Paper 110-2011

To Hash or Not To Hash! That is the Question!! When it Comes to SQL!!!

Mustaq Ahmad, Merck Sharp & Dohme Corp., Upper Gwynedd, PA

ABSTRACT

This paper demonstrates how to compare the efficiency of using the hash object when the same results can be produced with PROC SQL. In the process of efficiency quantification, other efficiency determinants such as data set size, compression, index, and environmental factors (remote PC versus local laptop) are accounted for in an experimental model with a simulation approach.

KEYWORDS

Hash object, SQL, Efficiency, Statistical Modeling, Data Compression, Index, Simulation.

INTRODUCTION

The hash object is used for table lookup, and in most cases has been preferred over MERGE, SQL, ARRAY, FORMAT, and SET with KEY= procedures due to its memory-resident searching method. In this approach, the hash object performs the look up not primarily by comparison between the keys but by direct addressing. If the data sets are small then all these table lookup procedures may have non-significant differences in regard to program execution time. But dealing with the large data sets raises questions regarding one procedure's effectiveness over the other procedure. Previous literature indicates that the hash object and PROC SQL approach is more efficient (i.e. less CPU and / or real time needed) than the MERGE procedure. The efficiency of other table lookup procedures like ARRAY, FORMAT, and SET with KEY= procedures varies in different situations. In these previous approaches, most of the efficiency comparisons were made using a single program with a DATA or PROC step and one execution of the program. No other factors or data set attributes was considered in the efficiency comparison approach.

In this paper, the efficiency of two procedures, the hash object and PROC SQL, is compared taking into consideration of other factors such as data set attributes (i.e. size, compression, index) and the environmental factor (i.e. remote PC versus local laptop). This comparison is performed using statistical analytical model (Generalized Linear Model, PROC GLM) on CPU and real time in a simulation approach.

APPROACH

OVERALL SETUP FOR THE STATISTICAL ANALYTICAL MODEL

In this setup, the hash object and PROC SQL procedures are handled separately. To build the setup, each procedure (hash object or PROC SQL) is used to build a table lookup using two SAS® input data sets which generate an output data set resulting from the specific procedure used. This pair of input data sets has differing attributes such as data set size, compression, and indexing. To check whether the two procedures are performing the exact same lookup, PROC COMPARE is executed on the output data sets to ensure exact match. Each table lookup is repeated five times and CPU and real time are generated for each execution to develop a simulation environment.

The efficiency endpoints, CPU and real time, are captured from the SAS Log for each procedure (hash object or PROC SQL), and an Analysis-Ready data set is created containing the values of CPU and real time for each specific procedure. The same data set also contains all the information for extraneous factors associated to the input data sets, procedure (hash object or PROC SQL) used, number of iterations, etc.

APROACH TO STATISTICAL ANALYSIS

Subsequently, the statistical analytical model using PROC GLM is applied to the CPU and real time for the two (hash object or PROC SQL) procedures. The detailed schema of analytical setup and analysis is provided in the following Figure 1 -

