Paper 119-2010

Blistering ETL Performance using the Intelligent, Dynamic and Parallel Capabilities of SAS®

David Logan, PBT Group, Johannesburg, South Africa

ABSTRACT

ETL from billion-record+ databases is a non-trivial task. By using the unique capabilities of SAS® to analyze the database in advance and then dynamically generating parallel SAS® jobs based on this interrogation, SAS® can effectively performance-tune itself each run for maximum benefit. Get in, get out, get the results. The net effect is to get the fastest possible results with the minimum window of ETL load on the source system. Scalable, efficient, and easy to develop. An added benefit is the data visualization possible post-analysis and pre-ETL to aid data quality checks.

Improving your ETL time (in the case study, by 92%), using existing resources, postponing expensive hardware upgrades, dynamically adjusting for ever increasing data volumes and, improving information productivity are powerful arguments for considering the adoption of this approach, where feasible, in your own environment.

INTRODUCTION

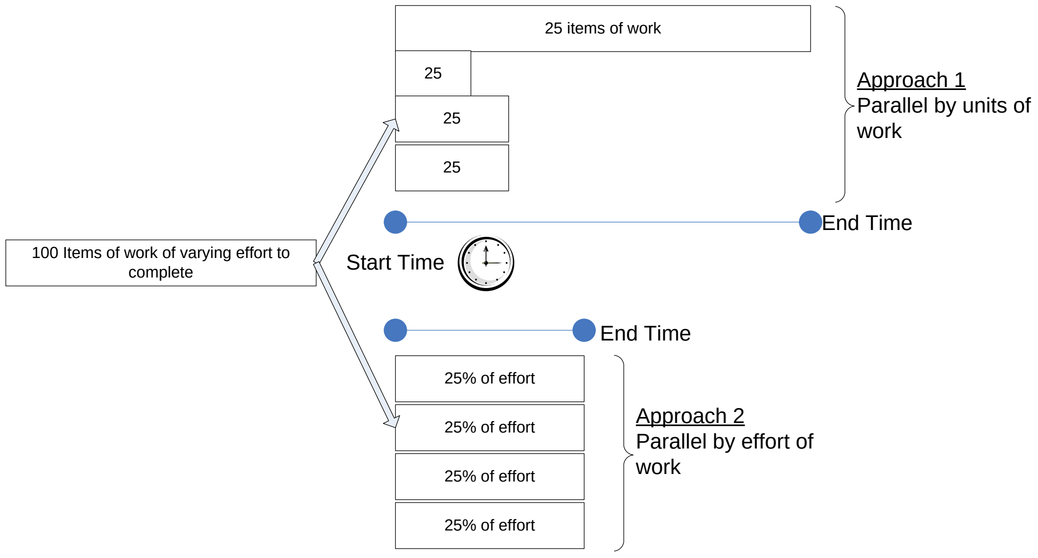
Initially the purpose of this paper was to communicate the huge performance benefits of running your SAS® ETL system with an intelligent, dynamic, parallel approach in a specific case study at a telco client. The approach has since been used with a variety of source databases with similar benefits, so I will explain from a conceptual point of view the principles and then finish with a brief overview of the initial case study. We start with some basic parallel principles and then follow the logic through into the Intelligent, Dynamic and Parallel phases of this paper. Then the case study will be used as an example of applying these principles.

Going at length into the exact detail of implementation as per the case study became an increasingly lengthy paper and the main thing to gain from reading this is an understanding of how to apply the principles for your own specific environment, so the intention is to keep it at a high-level conceptual explanation whereby you can possibly see a scenario in your own environment where it can be applies successfully, too.

PARALLEL PRINCIPLES

With multi-processor machines and partitioned databases with billions of rows there is frequently a requirement to extract a specific subset of data from the database on a regular basis. See figure 1 below for an example of 2 approaches to this task using parallel processes.

Figure 1. Comparison of different approaches to working in parallel



Note the difference in end time.

In Figure 1 above, you start with 100 "items" of work to perform.

