

Replacing Oracle with the SAS Scalable Performance Data Server: Improved Performance, Enhanced Functionality, and Increased Efficiency.

Jay L. Stevens, The Prudential Bank

Abstract

Although the SAS Institute has been very successful in building high functioning gateways to external database repositories, it is almost impossible to take full advantage of the SAS system through these interfaces. Many experienced SAS users have long recognized that SAS is vastly more efficient and functional when used against SAS datasets. Partially due to their origin as transaction processing systems, most of the existing relational databases are overly complex and inefficient when used as the repository for a data warehouse or datamart. With the advent of the SAS Scalable Performance Data Server, a viable high-performance alternative to Oracle and the other Relational Database Management Systems (RDBMS) is available to the enterprise. This paper will review the process and advantages of migrating from Oracle to SAS/SPDS including:

- Overcoming Obstacles to Migration
- Performance & Productivity Gains
- Data Loading and Transformation Efficiencies
- Security and Auditing Considerations

Introduction/Background

In 1995, the Prudential Bank's Credit Card Portfolio Analysis group began to use the SAS system as its primary analysis tool (migrating from a combination of mainframe-based SQL on DB2 and Windows-based SPSS). This credit card

datamart was planned as an in-house initiative to improve performance, flexibility, and control. The original configuration called for a Sun Microsystems SPARC 1000-E Server (with approximately 60 GB of usable disk capacity after RAID 0+1 implementation, 4-60 MHz SuperSPARC processors, and 1 GB of RAM). The Sun server ran the Oracle 7.1.3 RDBMS and version 6.12 of the SAS System with SAS/Access for Oracle using the Solaris 2.4 operating system. For ease of conversion, phase 1 of the data mart called for a complete replication of the current DB2 database structure into the Oracle system.

System users, members of the Portfolio Analysis and Information Support departments, ran SAS directly on the server via X-windows terminal emulation. Information Support programmers used SAS/Access SQL pass-through views and queries in combination with Procs REPORT, PRINT, TABULATE, and DATA STEP to complete reporting requests. Portfolio Analysts used SQL pass-through to extract data from the Oracle tables into SAS datasets for data transformation / manipulation and analysis.

Why Oracle?

In planning our original implementation the IT department suggested that SAS be used as an end-to-end solution, to be used as both the repository for the data as well as the analytical and reporting tools. As the manager of the Portfolio Analysis group (and its primary database user) and project manager for the data mart project, I decided against a complete SAS solution for the following reasons:

- 1) At the time, SAS was unable to overcome the 2 GB file size limitation imposed by the UNIX operating system. Our largest table at the time was expected to reach over 3 GB in size. Oracle overcame this obstacle by partitioning data into multiple physical files and then united them into one logical unit.
- 2) SAS was, and still is, a single-threaded application and would not take full advantage of SMP (Symmetric Multiple Processing) hardware. This would be necessary for some of our larger reporting and analysis tasks. Oracle was able to take advantage of multiple processors through parallel processing.
- 3) The least significant. SAS was only able to support variable (column) names of 8 characters. The DB2 system that we were migrating from supported up to 32. The programming group felt that this might be a significant barrier to widespread acceptance of the system.

For these reasons, it was decided that we would use the Oracle RDBMS as the database repository and use SAS as the analysis and reporting tool.

SAS Preprocessing for Oracle

Because of the complexity of the operational source data structures (mainframe VSAM data files from the Bank's credit card processor) it was necessary to use SAS to preprocess and prepare the data for Oracle. Oracle's SQL*Loader product did not have facilities to read and write some of the mainframe data types used in the file.

SAS DATA STEP programs were used to read the credit card masterfile data from 3480 cartridge tapes. These SAS programs created multiple ASCII flat files that contained the data needed to load the Oracle tables.

Resource Hog, Performance Pig

Almost from the beginning we began to experience performance problems related to the Oracle data server. We immediately noticed that SAS was able to read the raw files from tape, perform conversions, and load the data to SAS datasets substantially faster than Oracle was able to load the tables directly from disk. Because of the ways in which Oracle preallocates and sets aside both memory and disk space, it reserved both for its exclusive use, consuming nearly a quarter of the available physical system memory.

The Problem with Oracle

In addition to these performance issues, the fact that the users needed to learn and use 2 different languages (SAS and the Oracle dialect of SQL) resulted in significant productivity impacts. These are common issues that all SAS/Access users have had to contend with at one time or another. These include the following:

- Because SAS procedures can not directly access the data, datasets must be filtered through SQL-passthru Views or must be read into SAS datasets (each of which adds additional processing time)
- There is limited support for debugging of SQL statements passed to Oracle via SAS/Access. (This often resulted in the programmer having to submit the query directly to Oracle from SQL*Plus in order to debug.)

