ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 4, pp: (68-74), Month: October - December 2015, Available at: www.researchpublish.com

% BigDataLoader: A SAS Macro to Migrate Big Data 99% Faster

Anant Sharma

Data warehousing & Business Intelligence, Wipro Technologies, Mumbai, India

Abstract: **The world as we know, is much more competitive as it was a decade back. Therefore, the restive companies are trying to lead their competitors and for that they need to have certain mechanism from which they can serve the best in their respective domains. The term Big Data can be inferred as the large amount of data stored among the databases and servers of these big Multinational companies (MNCs), which is basically used to identify patterns which can influence future predictions. As per the Oracle survey, 90% of the world's data has been coagulated in the last two years. MNCs are trying to hold this data into their Data warehouses and thus making it available for data mining and analytics. In such scenarios, there is an insisting need of some powerful tools and programs which can handle such huge data. One such tool is provided by SAS (Statistical Analysis System) company to handle the migration of the Data into Data warehouses from different sources. This process is commonly known as extraction, transformation and loading (ETL). But sometimes these tools are further optimize to increase their efficiency in terms of time and efforts. One such program has been shared in this paper as a motive to decrease the loading cycle of Big Data and save time, efforts and money.**

Keywords: **SAS; Data warehouses; Big Data, ETL.**

1. INTRODUCTION

The era of new technology in which need of Enterprise Data Warehouses are growing rapidly, MNCs are trying to store their high volumes of data for future predictions and analysis to sustain in the competitive environment. One such suite is provided by SAS to ease the process of migrating high volume of data into Databases. Teradata, among them is the most popular and powerful database for Data warehousing implementation because of its index based architecture which makes it highly efficient for retrieving data. On the other hand, OLTP systems are commonly configured on the Oracle. Henceforth, in this paper we have considered Source database as Oracle and Target database as Teradata.

Need of the hour is to make a program that not only eliminates the efforts of a developer to facilitate the data migration, but also to reduce the time complexity. The common difficulty among the IT companies are usually the hardware configuration problems, such as, less server space, to much time consuming, direct proportionality of jobs with the number of tables, etc. Due to these problems, there is an imperative need of a program to trim down all these obstructions.

In a typical Data warehouse implementation for MNCs, they tend to have millions and billions of records which is most of the time unstructured data. Some of the tables might hold up to 1000Cr+ records. It is then necessary to migrate this data into a Data warehouse in a structured form so that various operations can be done upon them. Prognosticate future will grant companies not only to expand themselves, but also help them to increase their revenue by investing less and targeting the right customers. But if the volume is accumulating everyday, then it is definite that nowhere in the future, this volume of data would become less. Henceforth, there is a certain need of such programs which can alleviate the entire process.

2. OVERVIEW

2.1. Assumptions:

- The target database which we are using is Teradata for our EDW implementation.
- The source is Oracle, and the tables have been built in Oracle by using Partitioning* concepts such as "Hash Partitioning", "Range Partitioning", etc.
- The tool for ETL is used as Base SAS/ SASEG/SASDI **(any one).

*It is important to note here that the tables in the source are partitioned which is always a best practice to follow. The reason behind this is that tables are easy to access and retrieving data.
**This a common algorithm for all the different type of databases and ETL tools irrespective of their functionality. The difference would be in the implementation of the algorithm.

2.2. Constraints:

- To load a partitioned table from Source to Target with 700 Cr. Records, it usually takes 3-4 days i.e. 90 hours (on an average).
- The developer has to make as many Jobs/Scripts as many the tables are present in the source.
- *The SAS server will takes up the space while executing the Job/Script which in this case will go up to 3- 4 Terabytes of memory.*

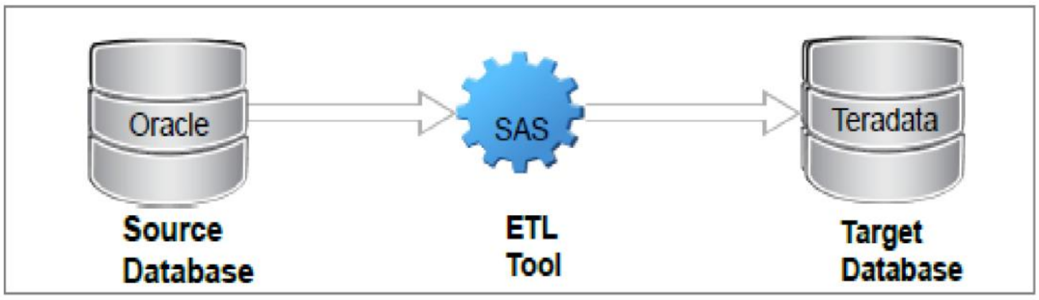

Fig 1: Movement of data from source to target

3. APPROACH

3.1. Traditional Approach:

The Traditional approach that would a project life cycle would follow is to open the SAS DI studio, make Source and target library, registering the source and target tables. This will be followed by the developer efforts for making a job by "drag & drop" facility using 'Table Loader' transformation, and at the last, the execution of the job.