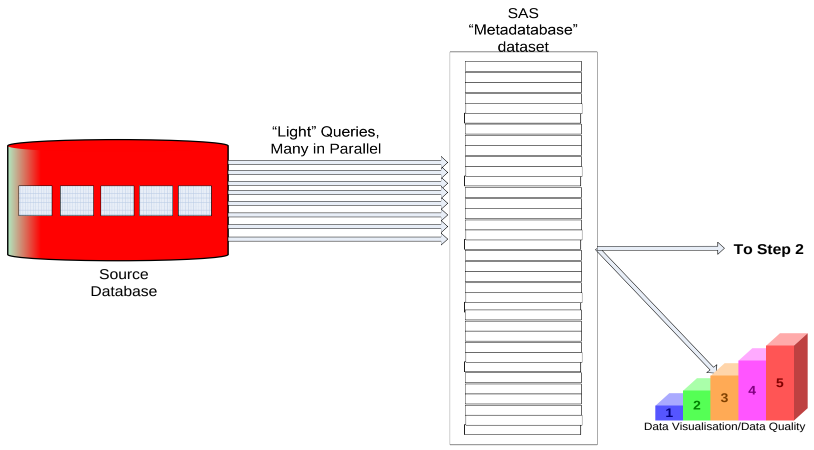
In the real world, as in the IT world, each of these items of work could require a varying degree of effort (and time) to accomplish. An example would an extract from a single database partition of 100 rows and another extract which return 1 Million rows. The effort involved is quite different! Simply splitting the work up into for e.g. 4 parallel processes would not necessarily take ¼ of the time as the completion time would be the length of the longest process (Approach 1 in diagram).

If there was a way we could "guesstimate" the amount of effort required for each item of work we could try and split the workload into 4 parallel processes with each having, ideally, 25% of the overall effort and elapsed time. Fortunately in SAS® we have at our disposal the ability to interrogate a wide variety of databases and the analytical power to turn this "guesstimate" into a reasonably accurate estimate.

STEP 1: GATHER THE INTELLIGENCE

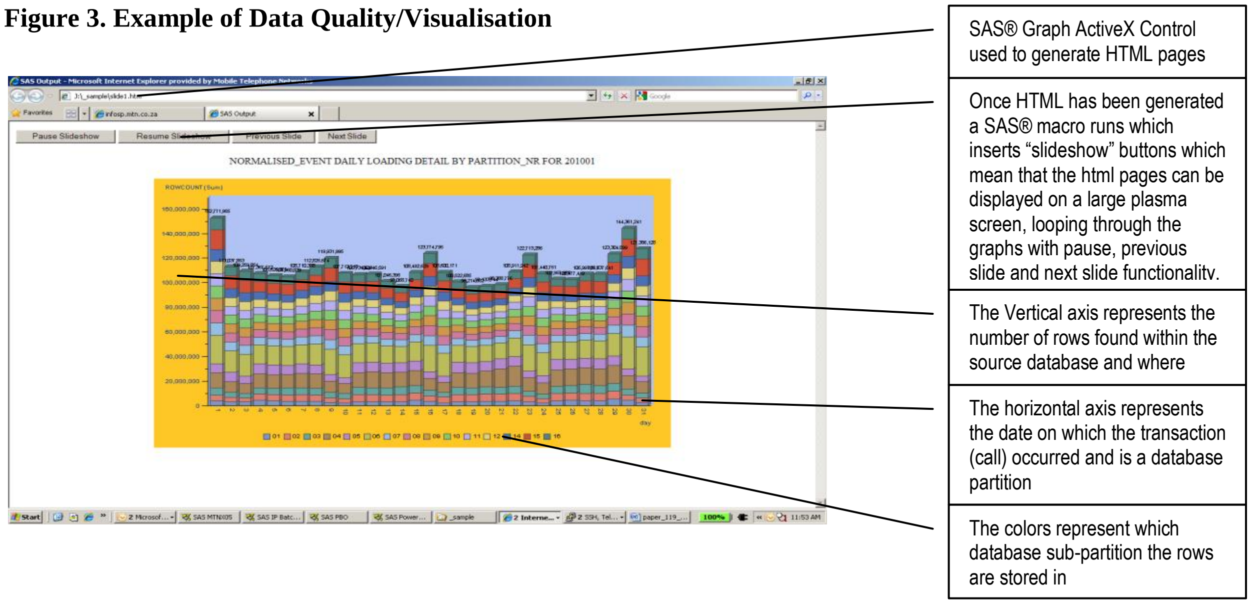
Ideally the processes to interrogate the database should be “light” processes, i.e. fairly fast and quick to return results. An example would be a simple count of the numbers of rows in each partition in a source database. However, depending on the intelligence “requirements” needed in order to be able to do workload estimates, this should be tailored to the specific problem at hand. See figure 2 below

Figure 2. Example of “Intelligence Gathering” on source database


SAS “Metadatabase” dataset
“Light” Queries, Many in Parallel
Source Database
To Step 2
1
2
3
4
5
Data Visualisation/Data Quality

In the Figure 2 above we run many “light” queries in parallel and then consolidate the results into a single SAS® metadata data set. From this data set we can then build the SAS® parallel processes (programs) with as equal a workload effort as possible, as opposed to an equal number of work items as explained in the parallel principles section previously. A major added benefit is the ability to generated data visualization and data quality graphs as early as possible in the process, *prior* to ETL. See Figure 3 below.

