Technical Paper

SAS® In-Database Processing

A Roadmap for Deeper Technical Integration with Database Management Systems

Table of Contents

Abstract

Significant trends in analytics and information management, along with regulatory pressures, demand new technologies that can turn increasing volumes of data into insight that can be exploited very rapidly. In addition, customer dynamics are changing more frequently so the need to develop and refresh analytic models that can be managed securely is accelerated.

SAS has announced a roadmap for enhanced partnership and deeper technical integration with database management systems (DBMS) providers that will enable SAS® Analytics and data transformation capabilities to be leveraged at the point of transaction rather than the typical extract, transform, and load (ETL) processes in use today. The SAS® In-Database technology, which allows SAS code to reside and execute inside a DBMS, streamlines the model management and scoring process from months to hours.

Introduction

The SAS In-Database technology described in this paper relates to a variety of environments, including grid, blade servers, and event management containers. The focus here is on its application to database management systems.

The goal of the SAS In-Database initiative is not only to achieve deeper technical integration with database providers, but to also extend this integration to a unique and differentiated value proposition that blends the best SAS data integration and analytics with the core strengths of databases.

This paper outlines the technology areas within the SAS® Intelligence Platform that are suitable for deep DBMS integration and relates these areas to higher-level process flows like data transformations, data mining, and model scoring implementations.

Key principles guide the implementation of SAS software that can run inside DBMS systems. Adhering to these principles ensures that the resulting software does the following:

- extends the DBMS SQL implementation to include key SAS business intelligence and analytic capabilities
- scales with the hardware and DBMS environment
- integrates with DBMS administration features
- maintains the reliability of the DBMS and integrity of the databases

Today, companies use SAS technologies for data integration, data summarization, data mining, forecasting, and statistical modeling to access information in DBMSs. Like all DBMS client applications, the SAS engine often must load and extract data over a network to and from the DBMS. This presents a series of challenges:

- **Network bottlenecks between SAS and the DBMS constrain access to large volumes of data**

  The best practice today is to read data into the SAS environment for processing. For highly repeatable processes, this might not be efficient because it takes time to transfer the data and resources are used to temporarily store in the SAS environment. In some cases, the results of the SAS processing must be transferred back to the DBMS for final storage, which further increases the cost. Addressing this challenge can result in improved resource utilization and enable companies to answer business questions more quickly.

- **The high cost of securing data across enterprise applications**

  More companies are designing solutions to address data governance and ensure regulatory compliance by reducing data movement. These companies rely on the security and auditing features of the DBMS to ensure that regulated data is not improperly accessed during automated business processes. Requiring large amounts of detail data to be extracted from the DBMS for processing by SAS raises the complexity of maintaining regulatory compliance. This can be particularly challenging for data preparation processes. By re-engineering SAS components to run inside the DBMS, elements of business processes can be executed in such a way that detail data can be summarized and analyzed without crossing the regulatory boundary created in the DBMS.

- **The high cost of converting a production SAS data transformation and model score to execute in a DBMS**

  Organizations using SAS software in conjunction with DBMS systems rely on the semantic power of SAS to develop processes that answer business questions. When they are confident that the model solution they have implemented in SAS is correct, they re-implement part or all of it using development tools that run natively inside their DBMS. They do this to address the challenges listed previously. Companies pay a high price both in terms of manpower and implementation time to remain competitive with market activity.

  By re-crafting elements of SAS so that they can run inside the DBMS and delivering appropriate tools to help manage these software artifacts, the distance between a SAS environment used for process development and a DBMS environment used for execution can be bridged. This reduces the effort and time required to implement these business processes in the DBMS environment.

SAS In-Database addresses all three of these challenges by moving the work closer to the data and increasing the intersection between the SAS and DBMS runtime environments.

Business Process Opportunities

Given the goals of SAS In-Database there are SAS technology areas that can be re-engineered to complete more processing inside the DBMS than they do today. Four data processing areas are frequently applied to large volumes of data stored in a typical DBMS: data preparation, modeling and scoring, analytics, and multi-step jobs (for example, multiple complex analysis steps combined with reporting).

Following is a brief description of these areas and a discussion of the SAS technology that can be re-engineered to deploy inside a DBMS.

Data Preparation

Data preparation includes the processes of cleansing, summarizing, and transforming detail and operational data into structures such as data warehouses and data marts that are suitable for BI reporting, data mining, scoring, and statistical processing.

Many DBMSs provide some support for data preparation, such as filtering, aggregation, denormalization, and so on. However, SAS uniquely supports operations such as sampling, variable binning, imputation, and variable selection, which are typically not as robust in the DBMS functionality.

Because the SAS language offers syntactic and semantic expressions that are not found in SQL, application programmers can implement more succinct, powerful data preparation programs than applications written in SQL. The opportunity is to enable this capability to run inside the DBMS.

Modeling and Scoring

Many enterprises that compete in today's market model business problems and opportunities and implement *model scoring*, which means applying the model to answer strategic questions. Applications for campaign planning, fraud detection, risk management, and customer retention require the processing of very large volumes of data that is stored in the DBMS.

Model training (development) typically involves many passes through the data. Once trained, the model score often depends on customer-specific information that is frequently under regulatory and security restrictions. In these cases, organizations typically standardize their regulatory compliance implementation on the DBMS. Extracting large amounts of such sensitive detail data from the DBMS raises complexity and increases compliance risks.