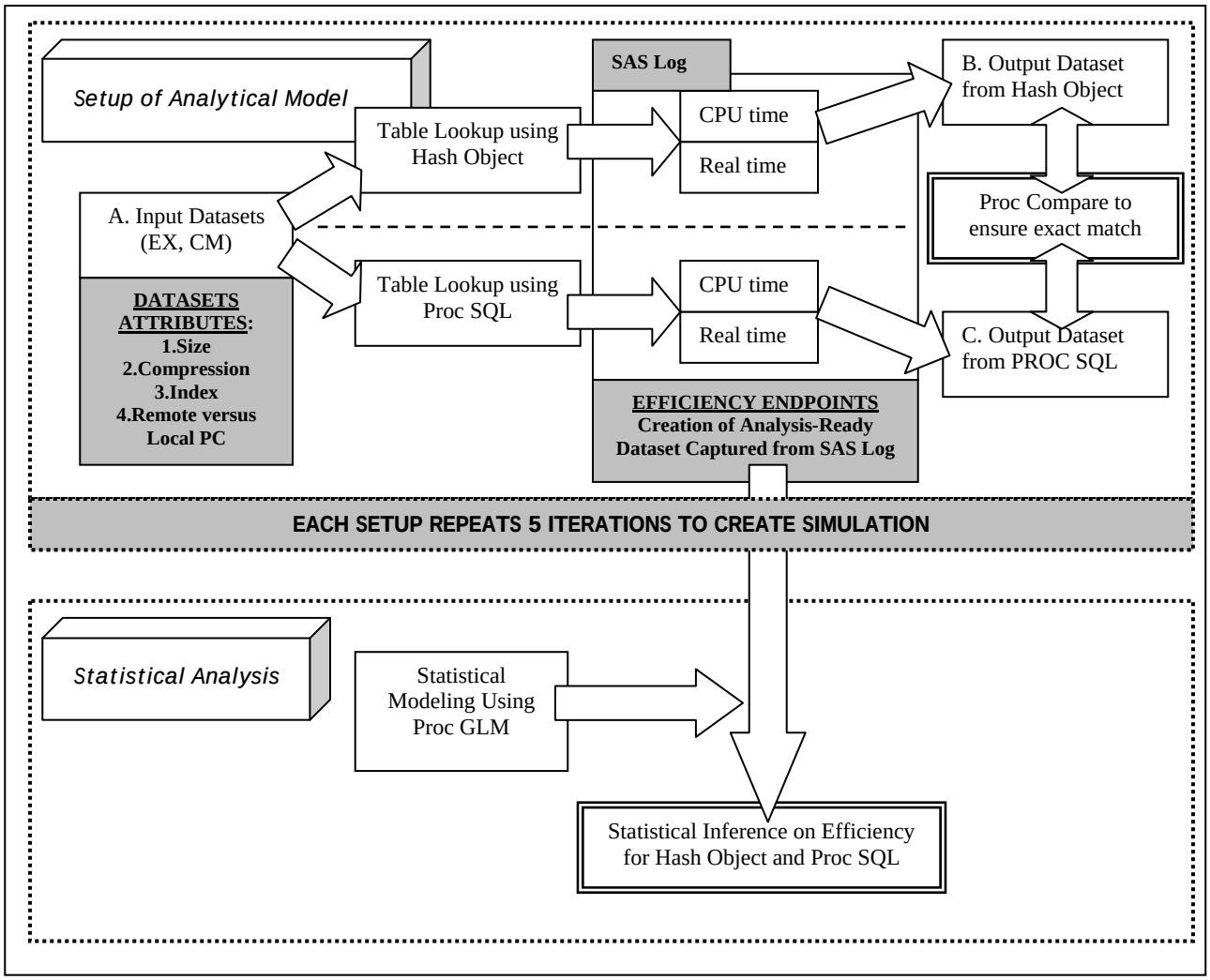


Figure 1: Schematic presentation of analytical model setup and analysis.

DESCRIPTION OF ANALYTICAL MODEL COMPONENTS

INPUT AND OUTPUT DATA SETS

The two input data sets, EX and CM, are used (A. in Figure 1). The EX data set contains the study exposure data and CM contains the concomitant medication data. EX is the larger data set, and CM is smaller one. These data sets are joined together using the hash object and SQL procedures. Both data sets for each procedure are joined using two key variables: SUBJ_ID and VISITNUM. Both data sets are not-sorted for the key variables. Also, while joining these two data sets (EX and CM), two additional variables, SPDYRLEP and EXSTDY, are used to restrict values - some are given upper and lower boundary values. Both procedures generate identical output data sets (B and C in Figure 1) as confirmed by PROC COMPARE. In the hash procedure, CM, the smaller data set, is used to create the hash object and put to the SAS buffer to make it memory-resident.

INPUT DATA SETS ATTRIBUTES

The larger input data set, EX, has the following attributes –

- Data set size: 5 gigabytes (GB) to 1GB. Each data set size with decrement of 1GB (five data sets).
- Data set compression: Compressed or not-compressed (two compression status).
- Data set index: Indexed or not-indexed (two index status)
- Data set variables: 1640836 observations (for 5GB data set) and 31 variables containing the key variables SUBJ_ID and VISITNUM

The smaller input data set, CM, has the following attributes –

- Data set size: ~16MB. (one data set).
- Data set compression: Compressed or not-compressed (two compression status).

- Data set index: Indexed or not-indexed (two index status)
- Data set variables: 3009 observations and 36 variables containing the key variables SUBJ_ID and VISITNUM

Each EX data set of varying size will be joined to the CM data set by key variables.

CODE CONSTRUCT

Figure 2 contains the programming code used to join the EX and CM data sets to produce the identical output data sets for hash object and PROC SQL –

```
/* EXCERPT OF THE CODE FROM %DT MACRO TO INDICATE JOIN BY SQL AND HASH OBJECT */
** JOIN BY PROC SQL **;
PROC SQL NOPRINT;
create table S_Dt&GIGA._&COMPRES._&INDX._&PC._&I (drop = ex_subj ex_visit) as
select a.* , b.*
       from   dtin_lib.&dtin_lg (rename =( subj_id=ex_subj  visitnum = ex_visit)) a,
              dtin_lib.&dtin_sm b
       where  a.ex_subj = b.subj_id and a.ex_visit = b.visitnum and
              50 < SPDYRLEP < 300 and  200 < EXSTDY < 400;
QUIT;
** JOIN BY HASH OBJECT **;
DATA H_Dt&GIGA._&COMPRES._&INDX._&PC._&I;
       if 0 then set dtin_lib.&dtin_sm ;
       declare hash cm(hashexp:7, dataset:"dtin_lib.&dtin_sm");
       CM.definekey('subj_id','visitnum');
       CM.definedata(all:'Y'); CM.definedone();

       do until(eof);
       set dtin_lib.&dtin_lg (where= ( 50 < SPDYRLEP < 300 and  200 < EXSTDY < 400))
              end=eof;
       if CM.find()=0 then output;
       end;
       stop;
RUN;
%if %upcase(&show_comp) = Y %then %do;
TITLE1 "COMPARE OUTPUT FROM SQL AND HASH PROCEDURE";
PROC COMPARE DATA=S_Dt&GIGA._&COMPRES._&INDX._&PC._&I
       COMPARE=H_Dt&GIGA._&COMPRES._&INDX._&PC._&I;
RUN;
TITLE;
%end;
```

```
/* EXCERPT 0F THE CODE TO SHOW THE CALL MACRO */
%DT (     no_compress  = y     /* SAS Options (compress=) usage control */
       , supp_mprint   = n     /* SAS Options (mprint) usage control */
       , dtin_lib      = Q:\ahmad\hash\final_datasets /* Library for input data sets */
       , dtin_lg       = ex1gb_notcomp_notindx /* Large data set (EX) */
       , dtin_sm       = cm    /* Small data set (CM) */
       , giga          = 1     /* Data set size (1GB) indicator */
       , compres       = NotComp      /* Data set compression status indicator */
       , indx          = NotIndx      /* Data set index status indicator */
       , pc            = Rem   /* Remote PC versus local laptop indicator */
       , iter          = 5     /* Procedure (Hash or SQL) iteration indicator */
       , show_comp     = y     /* Proc COMPARE output control */
       , debug         = n     /* Program execution and debugging control */
       , list_out      = y | Q:\ahmad\hash\list_output
                               /* Generating SAS .lst output control */
       , log_out       = y | Q:\ahmad\hash\log_output
                               /* Generating SAS Log control */
       , anal_out      = Q:\ahmad\hash\anal_ready_output
                               /* Library for analysis-ready data sets */
       , outfile_nm    = NotComp_NotIndx_Rem
                               /* Name of the analysis-ready data set */
       , append        = y
                               /* Control for append results to the existing data set */
       , delim         = |     /* Delimiter for list_out & log_out parameter */
       , caplog        = y     /* SAS Log capture to create 'Analysis-Ready' data set */
    );
```