3.1.1. Limitation:

- The developer has to make two logical library in SAS both for Source and Target.
- Registering the tables in those libraries respectively.
- Direct proportionality of job with the source table.
- The job will execute for 3-4 days on an average.

The figure below shows the flow of the job to visualize how the data will move from source to target.

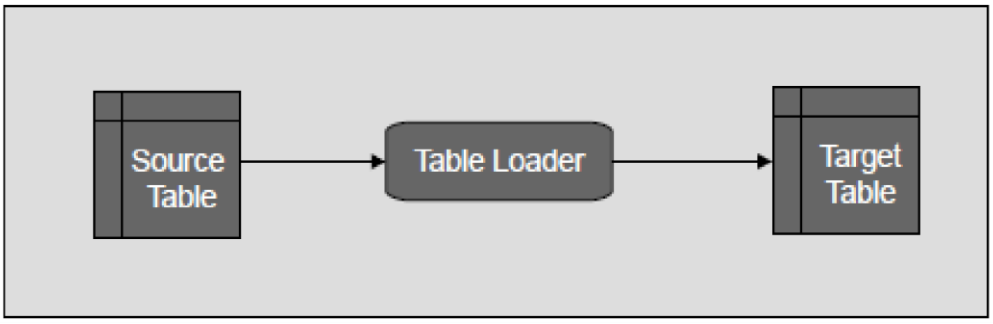

Fig 2: A typical SAS job

3.2. %Big Data Loader Approach:

The macro approach to deal this scenario is much more dynamic in nature i.e. this macro can be used for multiple tables at run time irrespective of their different structures. Hence, it has unified the power of SAS in one process.

3.2.1. The Flow of the Macro:

To understand the functionality of the macro, first we need to understand the flow of the macro.

Figure below shows the process in diagrammatic format.

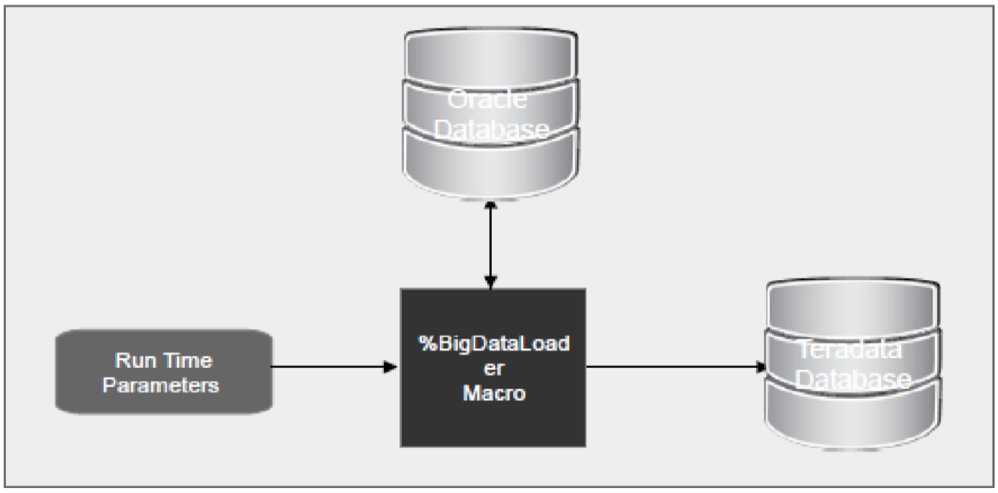

Fig 3: Process flow of the Macro

3.2.2. Run Time Parameters:

For this script we have to make two tables. One is Source_Metadata and other is Job_Control_Table. Besides this, only one parameters will be passed by the user at run time, i.e. the location path of the tables.

For the Source_Metadata, design the table as:

- *Src_Lib* : Specify any random but unique source library reference name.
- *Land_lib* : Specify any random but unique landing library reference name.
- *Src_Authdomain* : Specify the Source authentication domain with respect to SAS.
- *Tgt_Authdomain* : Specify the target authentication domain with respect to SAS.
- *Src_Path* : Specify the Source path with respect to SAS.
- Tgt_*Server*: Target teradata server details with respect to SAS.