Analytics

Many organizations that use detail data to analyze and predict trends already rely on SAS to provide highly accurate predictive, optimization, and time series forecasting models. To boost performance and flexibility, the data is often transferred over a network from the DBMS into a SAS data set to perform these analyses. Performing these analytic processes on the data from inside the DBMS allows the large volume of data involved in these analytics to remain stationary, thus saving valuable processing time.

Multi-step SAS® Jobs

Data preparation, modeling and scoring, and data analysis processes can be independent processes or combined into complex multi-step jobs or services that can be run in batch, as stored processes, or as dynamic code streams. When some or all of the processes in these complex jobs run inside the DBMS, valuable time is saved and network bottlenecks are reduced. These jobs can also include reporting steps that use SAS reporting and graphics technologies. Web browsers, Microsoft Office applications, and other business applications can be used to deliver SAS jobs to the business analyst's desktop.

SAS® In-Database Technologies

The following SAS technology areas yield significant acceleration when they are re-engineered to run inside a DBMS:

- SAS functions, SAS formats, the SAS DATA step language, and many *SAS procedures* (which are software modules that encapsulate a related set of functions—similar to the concept of an SQL stored procedure) are used in data preparation processes. These processes filter, aggregate, denormalize, and sample large tables in a variety of business processes. Of particular interest, with respect to running SAS inside the database, is the process of building analytic data marts, which are commonly built by analyzing tables that use the same or similar code repeatedly while changing only a few options or input parameters. The ability to perform these tasks from inside the DBMS significantly reduces bottlenecks that result from moving the data over a network and fosters a single environment for collaborative SAS processing by multiple SAS programmers.

- Predictive and descriptive model training typically requires many passes over the prepared, denormalized data. SAS analytic and statistical functions as well as SAS procedures that execute repeatedly in model training can be moved inside the database. SAS® Enterprise Miner™ procedures that are candidates for inclusion are DMDB (Data Mining Data Base), DMREG (Regression), NEURAL (Neural Network), DMVQ (Clustering), and others. Possible SAS/STAT® software procedures include LOGISTIC, CATMOD, GENMOD, GLM, and many more.

- Scoring models (those that are trained and which produce appropriate scoring functions) will be saved in the database and can execute directly in the database using a library of SAS data manipulation and raw statistical functions. SAS models can also be coded in a new SAS language that is under development, SAS TSPL which can execute inside the database. In some cases, raw C code that can be compiled by the database's tools and executed inside the database is used for scoring.

- All of these programming elements can be combined in multi-step SAS jobs that implement complex processes. These jobs run all or in part inside the database. The jobs can be combined with SAS reporting capabilities to deliver analysis, business trends, and forecasting insights directly to the interested parties throughout an enterprise. These jobs, which implement complex processes, can be run repeatedly on one or more data tables.

In all of these cases, it is advantageous to execute the code that manipulates the data as close to the data as possible to minimize data movement and keep the detail data inside the boundaries of the DBMS that manages it.

The subsequent examples clarify these points.

The following SAS code example uses a SAS format and the SAS SQL procedure to select a subset of DBMS table data. A SAS format, $region, is used here to group rows of the dbms.sales table into discrete sales regions. The format is used in an SQL GROUP BY clause.

```
proc sql;
   select zip, sum( sales ) as sale_sum from dbms.sales
      group by put(zip, $region.);
quit;
```

The SAS PUT function is used to perform the actual formatting. This programming pattern occurs frequently in SAS code. Today, all the rows of dbms.sales must be extracted from the DBMS and transferred to SAS in order to transform zip codes into regions and complete the grouping operation. However, if the execution of the PUT function that uses the $region format occurs inside the DBMS environment, only the aggregate result set is returned to the SAS environment.

The following data preparation scenario uses the SAS FREQ procedure. PROC FREQ tabulates the frequency of specified data values that occur in a database table. Here PROC FREQ tabulates the percentages of males and females in a table called “students.”

```
proc freq data=students;
   table gender;
run;
```

It is possible to express part of this tabulation as an ANSI SQL statement that can be executed by the DBMS. PROC FREQ can push such an SQL statement to the DBMS and extract just the aggregate results, rather than extracting the entire student data table as input to PROC FREQ. By processing inside the DBMS and pulling just the result set into the SAS session, the amount of data that is pulled from the database is reduced to only a single table that contains one variable (“gender”) and values for only the count, percentage, cumulative count, and cumulative frequency of males and females in the student body.

Re-engineering for SAS® In-Database

To implement SAS In-Database processing, specific SAS components, including SAS procedures such as PROC FREQ and other SAS software components, such as functions and formats, will be re-engineered to invoke SAS subsystems that are integrated into the database. These subsystems are implemented using the SAS Threaded Kernel (TK) framework, which provides a lightweight, low-level programming environment that can be hosted on most hardware environments that run popular operating systems such as Windows, UNIX, Linux, and z/OS. This enables many parts of complex, multi-step SAS jobs to be executed inside the database with little or no data movement.

For the sake of this discussion, suppose the following TK subsystems are needed to support the re-engineered SAS processes:

- TKFORMAT—components in this subsystem implement formats supplied by SAS as well as custom formats created with PROC FORMAT. As discussed previously, SAS formats provide a powerful way to group individual data values or filter data using an SQL WHERE clause.

- TKFUNCTION—components in this subsystem implement SAS DATA step language functions and CALL routines that are used in data preparation, data transformation, and model training.

