Paper 153-27

Autotrader.com: From Chevette to Corvette

Jenine Eason, Jerry Johannesen, Brad Goldman, Lars Olufsen, & Jeff Chang
Autotrader.com, Atlanta Georgia

ABSTRACT

AutoTrader.com has been extremely successful in applying SAS to manage and report voluminous business information since April 1999. Since receiving the Enterprise Computing Award at SUGI 25, there have been many overwhelming successes as well as a few bumps in the road.

This paper will review the challenges and solutions that the development team at AutoTrader.com dealt with while successfully rolling out their project. For example, understanding the dot.com reporting pitfalls - what appeared to be straightforward reporting was revealed to be treacherous and vague. Forecasting storage capacities became a never-ending challenge. Creating timely reporting for the user seemed easily deliverable - but as the site grew and our audience became more sophisticated, needs changed…quickly.

Autotrader.com successfully rose to each new level. This presentation will layout the map for collecting data from multiple sources so that our clients had one reporting source. It will explore discoveries using advanced SAS tools, Java drill down graphs, sophisticated MDDBs, ODS, dynamic Internet front ends, and HTML. An increasing demand for advanced site usability and a more professional look prompted us to delve deep into these advanced SAS tools.

INTRODUCTION

The main source of the company's reporting is provided through an Intranet Unix based site. At the time Autotrader.com received the SAS Enterprise Computing award, the company was small and reporting needs limited. The Intranet platform was a perfect way for management to locate web site traffic reports at their leisure. While the Intranet site was successful, it used little of the current SAS tools for reporting. SAS was used primarily to crunch data. Reports maintained on the site were nothing more than SAS datasets converted to Excel spreadsheets, one dimensional graphs, and highly complicated outputs using SAS and endless PUTs and MDDBs.

THE NEED FOR SPEED

There were many factors that forced the AutoTrader.com web site and analysis into overdrive. Growth in the web site audience, growth in the number of web servers, growth in the number of employees and the addition of a nationwide sales force, increased advertising partnerships, growth in the number of cars and dealerships added to our database, and the sheer volume of the daily web log files to be processed. In May 1999, traffic reporting was performed using SAS version 6. 1.2 million people visited the web site each month, they viewed 6.6 million pages and performed 1.1 million used car searches. One to two Unix servers hosted the web site and the daily SAS processing of traffic was performed on an old email server, namely a Compaq PC.

The web traffic multiplied significantly as a result of Autotrader.com's Super Bowl ad in January of 2000. The site began setting new traffic records each month. In the spring of 2000 Autotrader.com received the SAS Enterprise computing award. At that time, traffic metrics had zoomed to over 4 million people viewing over 80 million pages and performing over 15 million used car searches per month. Web servers were multiplying like rabbits to keep ahead of the demands on the web site. 20 million rows of web log files were read each morning. The daily traffic reports were posting later each day. In an effort to relieve this, SAS was migrated from a PC to a Sun 4000 4-way server. The movement of the daily analysis and reporting SAS application from one platform to another took less then a week thanks to SAS programs being portable across operating systems – only the Filenames and Libnames needed to be updated to point to the new disk array names. Everyone was happy with the reduction in nightly processing time, but growth continued.

One year later, in the spring of 2001, traffic had steadily worked its way up to 5.7 million people viewing over 140 million pages and performing over 25 million searches each month. The slice and dice OLAP reporting using MDDBs had gotten too slow, and the nightly processing was again running too long and producing results later and later. The CPUs were upgraded to faster processors and image logging was dropped from the web logs to improve performance. This helped for a while, but was not enough to keep up with growth. The next attempt to improve speed was fine-tuning our SAS applications.

The solution to our MDDB performance was to break the MDDBs up into seven smaller MDDBs and tie it all back together seamlessly using the Hybrid OLAP feature. This increased performance by breaking a very large MDDB, over 2 gigs, into multiple smaller MDDBs around 50 Megs. Information on how to do this can be found in SAS proceedings from SUGI24, #139 and SUGI25, #133. Dial in users were still reporting unreasonable slowness. Therefore a SAS/Intrnet version of this information without MDDBs, described later, was developed.

The solution to our nightly run time issue was to take the single threaded process and make it multi-threaded. Although SAS does not natively multi-thread its processes in version 6 or 8, a process was developed to do this. Since web logs were being read from dozens of servers each night, SAS was used to start up a separate SAS session to read each of the servers. Once they all completed, the results were gathered together. In order for this method to be useful, there must be multiple CPUs. This divide and conquer approach had the added benefit of dynamically listing every web server in production. Previously, the server list was constantly updated manually.

The first step in multi-threading the application was to hop out to the operating system and collect a list of all of the servers. To do this, a pipe command option on the Filename statement was issued to command the operating system:

```
data mtd.howmany;
  filename pipefile pipe 'ls -trl /Business Intelligence/logs';
  infile pipefile pad;
    input @55 machine $24.;
    nummach+1;
    call symput('machines',nummach);
run;
```

The data step collected an overall count and a list of server names to be used. This technique of gathering information to drive subsequent processes is known as data driven programming. A permanent SAS dataset was created and used to determine when all of the individual processes complete, how many rows each read, and the completion time for each process. Then the dataset is used to drive a loop, which generates SAS code to process each server's logs, and execute each program. The macro for doing this step of the process is shown in Appendix A. The core read routine is read as an external file and written back out under a new name, with Libname and Filename

statements specific to each individual web server. Care must be taken to keep each set of results in a separate library or dataset in order to avoid sharing conflicts of multiple processes trying to write to the same dataset. Then each individualized program is executed in the background by the statement:

```
x "/sas/sas -memsize 96M -sysin
/links/code/daily/jobs/run&&i..sas &";
```

As each program finishes executing, a summary dataset is created for each process. These summaries are examined periodically to determine when each machine has been read.

