engine chooses to request a detail table to be returned to the SAS server so that SAS can perform the join. This points out the importance of fully understanding the database structure and the business requests expected from the database.

DATABASE OPTIONS

There are a couple of options that can be used on the database side to improve the performance of Information Maps. Recall that the Libname Engine decides how to request the data from the RDBMS, whether to send one query and let the database do the processing or to send multiple queries of single table fetches and join the results on the SAS side. If there is only one table to query from, then the Libname Engine has no issue; it sends a single query to the database.

The Libname Engine can be fooled into thinking it's accessing a single table through a database view. One can request that a DBA add a view to the source database. This should be set up to join the key tables together. Most people understand that a view will not improve performance, it just makes the query simpler in the *select* and *where* causes of the SQL call. More important, the Libname Engine thinks it's fetching from a single table and sends the entire request to the database where the database optimizer will reform the query to join the necessary tables without SAS pulling back full tables to the SAS server.

Another approach, similar to the database view, is to create a stored procedure on the database. In some more complicated database structures, it may be necessary to create multiple sub-queries or other more complex code. In this case, one can request a database stored procedure be created. A good developer or DBA can create one and exercise performance tests to assure that the stored procedure is tuned for optimal performance. Once created, it can look just like another table in the same way as a database view. Again the Libname Engine sees it like a single database table.

Both of the above options can work very well. For the user (and the Libname Engine) both can be set up to look like a single table. However, with either option, one may need to provide justification for these changes to the DBA who is responsible for those source databases.

SAS OPTIONS

Sometimes there are tight controls on corporate databases inhibiting implementation of the above database options. Also, depending on work assignments or priorities, a DBA may not be able to commit time to analyzing and creating views or stored procedures. In these cases, one still has a couple of SAS options. In the SAS framework there are similar features to the database options that can help improve the performance of Information Maps. Through the use of SAS SQL views and SAS Stored Processes one can effectively define the query path for the Libname Engine, avoiding the large table requests which in turn will improve the performance of the Information Map.

SAS SQL VIEWS

As with a RDBMS, SAS SQL views can define the SQL to be used. Two main differences are that they don't require a DBA and the code exists on the SAS server, not on the RDBMS server. Care is still needed to assure that the query performs well. The same issue regarding the Libname Engine still exists. If one uses an implicit SQL call to create the view, the Libname Engine still gets involved. If one uses an explicit query, the Libname Engine is bypassed and the SQL is passed directly to the RDBMS. The two possible paths are depicted in Figure 6.

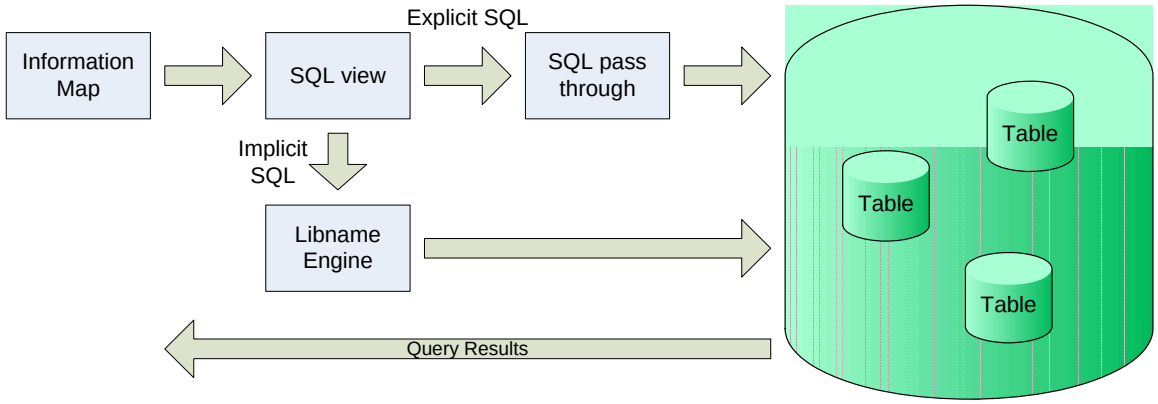

Figure 6

Creating a SQL view using an explicit query allows one to tune the query to the database as well as include rather

complex SQL logic including nested queries. The key is to test the queries to assure that they have reasonable response times from the database. A side benefit of the SQL view is that one can build it using the EG client. In EG, one can build the view into a permanent directory and test the query performance with classic SAS programs. Once the views are tested, one can then register them in Management Console like any other SAS library. Once registered in Management Console, one can then create an Information Map based on that SQL view. To the user creating the Information Map, it appears as a single table with all of the columns that were part of the defined view.

In its simplest form the creation of the SQL view with an explicit query would look like this:

```
libname home '/home/sasdata';
run;
proc sql;
  connect to oracle as clms
    (path='CLAIM' user='jdoe' password='mypasswd');
  create view home.clm_view as  /* library.viewname */
    select *
         from connection to clms
        (
         select *
           from clmlines.claims /* schema.tablename */
        );
  disconnect from clms;
quit;
libname home clear;
run;
```

There are no limits to the complexity of the query, but keep in mind that if the ultimate goal is to use the SQL view in an Information Map accessed via the IDP and WRS, one should strive to have a query that runs in under 20 seconds.

