Paper 79-28

UNLOADing Data From Informix

John E. Bentley and Bala Pedi, Wachovia Bank, Charlotte, NC

Abstract:

The SAS/Access Interface to Informix provides the ability to run native Informix stored-programs and SQL commands in SAS and thereby let programmers to take advantage of the power of the database engine itself. When working with a parallel database in a data warehouse or data mart environment, this is approach is almost mandatory for getting reasonable query performance. The issue, though, then becomes landing the results set as a SAS data set. Streaming millions of records from numerous database nodes back to the SAS node for sequential writing to disk is a slow process. Performance is degraded further when there are other users doing the same thing. To solve this problem, we can use a SAS macro that takes advantage of Informix's Pload utility to first land the data in parallel as a set of flat files and then input them into SAS. The result is a performance boost that cuts the clock time for creating a SAS data set by 30% or more. Although this paper is written from an Informix perspective, it may be used as a template for Oracle, DB2, and other databases that have a bulk unload facility.

Disclaimer: The views and opinions expressed here are those of the author and not his employer. Wachovia Bank does not necessarily use the macro discussed here.

The Problem

Massively parallel processing (MPP) systems are very common (mandatory, some would argue) for terabyte-class data warehouses. These systems require database software that is specially designed to reduce load time, query response time, and administrative overhead. Commercial database management systems (DBMS) for MPP data warehouses include IBM's DB2 Enterprise Edition and the Informix Extended Parallel Server. The Teradata database is designed specifically for parallel data warehousing, but Oracle takes a different approach by using a proprietary version 9i technology called Real Application Clusters to provide multi-node scalability instead of offering a separate database product. (For an overview of parallel processing, see "An Introduction to Parallel Computing" in the SUGI 25 Proceedings.)

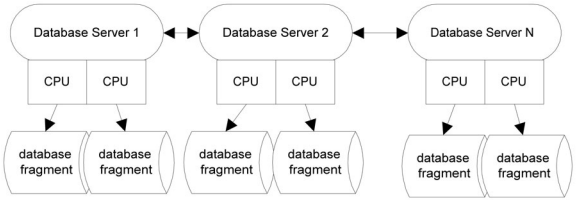
By using the appropriate interface from SAS/Access, SAS Software provides easy access to each of these parallel databases. With SAS/Access, much or even all of the processing is handed off to the underlying DBMS so that the parallel processing capabilities of the database can be exploited. Joins to multiple tables are passed directly to the database so that individual rows of data don't have to be extracted and processed by SAS itself. For loading data, SAS uses the bulk loader utility available with the database to load multiple records as a single unit. The dynamic LIBNAME make it simple to use a DATA or PROC step to list contents, read, create, update, or delete DBMS tables.

Although SAS is a great tool for working with Informix XPS particularly, the author has found there to be a performance problem when a large query results set is landed as a SAS-format data set.

A Little Background on Informix XPS

Owned by IBM Corporation since April 2002, the Informix Extended Parallel Server (XPS) database is designed specifically for MPP systems. With its shared-nothing architecture, the database is a perfect fit for MPP's shared-nothing hardware architecture. A copy of the software runs on each processor in a multiprocessor server/node with the database fragmented among sets of storage disks and each set is associated with one node. Data is not shared across the servers, but instead the XPS software tracks which data is assigned to each server. During processing, the individual servers cooperate with each other to provide a single more powerful database server.

Figure 1. Informix XPS Configuration



Informix XPS is designed to provide the data source for decision support applications but not OLTP. It can support OLAP, but better solutions exist for storing frequently accessed summarized data. XPS is excellent at handling queries that are typical of those submitted for decision support purposes, such as monthly performance reporting. These queries may require full table scans, multiple table joins and the creation of temporary memory tables to hold interim results. Response time is measured in minutes or possibly hours, not the seconds needed for acceptable OLAP and OLTP performance

When a query is submitted to XPS, the primary server determines the location of the requested data and sends the request to that server for simultaneous execution and return of the results set. If a query accesses a table that is fragmented across servers, then the I/O request is distributed to the participating servers and the results set is returned to the primary server.

