

A Comparison of Techniques for Extracting and Scoring data from a Teradata Data Base using SAS

Randy T. Rist, Ph. D.
JCPenney Co. Inc.
IS Merchandising
Statistical Modeling

This paper compares six (6) of seven (7) Teradata Access Methods using the BTEQ and FASTEXPORT Teradata utilities and the NCR-ODBC drivers in conjunction with SAS PROC ACCESS/ODBC. These Teradata Access Methods have been paired with three ways of writing SAS code to extract and score customer data.

This study suggests that the worst possible way to extract and score data from the Teradata database is to use the BTEQ utility to write an ASCII flat file, use SAS to re-format the flat file to a SAS data set and then execute a SAS scoring program. On the test data, this approach uses 17 X more storage and 6.2 times more elapse time than the best way tested. The best approach uses FASTEXPORT with a customized OUTMOD exit written in 'C' to write binary data to a Unix standard out and the SAS *infile* statement to read the binary data from the Unix standard out file through an *unnamed pipe*. This procedure is 6.2 times faster and uses 1/17th the amount of storage space of the worst possible approach.

Two or three promising methods are yet to be explored. Some actual test code is included with this paper.

INTRODUCTION

Scoring is the assignment of a value to a customer based upon a weighted combination of customer attributes. At JCPenney, in Target Marketing and Catalog, these attributes are stored in a Teradata relational database. In the Teradata, as in other relational databases, data is stored in a way which conserves both space and reduces access time. Unhappily, these goals do not produce a data structure suited to a scoring process.

To illustrate, the entity sales summary table is organized with three columns, one for a customer id, one for an associated entity in which one or more purchases are made and one for the total number of dollars spent by the customer. Although this organization may be superior for speed of access and may minimize storage space requirements, it is difficult to apply all but the simplest scoring equations to data so organized.

Scoring equations may require the use of cross products between two or more variables. For instance:

$$y = 5 + .78*ent001 + 2.51*ent001*ent099$$

shows the use of cross products between dollar purchases in entity one and entity 99. To compute the value of 'y' - the customer id's 'score' - the value of entity 1 and entity 99 must be present in memory at the same time. A program must be written which will retrieve the data for each customer, restructure the data so an individual's dollar purchases for each entity in which he has made a purchase is held in memory. Any entity in which the customer id has not made a purchase will not be stored in the data base, yet if that entity appears in the equation, it must be given a value. In this case a value of zero (0).

Restructuring data from a series of n rows for each person to a single row for each person scored is called often called pivoting.

Besides the program for restructuring the data, scoring must retrieve the data from the Teradata database. Retrieval from databases often depends upon the existence of proprietary retrieval utilities. There are three basic proprietary retrieval utilities for Teradata databases. The first is BTEQ, the second is FASTEXPORT and the third is the NCR-ODBC drivers. All three employ SQL (Standard Query Language) syntax for data retrieval.

Each of these three utilities has its own advantages and disadvantages that differ along the dimensions of *output, internal Teradata processing limitations and interface characteristics*. For example, BTEQ can produce output in binary and alphanumeric character format. The character output can be redirected from standard out to a flat file. Binary output can be sent to a flat file only. Standard out is not available.

FASTEXPORT produces binary output to a flat file, yet it has an exit that can be used by a 'C' programmer to redirect the output, either character or binary to standard out.

Regarding Teradata processing limitations, a BTEQ query may generate up to 16 'sessions', a FASTEXPORT query may produce 64 sessions and an ODBC query can generate only 1 session.

BTEQ has the most liberal interface limitations of the two Teradata utilities. It produces alphanumeric character formatted output to standard out and the standard out can be redirected to a flat file without additional programming. FASTEXPORT on the other hand, requires additional 'C' programming to accomplish the same variety of output vectors.

The ODBC drivers require a user interface. The variety of output choices for ODBC depends upon the user interface.