Figure 3. Example of Data Quality/Visualisation



SAS® Graph ActiveX Control used to generate HTML pages

Once HTML has been generated a SAS® macro runs which inserts “slideshow” buttons which mean that the html pages can be displayed on a large plasma screen, looping through the graphs with pause, previous slide and next slide functionality.

The Vertical axis represents the number of rows found within the source database and where

The horizontal axis represents the date on which the transaction (call) occurred and is a database partition

The colors represent which database sub-partition the rows are stored in

Figure 3 (above) essentially gives you a visual representation of the physical data in the source database. Note that adding “slideshow” functionality to SAS® ActiveX Graphs really enhances the presentation of the information. I am indebted to Nisha Inarman of MTN South Africa for developing the SAS® macro functionality which inserts the necessary HTML lines into the HTML files produced by SAS® Graph. Now that we have the required “meta-database” information in a SAS® data set *and* visually, it should be easy to see that, depending on the extract to be undertaken, intelligent decisions can be made around how to structure the extract dynamically and load balance the parallel extract streams according to a work effort estimate.

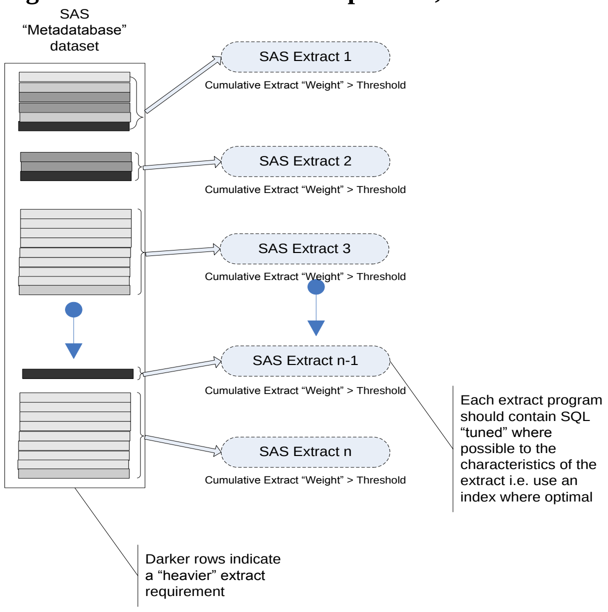
STEP 2: DYNAMICALLY CONSTRUCT THE PARALLEL EXTRACTS

In this step we have already gathered the necessary information from the database itself which we require in order to construct the SAS® extract programs which will be run in parallel. The optimal solution would be

Total effort (est time) = Effort (Est time) per process

No of Parallel Processes

Figure 4. Construction of <n> parallel, load-balanced extracts


SAS "Metadatabase" dataset
SAS Extract 1
Cumulative Extract "Weight" > Threshold
SAS Extract 2
Cumulative Extract "Weight" > Threshold
SAS Extract 3
Cumulative Extract "Weight" > Threshold
SAS Extract n-1
Cumulative Extract "Weight" > Threshold
SAS Extract n
Cumulative Extract "Weight" > Threshold
Each extract program should contain SQL "tuned" where possible to the characteristics of the extract i.e. use an index where optimal
Darker rows indicate a "heavier" extract requirement

In Figure 4 above, a "master" SAS®program is used to scan through the data set which was created in Step 1. Using for e.g. the no. of records to be extracted for each unit of extract work to be done, we can assign a relative "effort" number to the unit of work. For E.g. if we were extracting 100 Million rows contained in 10 partitions (units of work) with 4 parallel processes then we would try and divide up the work as close to 25 Million rows each as possible. In practice, it depends on what the key factors are in determining the length of time a single extract will take, which is why it is crucial to get the right metadata from the database in the first place.

In the diagram above the relative job weights are indicated by the shade of the rows in the SAS® "Meta-database" data set and a relatively large number of low job weights are incorporated into a single extract whereas a single extract with a large job weight may have a process of its own.