- TKSCIENCE—components in this category implement SAS Analytics capabilities. These include select SAS procedural computations from Base SAS®, SAS/STAT® software, and other SAS analytical products, which provide the most sophisticated and up-to-date analytic techniques, methodologies, and practices.

- Table Server Programming Language (TSPL)—components in this subsystem implement a derivative of the SAS DATA step programming language. TSPL is a new programming language (with syntax like SAS DATA step) that is introduced in SAS 9.2. TSPL includes the syntax and semantics needed to implement analytic models and scores, including those created using SAS Enterprise Miner.

Because these subsystems are implemented in SAS TK, they can be installed inside the DBMSs offered by the major DBMS vendors. They are then available to perform some or all of the processing requested by the client. For example, when SAS procedures push down SQL, or when some other SAS process is requested from a client, the processing occurs inside the DBMS where the subsystems are deployed. Appropriate output is returned to the calling client, a SAS session, for example, thereby reducing data movement.

In a previous example, we saw that PROC FREQ can push an SQL statement to the DBMS to perform initial processing of the input table inside the database and only extract the summarized results. SAS analytic procedures such as PROC REG can invoke the TKFUNCTIONS and TKSCIENCE subsystems inside the DBMS. The result is that these sophisticated statistical algorithms execute in the same process context that manages the data. There is no context "hop" and much less data movement is necessary to obtain a result.

More information about how SAS procedures can redistribute work is presented later in this paper.

SAS® In-Database Architecture

Understanding the general DBMS architecture helps to simplify the concepts around this new architecture.

Commercial database management systems can be thought of as a collection of processing nodes. These nodes can map directly onto blade hardware architecture, an SMP computer, or onto a pure software system that maps less directly to a standard hardware configuration. Regardless, most DBMS systems can be conceptualized as some combination of three abstract node types: SQL nodes, data nodes and head nodes.

- **An SQL node is the client's gateway to the data in the DBMS.**

  SQL commands provide a host of functionality to administer and monitor the DBMS environment, define and alter database structures, and queries to retrieve the data in some form.

  Stated simply, the SQL node receives SQL statements from the client and parses them to form execution plans that will do the work on the specified data and return the desired results to the client. In our conceptual DBMS architecture, there is only one SQL node.

- **Data nodes manage the physical data maintained by the DBMS.**

  In our conceptual DBMS model, there are many data nodes. The physical data is partitioned among the data nodes in a way that is optimized for the specific DBMS. Once the SQL node has formulated the execution plan, the plan is routed to the data nodes that manage the data partitions targeted by the SQL statement.

  Each data node performs the execution plan it is given and forms a result set. The result set from each data node is returned to the SQL node, where it is merged with the result set from the other data nodes. The final merged result set is then returned to the client. By distributing work this way across the available data nodes, work can be scaled inside a DBMS as data volumes grow. Increasing DBMS data capacity becomes a matter of increasing the number of data nodes.

- **Head nodes are specialized software processing environments inside the DBMS.**

  Head nodes are used to host applications that may not typically be thought of as part of database management systems. Head nodes extend the functional capabilities of the DBMS to include sophisticated and specialized software capabilities that might otherwise be beyond the domain of commodity DBMS features. Examples are Web application servers and business intelligence solutions.

  Head nodes are often implemented using discrete hardware and operating system configurations (for example, an Intel processor running Windows XP). These systems might be installed directly in the DBMS chassis or as a separate computer system that is directly connected to the DBMS chassis via a high-speed network connection. The result is a processing node that is tightly coupled to the DBMS SQL processing environment. A head node can send commands to the SQL node and vice versa. Also, head nodes can be managed and monitored using the DBMS administration tools.

The concept of an SQL user-defined function (UDF) is important for understanding SAS In-Database processing because it is the very mechanism that enables work from SAS programs to be executed inside the DBMS. A UDF is simply a packaged routine that can be invoked from SQL statements. Most DBMSs provide a framework for packaging code into executable modules, storing them inside the database, and making them available to clients of the DBMS. In short, UDFs make it possible to extend the SQL semantics implemented by the DBMS. More information about how UDFs implement SAS In-Database processing is presented later on in this paper.

Running SAS® Subsystems Inside

The following examples use the generalized three-node architecture with UDF support to show how the SAS subsystems can be run inside the DBMS. SAS formats are invoked by SAS code with the SAS PUT function. User-written SAS formats are created by the SAS FORMAT procedure.

The SAS PUT function takes two input parameters: the value to be formatted and name of the SAS format to be used. The SAS PUT function is expressed as:

```
put(value, format);
```

The output is the formatted result.

Most DBMSs support SQL UDFs that take a similar form, that is, a function name and some input parameters. These are called *scalar UDFs* in most DBMSs. Now suppose the TKFORMAT subsystem (described earlier) is installed inside our DBMS. A scalar UDF, named SAS_PUT, is defined in the DBMS that calls into the TKFORMAT subsystem to access the SAS functionality that enables a SAS format to be used to qualify data.

The user-defined “SAS_PUT” function takes the same input parameters and returns the same output value as the PUT function in SAS code. And because the TKFORMAT subsystem is installed inside the DBMS, a client can send down SQL syntax like the following:

```
select * from mytable where sas_put(zip, $region.) = “Southeast”
```

Figure 1 extends our DBMS architecture model to further explain how SAS formats deployed inside the DBMS relate to the DBMS client and other processing nodes.