The requirement to restructure the data and the variety of data retrieval options provided by the two Teradata utilities and the NCR-ODBC drivers affords an opportunity to evaluate the performance of various combinations of restructured code and retrieval options involved in potential scoring scenarios.

This report presents Scoring Elapse Time and Maximum Bytes Stored when using BTEQ queries writing both character and binary formatted files, BTEQ queries writing character data to pipes, FASTEXPORT queries and NCR-ODBC drivers processing SQL queries via the SAS Proc Access procedure.

Even though the tables and SQL queries are identical for all Teradata access methods, scoring procedure elapse times vary by a factor of 6.2. Maximum File Size stored can vary by scoring procedure by a factor of 17.

METHOD

MEASURES

The methods presented here are used to establish the performance levels of various scoring procedures along two dimensions, Scoring Elapse Time and Maximum Bytes Stored at any one time.

Scoring Elapse Time is time it takes for the result set of a standard SQL query to be returned, scored and stored.

Maximum Bytes Stored is the greatest number of bytes stored on disk at any one time.

The shorter the Scoring Elapse Time the better the performance. The fewer the number of bytes stored at any one time the better the performance.

DESIGN

The design characteristics have been determined by the nature of the problem and contemporary practice at JCPenney Catalog. At one time, Catalog had a 'Three Landings' procedure for scoring their data. A result set was placed in a flat file using a Teradata Access Method. The flat file was read by SAS and converted and stored as a SAS Data Set. SAS reads the SAS Data Set, the data is scored and a SAS Data Set containing the customer scores. This is a 'Three Landings' procedure because the data is stored first as a flat file, second as a SAS Raw Data Set and third as a SAS Scored Data Set.

A 'Two Landings' procedure for scoring data can be conceived in two ways. The first is a flat file being produced by a Teradata Access Method, followed by a SAS job reading the flat file, scoring the data and producing a SAS Scored Data Set. One flat file and one SAS Scored Data Set exist at the same time, therefore a 'Two Landings' procedure. The second way to produce 'Two Landings' is by having a Teradata Access Method pipe the SQL query result set to a SAS job which reads the data and produces a SAS Raw Data Set. Then a second SAS job reads the SAS Raw Data Set and produces the second SAS Scored Data Set. Both 'Two Landings' procedures are included in this studies design.

A 'One Landings' procedure for scoring data can be built by having the Teradata Access Method pipe the SQL query result set to a SAS job which restructures the result set, scores it and produces a SAS Score Data Set.

'Landings' refer to the number of files containing an SQL query's result set or a result set's transformation produced by a scoring procedure.

There are three basic Teradata access mechanisms. They are BTEQ, FASTEXPORT and NCR-ODBC drivers. Both BTEQ and FASTEXPORT can vary the result set output vectors. Each of these access mechanisms has more than one output vector. And each has a different syntax or requires more or less programming skill and knowledge to execute. For the purposes of this study, a basic Teradata access mechanism and each mechanism's various output vectors are called Teradata Access Methods.

Seven (7) Teradata Access Methods have been identified, six have been tested in this study. The seven (7) Teradata Access Methods are:

BTEQ-C:	BTEQ with Character output.
BTEQ-B:	BTEQ with Binary output.
FEXP-B:	FASTEXPORT with Binary output.
BTEQ-CP:	BTEQ with Character output piped.
BTEQ-BP:	BTEQ with Binary output piped.
FEXP-BP:	FASTEXPORT with Binary output piped.
NCR-ODBC:	NCR-ODBC using SAS PROC ACCESS ODBC interface to produce SAS Data Set output.

Pipes are a technical means of passing data between two Unix processes. There are two kinds of pipes. The first is a 'named pipe.' A named pipe is created by the 'MKNOd' NCR Unix command. It is referenced by two or more Unix processes operating simultaneously, as though the named pipe were a file. The second kind of pipe is the 'unnamed pipe.' When using an unnamed pipe, one process reads from the standard output of another process. Named pipes are not used in this study. All references to pipes refer to the unnamed variety.