Note that the programs have *not yet been submitted for processing*, basically what happens is that SAS® programs are dynamically written as the normal "*.sas" text files, each of which %includes a pre-written SAS® program/macro which performs a parameter-driven **unit** of extract work. In the case where we are extracting from various partitions the parameters can be the name of a SAS® data set which contains the parameters for multiple units of work or the parameters themselves. E.g. extract_sample_<job number>.sas is created and contains the code below

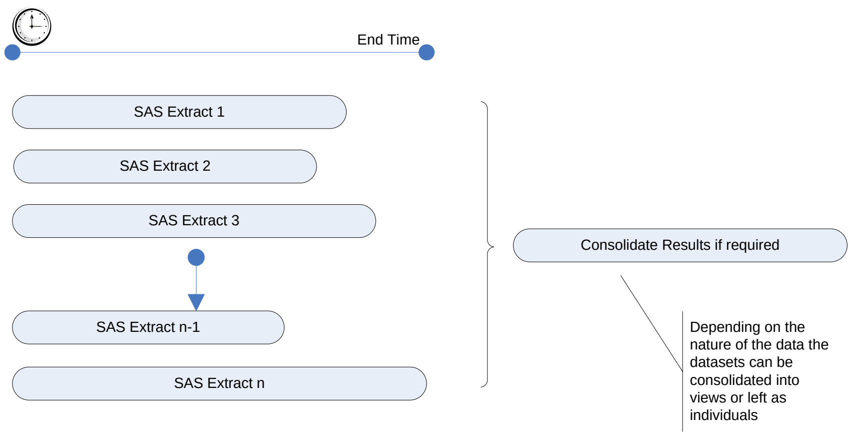
```
%include "<program directory>/unit_of_work.sas";
*Calls the macro in the above program with parameters;
%unit_of_work(<parmameter1=X>,<parameter2=Y>;
```

In this way the single unit of work program can be individually tested on small volumes and with various parameters to tune the job weighting algorithms. Now that we have performed the "intelligent" and "dynamic" steps without actually doing any extract work as yet, we can now run the SAS® programs created in parallel.

STEP 3: PARALLEL PROCESSING

In the previous 2 steps we spent a fair amount of time thinking about the optimal way to perform our ETL. Now we need to see if the benefits have been worth the effort. (Hint, They are!)

Figure 4. Execute the programs in parallel


End Time
SAS Extract 1
SAS Extract 2
SAS Extract 3
SAS Extract n-1
SAS Extract n
Consolidate Results if required
Depending on the nature of the data the datasets can be consolidated into views or left as individuals

Unfortunately the client has only a small number of SAS® products licensed, and there was no SAS® Data Integration Studio, LSF Scheduler or any tool driven way to submit these programs once we had created them. *Note Opportunity to incorporate this into a SAS® Data Integration Studio environment?*

The result was the creation of a %subjob macro which could submit SAS®code which does the following steps

- Create .bat/.sh script depending on whether Windows/Unix
- Execute program passed as a parameter
- Write results to <yyyymmdd>_<program_name>.log.<ok/error>
- Email status to submitting user upon completion, attaching log if error

In Figure 4 above each SAS® program generated in the previous step is now submitted as an independent/parallel process. Checking for a *.log.ok program for each of these programs would then signify that the complete batch has finished successfully at which time any integration/post-extract dependancies could be completed.

