Paper BI-02 2014

Efficiencies using SAS® and Netezza®

Rachel Rabaey, OLSON 1to1, Minneapolis, MN

ABSTRACT

IBM’s Netezza® data warehouse is specifically optimized for large amounts of data and high performance analytics. As SAS users, utilizing these features requires some knowledge and a few changes to codes. This paper will cover efficiencies such as Bulk Load and Bulk Unload, usage of temporary tables and how to utilize pass-through SQL to move data processing to a place where run times can be reduced by 30% or more.

INTRODUCTION

Being able to utilize the data stored in a Netezza data warehouse from your SAS® environment allows you to take advantage of Netezza’s fast processing capability. By doing as much of the data organization and summarization in Netezza, you can dramatically reduce your run-times.

GETTING STARTED

The first step when working with data in Netezza is to connect to the database. This can be achieved either via a LIBNAME statement or within a PROC SQL statement.

All of the following code assumes that a connection has been made to the Netezza database via a LIBNAME statement. A sample LIBNAME statement is below:

```
LIBNAME sandbox NETEZZA SERVER=’NZServer’ USER=NZUser DATABASE=NZDatabase
PASSWORD=NZPassword;
```

To connect to the Netezza server via a PROC SQL statement, the connect statement is added to the beginning of the code.

```
PROC SQL;
CONNECT to Netezza (SERVER = ‘NZServer’ DATABASE = NZDataBase USER = NZUser
PASSWORD = NZPassword CONNECTION = GLOBAL);

<SQL Code here>

QUIT;
```

BULKUNLOAD/BULKLOAD

The fastest way to move data between the Netezza server and the SAS server is with usage of the Bulkunload or Bulkload options. The option can be stated in either a data step or procedure.

The option is placed directly after the name of the Netezza table. If your data has null or missing values, the option BL_USE_PIPE set to ‘No” will force SAS/ACCESS Interface to Netezza to read data from a flat file. If the BL_USE_PIPE=NO is not used when moving data with null values, the code will error out due to consecutive delimiters.

EXAMPLES:

Copy a table from Netezza to SAS	`PROC SQL;`
	`CREATE TABLE MyLib.aggregation as`
	`SELECT * FROM sandbox.aggregation`
	`(BULKUNLOAD=YES BL_USE_PIPE=NO)`
	`ORDER BY personid;`
	`QUIT;`

Copy a table from SAS into Netezza	PROC SQL;
	CREATE TABLE sandbox.tags
	(BULKLOAD=YES BL_USE_PIPE=NO)
	as
	SELECT *
	FROM MyLib.tags;
	QUIT;

Append data from a SAS table into an existing Netezza table	PROC APPEND BASE=sandbox.Cust_List
	(BULKUNLOAD=YES BL_USE_PIPE=NO)
	DATA=MyLib.New_Cust FORCE;
	RUN;

EXECUTE VS SELECT * FROM CONNECTION

Creating a table in the Netezza Sandbox and copying it over to the SAS server using the BULKUNLOAD option compared to writing directly to the SAS table can also decrease run times.

In the example below, the main query is exactly the same. The difference is how the data is moved from the Netezza sever to the SAS server. When using the Select From Connection methodology, each row is written to your file one line at a time. The BULKLOAD option allows SAS to use Netezza's Remote External Table interface to move data from one server to the next.

EXAMPLE:

Select * from Connection	Execute

PROC SQL;
CONNECT TO Netezza (SERVER = 'NZServer' DATABASE = NZDataBase USER = NZUser PASSWORD = NZPassword CONNECTION = GLOBAL);
 	
CREATE TABLE MyLib.LargeFile as
 	
SELECT * FROM connection to Netezza
(
SELECT pa.BrandID
    , pa.customernumber
...
FROM DB.dbo.personalias pa
INNER JOIN DB.dbo.person p ON pa.personid = p.personid
...
ORDER BY pa.CustomerNumber
);
QUIT;
	PROC SQL;
	CONNECT TO Netezza (SERVER = 'NZServer' DATABASE = NZDataBase USER = NZUser PASSWORD = NZPassword CONNECTION = GLOBAL);
	