Figure 2: SAS Programming code excerpt (%DT macro) to execute the hash object and PROC SQL procedures; the parameters to execute the procedures are presented.

CREATION OF ANALYSIS-READY DATA SET

The Analysis-Ready data set is created from the SAS log generated from the execution of the %DT macro presented in Figure 2. This data set has a specific number of columns and rows and contains all the data points to perform a statistical analysis using PROC GLM. The following Table 1 describes the determinants of the Analysis-Ready data set structure (columns and rows) and relevant information.

Procedure Name	**Input Data Set Attributes**					
Hash object & PROC SQL	Data Set Size	Compression	Index	Remote PC vs. Local Laptop	Iterations	TOTAL NUMBER OF ROWS IN ANALYSIS-READY DATA SET
2 procedures	5GB, 4GB, 3GB, 2GB, 1GB,	Yes / No	Yes / No	Remote PC / Local Laptop	5 iterations	
(2)	(5)	(2)	(2)	(2)	(5)	(2*5*2*2*2*5) = 400

Table 1: Determinants of columns and rows for Analysis-Ready data set.

STATISTICAL ANALYSIS AND SIMULATION

THE TEST HYPOTHESES

Two tailed, two sample, hypotheses testing is performed for statistical analysis approach.

The Null Hypothesis: In this hypothesis, mean CPU or real time for execution of hash object and PROC SQL is same i.e. there is no difference in mean CPU or real time.

Hence H_0: $\mu_{\text{CPU Time from hash object}} = \mu_{\text{CPU Time from PROC SQL}}$ (for CPU Time) or
H_0: $\mu_{\text{Real Time from hash object}} = \mu_{\text{Real Time from PROC SQL}}$ (for Real Time)

The Alternative Hypothesis: In this hypothesis, mean CPU or real time for execution of hash object and PROC SQL is significantly different i.e. there is significant difference in mean CPU or real time.

Hence H_A: $\mu_{\text{CPU Time from hash object}} \neq \mu_{\text{CPU Time from PROC SQL}}$ (for CPU Time) or
H_A: $\mu_{\text{Real Time from hash object}} \neq \mu_{\text{Real Time from PROC SQL}}$ (for Real Time)

THE EQUATION FOR STATISTICAL MODEL

To model the CPU and real time, the following respective Generalized Linear Models (GLM) is used –

$$Y_{\text{CPU Time}} = a + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + e$$
$$Y_{\text{Real Time}} = a + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + e$$

Where b_1 to b_5 are different coefficients for CPU and real time.

DEPENDENT VARIABLE

The CPU and real time are considered as continuous variables and are treated as the dependent variable in the statistical model. The CPU and real time are considered separately so two separate models are built.

INDEPENDENT VARIABLES

The attributes of the EX data set are considered as the independent variables for the model. These independent variables are procedure (SQL/hash), data set size, compression, index, and remote PC versus local laptop. Table 2 presents the description of the independent variables.

	Independent Variables				
	x_1	x_2	x_3	x_4	x_5
	Procedure	Data set size	Compression	Index	Remote PC vs. Local Laptop
Variable Characteristics	Categorical	Ordinal	Categorical	Categorical	Categorical
Category Indicator in the Model	SQL=0 Hash=1	5GB = 0 4GB = 1 3GB = 2 2GB = 3 1GB = 4	Yes = 1 No = 0	Yes = 1 No = 0	Remote PC = 0 Local Laptop = 1

Table 2: Description of the independent variables to use in the statistical model.

Depending on information presented in Table 1 and 2, a tabular structure of the Analysis-Ready data set is presented in the Appendix 1.

RESULT

DEFINITION AND SCOPE OF ANALYTICAL SETUP

To test the hypotheses, a controlled approach is applied to collect the CPU and real time. There is no missing data for CPU and real time and for combinations for each categorical / ordinal levels of independent variables, there are equal number of CPU and real time. Thus, the analytical setup can be considered as balanced design in an experimental setup. For building the analytical model, the differential effect of SQL and hash object on CPU and real time is considered as the main effect and this effect is determined while adjusted for other covariates as data set size, compression status, index status, and remote or local laptop usage. Thus, in the final GLM model, variable reflecting SQL/hash effect on CPU and real time is included because that is the main effect is to be determined.

PREPARATION OF THE ANALYSIS-READY DATA SET

In the analysis-ready data set the dependent variable, CPU and real time, is captured as character variables from the SAS log in the form of nn:nn:nn.n. These variables are transformed to numeric variables and for CPU and real time the assigned units are seconds and minutes respectively. No additional change is performed to the categorical / ordinal independent variables.