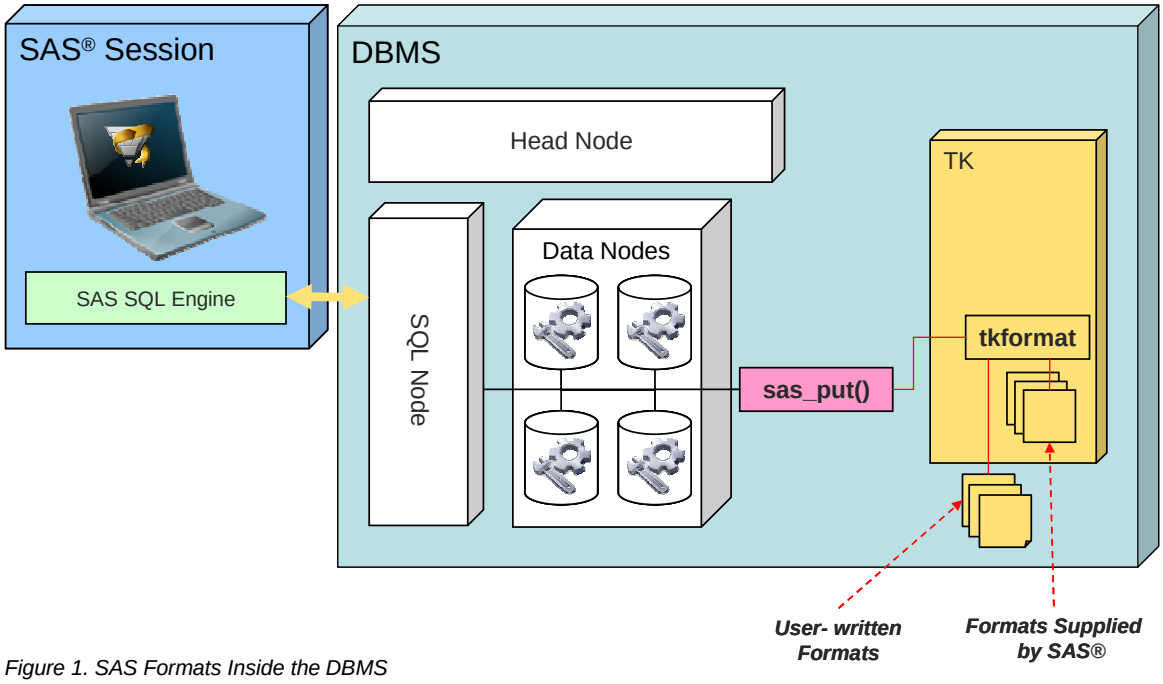

Figure 1. SAS Formats Inside the DBMS

For example, assume a SAS session executes the SAS SQL procedure with the following statements:

```
proc sql;
   select * from DB.T where put(zip, $region.)  = “Southeast”;
quit;
```

In this case, DB is a schema in the DBMS and T is a table in DB. PROC SQL realizes that the SQL is to be executed against a DBMS and passes the SQL to the underlying SAS engine. Then the following events take place:

1. The SAS engine finds the PUT function in the SQL expression and turns it into a SAS_PUT UDF reference.

2. The SAS engine passes the SQL statement to the SQL node in the DBMS.

3. The SQL node parses the SQL statement and forms an execution plan; this plan includes invoking the SAS_PUT UDF.

4. The execution plan is sent to each of the data nodes that manages table T.

5. As the execution plan evaluates the WHERE clause on each of the data nodes, it invokes SAS_PUT for each row processed.

6. Each data node forms a result set of rows that meet the WHERE criteria from T for the data partitions it manages. Each data node’s result set is returned to the SQL node.

7. The SQL node merges all the result sets into a single result set that is returned to the SAS session where PROC SQL is executing.

Note the three significant benefits of this process:

- The SAS format is evaluated in the same context that manages the physical data, that is, inside the DBMS.
- The work required to evaluate the SAS format for each row of table T is distributed across DBMS processing resources (data nodes) by the DBMS itself.
- Only the rows in the DBMS table that match the WHERE clause criteria are extracted from the DBMS.

This is all possible because the TKFORMAT subsystem is installed inside the DBMS. In fact, TKFORMAT supports more than just SAS formats that are defined using PROC FORMAT. SAS ships with hundreds of formats already defined. Formats supplied by SAS are deployed into the DBMS as a small set of runtime modules. Formats created by PROC FORMAT, in contrast, can export these user-written format definitions as XML.

These XML streams can then be stored inside the DBMS as database objects. The TKFORMAT components know how to identify a user-written format and find its XML definition inside the DBMS. The details of how these XML definitions are created, copied in the DBMS, and subsequently managed there as database objects, is beyond the scope of this paper. But the process is defined and can be implemented in such a way that new user-written SAS formats can be created and deployed with simple SAS tools. And the XML streams stored in the DBMS can be managed like any other database object.

Because the processing of both user-written formats and formats that are supplied by SAS is encapsulated into a single subsystem (namely TKFORMAT), only a single UDF, the SAS_PUT UDF, is required to make all of these formats available to the SQL and data nodes in the DBMS.

Consider these DBMS UDFs that map to multiple SAS functions of similar type to be “universal UDFs.” The SAS_PUT and other “universal UDFs” discussed below make diverse SAS functionality available to the DBMS through only a few SQL language extensions.