The macro, which checks for completion of each process, is shown in Appendix B. A first attempt was made to append a row to a summary dataset when each process completed, but this frequently failed due to dataset reservation conflicts as most processes finished at the same time. The SAS Institute supplied the guts of the macro used in this process (CHECKDS), which dynamically builds a list of valid SAS datasets created as each job finished. The combination of these rows is compared to the original list of servers in order to determine when all server logs have been read. The results are combined for further processing.

By using HOLAP and innovative programming techniques to multi-thread the nightly processing, AutoTrader.com has been successful in keeping pace with the web site traffic growth over time.

ADVANCED REPORTING – SAS, FROM START TO FINISH

In the beginning, reporting on the Intranet was very limited and very unsophisticated. The simplest of graphs were used. Most of the standard reports consisted of .CSV files uploaded to the server. Advanced data steps with complicated PUTs littered the site. MDDBs were the most advanced reports of our Intranet site because they allowed for user interaction.

ALONG CAME VERSION 8 AND ODS

The lure of ODS was almost immediate. Standard reporting tools such as Proc Report, Proc Tabulate and Proc Print were encased with ODS, providing a very professional looking HTML output. The users were delighted.

Below is an example of a report prior to applying ODS and a view of the results with ODS.

Before with a .CSV file

	A	B	C	D	E	F
1	Internal 001			pagenames_location.sas		
2						
3	Source data file: loc.w_file					
4						
5		REQUEST	PAGE NAME	COUNT	PERCENT OF TOTAL	
6						
7	Location:					
8		/				
9		/admin/dealer/	Auction validat	8	0.00%	
10		/admin/messa	Search Messa	3	0.00%	
11		/admin/site_in		2	0.00%	
12		/admin/view/in	inView: Site Cr	3	0.00%	
13		/aol/index.jtmp		488	0.00%	
14		/broadcast/wft		1	0.00%	
15		/careers/index	Employment C	3262	0.00%	
16		/credits.jtmpl	Credits: About	1013	0.00%	
17		/dealer/index.jt		76	0.00%	
18		/decision/inde		5421	0.01%	
19		/finance_insura		12	0.00%	
20		/finance_insura		5	0.00%	

After using ODS

AutoTrader.com

Location	Page	Name	Page Views	Percent of
AdManager	/admin/dealer/index.jtmpl	Ad Manager: Home Page	2,360	
	/admin/dealer/inv_login.jtmpl	Ad Manager: Member Log In	103	
	/admin/dealer/processlogin.jtmpl	Ad Manager: Process Login	70	
	/admin/dealer/resign_in.jtmpl	Ad Manager: Member Log In	105	
	/admin/dealer/signup.jtmpl	Ad Manager: Dealer Signup	65	
	/admin/index.jtmpl	Admin: Home Page	234	
			2,937	
Calculators	/finance_insurance/calculators/calculators.jtmpl	Calculators	460,918	
			460,918	
Cars Direct	/newcar/index.jtmpl	CarsDirect.com	229,639	

The coding for this change is actually very simple. A basic Proc Report in an ODS HTML shell. This sample stipulates where the html output should be placed. A title/logo is added for consistency across all reports. The style, barrettsblue, is a system provided standard.

```
ods html
 file="/reports/Pagename/pagenames.html"
 headtext='<TABLE ALIGN=CENTER
 <TR><TD ALIGN=CENTER>
 <Img src="../../atlogo_448x54.gif"
 border="0"></TD></TR></TABLE>'
 style=barrettsblue ;

    <Proc Report Code>

ods html close;
```

Incorporating a reporting standard on the Intranet site required settling on a basic reporting view. This is most easily and professionally accomplished utilizing ODS. The same titles and styles are used throughout. There is no longer a need for Excel-converted tables or tab delimited files created with PUTs. This gives the site a far more advanced look while guaranteeing the consistency of reports created by different developers.

DOWNLOADING FILES

Another useful tool for our users is embedding a downloadable file in the title of a report. While reports on the Internet may look good, manipulating the data for further use was another challenge. By creating a comma-delimited file from the same input as the html, a user can simply click on the hyperlink to open spreadsheet for further use of the data.

Having created a .CSV file (or other comma delimited file), simply create a reference to it in the title field in ODS output. The line below creates the html results above.

```
title h=2 "Click <a href="/outputfile.csv">here</a> to
download this file.";
```

DEALER LEADS REPORTING

One of the fundamental ways that Autotrader.com sells itself to dealers as superior to traditional classified advertising is reporting. The reporting Autotrader.com provides lets the dealer know exactly what they get for their dollar. Unlike a newspaper or radio ad, Autotrader.com can tell a dealer exactly how many times their cars are viewed, how many people sent emails, asked for a map, etc. These metrics are vital in convincing dealers to buy our products. The reporting system draws from a large number of data sources, combines them over different time

periods, creates “nice” HTML output, provides an interactive system to access them, and emails them out to dealers.

COLLECTING DAILY STATISTICS

Each morning, an automated job kicks off the daily driver program. This program sets an autocall library, and a set of macro variables that contribute to the input and output datasets for that day. It then calls a series of macros, each one corresponding to a given input source or kind of metric being gathered. All this daily data is merged together in a dataset, with one row per dealer, and one variable for each kind of metric or flag needed.