All Teradata Access Methods execute the same SQL query. The query is:

```
Select cust_id, entity, entity_sales
From ENTITY_SALES_SUMM
Where cust_id IN(select cust_id from RFM where score=555)
Order by cust_id;
```

Briefly, this query examines a table called RFM in the Teradata Test data base, selects all customer identifiers whose associated RFM score is 555, and uses these customer identifiers to access and return the customer identifier entity value and dollar amount purchased from the entity sales summary table. Cust_id is the primary key for the Entity_Sales_Summ table.

The basic design matrix for this study consists of three levels of landings and seven levels of Teradata Access Methods. This produces a matrix of twenty-one (21) cells. Each cell represents a scoring procedure. A scoring procedure is constituted of a particular Teradata Access Method and one or more SAS programs required to produce the number of data 'Landings'.

Because the NCR-ODBC drivers can only activate a single Teradata session, all access methods are studied with Teradata sessions set to 1.

DATA COLLECTION

The tables exist on a two Teradata node, one SAS node test machine. The total number of observations in the Entity_Sales_Summ table corresponding to all customer identifiers whose RFM score is equal to 555 is 1,597,331. These 1.6 million observations represent the data collected on 144,124 customers for 62 active retail entities.

Three test macros are written in the SAS macro and basic language. The test macros contain the SAS Data Steps that are the components of the scoring procedures. In each macro, the starting time and end time of each SAS component is recorded and the elapse time calculated and stored. Each macro contains a loop and executes 25 times. Between each Component, a 30 second delay is programmed. This is done to make certain that the Teradata database has completed all its house keeping duties before the next component of the test is initiated.

The intent was to construct a single macro containing all the components. However, the original NCR-ODBC drivers were too slow to be included so they were omitted in the first macro. When the updated NCR-ODBC drivers were available, a separate test loop was built to test them. Finally, a third test loop macro was built when it was learned how to write the FASTEXPORT OUTMOD procedure that sends binary data to the standard output file for use by unnamed pipes.

All performance data is stored in a SAS Library. It is then evaluated using the SAS Univariate Procedure. The raw performance data can be found in Appendix C. The summarized data can be found in Tables I-VI in the Results section.

PERFORMANCE DISPLAY AND EVALUATION

The primary performance evaluation data is displayed as a three-dimensional graph (FIGURE 1). The vertical axis (z) representing Scoring Elapse Time, the seven valued blocks on the x axis representing Teradata Access Methods, and the three valued blocks on the y axis representing both the number of landings or files produced and roughly the number of bytes stored at one time.

Supporting evidence for the elapse time and number of bytes stored can be found in the accompanying tables.

For ease and accuracy of interpretation, if any scoring procedure elapse time is within fifteen seconds of another, it is fair to say that there is no difference between the two procedures. Otherwise, the elapse times of the procedures being compared should be considered different from one another.

RESULTS

TABLE I

Teradata Access Methods Elapse Time Data

Summarized for 25 Trials

OBS	Process Function	Number of Trials	Mean Elapse Time(sec.)	Std Error of Elapse Time Mean	Elapse Time Std Dev
1	MAKE-CHARDATA	25	920.00	11.5964	57.9821
2	CHARFLAT-CED	25	124.58	0.7620	3.8100
3	SAS.CED-SAS.SCR	25	47.67	0.1115	0.5574
4	PIPE-SAS.CED	25	1026.39	7.1559	35.7796
5	PIPE-SAS.SCR	25	1009.12	7.3146	36.5729
6	BTEQ-Bin-Export	25	257.88	2.3585	11.7923
7	BTEQ-BINFLAT-CED	25	103.58	1.9984	9.9921
8	FEXP-Bin-Export	25	180.15	5.7380	28.6901
9	FXP-FBINFLAT-SCR	25	67.11	3.4479	17.2396
10	NCRodbc.CED	25	448.60	1.9059	9.5293
11	NCRodbc.SCR	25	458.26	1.9286	9.6428
12	FEXP Bin-CED	25	196.09	2.5881	12.9433
13	FEXP Bin-SCR	25	169.74	1.1148	5.5741