The SAS variables in analysis-ready data set are –

	Dependent Variable	**Independent Variables**				
	y	x_1	x_2	x_3	x_4	x_5
	CPU and Real Time	Procedure	Data set size	Compression	Index	Remote PC vs. Local Laptop
SAS Variable Name	CPUTIME_SEC REALTIME_MIN	PROC	DATA_SIZE	COMPRESS	INDEX	PC
SAS Variable Value	Numerical & Continuous	0,1	0,1,2,3, and 4	0,1	0,1	0,1
SAS Variable Value Definition		PROC=0(SQL) PROC=1(Hash)	DATA_SIZE=0(5GB) DATA_SIZE=1(4GB) DATA_SIZE=2(3GB) DATA_SIZE=3(2GB) DATA_SIZE=4(1GB)	COMPRESS=0(Not Compressed) COMPRESS=1(Compressed)	INDEX=0(Not Indexed) INDEX=1(Indexed)	PC=0(Remote) PC=1(Local Laptop)
Variable's Role in the GLM model		Predictor for Primary Effect for the GLM Model	Used for Covariate Adjustment	Used for Covariate Adjustment	Used for Covariate Adjustment	Used for Covariate Adjustment
		Primary Effect	Covariates			

Table 3: Description of the variables in analysis-ready data set.

There are five ordinal groups for the data set size. To interpret the effect of the data set size on the CPU and real time, dummy coding is introduced for the variable DATA_SIZE. The framework for the dummy coding for data set size is –

	Data Set Size				
Variable (variable value) [Data Set Size]	**1GB**	**2GB**	**3GB**	**4GB**	**5GB**
DATA_SIZE (0) [5GB]	0	0	0	0	1
DATA_SIZE (1) [4GB]	0	0	0	1	0
DATA_SIZE (2) [3GB]	0	0	1	0	0
DATA_SIZE (3) [2GB]	0	1	0	0	0

Thus the effect (b_2 coefficient) of independent variable X_2 can be interpreted as the additive effect of – b_{2i} DATA_SIZE (0) [5GB] + b_{2j} DATA_SIZE (1) [4GB] + b_{2k} DATA_SIZE (2) [3GB] + b_{2l} DATA_SIZE (3) [2GB] (Where b_2 is replaced by b_{2i}, b_{2j}, b_{2k}, and b_{2l} and these coefficients may or may not be different). For other covariates (PROC, COMPRESS, INDEX, and PC), the values are dichotomous and one coefficient is appropriate to build the GLM model.

FITTING THE SIMPLE LINEAR REGRESSION MODEL USING GLM AND CHECKING OF THE ASSUMPTIONS

To perform the preliminary exploration of the collected data several Simple Linear Regression (SLR) models are developed to determine the effect of individual independent variable on CPU and real time. For each fitted model, the parameter estimate, p-value, and R^2 are determined. In SLR approach it is evident that PROC variable has no significant effect on the CPU and real time. But, DATA_SIZE, COMPRESS, INDEX, and PC have significant effects on the CPU and real time at the $\alpha = 0.05$ level and considered as the significant predictors for the CPU and real time. For the real time, these independent variables individually explain between 3% to 13% of the total variance. For the CPU time the same variables explain between 4% to 14% of the total variance.

Even some of the independent variables became significant in the SLR models, checking for the model assumptions is necessary to determine the validity of the SLR models. Residual plots are performed to check the assumptions required for SLR models. In the residual plots, residual data points are plotted against the predicted values and the independent variables to check for normality, linearity, and heteroscedasticity. Existence and independence assumption can not be checked and assumed to be present. From the residual plots it is verified that normality and linearity assumptions are satisfied and heteroscedasticity is not satisfied. To address the assumption of heteroscedasticity, the dependent variables, CPU and real time, are transformed using LOG (natural log with base e) function and residual plots are generated. LOG transformation addressed the issue of heteroscedasticity and thus following two new variables are created in the analysis-ready data set –

```
REALTIME_MINLOG = log(REALTIME_MIN)
CPUTIME_SECLOG  = log(CPUTIME_SEC)
```

Subsequently, these two variables are used as dependent variables for future model determination.

FITTING MULTIPLE LINEAR REGRESSION MODEL USING GLM AND CHECKING OF INTERACTIONS BETWEEN COVARIATES

As determination of the primary effect of PROC (SQL/hash) on the CPU and real time while the effects of other covariates are fixed is the main hypotheses of this paper, a Multiple Linear Regression (MLR) model is developed. In process of building the MLR model, the effect of interaction terms are explored. The approach to explore the interaction terms is to determine whether the effect of PROC variable on CPU and real time (log transformed) depends on ordinal values of DATA_SIZE used. Interactions are explored for different combinations of pair of independent variables (covariates) using the plotted graphs and the variable pairs which show interaction effects are DATA_SIZE*COMPRESS, DATA_SIZE*INDEX, DATA_SIZE*PC, COMPRESS*INDEX, COMPRESS*PC, and INDEX*PC. The subsequent MLR models are built using GLM procedure with and without interaction terms.

The MLR model without interaction terms provides evidence that the primary effect of PROC (SQL/hash) on CPU and real time is significant (at the $\alpha = 0.05$ level) after fitting the covariates in the model. The R^2 is increased significantly in comparison to the SLR model. For MLR model using real and CPU time the R^2 values become 0.72 and 0.68 respectively. Thus, the model without interaction terms explains 72 percent of the total variance of real time and 68 percent of the total variance of CPU time respectively. Beyond the primary effect, this MLR model also provides similar trend of significant effect between real time and CPU time. For both real time and CPU time as dependent variable, the covariates COMPRESS, INDEX, PC also become significant at the $\alpha = 0.05$ level. The covariate DATA_SIZE is significant for larger data sets (3GB – 5GB).