COMBINING TIME PERIOD DATA

Reports are created based on a given week, month, month-to-date, or last month+month-to-date. All these are driven off of the same data collection system. Each has its own driver program that tells it where the relevant data directories and date boundaries are. The daily datasets are all named with a yymmdd suffix, so it is easy to parse a directory listing to get the correct datasets. The appropriate datasets are appended, and a giant SQL statement is executed that calculates sums and averages across the time period.

CODE OPTIMIZATION

The code that ran this system each day had become convoluted, confusing, and cumbersome. In mid-2001, the code was revisited and 99% rewritten. Despite the rapidly increasing size of data, reporting needs, and the level of customization, the daily load time was reduced from 4 hours to 30 minutes. 35 macros became 14, 150 datasets per month became 70. The code is easier to understand, takes less time to run, less time to maintain/improve, and less disc space.

OUTPUTTING THE REPORT

The reporting macro is essentially a big data loop. One of the variables in the input dataset is the Unix output file location. This is used to create ~50,000 HTML reports each day.

```
DATA daily.reportout;   SET daily.result;
  file dummy filevar=dlrfile;
  by dealrid;
```

…hundreds of statements like:

```
put '<p>Customer requests for new car details page</td>';
put '<td>';
put '<p align="RIGHT">' count comma9.0 '</td></tr>';
```

Although the reporting macro is cumbersome to maintain, it gives exact control over every aspect of the output.

POST-REPORTING

The process of reporting also creates a final dataset with all the statistics over a time period for each dealer. The datasets are dumped into Oracle for use by other groups within Autotrader.com. If it is the monthly report, then the dataset is copied into a special directory, and used as input to other monthly programs.

In order to allow interactive access, indexes are built on certain key fields:

```
Proc SQL;
 Create index city on mtd.mtdrpt(city);
 Create index addre on mtd.mtdrpt(addre);
 Create index state on mtd.mtdrpt(state);
 Create index dlrname on
  mtd.mtdrpt(dlrname);
 Create index zip5 on mtd.mtdrpt(zip5);
 Create unique index dealrid on
  mtd.mtdrpt(dealrid);
Quit;
```

HTML INTERFACE

Everyone has to be able to access the reports via our intranet site. Selecting from 50,000 reports requires an interactive interface. An HMTL front end allows users to look up a report in any given time period by dealer ID, dealership name, or any element of the address. An HTML form passes the parameters to a SAS broker program. This broker program looks up the passed parameters against the indexed dataset created above.

If the result is one report, it loads a redirect script that passes the user to the correct report.

```
data _null_ ;
  file _webout;
  put '<html>';
  put "<META HTTP-EQUIV='refresh'
   CONTENT='0;URL=&location'>";
  put "<h2>Redirecting you to the
   report...</h2>";
  put "</html>";
run;
```

If there is more than one report (for example the user asked for all dealers in Georgia), a separate HTML screen is created. Again using put statements, it writes a line for each dealer fulfilling the criteria.

```
put '<td align="center"><font size="5"><b>
 Customer Activity Summary</b></font></td></tr></table>';
 put '<table width="95%" border="1"> ' ;
end;
put '<tr>' / '<td><a href=http:/' location '>' dealrid '</a></td>' ;
put '<td>' name '</td>' ;
put '<td>' city '</td>' ;
put '<td>' state '</td>' / '</tr>' ;
```

Note that the location is put as a hyperlink, so the user can simply click it to bring up the report they need.

EMAILING AND FAXING REPORTS

Autotrader.com emails and faxes approximately 20,000 dealer reports monthly. In early 2001, the full process to get monthly reports into dealer’s hands took approximately 10 days. Today, they get these reports by noon of the first day of the month. Processing has been smooth enough that Autotrader.com will begin emailing reports every week, or every day by mid-2002. Working closely with Unix and Windows administrators to ensure the below code produced the correct results, customer emails are produced. SAS controls the “from” field in the filename statement, however, many receiving servers look at the “sender” field, which needs to be spoofed at the Unix level (See Tech Support Notes #500 and #3691 for more details on this known bug).

```
%macro runem ;
%do i=1 %to <nobs in dataset>;
 data _null_;
  pointer=0+symget('i');
   set <dataset> point=pointer;
   call symput('Unixfile',compress(dlrfile));
   call symput('email',compress(e_mail));
 stop;
run;

filename outemail email "&email"
 from="dealerleads@autotrader.com"
 subject="Your January, 2001 AutoTrader.com
   Leads" type="text/html";

data _null_; file outemail;
   Infile "&Unixfile" length=ln;
   input;
   put _infile_;
run;
```

```
%end;
%mend;
%runem;
```

Faxing is also a big part of customer reporting. The fax server that is in place at Autotrader.com will support any Microsoft Office format attachments, but cannot send HTML formatted files. The solution was to use SAS to get one HTML format file at a time from the server (using FTP filename method) and run a Microsoft Word macro (as a DDE link from SAS) to convert the file into a Word document format, and send the fax.

```
%macro runfax ;
%do i=1 %to <nobs in dataset>;
<prepare code as above in emails>
<code to ftp file from unix to PC referencing the fax server>

filename cmds dde 'winword|system';

data _null_; file cmds;
  put '[ToolsMacro .Name = "htm2doc", .Show=Normal,
       .Run]';
run;

filename test email 'testuserid' attach=("&dlrfile")
subject ='Your Report from AutoTrader.com';

data _null_;
  file test to=&deal_fax;
run;
%end; *do loop;
%mend runfax;
%runfax;
```

See Appendix E for the Word Macro (htm2doc).