TABLE II

Teradata Access Methods Elapse Time Data

Most Extreme Trial Value Removed

OBS	Process Function	Number of Trials	Mean Elapse Time(sec.)	Std Error of Elapse Time Mean	Elapse Time Std Dev
1	MAKE-CHARDATA	24	913.80	10.2152	50.0443
2	CHARFLAT-CED	24	124.21	0.6892	3.3765
3	SAS.CED-SAS.SCR	24	47.63	0.1048	0.5135
4	PIPE-SAS.CED	24	1022.73	6.4086	31.3958
5	PIPE-SAS.SCR	24	1003.62	5.0275	24.6298
6	BTEQ-Bin-Export	24	256.57	2.0438	10.0123
7	BTEQ-BINFLAT-CED	24	102.63	1.8304	8.9673
8	FEXP-Bin-Export	24	178.67	5.7806	28.3189
9	FXP-FBINFLAT-SCR	24	64.95	2.8087	13.7597
10	NCRodbc.CED	24	447.49	1.6178	7.9257
11	NCRodbc.SCR	24	457.24	1.7114	8.3843
12	FEXP Bin-CED	24	194.34	1.9896	9.7471
13	FEXP Bin-SCR	24	169.16	1.7731	4.8487

TABLE III
Selected Comparisons Among Teradata Access Methods
And Scoring Components Using Mean Trial Elapse Time Data
From Table I.

Access Method	Description	Elapse Seconds
BTEQ - Character	Teradata to Ascii Flat File	920 secs
BTEQ - Binary	Teradata to Binary Flat File	257 secs
FastExp - Binary	Teradata to Binary Flat File	180 secs

Access Method to SAS data set	Description	Elapse Seconds
BTEQ CHAR to SAS DS	BTEQ Ascii Via Pipe to SAS DS	1,026 secs
BTEQ CHAR to SAS SCR	BTEQ Ascii Via Pipe to Score	1,009 secs
ODBC Drivers - SAS DS	SAS Proc Access - old drivers	4,917 secs
ODBC Drivers - SAS DS	SAS Proc Access - new drivers	448 secs
ODBC Drivers - SAS SCR	SAS Proc Access to Score	458 secs
FEXP Binary to SAS DS	FEXP Binary Via Pipe to SAS DS	194 secs
FEXP Binary to SAS SCR	FEXP Binary Via Pipe to SAS SCR	169 secs

Flat File to SAS	Description	Elapse Seconds
Flat CHAR to SAS DS	Char Via 'Infile' to SAS DS	124 secs
Flat BIN to SAS DS	Bin Via 'Infile' to SAS DS	104 secs
Flat BIN to SAS SCR	Bin Via 'Infile' to Score	67 secs

SAS DS to SAS Score DS	Description	Elapse Seconds
SAS DS -Scoring -SAS DS	SAS DS Via 'Set' to SAS Score	48 secs

TABLE IV
A Comparison of Selected Scoring Techniques
Using Different Combinations of Teradata Access Methods
And Scoring Components.
Mean Trial Elapse Time Data From Table III.