For the MLR model with selected interaction terms, similar trend of significance is found. The R^2 increased slightly in comparison to the MLR without interaction terms. For real time and CPU time it is 0.88 and 0.87 respectively. The primary effect of PROC(SQL/hash) on CPU and real time remains significant as it is the same for larger data set categories of covariate DATA_SIZE. In this model, the effects of COMPRESS, INDEX, and PC is significant at α = 0.05 level when CPU time is the dependent variable and not significant for real time as the dependent variable. COMPERSS*INDEX, COMPRESS*PC, and INDEX*PC interactions are significant for both CPU and real time as dependent variables.

AUTOMATED MODEL SELECTION

To verify the validity of the above mentioned MLR models with and without interactions terms, a set of automated model selection techniques are used. Among the techniques available following are the preferred ones –

- Adjusted R^2
- Mallows' C(p) Statistic
- Akaike Information Criterion (AIC) and
- Automated Covariate Selection Using Forward, Backward and Stepwise procedures with GLMSELECT

Using the adjusted R^2, C(p) statistic, and AIC the best fitted predictors are identified in hierarchy and depending on all three selection techniques model with PROC, DATA_SIZE, COMPRESS, INDEX, and PC as predictors becomes the model of choice. The best fitted model is determined by highest adjusted R^2, lowest Mallows' C(p) and AIC. This approach is used to determine the individual effects only.

Automated model selection with interaction terms is not possible using PROC REG. A procedure, PROC GLMSELECT, is available in SAS/STAT® software that performs model selection in the framework of general linear models. When it comes to perform model selection with interaction terms within the model, this procedure is of choice as it can accommodate interaction terms in the model.

To build the model using PROC GLMSELECT, all the covariates are included in the model to determine the individual effects of these covariates on the CPU and real time. This approach is also supported by the previous findings when only covariates (PROC, DATA_SIZE, COMPRESS, INDEX, PC) with individual effects are used in the GLM model. Also, all the possible pair-wise combinations of covariates are included in the model to determined the best possible regression model fit using the individual and interaction terms. The stepwise selection procedure is used with best regression fit selection using significance level (SL) as selection procedure and with SLENTRY = 0.15 and SLSTAY = 0.25. To include the PROC, DATA_SIZE, COMPRESS, INDEX, and PC in the model INCLUDE=5 is used. The selected covariates (individuals and interactions) are determined for CPU and real time as dependent variables and the selected covariates are used for creation for the final model using PROC GLM. The output of PROC GLMSELECT showed the selected covariates are different depending on the CPU and real time as dependent variables. The difference in the selected predictors for CPU and real time are reflected in the following final model.

THE FINAL MODEL

The final model is built depending on the findings from the GLMSELECT. PROC GLM is used to determine the covariate effects. The findings of the final MLR model are –

Independent Variables	Parameter Estimate (Log Transformed)	p-value

Final MLR model with interaction terms: Real time as dependent variable
`PROC GLM DATA=ALL;`
`  class proc data_size compress index pc;`
`  model realtime_minlog = proc data_size compress index pc`
`  proc*index data_size*compress data_size*index`
`  data_size*pc compress*index compress*pc index*pc`
`  / solution;`
`QUIT;`		

	R^2=0.89 / Intercept=-4.869	
Individual Effects:		
PROC (SQL)	1.702	<0.0001
DATA_SIZE (5GB)	0.927	0.0017
DATA_SIZE (4GB)	0.693	0.0190
DATA_SIZE (3GB)	0.464	0.1153
DATA_SIZE (2GB)	0.075	0.7984

COMPRESS (Not Compressed)	-0.248	0.3137
INDEX (Not Indexed)	0.818	0.0020
PC (Remote)	0.087	0.7249
Interaction Effects:		
PROC(SQL)*INDEX(Not Indexed)	-1.310	<0.0001
DATA_SIZE(5GB)*COMPRESS(Not Compressed)	0.011	0.9714
DATA_SIZE(4GB)*COMPRESS(Not Compressed)	0.315	0.2851
DATA_SIZE(3GB)*COMPRESS(Not Compressed)	0.101	0.7311
DATA_SIZE(2GB)*COMPRESS(Not Compressed)	0.783	0.0081
DATA_SIZE(5GB)*INDEX(Not Indexed)	0.618	0.0362
DATA_SIZE(4GB)*INDEX(Not Indexed)	0.572	0.0525
DATA_SIZE(3GB)*INDEX(Not Indexed)	0.676	0.0220
DATA_SIZE(2GB)*INDEX(Not Indexed)	0.033	0.9107
DATA_SIZE(5GB)*PC(Remote)	0.519	0.0786
DATA_SIZE(4GB)*PC(Remote)	-0.075	0.7994
DATA_SIZE(3GB)*PC(Remote)	0.047	0.8721
DATA_SIZE(2GB)*PC(Remote)	-0.548	0.0634
COMPRESS(Not Compressed)*INDEX(Not Indexed)	2.868	<0.0001
COMPRESS(Not Compressed)*PC(Remote)	0.858	<0.0001
INDEX(Not Indexed)*PC(Remote)	3.071	<0.0001

Final MLR model with interaction terms: CPU time as dependent variable
`PROC GLM DATA=ALL;`
`  class proc data_size compress index pc;`
`  model cputime_seclog = proc data_size compress index pc`
`  proc*index data_size*index compress*index compress*pc`
`  index*pc / solution;`
`QUIT;`		