MDDBS & SAS/INTRNET APPLICATIONS

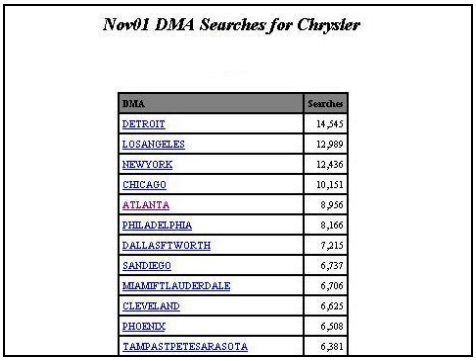
Use of MDDBs on the Management Dashboard was a necessity right from the beginning. One of the leading reporting needs is the dynamic reporting of vehicle searches performed on Autotrader.com. As the site grew, the MDDBs became cumbersome. Many users access the Management Dashboard from remote sites using dial up service and slower modem speeds. For many of them, report return times became unacceptable. Some of the MDDBs were no longer Dial In Friendly. A different approach was needed. The answer was found using a SAS/Intrnet application to replace the CPU heavy MDDB.

A vehicle searches SAS/Intrnet application was developed to provide the same functionality as the vehicle searches MDDB. This includes having the same hierarchical structure and data filters. Since the MDDB is dynamic, CPU is required to summarize the data at the users request. On the SAS/Intrnet application, the data is pre-summarized and stored in permanent SAS datasets. As the user selects a particular hierarchy, a pointer goes to the applicable pre-summarized dataset. The CPU time required by the MDDB (causing the user to sit and wait) is eliminated by having built permanent datasets. If a filter is requested, the user is again pointed to the appropriate pre-summarized dataset, and a "where" clause is utilized to pull out only the desired rows. The application then sorts and summarizes this even smaller dataset, and the CPU required is drastically reduced.

The SAS/Intrnet application incorporated dynamically creates web pages. Through ODS and the _webout capability, the user is presented with a report that is further drillable. This can be seen via an example.

FRONT END

DRILLABLE REPORT



Nov01 DMA Searches for Chrysler

DMA	Searches
DETROIT	14,545
LOSANGELES	12,989
NEWYORK	12,436
CHICAGO	10,151
ATLANTA	8,956
PHILADELPHIA	8,166
DALLASFTWORTH	7,215
SANDIEGO	6,737
MIAMIFTLAUDERDALE	6,706
CLEVELAND	6,625
PHOENIX	6,508
TAMPASTPETESARASOTA	6,381

Creating a URL and passing a “hidden” variable as a column within the dataset that is being sent to _webout accomplishes the process. (NOTE: In order for SAS not to interpret special characters found in an href= .html statement (<, >,/,", etc) the "protectspecialchars=off" option is used in a proc template.) The URL created contains a field that acts as a "place holder" within the application and also defines what variables will pass into the "where" clause for the next level of summarization detail. This allows for only one SAS/Intrnet program instead of one for each step in the drill down process. In other words, when the first input screen is selected, the counter variable passed is "one". The program in a macro reads this value IF-THEN to ensure the correct level of summarization is taking place. Once level "one" produces an output screen (that is dynamic to drill down further), that "input" screen now will pass the hidden variable with a value of "two".

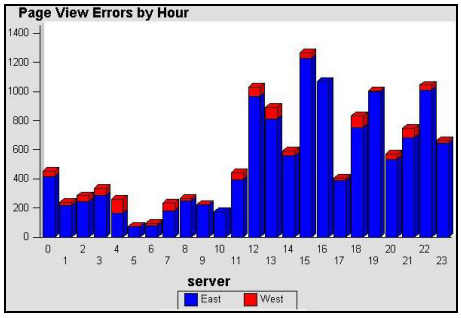
While MDDBs are still very much part of the reporting on the Management Dashboard, utilizing some tricks with a SAS/Intrnet Application has solved some of the problems with size experienced with growth.

The full program can be found in Appendix C.

JAVA DRILLDOWN GRAPHS

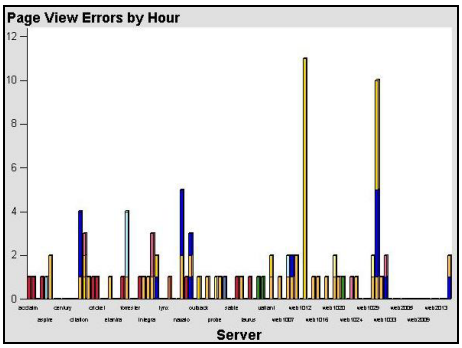
Management likes graphs. Interactive visuals delight them. One of the most vital reporting needs is the web server activity. This need includes reporting from many different angles. SAS’s drillable graphs were handy but only got us half way there.

The graph below shows errors per hour. SAS's drillable graphs allow us to see these errors split into regions.



If you were to drill into this graph using typical SAS graph, you would just see a split out of the 2 regions. Our needs were much more sophisticated. Viewing the actual servers within a region was what was needed.

To do this, 24 HTML graphs, one for each hour of the day, had to be created. The secret is to reference each of the html links as an option in a PROC GCHART. The graph below shows the results for one of the hours above. Drilling down on an individual hour in the first graph links you to this more detailed graph.



Using this embedded html graphing method gives Autotrader.com a far more advanced look and is very user friendly.

The code can be viewed in Appendix D.