Three Landings	Description	Component Elapse Seconds	Total Seconds
BTEQ Character	Flat-SAS-Score	920 + 124 + 48	1,092 secs
BTEQ Binary	Flat-SAS-Score	257 + 124 + 48	427 secs
FASTEXP Binary	Flat-SAS-Score	180 + 124 + 48	352 secs

Two Landings	Description	Component Elapse Seconds	Total Seconds
BTEQ Char PIPE	SAS-SAS Score	1,026 + 48	1,074 secs
BTEQ Char	Flat-SAS Score	920 + (124*.64)	999 secs
BTEQ Bin	Flat-SAS Score	257 + 67	324 secs
FASTEXP Binary	Flat-SAS Score	180 + 67	247 secs
ODBC	SAS-Score	448 + 48	486 secs

One Landings	Description	Component Elapse Seconds	Total Seconds
BTEQ Char Pipe	BTEQ PIPE-SAS Score	1,009	1,009 secs
ODBC	Proc Access-Score	458	458 secs
FASTEXP Bin Pipe		169	169 secs

TABLE V

**Rows and Bytes of Data Stored for
Teradata Access Methods and Scoring Components.**

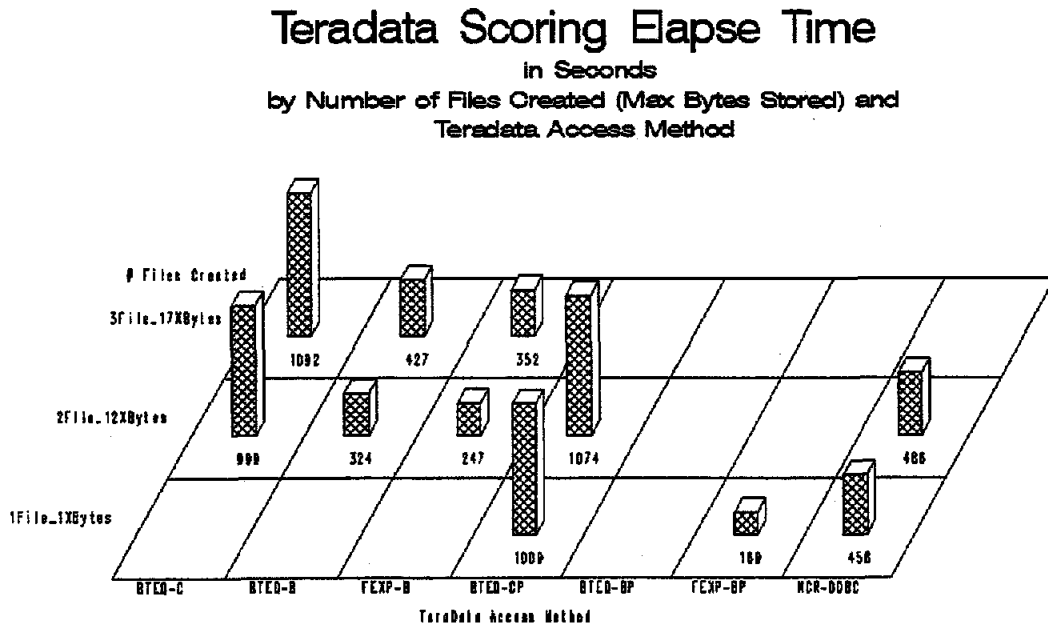
Access Method and/or Scoring Component	Description	Rows Stored	Bytes Stored
BTEQ Character Data	Character Data Flat File	1,597,331	47,921,323
BTEQ Binary Data	Binary Data Flat File	1,597,331	20,765,303
FTEQ Binary Data	Binary Data Flat File	1,597,331	17,570,641
SAS Formated Data	SAS formatted Data Set	1,597,331	38,723,584
SAS Scored Data	SAS Score Data Set	144,124	3,506,176