If the query requires data from multiple tables spread across multiple servers, the primary server develops a "query plan" and divides it into sub-plans based on the fragmentation schemes of the tables involved and the availability of resources for connecting the participating servers. Each sub-plan is processed simultaneously and memory is used intensively. Because each sub-plan retrieves only a part of the requested data, processing time is drastically reduced.

The need for SAS

Informix XPS is a powerful database engine but doesn't have an interface that makes its power readily available to users. Its non-GUI native SQL editor resembles a line editor from the late 1980s. It also has no graphic and advanced analytical features. So SAS is needed to exploit and leverage the database's power.

In its role as a decision support application SAS excels at querying databases and transforming the results sets into actionable information. Informix XPS provides good performance for processing queries and retrieving results into a temporary memory table, and SAS provides exceptional performance for transforming and analyzing SAS format data sets.

The problem is moving the data from the query results set into a SAS data set. When creating a large dataset ("large" being a relative concept based a combination of the number of records and fields per record), using PROC SQL and the CREATE TABLE statement results in noticeably slow performance. "Using SAS to Extract from a Parallel RDBMS" in the Fall 2002 issue of The SESUG Informant, looks at the different ways to land an Informix query results set and gives some performance comparisons.

The Solution

The %UNLOAD macro is based on two features of Informix XPS.

1. It can store query results sets in temporary memory tables.
2. It has a high-performance parallel loader and unloader "utility" called Pload/XPS.

Pload is not a separate utility but instead is a standard set of SQL commands that move data into or out of the database in parallel and in the process transforms it between the Informix raw representation and delimited ASCII, fixed ASCII, or EBCDIC format.

The key feature of Pload is its use of an Informix External Table. An external table is a sort of logical link to a flat file or set of identical flat files accessible to Informix and having a file structure mirroring that of a table within the database. The file structure is implicitly defined when the external table is created:

```
CREATE EXTERNAL TABLE <name>
SAMEAS <table>
```

The external table is used to as the source or target to load or unload the database table in a constant stream. The %UNLOAD macro described below creates an external table with a structure identical to a temporary Informix memory table that is holding the query results. Processing is speeded up by having Pload unload simultaneously to multiple identical tables, which differ from one another only by a numeric suffix attached to their common name. The number of files to be created is embedded in the SQL, and the best number depends on the size of the results set. The sample code uses 4, but the value could be a parameter.

The %UNLOAD macro's program flow is:

1. Using PROC SQL, extract data from the database into a memory table.
2. Create an external file with a structure that mirrors that of the memory table.
3. Load the external file as a set of delimited flat files.
4. Create SAS data set containing a two record sample from the memory table.
5. QUIT the SQL procedure.
6. Run PROC CONTENTS against the sample data set, saving the output.
7. Sort the Contents output so that the order of the records—one record per variable—is the same as the physical order of the data set.
8. Write a SAS program that will read in the set of flat files and create a permanent SAS data set. To get the variable names and informats, use a SET statement to read the data set containing the Contents output.
9. Run the program.

Sample program code is attached.

Here are some performance timings comparing the PROC SQL CREATE TABLE AS syntax with the UNLOAD macro. Each query was run three times; the values shown are an average.

Table 1. Performance Timings

	Query Time	Results Set	Create Table time	%UNLOAD time
Monthly Analysis and Reporting	0:04:53	5.8M rows and 30 columns	0:12:23	0:09:11
Monthly Reporting	1:44.92	3.6M rows and 17 columns	0:17:22	0:08:31
Predictive Modeling	1:12:26	6.8M rows and 29 columns	O:08:07	0:03:36

Summary

Without using the Pload utility, unloading an Informix XPS query results set of millions of records and dozens of fields can be time and resource intensive. It's common for users to land several large data sets in a single program, and they ask "We have this powerful system. Why does this take so long?" So once again, it's SAS to the rescue with a surprisingly straight-forward solution to a frustrating problem. With the %UNLOAD macro program, we've accomplished a number of things:

1. We take advantage of a powerful Informix capability;
2. We automate the solution and remove it from user intervention; and
3. We reduce the time the time needed to create a SAS data set by 30-50 percent or more.
4. We free up system I/0 resources.