	R^2=0.88 / Intercept=-1.1057	
Individual Effects:		
PROC (SQL)	0.937	<0.0001
DATA_SIZE (5GB)	0.900	<0.0001
DATA_SIZE (4GB)	0.647	<0.0001
DATA_SIZE (3GB)	0.493	<0.0001
DATA_SIZE (2GB)	0.310	0.0109
COMPRESS (Not Compressed)	-0.652	<0.0001
INDEX (Not Indexed)	0.911	<0.0001
PC (Remote)	-0.788	<0.0001
Interaction Effects:		
PROC(SQL)*INDEX(Not Indexed)	-0.703	<0.0001
DATA_SIZE(5GB)*INDEX(Not Indexed)	0.313	0.0681
DATA_SIZE(4GB)*INDEX(Not Indexed)	0.340	0.0478
DATA_SIZE(3GB)*INDEX(Not Indexed)	0.293	0.0881
DATA_SIZE(2GB)*INDEX(Not Indexed)	-0.035	0.8367
COMPRESS(Not Compressed)*INDEX(Not Indexed)	0.828	<0.0001
COMPRESS(Not Compressed)*PC(Remote)	1.666	<0.0001
INDEX(Not Indexed)*PC(Remote)	1.895	<0.0001

Table 4: Parameter estimates and p-values for final MLR model with interactions.

The findings of the final MLR model with interactions is similar to the MLR model with interactions described previously with some minor differences. The final MLR model has slightly higher R^2. In the final model PROC*INDEX is included which was not present in the previous model. Some of the interaction terms are omitted in the final model depending on the outcome of automated model selection using PROC GLMSELECT. Findings are comparable between the final model and the previous GLM model. In final model, primary effect of PROC is significant as the previous model. In both models individual effect of DATA_SIZE (larger data set size) and INDEX are significant. In the final model the interaction between PROC*INDEX becomes significant. Other interaction terms – COMPRESS*INDEX, COMPRESS*PC, and INDEX*PC are significant in both the final and previous models.

THE INTERPRETATION OF THE PRIMARY EFFECT IN THE FINAL MODEL

The primary effect of PROC(SQL/hash) can be summarized in the following tables –

Dependent Variable: Real Time					
Effect of PROC(SQL/hash) on real time (minutes) when data set size is 5GB, not compressed, and remote PC is used					
Independent Variables & Intercept	**Model Estimates**	**X_i (SQL)†**	**Estimate*X_i (SQL)**	**X_i (Hash)†**	**Estimate*X_i (Hash)**
Intercept	-4.869	1	-4.869	1	-4.869
PROC (SQL)	1.702	1	1.702	0	0
DATA_SIZE (5GB)	0.927	1	0.927	1	0.927
DATA_SIZE (4GB)	0.693	0	0	0	0
DATA_SIZE (3GB)	0.464	0	0	0	0
DATA_SIZE (2GB)	0.075	0	0	0	0
COMPRESS (Not Compressed)	-0.248	1	-0.248	1	-0.248
INDEX (Not Indexed)	0.818	1	0.818	1	0.818
PC (Remote)	0.087	1	0.087	1	0.087
PROC(SQL)*INDEX(Not Indexed)	-1.310	1	-1.310	0	0
DATA_SIZE(5GB)*COMPRESS(Not Compressed)	0.011	1	0.011	1	0.011
DATA_SIZE(4GB)*COMPRESS(Not Compressed)	0.315	0	0	0	0
DATA_SIZE(3GB)*COMPRESS(Not Compressed)	0.101	0	0	0	0
DATA_SIZE(2GB)*COMPRESS(Not Compressed)	0.783	0	0	0	0
DATA_SIZE(5GB)*INDEX(Not Indexed)	0.618	1	0.618	1	0.618
DATA_SIZE(4GB)*INDEX(Not Indexed)	0.572	0	0	0	0
DATA_SIZE(3GB)*INDEX(Not Indexed)	0.676	0	0	0	0
DATA_SIZE(2GB)*INDEX(Not Indexed)	0.033	0	0	0	0
DATA_SIZE(5GB)*PC(Remote)	0.519	1	0.519	1	0.519
DATA_SIZE(4GB)*PC(Remote)	-0.075	0	0	0	0
DATA_SIZE(3GB)*PC(Remote)	0.047	0	0	0	0
DATA_SIZE(2GB)*PC(Remote)	-0.548	0	0	0	0
COMPRESS(Not Compressed)*INDEX(Not Indexed)	2.868	1	2.868	1	2.868
COMPRESS(Not Compressed)*PC(Remote)	0.858	1	0.858	1	0.858
INDEX(Not Indexed)*PC(Remote)	3.071	1	3.071	1	3.071
			∑(Estimate*X_i) for SQL = 5.052		∑(Estimate*X_i) for Hash = 4.660
∑(Estimate* X_i) for SQL - ∑(Estimate* X_i) for Hash = 5.052 – 4.660 = 0.392 Inv(ln) of 0.392 = 1.48 (Value obtained from inverse of natural log)					
Interpretation$^{\pi}$: For Not Indexed data set: To perform SQL procedure compared to hash procedure will take 1.48‡ minutes more when adjusted for data set size, compression status, and independent of procedure run on remote PC or local laptop. For Indexed data set: To perform SQL procedure compared to hash procedure will take [Inv(ln) of 1.702] or 5.48§ minutes more when adjusted for data set size, compression status, and independent of procedure run on remote PC or local laptop.					
† X_i = Dummy coding indicator. $^{\pi}$ Interaction between PROC(SQL) and INDEX(Not Indexed) become significant and thus effect of data set indexing on real time varies depending on indexed or not-indexed status of the data sets. ‡ Calculated from this table using dummy coding and coding is showed in table. § Calculated from parameter estimate for PROC (SQL) and dummy coding is not showed in tabular form.					