SAS® Model Scoring and Data Transformations Inside

Scoring models and data transformations are deployed inside the DBMS in much the same way as SAS formats. For example, mathematical functions used in data transformations are deployed in the TKFUNCTION and TKSCIENCE subsystems mentioned earlier. Scoring is implemented using the new SAS language TSPL. Figure 2 shows the architectural relationships between the DBMS nodes and the TSPL subsystem deployed inside the DBMS. This relationship is similar to the one in Figure 1, which shows SAS formats and the TKFORMAT subsystem.

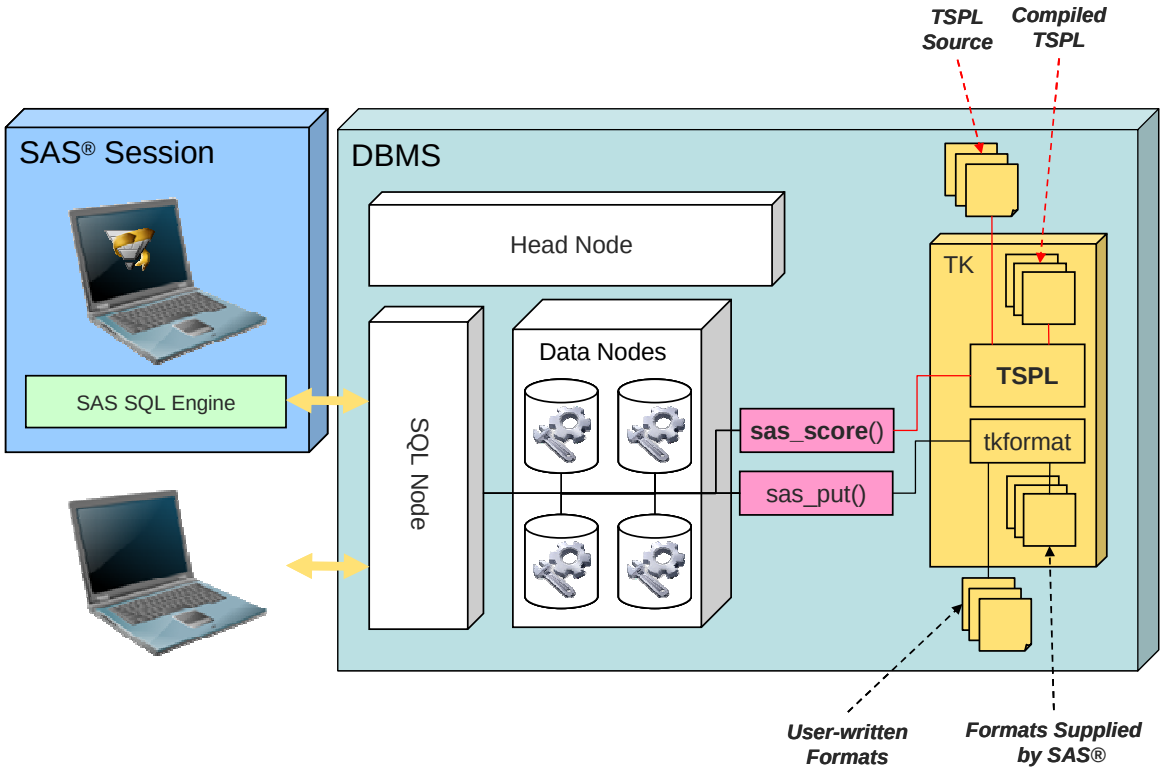

Figure 2. SAS Scoring Inside the DBMS

The universal UDF in this case is called SAS_SCORE. This UDF is slightly different in that it can take a table or result set name as input, and the output is a result set. Call this a table UDF (compared to a scalar UDF like SAS_PUT UDF, which outputs a single value). The input parameter list of the SAS_SCORE UDF varies according to the DBMS and the model being executed. Regardless, the input parameter list of SAS_SCORE can be broken into two sets: the name of the scoring function to be run and the table or result set containing the rows to score. The model that is executed determines the structure of the output result set of the SAS_SCORE UDF.

Two forms of TSPL programs are shown in Figure 2: TSPL source code and compiled TSPL. TSPL source is similar to any other SAS code stream; it is a character stream comprising TSPL source statements. Each module of TSPL source code encapsulates a complete scoring model that is created using SAS Enterprise Miner tools. (The details of exactly how this is done are beyond the scope of this paper.) TSPL source modules are stored as character streams in a database. This enables them to be managed like any other database object.

Compiled TSPL programs are created by compiling a TSPL source module that is generated by SAS Enterprise Miner. They do not contain TSPL source statements. Compiled TSPL programs are shared object libraries that are stored inside the DBMS. SAS provides a collection of tools to simplify and automate the process of converting a TSPL source program to compiled TSPL. Some of these tools integrate with specific DBMS vendor tools. When SAS Enterprise Miner integrates with the SAS In-Database components discussed here, the process of deploying a scoring model from SAS Enterprise Miner to a DBMS will be highly automated. SAS Model Manager will also support registering the champion TSPL model with the universal SAS_SCORE UDF for more automated scoring.

While there are two forms of TSPL programs that can be deployed inside a DBMS, only one form is used by any particular DBMS. Exactly which is used depends on the implementation details of the internal architecture in the DBMS. TSPL source programs are used in DBMSs that provide a rich UDF runtime environment with the system functionality that

dynamic compilation engines require, such as global memory services and file I/O. Compiled TSPL programs are used in DBMSs that provide a lightweight UDF runtime environment. The tradeoff between these two approaches is ease of deployment with TSPL source and lower execution overhead with complied TSPL.