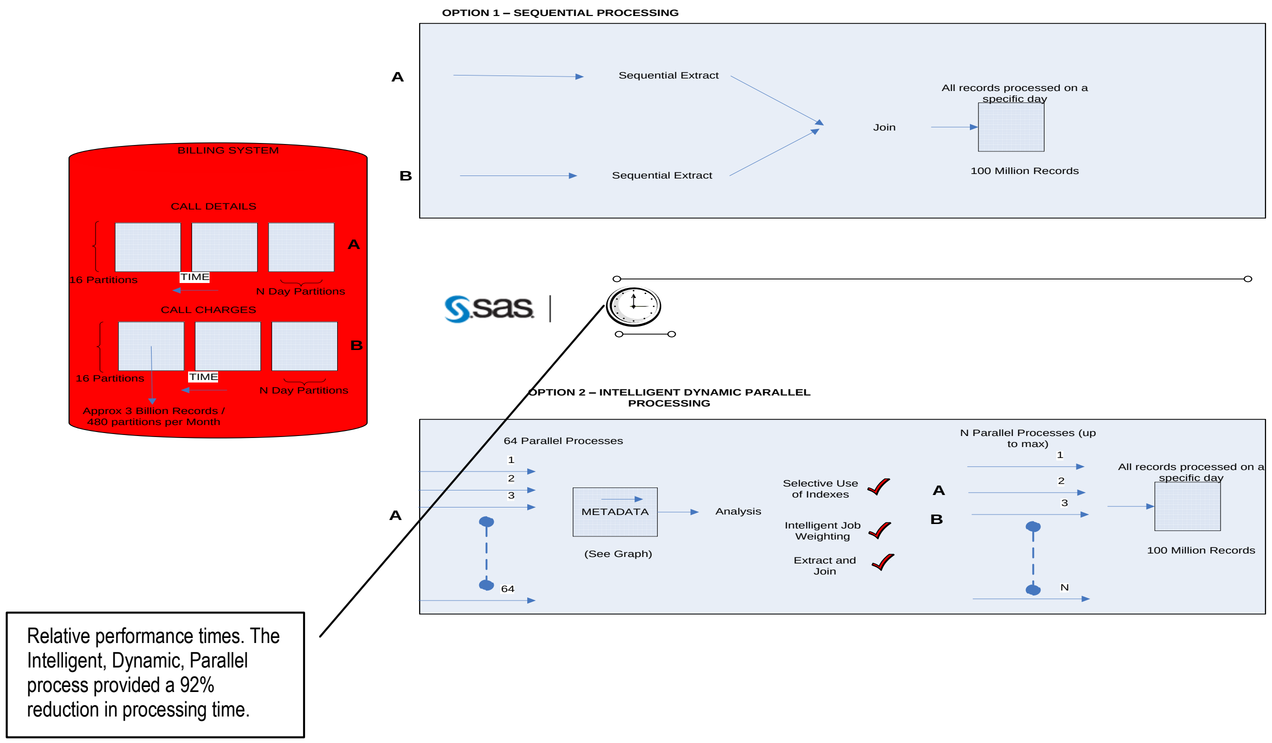
Now that we have explained how to implement each of the intelligent, dynamic and parallel concepts let’s review the actual case study which drove this design. Initially the case study was the main focus of this paper but I’ve re-used the concepts repeatedly for a variety of ETL issues and the concept is more important than the specific approach. The case study itself demonstrate the huge benefit of this approach in a real-world situation

CASE STUDY: 200 MILLION+ROWS FROM 18 BILLION ROW DATABASE (TELCO)

PROBLEM STATEMENT

See Figure 5 (below). The source database contains approx 3 Billion call detail records and 3 Billion equivalent call charge records per month. (See A and B within database icon in diagram.) We need to extract the approx 200 Million rows which were added to the database on a particular day. Call records can be delayed and are added into the relevant call date partition and call type sub-partition i.e. they are deposited anywhere amongst approx 3 months worth of data and partitions (around 18 Billion records and 3000 partitions). The call detail and call charge components have to be joined to derive a composite record for EDW purposes. It's a non-trivial problem to resolve.

Figure 5. Case Study Overview


OPTION 1 – SEQUENTIAL PROCESSING
A
Sequential Extract
B
Sequential Extract
Join
All records processed on a specific day
100 Million Records
BILLING SYSTEM
CALL DETAILS
A
16 Partitions
TIME
N Day Partitions
CALL CHARGES
B
16 Partitions
TIME
N Day Partitions
Approx 3 Billion Records / 480 partitions per Month
sas
OPTION 2 – INTELLIGENT DYNAMIC PARALLEL PROCESSING
64 Parallel Processes
1
2
3
64
A
METADATA
(See Graph)
Analysis
Selective Use of Indexes
Intelligent Job Weighting
Extract and Join
N Parallel Processes (up to max)
A
B
1
2
3
N
All records processed on a specific day
100 Million Records
Relative performance times. The Intelligent, Dynamic, Parallel process provided a 92% reduction in processing time.

APPROACH

The Intelligent, Dynamic and Parallel appeared to be a worthwhile option to explore. Based on the first, sequential approach, we had some useful statistics (from the SAS® logs and data sets) around the types of queries which worked better and at which volumes (especially in regard to using indexes). The sequential approach was taking up to 12 hours from midnight to run and varied quite widely. (The longer a process runs the more inconsistent it gets as the contention on the hardware varies, particularly as the online day approaches)

STEP 1: GATHER THE INTELLIGENCE

64 Parallel processes were submitted which took approx 15 min to run. These processes would count the number of rows by fileid, where fileid was a column which could be used to identify *when* a record was processed on the database. As fileid was an indexed column on the source database this would prove to be an added advantage later. Immediately after this scan the ETL work to be done would be reduced to *only* those partitions where we know data to exist for a particular process date. In addition should be know we will only return a relatively small number of records, the SQL query can be constructed with *exactly* those fileids which we know to exist.