In addition to the performance and productivity issues, we encountered a number of problems related to the relative complexity of Oracle and its origination as an OLTP (Online Transaction Processing) system. By default, the Oracle data server is tuned to perform as a transaction processing system. However, most of the overhead associated with full-relational functionality is unnecessary in a decision support environment.

A Complete SAS Solution

In 1996, the SAS Institute announced a joint development effort with Sun Microsystems for a high-performance parallel data server which would serve SAS datasets, the Scalable Performance Data Server, or SPDS. With SPDS, SAS had overcome all of our original objections to using the SAS system as a complete end-to-end data mart solution.

As soon as possible we signed up to beta-test the SPDS. Since that time we have moved from version 1.0 beta to production versions 1.0, 1.1, and 1.2. We are currently beta-testing SPDS version 2.0.

The SAS Scalable Performance Data Server resolves all but one of our original concerns regarding SAS. It supports file sizes greater than 2 GB through file partitioning. In addition, at every opportunity SPDS exploits SMP hardware to parallelize its data retrieval and indexing operations. The only other obstacle, the lack of support for long variable names, became a non-issue when the programming staff realized the other productivity gains that SPDS made possible.

Overcoming Obstacles to Migration

There are many obstacles to implementation of an Oracle to SPDS migration. These include:

- Concerns related to migration of data and processes.
- Concerns related to SPDS' stability as an enterprise-strength alternative to Oracle.
- Concerns related to the ongoing war between Relational DBMS fanatics and SAS dataset fanatics.

Process and Data Migration

The migration of processes and data includes the conversion of any control programs, conversion of load and transformation programs, process changes, and programming language adjustments for end-users.

Most of these issues were minimized by the fact that SAS already played a major role in our systems. Since SAS was used to read the data from tape and perform transformation and data type conversions, the transition from loading to Oracle to loading to SPDS was almost as simple as modifying a `DATA _NULL_` statement to read `DATA1.MAINDATA`. There was some additional "tweaking" to the original data step programs however these updates were minor and program conversion was completed and tested in less than 2 weeks.

Once the data was natively available to SAS via SPDS, the changes to programming practices were almost second nature. Eliminating Oracle eliminated the SQL pass-through coding step.

Security and Auditing Concerns

When we began discussions about replacing Oracle with SAS/SPDS, one of the first issues raised was regarding Security and Auditing. This was due primarily to a misunderstanding regarding the differences between SAS and SPDS.

Our operations are under nearly constant scrutiny from Internal Auditors, external auditors, Bank examiners from the OCC, the FDIC, etc. Thus, the regulatory environment is rather strict. SPDS was able to meet almost all of the security requirements (the only exceptions being the need to prohibit users from rotating between 2 different passwords and a requirement that the password contain at least 1 numeric).

The audit logs generated by SPDS provide detailed information about all server events: Failed passwords, system errors, User Logins and Logouts, etc. In addition, an extensive audit trail is generated which shows all actions taken on data tables under SPDS control. These logs are easily read into SAS datasets. SAS can then be used to generate the required monthly and quarterly Audit reports reviewing database access.

Relational Religion

As many supporters of end-to-end SAS solutions have learned over the years, a number of the arguments against SAS as a "database" are based on strongly held beliefs regarding the suitability of Relational Database Management Systems for data warehousing and decision support applications.

For our purposes and, I venture to guess, for most data marts in which SAS is the primary analysis and reporting tool, there is very little

added functionality that a "true Relational" system can offer over the SPDS. To the contrary, it is my assertion that full featured Relational Database systems such as Oracle are overly complex for decision support tasks and introduce unneeded overhead associated with those features.

In several postings to the SAS-L newsgroup, I solicited comments regarding the value of RDBMS over SPDS specifically for data warehousing applications. Some of the respondents cited the limitations of SAS (not SPDS) which have already been discussed in this paper (lack of support for SMP hardware, advanced indexing structures, etc.). The remaining comments highlighted the very same features of relational database systems that we viewed as unnecessary for decision support:

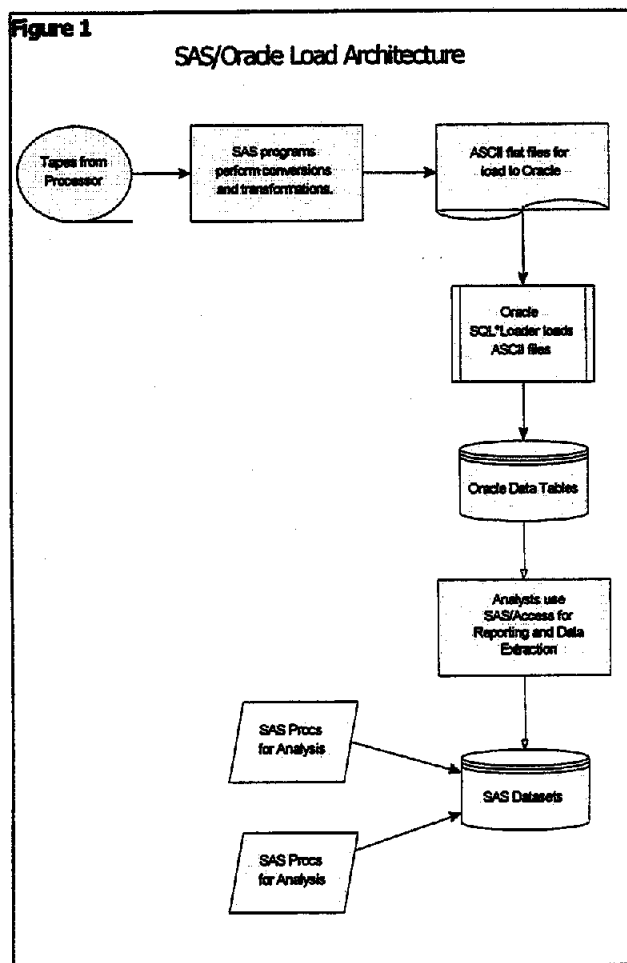
- Transaction Rollback capabilities.
- Referential Integrity controls such as cascading updates and deletes.
- Oracle's "ReDo" logging.

Performance

Faster Loading Processes

Because of Oracle's limited ability to convert EBCDIC mainframe data types, our process necessarily included SAS as a preprocessor to Oracle. SAS was used to read the data from 3480 cartridge tapes and convert the data to a format readable by Oracle. This included conversion from mainframe unsigned packed data, signed packed data, integer binary, hexadecimal, and regular EBCDIC text data as well as relatively complex transformations requiring the use of IF - THEN programming and DO loops, etc. It was impossible for Oracle alone to meet the business requirements for the

load process. A SAS _NULL_ data step program

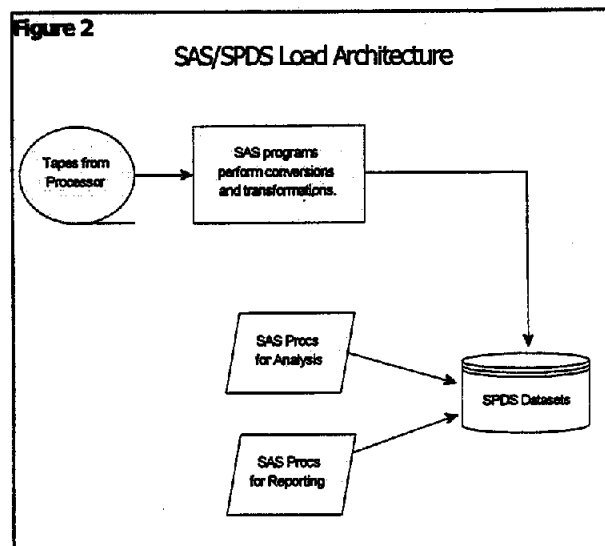


generated text flat files.

The ASCII flat files created by SAS were then loaded into Oracle using SQL*Loader's *direct path* load which writes database blocks directly without the use of SQL Insert statements. The Oracle load was the most time consuming segment of the load process. Total elapsed time for the entire automated process, which launched multiple instances of the SQL*Loader to take advantage of multiple processors, was 12 hours.

Once the data was loaded and indexed in Oracle, analysts submitted SAS code that would either extract data directly to SAS datasets or generate reports using Proc Report and SQL-passthru views. (See **Figure 1**)

The added step of writing data to flat files and then loading them to Oracle represented an additional step that was, with SPDS, unnecessary. Once SPDS was installed and tested, we were able to modify the load



routes. Instead of writing to ASCII flat files, the SAS program that performed the data conversion and transformations was easily able to output multiple SAS/SPDS datasets.

This new process resulted in a significantly faster load of the warehouse (total time to load and index for SPDS is approximately 8 hours, a 25% reduction over Oracle). A comparison of the two data loads is shown below. As can be seen in **Table 1** below, even though the volume and complexity of the work is much greater for SPDS, it is much more efficient than the Oracle process. The SPDS load is accomplished in half the time of the Oracle load.