References and Resources

Bentley, John E. (2000) "An Introduction to Parallel Computing." SUGI 25 Proceedings.

Bentley, John E. (2002) "Using SAS to Extract from a Parallel RDBMS." The SESUG Informant, Vol. 4 No. 2, Fall 2002. http://www.sesug.org/newsletters/fall2002.pdf

Informix Software (1996) Informix-OnLine XPS Feature Enhancements Guide. Informix Press, Menlo Park CA.

Informix Corporation (2000). Informix Guide to SQL. Prentice-Hall PTR, Upper Saddle River NJ.

About the Authors

John E. Bentley has used SAS Software since 1987 in the healthcare, insurance, and banking industries. For the past six years he has been with the Corporate Data Management Group of Wachovia Bank with responsibilities for supporting users of the bank's data warehouse and data marts and managing the development of SAS client-server applications to extract, manipulate, and present information from them. John teaches a short course in parallel processing with SAS Software and regularly presents at national, regional, and special interest SAS User Group Conferences and local SAS User Group meetings. He is chairing the SUGI 28 Systems Architecture section and organized the Weekend Workshops at SESUG 2002.

Bala Pedi is a database administrator with many years' experience with the Informix Extended Parallel Server. For the past five years, he has been the Senior DBA with Wachovia Bank's Data Warehousing Group with responsibilities for maintaining, upgrading, and enhancing the Bank's customer information data warehouse