Aside from the particular aspects of the SAS_SCORE UDF and the TSPL subsystem described in the preceding sections, the work flow from a SAS client invoking a scoring model that executes inside the DBMS is very similar to the SAS format case examined earlier in this paper. That is:

1. An SQL expression that references the SAS_SCORE UDF is submitted to the DBMS. If the SQL is submitted from SAS, it is submitted using the SAS SQL procedure. If the SQL is submitted from a database client, then the client's DBMS API is used.

2. The SQL node in the DBMS receives the SQL statement, parses it, and forms an execution plan; this plan includes invoking the SAS_SCORE UDF.

3. The execution plan is sent to the data nodes in the DBMS.

4. As each data node processes the execution plan, it invokes SAS_SCORE for each row processed.

5. Each data node forms a result set of the rows processed.

6. Each data node's result set is returned to the SQL node.

7. The SQL node merges all the result sets into a single result set that is returned to the calling client or stored in the DBMS.

The benefits of this process are the same as for the SAS_PUT UDF process described earlier; that is, the SAS_SCORE evaluation occurs in the DBMS where the data is stored. The evaluation work is distributed across the DBMS processing resources by the DBMS and only a single result set is returned to the client.

SAS® Analytics Inside

SAS provides an extensive array of analytic algorithms and methodologies to today's market. In SAS, these analytic algorithms are encapsulated into SAS procedures (or PROCs) as mentioned earlier. The following are some examples of SAS procedures:

- PROC FREQ produces one-way to n-way frequency and cross-tabulation tables.

- PROC SUMMARY provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.

- PROC LOGISTIC performs conditional logistic regression analysis for case-control studies and exact conditional logistic regression analysis.

- PROC DMDB creates a data mining database (DMDB) from an input data source. It also compiles and computes metadata information about the input data based on variable roles and stores it in a metadata catalog.

In traditional SAS programs, PROCS are called as part of a workflow that usually includes SAS DATA steps along with PROC steps, which invoke SAS procedures. For example, a very basic SAS program includes a DATA step that creates the SAS data set from raw data files and a PROC step that invokes an appropriate SAS procedure to process the SAS data set.

A PROC invocation typically takes one or more tables as input and produces a new table as output. Various programming options and control statements can also be specified on the PROC invocation. These dictate how the PROC does its work within the boundaries of its particular analytic role. Key SAS analytic PROCS will be re-engineered so that some of the work is delegated to run inside the DBMS. The part of a PROC that runs inside the DBMS varies from PROC to PROC and depends on what the PROC does.

There are two general-use cases that dictate how much work is delegated to the DBMS. In the first scenario, data is summarized or transformed in a way that can be expressed using ANSI SQL. The PROC generates an SQL expression that is passed to the SAS engine which, in turn, passes the SQL to the DBMS for execution.

The following example uses PROC FREQ:

```
proc freq data=dbms.students;
   table age * sex;
run;
```

In this example, PROC FREQ performs a two-way frequency tabulation on the age and sex columns found in the dbms.students table. When a SAS program invokes this step, PROC FREQ generates SQL that collects univariate statistics (count, min, and max) on the age and sex columns. The generated SQL is passed down to the DBMS SQL node for evaluation. Of additional interest is the fact that SAS formats are used in the generated SQL expression. As already noted, SAS formats execute inside the DBMS using the embedded SAS subsystem TKFORMAT.

When processing is complete inside the DBMS, the result set of the SQL expression is returned to the SAS session and PROC FREQ continues processing the result set to form the final output table as a SAS data set or some other SAS format. Not all PROCS behave this way. But in general, depending on the particular PROC and the options and statements that are specified for the PROC (which affect the flow of control), one of the following happens:

- The SQL result set is returned to the SAS session context for further analysis by the PROC or by other PROCS.
- The SQL result set needs no further processing and is either stored in a database table or returned to the SAS session context for processing or stored as a SAS data set.
- The result set represents an intermediate stage of the analyzed data, which is then temporarily stored in the DBMS for further processing by the procedure from inside the database.

In each of these cases, the amount of work delegated to the DBMS is different. For the last case, the procedure delegates more work to the DBMS; for example, further processing is performed on the temporary table to create yet another temporary table or the final result set. The DBMS then either returns the final result set to the SAS session that is running the procedure or stores it in the database as a permanent table.

The other general-use case that affects the amount of work that is delegated to the DBMS is when the data manipulation that a SAS PROC needs to do cannot be expressed using ANSI SQL. SAS PROCS that implement specialized analytic functions usually use lower-level programming languages, such as C, to express the associated algorithms more effectively than SQL. Therefore, they are re-engineered to generate SQL that calls into the SAS subsystem, for example the TKSCIENCE subsystem, using a UDF.

PROC NEURAL is an example of a procedure that uses C algorithms instead of generating pure ANSI SQL. PROC NEURAL fits a flexible nonlinear response model for estimating values and posterior probabilities; PROC NEURAL can be used to predict such things as which customers will respond to a marketing campaign, which patients will respond to medical treatments, and to estimate time to failure in manufactured components.