4-SPEED TO AUTOMATIC USING SAS

One of the major accomplishments of the group was to move from a somewhat manual process to a fully automated one. One of the main reasons the Management Dashboard is so dynamic is that SAS is utilized from start to finish. SAS is so capable in so many areas that use of other platforms is unnecessary. Using specific SAS tools and tricks at crucial steps allows processing to flow seamlessly.

Many things have been learned along the way. In the beginning, there were many steps running independently of each other. Workspace was not efficiently utilized and controlled. Restarts of daily processes caused many headaches, were inconsistent and could trigger problems in other areas. There is nothing worse than thinking a process was running effectively overnight, than to come in the next morning to no updates and hours of backpedaling. A more automated system using more SAS tools was needed.

Today, updates to the site are 99% automated. Absolute control is obtained using SAS macros to control process pathing and kick-offs. As more and more individual programs were added to the overall process, this became increasingly necessary.

One of the most effective "tricks" added to the process was to flag system failures, stop the processing at proper points at these occurrences, and even notify contacts when these problems occur. Using the statement below creates a flag when an unacceptable error occurs and notifies the contact.

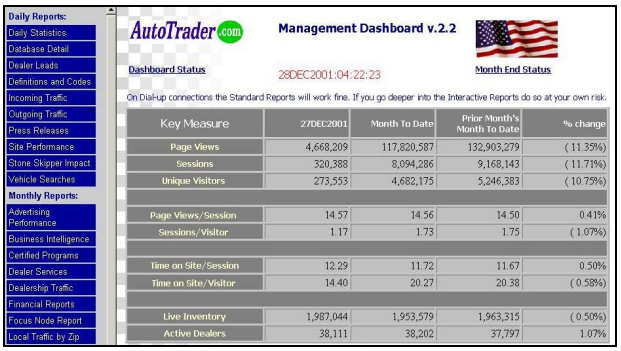
```
%if &syserr > 3 %then %let failed=Y;
data _null_;
 failing='yes';
 filename test email
  ("john.smith@autotrader.com")
  subject="&mess";
 file test;
 put "Daily Update Program Stopped at
  &sysdate &systime" ;
 if failing='yes' then abort abend;
run;
```

Problems still occur. This is inevitable with massive growth. Such standards set within the ever-changing process flow allows the Business Intelligence group to manage and maintain control.

MANAGEMENT DASHBOARD REDESIGN

All output for Business Intelligence goes to the Management Dashboard. It has grown up in piecemeal. As new reports were added, many initially adhocs, layers continued to build, some not as logical as others. Business Intelligence has decided to redesign the system to address some of the weaknesses and add more functionality.

NAVIGATION: FRAMES TO JAVASCRIPT



The Management Dashboard is what Autotrader.com refers to its website used for the company's reporting. It is everyone's link to information about the company. Everything from website traffic to vehicle search information is maintained on the Management Dashboard. This site was created using frames with the menu running along the left side of the screen. The user can choose from a variety of monthly or daily report tabs, and the corresponding list of reports will appear in the main part of the window. This gets the user to the report, but there are some problems with it.

The reporting is no longer limited to daily and monthly reports, but incorporates many other time frames and reporting needs. As our report output shifts and grows to meet the business needs, the actual tabs no longer correspond well with output. Frames also do not allow bookmarks to the sub page. This means users have to renavigate each time they want to access a different report. This also makes it harder to report bugs and/or forward information about these sub pages. The URLs shown and the HTML title for the page do not correspond to the actual report. It was time to upgrade.

Navigation in the new and improved Management Dashboard will be done primarily through a JavaScript toolbar on the top of the screen. This toolbar will have the main categories of reports, with dynamic pull down menus and submenus making it far easier to

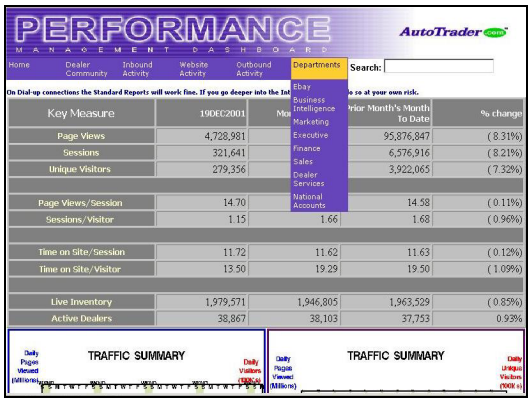
locate reports. The JavaScript code to apply this can be found on many sites. It is important that the system be very updateable. To this end, a SAS dataset of all reports created is maintained along with the location in the menuing system. Daily, a SAS program will dynamically create the JavaScript code that drives the menuing system. If a new report is added, an observation is added to the dataset and the menu will update itself.

RECATEGORIZING REPORT TABS

At present, there are 22 tabs on Management Dashboard. These tabs no longer break our reports in to clean subdivisions. It is often unclear what tab will bring you to a given report. With the new pull-down menus, the 22 tabs are collapsed into 5 tabs. With the JavaScript pull down menus allowing the collapse of many sub areas under one main category, navigation should be more straightforward. An added benefit in eliminating the frames method is the 25% increase in the amount of screen width available for reports.

The introduction of a search engine is another aid to navigation to be incorporated into the Management Dashboard. The user can type keywords and get back a list of reports that fit those criteria. A free search engine called ht://dig is currently being evaluated. Other engines are being considered. SAS reporting will have to be updated to include meaningful <title> and <meta> tags so bookmarking and searching will work properly.