	EXECUTE
	(
	CREATE TABLE sandbox.LargeFile as
	
	SELECT pa.BrandID
	    , pa.customernumber
	...
	FROM DB.dbo.personalias pa
	INNER JOIN DB.dbo.person p ON pa.personid = p.personid
	...
	ORDER BY pa.CustomerNumber
	)
	BY NETEZZA;
	DISCONNECT FROM NETEZZA;
	QUIT;
	
	PROC SQL;
	CREATE TABLE MyLib.LargeFile as
	SELECT * FROM sandbox.LargeFile
	(BULKUNLOAD=YES BL_USE_PIPE=NO)
	;
	
	DROP TABLE sandbox.LargeFile;
	
	QUIT;

Sample Elapsed hh:mm:ss.ss = 0:08:43.79;	Sample Elapsed hh:mm:ss.ss = 0:03:20.13;

By splitting the process into three steps, the process was reduced to one-third of the time.

An added benefit is that names of tables and fields in the Netezza database may be longer than the SAS constraint of 32 characters. When you use the EXECUTE statement, the variables can be renamed to a shorter and SAS appropriate name.

DROPPING TABLES

In SAS, tables can be overwritten without an error or warning. This is not the case in Netezza. If a table exists, it cannot be overwritten and must be dropped.

A “drop table” command in PROC SQL will delete the table. However, if the table doesn’t exist, the command will result with a warning in your SAS code.

A solution to this is to use a simple macro that only drops a table if it exists.

```
%macro droptable(dataset=);
%if %sysfunc(exist(&dataset))=1 %then %do ;
    proc sql; drop table &dataset; quit;
%end;
%mend droptable;

%droptable(dataset=sandbox.MyFile);
```

This macro is beneficial when working on larger projects, since it’s easy to pick and choose which tables you need to drop. It’s also a good practice to call the macro both at the beginning and at the end of your code in the off chance that your code didn’t complete during the prior execution.

TEMPORARY TABLES IN NETEZZA

If your setup with Netezza does not allow you to create tables, temporary tables may come in handy when processing large amounts of data or when doing aggregation or joins.

Netezza temporary tables are kept only during an active session. This means that that all processing of the temporary tables must take place before SAS disconnects from the Netezza.

In the example below, two temporary summary tables are created in the Netezza environment and then joined to a customer table to create the final results. The final table is created using the Select from Connection methodology to create a table on the Work drive.

Once the Disconnect statement is run, both temporary tables have been dropped and no longer exist in the Netezza or SAS environment.

EXAMPLE:

```
PROC SQL;
CONNECT to Netezza (SERVER = ‘NZServer’ DATABASE = NZDataBase USER = NZUser
PASSWORD = NZPassword CONNECTION = GLOBAL);

EXECUTE
(
CREATE TEMP TABLE SalesSummary as
SELECT customernumber
    , sum(SalesAmount) as Total_Sales
FROM DB.dbo.Transactions
GROUP BY customernumber
)
BY NETEZZA;

EXECUTE
(
CREATE TEMP TABLE RewardSummary as
SELECT customernumber
    , sum(RewardAmount) as Total_Rewards
FROM DB.dbo.Rewards
GROUP BY customernumber
)
BY NETEZZA;
```

```
CREATE TABLE work.Results as
SELECT * FROM connection to Netezza
(Select c.Customernumber
    , s.Total_Sales
    , r.Total_Rewards
FROM DB.dbo.Customer c LEFT JOIN SalesSummary s on
c.customernumber=s.customernumber
LEFT JOIN RewardSumary r on c.customernumber=r.customernumber);

DISCONNECT FROM NETEZZA;
QUIT;
```

CONCLUSION

IBM Netezza data warehouse appliances are becoming more common and combining the Netezza processing power with the flexibility of SAS will continue to provide great results. Understanding how to move files between the two systems using the bulk load options, how the Execute command keeps all the processing within the Netezza server and understanding temporary tables can make an impact on your productivity.

REFERENCES

- “Leveraging IBM Netezza Data Warehouse Appliances with SAS: Best Practices Guide for SAS Programmers;” Available at http://www.as.com/partners/directory/ibm/NetezzaDWAppliances-withSAS.pdf
- “SAS/ACCESS 9.1.3 Supplement for Netezza: SAS/ACCESS for Relational Databases;” Available at http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/access_netezza_9933.pdf

CONTACT INFORMATION

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

Rachel Rabaey
OLSON 1to1
Minneapolis, MN
rrabaey@olson.com or rrabaey@gmail.com