TABLE VI

**Mean Rows and Bytes of Data Stored for
Seven Teradata Scoring Methods
By Number of Landings**

Teradata Scoring Methods	Description	Maximum Rows Stored	Maximum Bytes Stored
Three Landings - Flat File and Two SAS Data Sets			
BTEQ-C	BTEQ Character - SAS DS - SAS SCR	3,194,662	68,686,629
BTEQ-B	BTEQ Binary - SAS DS - SAS SCR	3,194,662	59,488,887
FEXP-B	FEXP Binary - SAS DS - SAS SCR	3,194,662	56,294,225
Two Landings - Two SAS Data Sets			
BTEQ-C	BTEQ-C SAS DS - SAS SCR	1,741,455	42,229,760
BTEQ-B	BTEQ-B SAS DS - SAS SCR	1,741,455	42,229,760
FEXP-B	FEXP-B SAS DS - SAS SCR	1,741,455	42,229,760
BTEQ-CP	BTEQ CP SAS DS - SAS SCR	1,741,455	42,229,760
BTEQ-BP	BTEQ BP SAS DS - SAS SCR	1,741,455	42,229,760
FEXP-BP	FEXP-BP SAS DS - SAS SCR	1,741,455	42,229,760
NCR-ODBC	OCBC SAS DS-SAS SCR	1,741,455	42,229,760
One Landing - One SAS Data Set			
BTEQ-CP	BTEQ-CP - SAS SCR	144,124	3,506,176
BTEQ-BP	BTEQ-BP - SAS SCR	144,124	3,506,176
NCR-ODBC	OCBC - SAS SCR	144,124	3,506,176
FEXP-BP	FEXP-BP - SAS SCR	144,124	3,506,176

FIGURE 1



The '-C', '-B', '-CP' and '-BP' suffixes following the BTEQ and FEXP roots on the Teradata Access Method mean character, binary, piped character and piped binary data respectively. As can be seen the least efficient method in terms of amount of data stored (file size) and the amount of elapse time required is using BTEQ-C utility to write character data to a flat file, reading the flat file, converting it to a SAS data step and then scoring.

The most efficient method in terms of amount of data stored (file size) and the amount of elapse time required is using the FASTEXPORT (FEXP-B) utility to write binary data to an unnamed pipe, which is read by SAS and then scored and placed in a SAS data set.

There are twenty-one possible cells in Teradata Scoring Elapse Time matrix. Only eleven of the twenty-one have been evaluated. Four of the empty or untested cells, those four in the back row (3File_17Xbytes row), do not seem worth pursuing. Either do those empty cells in the middle row (2File_12Xbytes row). However, the 1File_1Xbytes row empty cells starting with BTEQ-B and including FEXP-B and BTEQ-BP are worth pursuing. In these cases, the Teradata utilities produce Binary output. After speaking with technicians from Teradata, it was learned that there may be some unsupported unpublished Teradata documentation showing how to write routines which will allow BTEQ to score binary data and either write the scored output to a flat file or to the standard out. This speculation is being researched.

The FEXP-B cell represents FASTEXPORT writing scored output using and OUTMOD exit.

Summary and Conclusions

The worst possible practice for extracting and scoring data using the Teradata Utilities and the SAS system is to:

1. Extract the data using character or ascii output.
2. Write more than one flat file output.
3. Convert the to a SAS formatted data set.
4. Score the converted data step.

The best possible practice for extracting and scoring data using the Teradata Utilities and the SAS system is to:

1. Select a Teradata Access Method that produces Binary output and sends it to Standard Out.
2. Read the Standard Out using SAS's *Unnamed Pipe* statements.
3. Read and score the data in one SAS data step.

A good intermediate practice for extracting and scoring data using the Teradata Utilities and the SAS system is to:

1. Use NCR's ODBC drivers and SAS's PROC ACCESS/ODBC.

Finally, it should be remembered that all tests were conducted with the Teradata 'Sessions' statement set to one (1). This is because the NRC ODBC drivers can and are limited to opening only one (1) session. The Teradata utilities, on the other hand, can open many more sessions simultaneously. In practice this seeming limitation of the NRC ODBC drivers can be circumvented by writing the SQL query so that more than one NCR ODBC request must be initiated to answer the query. It also should be noted that there is a point not only of diminishing, but degraded returns when increasing the number of sessions. More sessions do not necessarily imply better performance.