Eventually the user will be able to navigate through the site in 4 ways: bookmarks, search engine, report category JavaScript selection, and departmental JavaScript selection.



OUTPUTTING REPORTS

A system is also being created whereby users can *subscribe* to given reports. The Management Dashboard user community has shown great interest in having reports directly emailed to them once they are available. (Sample emailing code is in dealer leads section earlier in this paper.) A SAS dataset is created with the user, report name, file location, email address, and time interval of report. Not only can their reports be found on the site but in their email mailbox.

DATA MART TO DATA WAREHOUSE

As Autotrader.com's web site has grown, the amount of data collected has grown exponentially. All this information requires analysis and reporting for daily, weekly, monthly levels, and custom time periods. All this information is currently stored in SAS datasets. These SAS datasets are created from a production database stored in Oracle. The Oracle data is constantly changing; therefore snapshots are taken nightly for reporting purposes.

We refer to the large number of SAS datasets created daily as a "Data Mart". While most of the company's reporting can be done using the Data Mart, other portions of the company's activities are not included. Billing and Financial data are amongst those. It is a goal to fold this information in with our current structure and create a true Data Warehouse. The ETL activity (extract-transform-load) of gathering these diverse information sources, cleaning them, and loading them into an integrated database will be a job for SAS. Information will be read from Oracle databases, flat files, Lotus Notes, SQL server, Apache web logs, and spreadsheets galore. SAS will be used to ensure consistency in field names, field types, and data quality, as information is load into the data mart. This brings into consideration typical data warehousing issues, such as who owns the data and is responsible for data integrity, which source is the best for customer name, address, and how frequently should the information be consolidated – monthly, weekly, daily, multiple times per day, or instantly as any source changes. Business rules remain a moving target. The amount of time and people required to do this successfully is large. These types of hurdles need to be overcome to build a successful all encompassing warehouse.

How do we get there? This first step is to create a table of contents for all the SAS datasets in the current Data Mart. Business rules for all the variables need to be laid out. This information will be posted on the Management Dashboard for all company employees to view. The principle here is to provide our customers a guide on how to access and pull an analysis together.

The Business Intelligence department is the only user of SAS at Autotrader.com. To do all the analysis to support the entire company would be overwhelming. By using SAS/Intrnet, MDDBs, drill down graphs and other interactive tools on the Management Dashboard, non-SAS users can do their own reporting and understand the business rules surrounding the information. A data warehouse insures these rules are consistent across the board.

The need to complete this task is obvious. Pulling together all this information will greatly improve the company's ability to determine its own health. Direct correlations can be drawn from one department's effects on another. Are marketing plans affecting revenues? Are new web designs increasing inventory? Are pricing changes affecting the number of clients? A warehouse will also further facilitate future revenue possibilities and data mining. Every department at Autotrader.com will benefit directly from a successful creation of a data warehouse.

ADVANCED DATA ANALYSIS

Reporting has become top notch, automated, and timely. A data warehouse links all the information together. Moving from data crunching to advanced data analysis is the next logical step. Two of the next opportunities at the forefront involve building a complete customer information system and market potential mapping.

Combining the multiple sources of customer information is a large challenge. With the help of an altered version of the sophisticated match/merge (Fuzzy Merge) routine provided by Charles Patridge, a significant amount of our files have been combined. The results are easily transported amongst many platforms for many departments' use. This data is logically the source for mapping advances. The mapping will allow the sales force to target potential golden opportunities.

The combination of the customer information from the many different angles also provides an excellent source for customer profiling, a 360 degree view of our customer. This is ideal for data mining for the cause and effect relationship of the customer for up-sell and cancellations understanding.

There is a true wealth of information stored within the vaults of data within Autotrader.com. Advanced Data Analysis will unleash some true opportunities moving Business Intelligence from a reporting department to a revenue-producing department.

CONCLUSION

The road has been long and exciting. SAS has provided Autotrader.com most of the tools needed to make a highly functional and respected intranet site, in a sense moving the Management Dashboard from a Chevette to a Corvette in management reporting. Autotrader.com looks forward to the enhancements coming with SAS version 9. The new SSA (SAS Scalable Architecture) may very well provide us with the next level of processing speed needed to stay ahead our continuing growth.

REFERENCES & RESOURCES

SUGI proceedings 139-24, Mark Moorman, SAS Institute Inc., The Art of Designing HOLAP Databases , 1999

SUGI proceedings 133-35, Ann Weinberger & Matthias Ender, SAS Institute Inc. The Power of Hybrid OLAP in a Multidimensional World, 2000

JavaScript: http://www.simplythebest.net/info/dhtmlscript79.html, simplythebest.net

The Fuzzy Feeling SAS Provides: Electronic Matching of Records without Common Keys; Charles Patridge, http://www.sconsig.com

ACKNOWLEDGMENTS

Special thanks to Jerry Johannesen for his vision, guidance, and allowing this team to develop our craft to new heights.