Table 1

Data Load Comparison		
	Oracle Load Process	SAS/SPDS Load Process
Source File Type	ASCII Flat File (Text only)	EBCDIC flat file (with packed data)
Source File Size*	122,670,000 bytes	307,170,000 bytes
Data Conversion / Transformation	None	Convert EBCDIC character to ASCII character; convert Packed-Signed numeric, convert Packed-Unsigned numeric, hexadecimal and integer binary conversions; transform calculated fields from existing data.
# of Variables (Columns)	195	276
# of Records Loaded	90,000	90,000
Total Elapsed Time	14 min 04 seconds	7 min 12 seconds

* For Oracle, *Source File Size*, represents the size of the SAS-preprocessed 90,000 record ASCII flat file. For SPDS it represents the size of a 90,000 record extract from tape of the raw EBCDIC file. Both files are approximately 1/2 of one cartridge tape volume. Each month-end load spans approximately 5-8 tapes.

Cutting Edge Optimizations

In addition to a much simpler, faster, and more efficient data load, SPDS offered us a high performance alternative for data query and retrieval. Recent enhancements added to SPDS in version 2.0 have shown much promise. Specifically, the addition of hybrid segmented bitmap indexes (a SAS innovation) have resulted in significant performance gains for some queries.

The bitmap indexes make powerful use of the file partitioning of datasets within SPDS. When querying an SPDS dataset with a WHERE statement on a bitmap indexed variable, SPDS will do a preliminary scan of the index to determine which physical segments of the data file contain records meeting the WHERE criteria. So, instead of scanning the entire dataset (i.e. all physical segments), in many cases the server is able to only scan 10% of the physical segments

to find the matching record set. In addition, this physical partitioning enables SPDS to make use of parallel processes to scan each physical segment. This is illustrated in the benchmark shown below:

	Oracle w/ B-tree Index	SPDS w/ Bitmap Index
Select query with Indexed WHERE clause which returns a small % of total records.	2 minutes 13 seconds	11 seconds

Improved Efficiency & Productivity

One World, One Software

Under the SAS / Oracle architecture, the programmers and analysts had to essentially maintain two sets of code as well as managing two different programming languages. With SPDS they are able to write SAS code in order to accomplish all of their needs:

- Data manipulation and Transformation
- Reporting
- Statistical Analysis using SAS procedures.

Free to Use SAS

By using the SPDS as the data repository instead of Oracle we are able to take advantage of the full range of the SAS system. One of the simplest examples of this is the DATA Step. The Data Step provides simple conditional IF-THEN-ELSE logic. It is one of the most basic elements of SAS and all programming languages. Oracle's PL/SQL (Procedural Language SQL) and some Oracle-specific functions and extensions allow more control than the ANSI standard SQL. However, it seemed inefficient to require the programming and statistics staff to learn a new language just to replicate the partial functionality available with SAS.

Conclusions

Many people, including the SAS community, acknowledge SAS as a great access,

analytic, and reporting tool but accede the data server realm to the likes of Oracle, Informix, and Sybase. SAS' Scalable Performance Data Server seems to be, unfortunately, one of the Institute's best-kept secrets. It offers enterprise-strength cutting edge technology that operates in seamless cooperation with the SAS system. Further, pricing for SPDS is comparable or better than the other database alternatives.

For our situation and environment, SPDS proved to be the logical choice and has resulted in improved performance, enhanced functionality, and increased efficiency. After an initial test period and much internal political negotiation, in March 1997, SPDS replaced Oracle as the central SAS data server for the Portfolio Analysis group. Since that time the database has been used extensively for data mining and analysis.

Due in part to our success with this architecture, we have recently installed a hardware upgrade (a Sun Ultra-Enterprise 4000). This new system will roughly quadruple the potential size of the datamart, double the memory and dramatically increase the overall computing power available. As we expand both the capacity of our systems and the scope of our reporting and analysis functions, SPDS 2.0 and beyond will continue to be our data server of choice.

Acknowledgments

I would like to thank the following people for their input and support for this paper and the Oracle to SPDS migration project:

- Eric Wagner, *Our SPDS Administrator*
- Amitava Ghosh, *The force behind SPDS*
- Eric Hunley, Ed Blair, & Bubba Talley, *SPDS Product Support and Management*
- Susan Peavy, Brian Santucci, & Mark Herman, *The Prudential Bank Portfolio Analysis & Information Support*
- Bruce Spengler, *VP of Business Systems, The Prudential Bank.*