There are a couple of minor notes about views. Data step views can only use the Libname Engine to access foreign data sources. The SAS/Access views also have limitations such as not supporting long variable names. As a result, the PROC SQL views are the preferred method for creating the views to be used by Information Maps.

SAS STORED PROCESSES

SAS stored processes are similar to the stored procedures in databases. A SAS stored process is SAS code or program flow that is registered in the Metadata server using the Management Console. Stored processes are very powerful in that they can have parameters for user input and produce various outputs such as datasets, reports or charts. Once created, they can be called from the IDP, EG, or the AMO to produce the same results. It enables many options for rendering complex reports across the SAS Intelligence Platform.

Stored processes have one main advantage over the PROC SQL views. They can perform other manipulations to the data through the use of other SAS data steps and procedures. This enables one to add complex business logic, filtering, or calculations that may be difficult or impossible in the PROC SQL view.

There are a few things to keep in mind when creating stored processes. Under normal circumstances, stored processes are executed under the Stored Process Server. When accessed through an Information Map, stored processes are executed under the Workspace Server. Although this might not be of great concern, it is important to know that the user session management is handled differently. Another point to keep in mind is that an Information Map can only accept and process a stored process if it meets key criteria. For example, the stored process must have the output type set to “none”. In other words, the stored process should not produce output on its own.

As with the PROC SQL view, the stored process can be created in EG. One can create the code and validate the performance to the RDBMS being queried. Once working optimally, it can be converted to a stored process and registered in the Metadata server using the Management Console. From there the Information Map can be created using the stored process as the source data. Some of the same considerations that apply to PROC SQL views also apply to stored processes. For example, if one develops an implicit query, the Libname Engine will still be invoked, potentially missing the performance gains desired.

MEASURING PERFORMANCE

These techniques can improve the performance of the Information Maps. There is no clear cut “best practice” for each situation. The “best practice” is to measure the performance of the various solutions to determine which best

suits the given database being queried. If one develops PROC SQL views or stored processes, the performance can be viewed in the log file within an EG session. Multiple runs should be made with different filters as tests. If one includes the sastrace option in the initial code, the log will reveal what queries were actually sent to the database. From this, it is possible to confirm that SAS is not requesting full tables back to the SAS server. By taking the resolved query to the DBA group, they can validate other efficiencies such as invoking database indexes. They can also review if new indexes would be useful for these upcoming common requests to be generated by the new Information Map.

CONCLUSION

The SAS Intelligence Platform provides, to a broader base of users, access to data that previously would require knowledge of database structures and programming languages. Now through the use of Information Maps and client tools (both web-based and desktop), data can be queried readily without these skills. Because of the ease with which queries can be made, performance on the back-end systems is even more important. For small data sources, even complicated queries will respond quickly. As data sources grow in size and complexity, so do the response times.

Information Maps make data requests easier. Because Information Maps use the Libname engine, queries against foreign data sources can extend longer than expected. As shown previously, there are a few options to improve performance of the data requests generated from an Information Map. This included such RDBMS techniques as creating database views and stored procedures. Since modifications to corporate RDBMSs aren't always practical, SAS options were explored.

When looking at the two SAS options to improve query performance, each has its merits. SAS SQL views (using an explicit SQL pass through) are easier to code, validate and implement, but are limited to just SQL operations. Stored processes offer far greater flexibility and power but take more care to create and implement. The database structure and desired output will help to determine which approach is best.

Another point to consider when exploring options to improve performance of the Information Map, is the database structure contained within the RDBMS. In some cases, modifications to the database structure can be made to more closely match the access requirements. Different database structures are suited for different uses. Transactional databases tend to be formed differently than Analytic databases. Even the requirements for Business Intelligence applications are different than mainstream analytics. By working with the business users, analysts, programmers, and DBAs one can usually find a structure that will meet the performance requirements.

REFERENCES

Using SAS® Information Map Studio to Create Information Maps: Course Notes, Course code SBIIMS, prepared 01Mar2007, ISBN 978-1-59994-348-0

SAS® Information Map Studio 3.1: Creating Your First Information Map. Available on the SAS web site at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/infomap_create_9683.pdf

Base SAS®: Guide to Information Maps. Available on the SAS web site at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_infomap_9951.pdf

SAS® Information Map Studio 3.1: Tips and Techniques. Available on the SAS website at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/infomap31_tips_9310.pdf

ACKNOWLEDGMENTS

The author would like to thank the members of his staff that answered technical questions and kept the SAS servers running while he tested many complex queries for this paper.

RECOMMENDED READING

SAS® 9.1.3 Metadata LIBNAME Engine: User's Guide (Second Edition). Available on the SAS website at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_libnameug_9302.pdf

SAS® 9.1.3 Language Reference: Concepts (Third Edition). Available on the SAS website at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_lrconcept_9196.pdf

SAS/Access® 9.1.3 for Relational Databases: Reference (Fifth Edition). Available on the SAS website at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/access_rdbref_10385.pdf

CONTACT INFORMATION

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

Jerry A. High
Blue Cross and Blue Shield of Minnesota
3535 Blue Cross Road
Eagan, MN 55122
Work Phone: 651-662-8548
E-mail: jerry_a_high@bluecrossmn.com