CONTACT INFORMATION

AutoTrader.com Your car is waiting.
5775 Peachtree Dunwoody Road
Suite A
Atlanta, Georgia 30342

Jenine Eason
(404) 843-7199
jenine.eason@autotrader.com

Jerry Johannesen
(404) 269-6986
jerry.Johannesen@autotrader.com

Brad Goldman
(404) 269-7712
brad.goldman@autotrader.com

Lars Olufsen
(404) 269-8758
lars.Olufsen@autotrader.com

Jeff Chang
(404) 269-8170
jeff.chang@autotrader.com

APPENDIX A – PROCESS SERVER LOGS

```
%macro launch;
%put Number of Machines is &machines ;
%do i=1 %to &&machines ;
 data temp;  set mtd.howmany;
   if _n_ = 0+&&i;
 call symput('machine',compress(machine));
  call
symput('lll',"'/scratch/"||left(machine)||"'"
);
 run;
 data _null_;
 call
symput('sublib',"/scratch/"||left("&machine")
);
  call symput('sublibnm',"libname m&&I
    '/scratch/"||left("&machine")||"';");
 run ;

x "mkdir &sublib &"; &sublibnm ;

***now read in the read method template, and
write out machine specific one ;
data inout ;
infile '/read2003.sas' lrecl=90 pad truncover
end=lastline;
  input @1 line $char89. ;
  file "/run&&i..sas" new;
  if _n_ = 1 then do;
**Establish pipe for each  machine here ;
  put "proc printto
log='/run&i._&&yyyymmdd..log' new;";
put "libname sub '&&sublib';";
put "Filename LOGS_IN pipe
'/usr/local/bin/gunzip -c /Business
Intelligence/logs/&&machine./access/*&&yy.&&m
m.&&dd.*gz';" ;
put "data _null_; call
symput('yymmdd',left('&&yymmdd')) ; " ;
put "call symput('machine',left('&&machine'))
; run; " ;
end;
put @1 line ;
if lastline then do ;
put "data sub.howmany;set mtd.dual;length
machine $24;machine='&&machine';";
put 'rawrows=0+&&rawrows ;' ;
put "complete=datetime() ; drop x ;format
complete datetime20.; output;";
put "run; proc print ;";
end;
run;
x "/sas/sas -memsize 96M -sysin /run&&i..sas
&";
x 'sleep 10';  ** Pause for 10 seconds so as
not to launch 90 sas runs at once;
%end;
%mend;
%launch ;
```

APPENDIX B – COMPLETION CHECKING

```
%macro checkds;
%let tmp=;
%let counter=0;
  %do i = 1 %to &&machines;
    %let rc=%sysfunc(exist(m&i..howmany));
     %if &rc = 1 %then %do;
      %let tmp=&tmp m&i..howmany;
      %let counter=%eval(&counter+1);
     %end;
     %if &counter=&machines %then %let
ISITDONE=YES;
  %end;
&tmp
%mend checkds;
data _null_;
 set %checkds ;
 success+1;
 rowtotal+rawrows;
 if success=0+&machines then isitdone='YES';
 else isitdone='NOPE';
 call symput('success',success);
 call symput('rowtotal',rowtotal);
run;
%mend checkho;
**Loop to check for completion every 10
minutes;
%macro complete;
%do %until("&isitdone" = "YES");
   x 'sleep 600';
%checkho;
%put The checkhowmany routine isitdone says:
&isitdone;
%put The checkhowmany routine rowtotal says:
&rowtotal;
%put The checkhowmany routine counter says:
&counter;
%put The checkhowmany routine Success says:
&success;
%end;
***Pause for a few minutes to let the
filesystem catch up;
 x 'sleep 120';
   *rc=system('sleep 180');
%put Now Its Done &&isitdone ;
%mend complete;
```