Contact Information

John E. Bentley
Corporate Data Management Group
Wachovia Bank
201 S. College Street
Mailcode NC-1025
Charlotte NC 28288
704-383-2686
John.Bentley2@FirstUnion.Com

```
/*************************************************************************
** UNLOAD.SAS
**
** Unload an Informix scratch table (memory table) into a SAS data
** set using the Informix Pload utility.
**
** Values passed in:
**  _memoryTable   - The name of the Informix scratch table containing the
**                   results set to be landed as a SAS data set.
**  _targetSASDSN  - The SAS data set name to be created.
**  _targetLibrary - The libref location of the new SAS data set.
**
*************************************************************************/
%macro unload(_memoryTable,_targetSASDSN,_targetLibrary);

options mprint;

%local _macroStartTime _macroEndTime
      _id _user _memoryTable _externalTable  _unloadDir
       _externalFN _program _schema;

%let _macroStartTime=%sysfunc(TIME());

%let _id=%substr(%sysfunc(round(%sysfunc(ranuni(0))*10000000,1)),2,6);
%let _user=%sysget(logname);
%let _memoryTable = %lowcase(&_memoryTable);
%let _externalTable=&_user._&_id;
%let _schema = X_%substr(&_id,1,6);

%let _unloadDir=/unload/unload8;
%let _externalFN = &_user._&_id..txt;
%let _program = &_user._&_id..sas;

%put _user_;

libname tmplib "&_unloadDir" ;

** Syntax check. ;
%if %length(&_memoryTable)=0 %then
  %do ;
    %put ERROR: MISSING NAME OF MEMORY TABLE.;
```

```
    %put ERROR: SYNTAX IS unload(memoryTable,targetDataset,targetLibref) ;
    %goto errout;
  %end ;
%else %if %length(&_targetSASDSN)=0 %then
  %do ;
    %put ERROR: MISSING NAME OF TARGET SAS DATA SET.;
    %put ERROR: SYNTAX IS unload(memoryTable,targetDataset,targetLibref) ;
    %goto errout;
  %end ;

** Create an Informix External Table.  An external table is a    ;
** data source that is not part of the database that can be      ;
** used to load and unload data.                                 ;
** Using the sameas specification gives it the same structure as ;
** the memory table holding the data.  Basically its a flat file ;
** The format and delimiter clauses are used to separate values  ;
** and the datafiles clause names the external files that are    ;
** opened when the external file is used.                        ;
** The disk method is used to write the data to CoServer 1 in    ;
** the directory specified.  The data is simultaneously written  ;
** to four files named the same but with a numeric identifier.   ;
** %superq prevents the %r from being resolved as a macro value. ;
execute(
  create external table &_externalTable sameas &_memoryTable
  using (
    format     'delimited',
    delimiter  '¶',
    datafiles ("disk:1:/unload/unload8/&_externalFN.%nrstr(%r(1..4))")
        )
) by informix;

** Load the external file. ;
execute (
  insert into &_externalTable
  select *
  from  &_memoryTable
) by informix;

** Drop the Informix external table.  The data has been landed ;
** into the files specified in the datafiles clause.          ;
execute ( drop table &_externalTable ) by informix;

** Create a SAS data set containing two obs.  Need this for ;
** proc contents.                                           ;
execute (
  select *
  from 2 samples of &_memoryTable
  into scratch my_samp_tmp
) by informix;

create table testsas as
  select *
  from connection to informix
    (
     select *
     from my_samp_tmp
    );

disconnect from informix;
quit;
```

```
** Get the variable names, formats, informats. Save to a data set. ;
proc contents data=testsas out=contents&_schema(replace=yes)
  nodetails noprint ;
run;

** Varnum is the physical position of the variables in the data set. ;
proc sort data=contents&_schema ;
  by varnum;
run;

** Write the first part of a DATA step. ;
data _null_; set contents&_schema end=eof;
  file "&_unloadDir/&_program";

  if _n_=1 then do;

    ** Concatenated filenames ;
    put 'filename in ("&_unloadDir/&_externalFN.1",
    "&_unloadDir/&_externalFN.2","&_unloadDir/&_externalFN.3",
"&_unloadDir/&_externalFN.4");';
    put ' ';

    ** Data statement ;
    if length("&_targetLibrary") > 0 then
      put "data &_targetLibrary..&_targetSASDSN; ";
    if length("&_targetLibrary") = 0 then
      put "data &_targetSASDSN  ;";

    ** Infile statement ;
    put @3 "infile in stopover dsd delimiter = '¶' blksize=32760
lrecl=32760;";
    put ' ';
  end;

  ** For cursor control. ;
  x=-1;

  ** Create INFORMAT statements so we know how to read the data. ;
  if informat='DATE' then
    put 'informat ' name   'MMDDYY12.; format ' name '  DATE9.; '  ;
  else if type = 1 & length = 8 & informl = 0 then
    put 'informat ' name  '11.;'  ;
  else if type=2 then
    put 'informat ' name  '$CHAR'     informl  +x  '.' informd ' ;'  ;
  else
    put 'informat ' name  informat  +x  informl  +x  '.' informd ' ;'  ;

  if eof then
    put ' ';
run;

** Now write the input statement. ;
data _null_; set contents&_schema end=eof;
  file "&_unloadDir/&_program" mod;

  if _n_=1 then put 'input';

  if type=2 then
    put @3 name "$";
```

```
  else
    put @3 name ;

  if eof then do;
    put @3 ';';
    put ' ';
    put 'run;';
  end;
run;

** Run the program we just wrote. ;
%include "&_unloadDir/&_program";

** Delete the work files and SAS data set. ;
x "rm -rf &_unloadDir/&_externalFN.*";
x "rm -rf &_unloadDir/&_program";

proc datasets library=work nodetails nolist ;
  delete  contents&_schema;
  run;
quit;

** On error, jump to here to end the macro.;
%errout:
;

%let _macroEndTime=%sysfunc(TIME());
%let _macroElapse=%sysevalf(&_macroEndTime - &_macroStartTime,integer);
%let _macroClockTime=%sysfunc(TIMEPART(&_macroElapse),time8.);
%put ***************************************;
%put Total Clock Time (hhmmss): &_macroClockTime;
%put ***************************************;

%mend unload;
```