STEP 2: DYNAMICALLY CONSTRUCT THE PARALLEL JOBS

Based on the intelligence gathered above, we now use this data to split up the work into <n> parallel jobs with as close to an equal share of the work as possible assigned to each job. The number of rows to be returned (more rows=”heavier” work) and whether or not the fileid index column can be used (index used=”lighter” work) would give a reasonable estimate of the workload involved in a single extract process from a single partition. The total work is then divided up into the <n>umber of processes to be submitted.

Hint: It might be best to get friendly with your DBA at this point as this can take the concept of “sweating your resources” to extremes. It helps to point out to your DBA that although you are doing a fair amount of work, you are doing the *minimum* amount of work in a smaller time window.

STEP 3: PARALLEL PROCESSING

Now that you have done everything you can to be as intelligent and dynamic as possible, execute the parallel jobs and review the results. In this case study processing time went from anything up to 12 hours to around 1 hour (15 min metadata/45min extract) so it *improved the completion time by around 92%.* In addition the time was extremely stable as, effectively it was re-calculating the optimal ETL characteristics every day before running.

CUSTOMER COMMENTS ON CASE STUDY

“*The complete replacement of our wholesale billing system called for a thorough audit of the new solution. This was achieved by comparing the sunset and sunrise systems on all aspects of charging and billing; with specific focus on the usage revenue line of the company: call data record rating. The tight window for audit and sign-off demanded daily dashboard publications based on previous day rating output. The improved completion time with regard to call data records extracts made it possible to meet the SLA requirement. Less time spent on extracting data, and more time on analysing the output, resulted in critical audit findings, which in turn, caused the sunrise system to implement with enhanced data integrity*.”

“***Blistering ETL performance****, in the end, played a pivotal role in the successful implementation of our new wholesale billing solution*.”

Alet Smith, *on behalf of MTN Powerbill Programme*.

BENEFITS

The **development effort** for the code involved in the case study was 1 person (myself) for 2 weeks. The ability of SAS® to access and analyze a wide variety of database information and, using SAS® macros to dynamically calculate, build and execute an optimal ETL approach make SAS® almost uniquely suited to an approach of this nature.

The **scalability** of this solution is fantastic, adding more hardware resources simply allows you to scale up the "parallelism" of the approach, splitting the same amount of work into more chunks. Alternatively large scale increases in volume result in a lesser increase in processing time for the same number of processes.

The **performance** in the case study speaks for itself, a 92% improvement in ETL processing time from a multi-billion row database created a positive "buzz" around SAS® at the client.

The **portability** of the concept has been extremely good, I have re-used this approach with multiple large-volume databases with varying characteristics, performance criteria and problem statements and each time it requires less development effort with similar results. The more often you do it, the easier it gets.

Cost-effectiveness has to be one of the major benefits of this approach, quick development time and making full-use of what are sometimes idle hardware resources, gets you the most "bang for your buck" from the hardware investment.

CONCLUSION

If performing the same task in *8% of the time, with existing resources, postponing expensive hardware upgrades, reducing time to business delivery and doing all of this in a relatively short period of time* is of interest to you then I hope you have enjoyed the paper and can see an application in your own environment.

Having worked in SAS® for around 15 years now, I would have to say that having conceptualized, designed and implemented the Intelligent, Dynamic, Parallel ETL process at a client, I am now wondering why I haven't always done it like this. It just seems like a natural fit with the unique capabilities and features of SAS®. I look forward to more opportunities to implement this approach and would be particularly be interested in anyone who would like to explore making this more of an easily customizable add-in/feature in the SAS®9.2. Data Integration Studio world.

Thanks for reading!

REFERENCES

ACKNOWLEDGMENTS

Many thanks to Nisha Inarman of MTN SA for the SAS® macro which incorporates slideshow functionality into the ActiveX graphs produced by SAS® Graph .

RECOMMENDED READING

CONTACT INFORMATION

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

Name	:David Logan
Enterprise	:PBT Group
Address	:Unit 3, Knowledge Park 3, Century Boulevard , Century City
City:	Cape Town, South Africa
E-mail:	:davidl@pbt.co.za
Web:	:www.pbt.co.za