APPENDIX C - SAS/INTRNET APPLICATION

```
 * Repeat this code for each combination of variables;
%macro heir;
%if &heir.=DMA, Make, Model %then %do;

%macro round;
%if &detail.=one %then %do;

data _null_;
file "dmamkmod_&_rmtaddr..html";
put '<html><head>';
put '<title>Vehicle Searches Reporting Tool</title></head>';
put '<body background="/images/chkrd_flg.jpg">';
put '<table width="95%" border="0" align="center">';
put '<tr><td width="62%" align="center">';
put '<img src="/images/atlogo_448x54.gif" width="448"
height="54"></td>';
put '</tr><tr align="center"><td width="62%"><br>';
put '<h1><font size="4">Vehicle Searches Reporting
Tool</font></h1>';
put '</td></tr></table>';
put '<form method=GET action="/cgi-bin/broker">';
put '<INPUT TYPE="hidden" NAME="_service"
VALUE="default">';
put '<INPUT TYPE="hidden" NAME="_program"
VALUE="sample.vsMDDB1_i.sas">';
put '<INPUT TYPE="hidden" NAME="detail" VALUE="two">';
put '<INPUT TYPE="hidden" NAME="heir" VALUE="DMA,
Make, Model">';
put '<INPUT TYPE="hidden" NAME="_debug" VALUE="2">';
put '<INPUT TYPE="hidden" NAME="date"
VALUE="'&date.'">';
put '<table align="center" width="90%" <tr align="center">';
put '<td width="50%" align="center">';
```

```
put '<h2><font size="2" color=black>Please Choose Make
Filter</font></h2>';
put '</td></tr><tr><td width="50%" align="center">';
put '<select name="make" size="1" >';
put '<option selected>Any Make';
**Start make list;
data _null_;
  file "dmamkmod_&_rmtaddr..html" mod;
  set vsMDDB.make_&date.;
   make1=compress('<option>'||make);
   put make1;
run;
**Finish the html;
data _null_;
file "dmamkmod_&_rmtaddr..html" mod;
put '</select></td></tr></table><br><br>';
put '<table align="center" width="50%"<tr align="center">';
put '<td width="50%" align="center">';
put '<h2><font size="2" color=black>';
put 'Press <input align="center" option="selected"
type=submit value="Run">';
put 'to view
report</h2></td></tr></table></form></body></html>';
run;

**Output to browser;

data _null_;
  file _webout;
  put '<html> <META http-equiv="Refresh" content="1; URL=';
  put "dmamkmod_&_rmtaddr..html";
  put '"></html>';
run;
%end;
  * Repeat this then/do for the number of drill downs;
%if &detail.=two %then %do;
 %if &make.=Any Make %then %do;
  proc sort data=vsMDDB.dmamkmod_&date. out=summary;
   by dma;
  run;
 %end;
 %if &make.^=Any Make %then %do;
  proc summary
data=vsMDDB.dmamkmod_&date.(where=(make="&make."))
nway;
   var searches;
   class dma;
   output out=summary(drop=_type_ _freq_) sum=;
  run;
 %end;

data summary; set summary;
  hdma='<a href="/cgi-
bin/broker?_service=default&_program=sample.vsMDDB_i.s
as   &detail=three&heir=DMA, Make,
Model&_debug=2&date='||"&date."||'&make='||"&make."||'&d
ma='||trim(dma)||'">'||trim(dma)||'</a>';
run;

ods html file=_webout;
proc print data=summary noobs label;
  title "&date. DMA Searches for &make.";
  var hdma searches;
  sum searches;
run;
%if &detail.=three %then %do;

 %if &make.=Any Make %then %do;
 proc sort data=vsMDDB.dmamkmod_&date.
   (where=(dma="&dma.")) out=summary;
   by make;
 run;
 %end;

 %if &make.^=Any Make %then %do;
 ods html file=_webout;

 proc print data=summary(where=(dma="&dma." and
make="&make."))
  noobs label;
  var model searches;
  sum searches;
  title "&date. &dma. Searches &make. Model Detail";
 run;
 ods html close;
 %end;

proc summary data=summary nway;
  var searches;
  class make;
  id dma;
  output out=summary sum=;
run;

data summary; set summary;
  hmake='<a href="cgi-
bin/broker?_service=default&_program=sample.vsMDDB_i.s
as
  &detail=four&hier=DMA, Make,
Model&_debug=2&date='||"&date."||'&dma='||trim(dma)||'&ma
ke='||trim(make)||'">'||trim(make)||'</a>';
run;

ods listing close;
ods html file=_webout;

proc print data=summary noobs label;
 title "&date. &dma. Searches Make Summary";
 var hmake searches;
 sum searches;
run;
%end;
%mend round;
%round;
%mend heir;
%heir;
```

APPENDIX D - JAVA DRILL DOWN GRAPHING

```
*create htmls for each hour;
data outdata; set indata;
  by hour;
  if hour = 1 then links = 'href=/graphs/hour1.html'; else
  *one for each hour  1 to 24;
  if hour = 24 then links = 'href=/graphs/hour24.html';
run;

ods html file=odsout
  parameters=("drilltarget"="_self")
  attributes=("codebase"="/sasweb/graph")
  archive="graphapp.jar";

goptions reset=all nodisplay device=JAVA;
axis1 minor=none
 value=('0' '1' '2' '3' '4' '5' '6' '7' '8'
```

```
  '9' '10' '11' '12' '13' '14' '15' '16'
  '17' '18' '19' '20' '21' '22' '23' '24')
proc gchart data=outdata ;
  vbar3d hour / nolegend sumvar=pcterr
  maxis=axis1 discrete frame cframe=white
  nolegend html=links;
ods html close;
%macro hours;
%do i=0 %to 23;
  ods html file="/graphs/hour&i..html"
   attributes=("codebase"="/sasweb/graph")
   archive="graphapp.jar";
*Make chart for each hour;
proc gchart data=serverdata(where=(hour=&i));
  vbar3d page /levels=900 sumvar=pcterr
  discrete frame nolegend woutline=1;
title "Hour - &i";
ods html close;
%end;
%mend hours;
%hours;
```

APPENDIX E – WORD MACRO

```
Sub htm2doc()
'
 htm2doc Macro
'
 ChangeFileOpenDirectory "C:\TEMP\"
 Documents.Open FileName:="Dec2001.htm",
 ConfirmConversions:=False,
 ReadOnly:=False,
 AddToRecentFiles:=False,
 PasswordDocument:="",
 PasswordTemplate:="",
 Revert:=False,
 WritePasswordDocument:="",
 WritePasswordTemplate:="",
 Format:=wdOpenFormatAuto
   With Selection.Font
      .Name = "Times New Roman"
      .Size = 12
      .Bold = False
      .Italic = False
      .Underline = wdUnderlineNone
      .StrikeThrough = False
      .DoubleStrikeThrough = False
      .Outline = False
      .Emboss = False
      .Shadow = False
      .Hidden = False
      .SmallCaps = False
      .AllCaps = False
      .ColorIndex = wdBlack
      .Engrave = False
      .Superscript = False
      .Subscript = False
      .Spacing = 0
      .Scaling = 100
      .Position = 0
      .Kerning = 0
      .Animation = wdAnimationNone
   End With
   Selection.WholeStory
   Selection.Font.ColorIndex = wdBlack
   ActiveDocument.SaveAs FileName:="Dec2001.doc",
FileFormat:=  wdFormatDocument,
LockComments:=False,
Password:="",
AddToRecentFiles:= True,
WritePassword:="",
ReadOnlyRecommended:=False,
EmbedTrueTypeFonts:= False,
SaveNativePictureFormat:=False,
SaveFormsData:=False,
 SaveAsAOCELetter:=False
 ActiveDocument.Close
End Sub
Sub savemail()
'
' savemail Macro
'
End Sub
```