The following is a PROC NEURAL step with many options specified for controlling the execution:

```
proc neural data=external.table  dmdbcat=d;
   input age income kids cars houses / level=interval id=demographcis ;
   input gender code1 - code500 / level=binary id=codes ;
   target promotionresponse / level=binary id=target ;
   hidden h1 number=3 activation=TANH ;
   hidden h2 number=3 activation=SINE ;
   connect demographics h1 ;
   connect codes h1 ;
   connect demographics h2 ;
   connect codes h2 ;
   connect h1 h2 ;
   connect h1 target ;
   connect h2 target ;
   initial ;
   train maxiter=100 estiter=1 outest=est ;
   save network=work.neural.net ;
   code catalog=work.neural.score ;
run;
```

The INPUT, TARGET, HIDDEN, and CONNECT statements define the model function; the INITIAL statement creates a starting set of approximately 1,034 parameter estimates. The TRAIN statement causes the model function evaluation to be repeated up to 100 times, and each evaluation requires a full pass through the data. The result of one iteration of the model function causes a new set of parameter estimates to be computed as input to the next model function iteration. By using TKSCIENCE to execute the model function inside the DBMS, processing cost and data governance concerns related to duplicating large data sets are avoided.

SAS® Servers Inside

The fourth SAS Intelligence Platform component targeted for SAS In-Database processing is the set of SAS Intelligence Platform servers that run SAS programs (or jobs). These servers include the SAS® Workspace Server, the SAS® Stored Process Server, and the SAS® Metadata Server. All of these servers can be run inside the DBMS on a DBMS head node. The services can be bound to UDFs and DBMS stored procedures so that SAS programs can be invoked via SQL commands or invoked directly from SAS clients that are connected to the DBMS via a LAN or WAN. Figure 3 depicts the architecture when the SAS Stored Process Server is deployed on a DBMS head node.

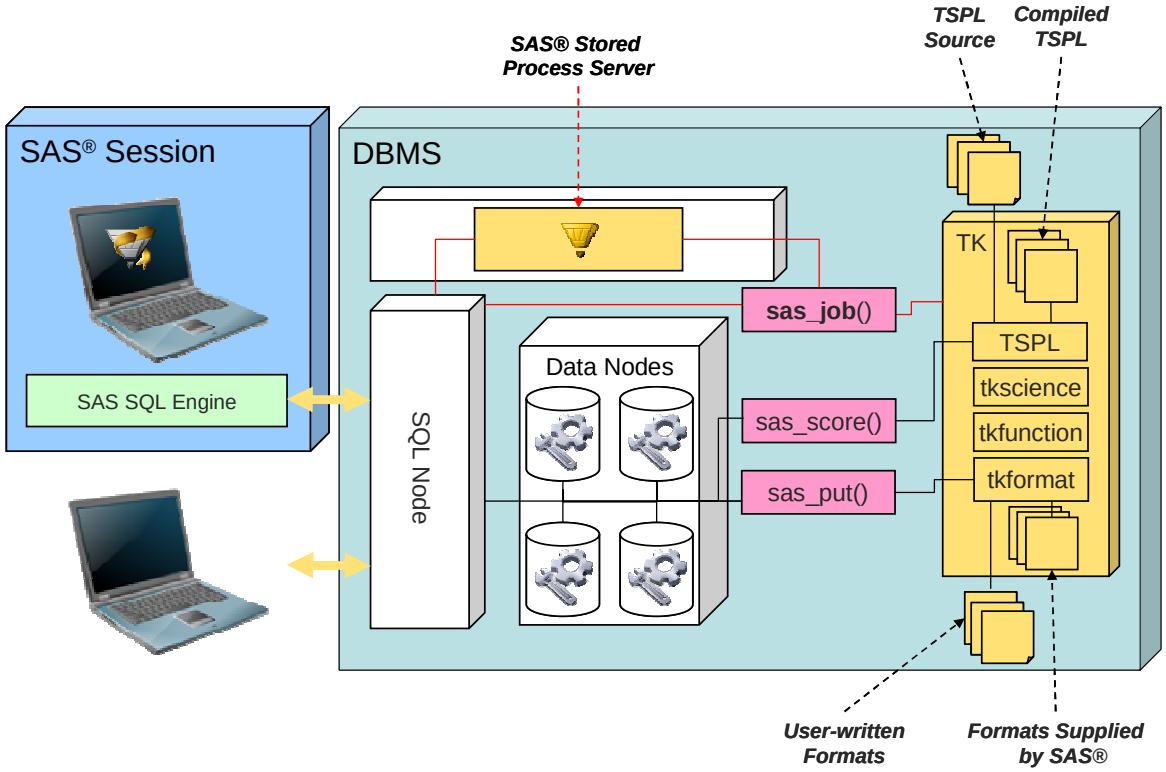

Figure 3. SAS Jobs Inside the DBMS

The universal UDF SAS_JOB represents a complex multi-step process that calls into all the SAS In-Database subsystems that can reside in the DBMS: formats (TKFORMAT subsystem), data transformation and model scoring (TKFUNCTIONS and TSPL subsystems), and analytics (TKSCIENCE subsystem).

Both SAS clients and DBMS clients can use the integrated SAS servers. A SAS client communicates directly with the SAS servers deployed on the DBMS head node. The SAS client can execute SAS jobs inside the DBMS by sending commands to the SAS servers that are running on the DBMS head nodes. When these SAS jobs reference SAS formats, scoring models, or SAS analytic procedures deployed in subsystems inside the DBMS, they execute inside on the data nodes as previously outlined.