Table 5: Interpretation of effect of PROC(SQL/hash) on real time adjusted for other covariates in the final MLR model.

Dependent Variable: CPU Time					
Effect of PROC(SQL/hash) on CPU time (second) when data set size is 5GB, not compressed, and remote PC is used					
Independent Variables and Intercept	**Model Estimates**	**X_i (SQL)†**	**Estimate*X_i (SQL)**	**X_i (Hash)†**	**Estimate*X_i (Hash)**
Intercept	-1.105	1	-1.105	1	-1.105
PROC (SQL)	0.937	1	0.937	0	0
DATA_SIZE (5GB)	0.900	1	0.900	1	0.900
DATA_SIZE (4GB)	0.647	0	0	0	0
DATA_SIZE (3GB)	0.493	0	0	0	0
DATA_SIZE (2GB)	0.310	0	0	0	0

COMPRESS (Not Compressed)	-0.652	1	-0.652	1	-0.652
INDEX (Not Indexed)	0.911	1	0.911	1	0.911
PC (Remote)	-0.788	1	-0.788	1	-0.788
PROC(SQL)*INDEX(Not Indexed)	-0.703	1	-0.703	0	0
DATA_SIZE(5GB)*INDEX(Not Indexed)	0.313	1	0.313	1	0.313
DATA_SIZE(4GB)*INDEX(Not Indexed)	0.340	0	0	0	0
DATA_SIZE(3GB)*INDEX(Not Indexed)	0.293	0	0	0	0
DATA_SIZE(2GB)*INDEX(Not Indexed)	-0.035	0	0	0	0
COMPRESS(Not Compressed)*INDEX(Not Indexed)	0.828	1	0.828	1	0.828
COMPRESS(Not Compressed)*PC(Remote)	1.666	1	1.666	1	1.666
INDEX(Not Indexed)*PC(Remote)	1.895	1	1.895	1	1.895
			∑(Estimate*X_i) for SQL = 4.202		∑(Estimate*X_i) for Hash = 3.968

∑(Estimate* X_i) for SQL - ∑(Estimate* X_i) for Hash = 4.202 – 3.968 = 0.234
Inv(ln) of 0.234 = 1.26 (Value obtained from inverse of natural log)

Interpretation[π]:
For Not Indexed data set:
To perform SQL procedure compared to hash procedure will take 1.26[‡] seconds more when adjusted for data set size, compression status, and independent of procedure run on remote PC or local laptop.
For Indexed data set:
To perform SQL procedure compared to hash procedure will take [Inv(ln) of 0.937] or 2.55[§] seconds more when adjusted for data set size, compression status, and independent of procedure run on remote PC or local laptop.

[†] X_i = Dummy coding indicator.
[π] Interaction between PROC(SQL) and INDEX(Not Indexed) become significant and thus effect of data set indexing on CPU time varies depending on indexed or not-indexed status of the data sets.
[‡] Calculated from this table using dummy coding and coding is showed in table.
[§] Calculated from parameter estimate for PROC (SQL) and dummy coding is not showed in tabular form.

Table 6: Interpretation of effect of PROC(SQL/hash) on CPU time adjusted for other covariates in the final MLR model.

DISCUSSION

Previously, programming efficiency comparisons were conducted among all the available table lookup approaches such as MERGE, ARRAY, FORMAT, SQL, hash, and SET with KEY= procedure. In this paper, only the efficiency comparison between the hash object and PROC SQL procedures is performed. The rationale for comparing these two procedures is that these procedures are very alike in terms of CPU and real time taken for program execution. The approach for statistical analysis in this experimental set up is chosen to differentiate any minor efficiency differences between these two procedures. In the approach mentioned in this paper, the program code and the experimental setup only compared the programming efficiency i.e. CPU or real time and does not account for the time needed to develop these procedure codes (human factor) or any other resource needed for code maintenance.

For model building, log (natural log) transformation of CPU and real time is performed to have better fitted model. R^2, adjusted R^2, significance level and other statistical criteria are used for automated model building. Pair-wise covariate interactions are included in the model for fitting the MLR models. More than two covariate interaction is not included in the model for possible over-fitting of the MLR models. The remote PC and the local laptop are chosen so that they are comparable in terms of processor speeds, RAMs, and disk storage spaces.

CONCLUSION

Although there are statistical significances for differences in effects of SQL and hash procedures on CPU and real time, in respect to real life execution time these differences are very minimal (Table 5 and 6). Thus the efficiency gain using SQL versus hash procedure is not very meaningful in daily life. Even both the procedures are comparable in efficiency, it is the end users who decide which procedure (the hash object or PROC SQL) is to be used to appropriately meet their needs considering the resource (time, personnel, computer configurations, etc.). The actual gain in programming efficiency can be achieved only by perceived experience from repeated use of related procedures over time. The result presented in this paper is to lay out guidance and provide a recommendation only. It is the end-user who will finally judge and choose the right procedure to achieve programming efficiency.