FASTEXPORT – binary out (fexp.cde)

These are the **FASTEXPORT** commands used to select customer identifying number, entity purchased and dollars worth of merchandise purchased from the **ENTITY_SALES_SUMM** table on the Teradata.

As with all commands, **BTEQ** and **NCR - ODBC**, the sessions have been set to *one (1)* in order to make the comparisons.

The “**EXPORT**” command indicates the ‘file’ to which the retrieved data is sent in record formatted binary. In this case, the ‘file’ is a ‘C’ coded and compiled **OUTMOD** exit to a ‘standard out’ pipe.

The SQL command selects customer identifier, entity purchased and dollars worth of merchandise purchased from the **ENTITY_SALES SUMM** table on the Teradata. The customer identifier is the primary key for the **ENTITY_SALES_SUMM** table. Each specific customer identifier is itself selected from the **RFM** table if the value of variable ‘**SCORE**’ is equal to 555. The data is ordered by customer identifier prior to being returned to the SAS node.

```
.LOGON rrist,xxxxxxxx;
.Logtable rrltmt.rrist;

.BEGIN EXPORT SESSIONS 1;

.EXPORT OUTFILE -/pipeout.so MODE RECORD FORMAT TEXT;

select cust_id, entity, entity_sales
from ENTITY_SALES SUMM
Where cust_id IN( select cust_id from RFM where score=555)
order by cust_id;

.END EXPORT;
.LOGOFF;
```

```
***** 'C' code used to compile OUTMOD EXIT TO PIPE *****;
***** file named 'pipeout.c' *****;
```

```

* * * * *
/*
/* Purpose This procedure is called by the Teradata FastExport */
/*          utility for each response row returned to the host. */
/*          The procedure examines each row and directs the output */
/*          to one streams */
/*
/** * * * * *
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>

```

```
/* Define the structure of the response row */
struct tranlog
```

```
{
    int          cust_id;
    short        entity;
    int          entity_sales;
}
```

```
;
FILE *file0;
```

```
static int recsize = 0 ;
static int counts[01] = 0;
```

```
int _dynamn(      EntryType,
                  StmtNo,
                  Resplen,
                  RespRec,
                  OutLen,
                  OutRec)
```

```
int          *EntryType;
int          *StmtNo;
int          *Resplen;
struct tranlog *RespRec;
int          *OutLen;
char         *OutRec;
```

```
{
```

```
// case on entry type
```

```
switch (*EntryType) {
```

```
case 1:
```

```
    // open file
    file0 = fopen("_f0.p", "w");          // datafile
    break;
```

```
case 2:
```

```
    // EOF for response data
    printf ("Records on named pipe = ", counts[0]);
    fclose(file0);
    break;
```

```

case 3:
    // WRITE RECORD;
    counts[0]++;

    recsize =    fwrite(RespRec,*RespLen,1,stdout );
    fprintf(stdout,"\n");

    break;

case 4:
    // no checkpoints to worry about
    break;

case 5:
    // DBC restart - close and reopen the output files
    break;

case 6:
    // Host restart same as normal since there are no checkpoints
    break;

default:
    printf("Invalid entry code = %d\n", *EntryType);
    break;
}
return(0);
}