Alternatively, a DBMS client can submit DBMS commands that reference SQL stored procedures. These stored procedures translate the request into commands that the SAS servers that are running on the DBMS head node execute. Again, when these SAS jobs reference SAS formats, scoring models, or procedures that run inside the DBMS, they execute on the DBMS data nodes.

The capability for SAS servers to run inside the DBMS results in a very powerful and flexible environment. SAS solutions are built on top of these servers. These solutions can be deployed on the DBMS head node along with the SAS Intelligence Platform servers. This creates a single, tightly coupled package for customers who choose to standardize a SAS solution on a partner's DBMS. Furthermore, this architecture enables SAS servers and SAS solutions to be administered and monitored from within the DBMS administration and monitoring domain, providing integrated system administration. By bringing all the SAS In-Database components into a single integrated system, performance is significantly increased.

SAS® In-Database Implementation Principles

When SAS In-Database processing is implemented for specific DBMSs, the details about how to execute them will be provided in additional technology papers. The focus of this paper is to describe the SAS roadmap for re-engineering SAS technology components to run inside the DBMS. Following is a summary of the key implementation principles exemplified throughout this paper. These principles are employed for all database management systems:

1. **Each SAS In-Database implementation is specific to a DBMS.**

   Each implementation treats the DBMS like any other hardware or operating system platform. SAS programs that are deployed as compiled modules, either as part of the SAS In-Database installation or site-specific customizations, are created using the compilers that are native to the hardware and operating system hosting the DBMS. Using the appropriate compilers enables the subsystems like TKFORMAT, TKFUNCTION, TKSCIENCE, and TSPL to be delivered as natively optimized shared object libraries.

2. **Maximize the manageability of SAS artifacts in the DBMS.**

   Components like user-written PROC FORMAT definitions, UDF and stored procedure definitions, TSPL source modules, and model scoring and data transformation metadata are persisted inside the DBMS as native database objects. This enables these SAS artifacts to be managed like any other database object using the administration tools of the DBMS. Adhering to this implementation principle ensures the integration goals of the SAS In-Database initiative.

3. **Use programming strategies that provide scalability.**

   The SAS In-Database implementations use a programming strategy that enables TKFORMAT, TKFUNCTION, TKSCIENCE, and TSPL subsystem functionality to operate on one row of data independently of any other rows, resulting in “shared nothing” scalability. Work done by all of these SAS In-Database modules is performed the same way on any data node in the DBMS, regardless of which partition of the data that the data node is managing. Consequently, the modules re-engineered for SAS In-Database processing scale seamlessly as data nodes are added to the DBMS. This implementation choice supports our initiative’s performance goals.

4. **Extend SQL capabilities by using SQL UDFs and stored processes.**

   The features and functions of SAS In-Database artifacts that are stored in the DBMS are available to the SQL node (and subsequently DBMS clients) with user-defined functions and triggers that launch stored procedures. In effect, this extends the SQL semantics of the DBMS such that functionality that was previously only available in a SAS session is now directly available via the database management system’s SQL processor for execution inside the DBMS.

5. **All SAS components deployed inside the DBMS must be implemented in such a way that they do not compromise the database management system’s reliability and integrity.**

   Organizations rely on database management systems to deliver 24x7 availability and maintain the correctness of the data stored in the DBMS. The SAS In-Database components are implemented in such a way that an error during execution does not disrupt DBMS service or compromise the integrity of the data.

SAS is committed to adhering to these principles when delivering SAS In-Database implementations for specific database management systems. Our commitment ensures that the resulting offering meets the quality expectations of all our valued customers.

Conclusion

SAS In-Database aims to create a tighter integration with commercial DBMSs to accelerate critical business processing. This is achieved with two key strategies: porting the implementation of select SAS components to the DBMS host platform as SAS TK subsystems and enabling SAS clients to call into the ported subsystems and dynamically push generated SQL directly to the DBMS for execution.

Among the set of SAS Intelligence Platform components targeted for SAS In-Database are:

- SAS formats
- Base SAS summary procedures
- SAS analytic procedures and functions
- Model scoring expressed in Table Server Programming Language (TSPL)
- SAS servers, such as the SAS Stored Process Server

Components that are deployed to run inside the DBMS are integrated with the database metadata so that they are available to be administered and monitored like any other database object.

The effort required to move processes implemented in SAS (as the customer's standard development environment) to the DBMS production environment is significantly reduced. The cost of managing both development and execution environments is reduced due to the tight integration between SAS and the DBMS. Hardware resources are conserved by reducing data movement and duplication. The processes that transform and summarize detail data are more efficient. Automated business processes are optimized, which enables companies to answer business questions more quickly.

All of these factors add value to the enterprise by

- Lowering total cost of ownership
- Scaling SAS processes seamlessly with the size of the database
- Increasing performance and reducing data movement
- Optimizing resource utilization across the end-to-end analytic/warehousing environment
- Enabling complex transformations to occur at the point of the transaction
- Streamlining data discovery and model development processes
- Supporting corporate governance processes
- Increasing process robustness by reducing implementation restrictions.

The mission for SAS has always been to provide our customers better answers faster. Adding technologies and architecture that accelerate the SAS Intelligence Platform functionality, such as SAS In-Database processing, is simply the next step in the continuation of 30 years of SAS leadership in innovation in data integration and analytics.