REFERENCES

1. Qi, Eric and Fikret Karahoda (2010). "TIPS AND TRICKS OF EFFICIENT SAS® PROGRAMMING FOR SDTM DATA". SESUG 2010 Annual Conference, Savannah, GA, September 26 - 28, 2010.
2. Muriel, Elena (2007). "Hashing Performance Time with Hash Tables". SAS® Global Forum 2007 Annual Conference, Orlando, Florida, April 16 - 19, 2007.
3. Secosky, Jason and Janice Bloom (2007). "Getting Started with the DATA Step Hash Object". SAS® Global Forum 2007 Annual Conference, Orlando, Florida, April 16 - 19, 2007.
4. Loren, Judy (2006). "How Do I Love Hash Tables? Let Me Count The Ways!". NorthEast SAS® Users Group Inc. (NESUG) Annual Conference, Philadelphia, Pennsylvania, September 17 - 20, 2006.
5. Cohen, Robert A (2006). "Introducing the GLMSELECT PROCEDURE for Model Selection". SAS® Global Forum 2006 Annual Conference, San Francisco, California, March 26 – 29, 2006.
6. Rohrbough, Rob (2005). "Table Lookups…You Want Performance?". Nebraska SAS® Users Group, November 12, 2005.
7. Dorfman, Paul M. and Gregg Snell (2003). "HASHING: GENERATIONS". SUGI 2003 Annual Conference, Seattle, Washington, March 30 - April 2, 2003.
8. Lafler, Kirk P. (2000). "Efficient SAS® Programming Techniques". SUGI 2000 Annual Conference, Indianapolis, Indiana, April 9 - 12, 2000.

ACKNOWLEDGEMENT

I would like to thank Robert Hoffman and Maryann Williams for all their assistance in reviewing this paper and providing the valuable suggestions.

CONTACT INFORMATION

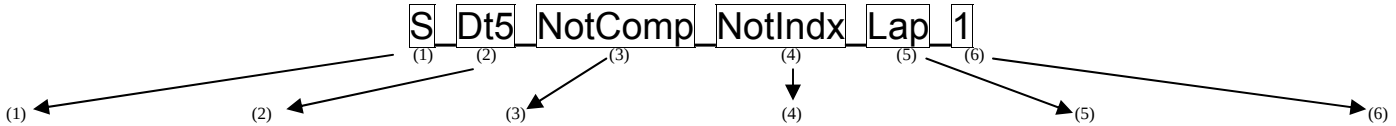
Your comments and questions are encouraged and appreciated. The author can be contacted at:

Mustaq Ahmad
Merck Sharp & Dohme Corp.
351 N. Sumneytown Pike
P.O. Box 1000, UG1CD-14
North Wales
PA 19454
mustaq_ahmad@merck.com

Appendix 1: Structure of Analysis-Ready Data Set.

Row #	Procedure (2)	Data Set Size (5)	Compress (2)	Index (2)	Remote PC vs. Local Laptop (2)	Iteration (5)	Procedure and Data Set Attribute Combined Indicator*	CPU Time (Sec)	Real Time (Min)
1	SQL	Dt5	NotComp	NotIndx	Lap	1	S_Dt5_NotComp_NotIndx_Lap_1	nn.n	nn.n
2	SQL	Dt5	NotComp	NotIndx	Lap	2	S_Dt5_NotComp_NotIndx_Lap_2	nn.n	nn.n
3	SQL	Dt5	NotComp	NotIndx	Lap	3	S_Dt5_NotComp_NotIndx_Lap_3	nn.n	nn.n
4	SQL	Dt5	NotComp	NotIndx	Lap	4	S_Dt5_NotComp_NotIndx_Lap_4	nn.n	nn.n
5	SQL	Dt5	NotComp	NotIndx	Lap	5	S_Dt5_NotComp_NotIndx_Lap_5	nn.n	nn.n
26	Hash	Dt3	Comp	NotIndx	Rem	1	H_Dt3_Comp_NotIndx_Rem_1	nn.n	nn.n
27	Hash	Dt3	Comp	NotIndx	Rem	2	H_Dt3_Comp_NotIndx_Rem_2	nn.n	nn.n
28	Hash	Dt3	Comp	NotIndx	Rem	3	H_Dt3_Comp_NotIndx_Rem_3	nn.n	nn.n
29	Hash	Dt3	Comp	NotIndx	Rem	4	H_Dt3_Comp_NotIndx_Rem_4	nn.n	nn.n
30	Hash	Dt3	Comp	NotIndx	Rem	5	H_Dt3_Comp_NotIndx_Rem_5	nn.n	nn.n
56	Hash	Dt1	Comp	Indx	Rem	1	H_Dt1_Comp_Indx_Rem_1	nn.n	nn.n
57	Hash	Dt1	Comp	Indx	Rem	2	H_Dt1_Comp_Indx_Rem_2	nn.n	nn.n
58	Hash	Dt1	Comp	Indx	Rem	3	H_Dt1_Comp_Indx_Rem_3	nn.n	nn.n
59	Hash	Dt1	Comp	Indx	Rem	4	H_Dt1_Comp_Indx_Rem_4	nn.n	nn.n
60	Hash	Dt1	Comp	Indx	Rem	5	H_Dt1_Comp_Indx_Rem_5	nn.n	nn.n
400	Hash	Dt1	Comp	Indx	Lap	5	H_Dt1_Comp_Indx_Lap_5	nn.n	nn.n

*Legend for Procedure and Data Set Attribute Combined Indicator:



Procedure	Date set Size	Compression	Index	Remote PC vs. Local Laptop	Iteration
S=SQL, H=Hash	Dt5=5GB, Dt4=4GB, Dt3=3GB, Dt2=2GB, Dt1=1GB	NotComp=Not compressed, Comp=Compressed	NotIndx=Not indexed, Indx=Indexed	Rem=Remote PC, Lap=Local Laptop	1=First, 2=Second, 3=Third, 4=Fourth, 5=Fifth

COPYRIGHTS