```

***** Compile and Link *****

```
cc -c pipeout.c
```

```
ld -dy -G pipeout.o -o pipeout.so
```

```

****SAS TEST CODE for FASTEXPORT unnamed Pipe test;

options errors=0      nonotes      nosource nomacrogen nosource2 linesize=75;
**rrist 010:00;
filename fexppipe pipe 'fexp < /home/rrist/outmod/fexpomod';

%macro testit3(inwork=);

proc delete data=work.timeData;
run;
%do I = %eval(1) %to %eval(25);

*****;
*** This uses NCR FASTEXPORT and an OUTMOD EXIT to write a SAS formatted
file;
*****;

Data work.timePscr(keep=step f);
  retain l f d 0;
  length Step $ 16;
  retain step;
  step='FEXP-PIPE-CED';
  f=(datetime());          ** begining seconds in the day;
  output work.timePscr;

data work.fexppipe(keep=custid entity dollars );
  infile fexppipe LENGTH=inptlen;
  RETAIN INPTLEN kp 0;
  format dollars dollar10.2;
  n+1;
  if inptlen ^=10 or n = 1 then do;
    input @1 text $ 4. @;
    kp=inptlen;
    end;
  if inptlen = 10 then do;
    input @1 custid ib4. @5 entity ib2. @7 dollars ib4.2 ;
    kp=inptlen;
    end;
  if inptlen ^=10 then delete;

data work.timePscr(keep=step f l dif);
  set work.timePscr;
  retain l d 0;
  l=(datetime());          ** ending seconds in the day;
  dif = l - f;              ** elapse time;
  step='FEXP-PIPE-CED';
  output work.timePscr;     ** output time and step name
information;
run;

proc delete data=work.fexppipe;

data work.waitamin;
  do _i_ = 1 to 0600000000;
    n+1;
  end;
run;

proc append base=&inwork..timedata data=work.timePscr;
proc delete data=work.timePscr;          run;

```

```

*****;
*** This use FEXP and PIPE to write a SAS SCORED and Formated File;
*****;
option nosource nosource2 nomacrogen;

Data work.timePscr(keep=step f);
  retain l f d 0;
  length Step $ 16;
  retain step;
  step='FEXP-PIPE-SCR';
  f=(datetime());          ** begining seconds in the day;
  output work.timePscr;

data work.fexppipe(keep=icustid score0 score1 );
  infile fexppipe LENGTH=inptlen;
  array dol(101) dol001-dol101;          ** define array dollar;
  retain icustid dol001-dol101 custid ent dollars score0 score1
INPTLEN kp n 0;

  format dollars dollar10.2;
  if inptlen ^=10 then do;

    input @1 text $ 4. @;
    end;
  if inptlen ^=10 then do;
    delete;
    end;
  else if inptlen = 10 then do;
    input @1 custid ib4. @5 entity ib2. @7 dollars ib4.2 ;
    if inptlen ^=10 then do;
      custid=icustid;
      link outit;
      delete;
    end;

    if N = 0 then do;          ** on first observation do;
      icustid=custid;          ** assign incoming custid to
      custid;
      n=1;
    end;
    if icustid= custid then do;          ** if custid EQ incoming
      custid, do;
      if ent < 1 or ent > 101 then ent=101;
      dol(ent)=dollars;
    end;

    else if icustid ^= custid then do;          ** if last custid NE current
      custid,do;
      if ent < 1 or ent > 101 then ent=101;
      link outit;          ** link to outit;
    end;
  end;

return;
outit:
  **place transformantions here;
  **place scoring equations here;
  score0 = 5*dol(05) + 2*dol(13) + 0;          ** scoring equation number 1;
  do _i_ = 1 to 101;          ** access every dol array
  position;
  score1=score1+dol(_i_);          ** scoring equation number 2;

```

```

    ** re initialize;
    dol(_i_)=0;
    '0' value;
    end;

    output work.FEXPPIPE;
set;

    ** re initialize;
    score0=0; score1=0;
of '0';
    ** assign;
    icustid=custid;
    custid;
    dol(ent)=dollars;
array pos; return;

    return;
run;
data work.timePscr(keep=step f l dif);
    set work.timePscr;
    retain l d 0;
    l=(datetime());
    dif = l - f;
    step='FEXP-PIPE-SCR';
    output work.timePscr;
information;
run;

proc append base=&inwork..timedata data=work.timePscr;
proc delete data=work.FEXPPIPE;

data work.waitamin;
    do _i_ = 1 to 0600000000;
        n+1;
    end;
run;

%end;

Proc Print data=&inwork..timedata;
var step f l dif;
title 'timing data';
run;

%mend;

%testit3(inwork=work);

```