For the Job_Control_Table, design the table as:

- *Src_Tab* : Specify the source table names
- *Lnd_Tab* : Specify the source table names
- *Src_schema* : Specify the source schema name
- *Lnd_schema* : Specify the target landing schema name
- *Load_Flg* : This is flag which tells whether the table is loaded or not.

For the execution of the macro, you just have to write,

%Big Data Loader (------- path where above two tables are present either in the form of SAS location for libname or user/password and server details);

3.2.3. Working & Architecture:

The script has a multiple session in-built mechanism. It can open multiple SAS sessions in parallel, this is one of the greatest feature of the script which makes it able to migrate a single table in different SAS sessions, thus decreasing the loading time effectively. Visualizing a source table with 32 partitions, then by this macro this source table will be loaded in the target database 32 times faster by loading 32 different tables at the same time. And at the end, these 32 tables will consolidate into a single table and temporary tables will be dropped automatically. This feature of the macro reduce the time by almost 99% which is very unique in its way. All this can be done from a single SAS session.
Also, this script is generic, i.e. it can facilitate the ETL process by migrating multiple tables sequentially, so there is no need to create 10000 jobs for 10000 tables. Irrespective of different structures of different tables, it will migrate the tables in the target database.

3.2.4. Advantages:

- Developer doesn't have to develop each job for each table.
- Irrespective of different structure, it will facilitate the process for multiple tables.
- The time and effort by this macro is reduced by 99%.
- A single SAS session can be use to invoke multiple SAS sessions, which makes it easy to control.
- Last but not the least, this macro performs on “Pull & Push” mechanism, i.e. the macro will not take any space onto the SAS server, whatsoever the size of the table is. Therefore, even if the table is of 4 TB and server configuration is only of 2 TB, this script will still perform.

3.3. Source Code:

```
%MACRO BigDataLoader;
libname tdaud “&Metadata_repository_path”;
%let aud_schema = tdaud;
PROC SQL noprint;
SELECT COUNT(1) INTO :CNT
FROM &aud_schema..JOB_CONTROL_TBL
Where (load_flg is null or load_flg = 'N');
SELECT upcase(SRC_TAB), upcase(SRC_SCHEMA),upcase(LND_TAB),upcase(LND_SCHEMA)
in to :sr separated by ':', :tg
SEPARATED BY ':', :s_schm separated by ':', :t_schm separated by ':'
FROM &aud_schema..JOB_CONTROL_TBL
Where (load_flg is null or load_flg = 'N');
select src_authdomain,tgt_authdomain, src_path, tgt_server
INTO :src_authdomain,:tgt_authdomain, :src_path, :tgt_server
from &aud_schema..SOURCE_METADATA;
QUIT;
%DO I = 1 %TO &cnt.;
%LET src_tab = %SCAN(&sr.,&I,':');
%LET tgt_tab = %scan(&tg,&i,':');
proc sql noprint;
CONNECT TO ORACLE(PATH=&src_path Authdomain=&src_authdomain );
create table part_nms as
select partition_name from connection to oracle
( select * from all_tab_partitions
Where table_owner = %unquote(%str(%')&S_SCHM%str(%'))
and table_name = %unquote(%str(%')&src_tab%str(%') );
```

```
select partition_name
into :partition separated by ' '
from part_nms;
select count(1) into :cnt_part
from part_nms;
select substr(partition_name,9)
into :task separated by ' '
from part_nms;
quit;
%do i =1 %to &cnt_part.;
%let var1 = %scan(&partition.,&i.,' ');
data &var1.;
length script $ 32676 ;
script=
"%unquote(%str(%""")
rsubmit %substr(&var1.,9) wait=no;"
libname ubi_lan1 teradata authdomain=&tgt_authdomain Server=&tgt_server schema=&T_SCHM
mode=teradata;
proc sql noprint;
CONNECT TO ORACLE (PATH=&src_path Authdomain=&src_authdomain);
create table ubi_lan1.&var1. (FASTLOAD=yes TPT=no sessions=10)
as
SELECT *
FROM CONNECTION TO ORACLE
(select * from &S_SCHM.&src_tab partition (&var1 ));
quit;
endrsubmit;
%str(%"));
run;
%end;
proc sql noprint;
select strip( partition_name) into :all_scripts separated by ' '
from part_nms;
quit;
data execution_script;
set &all_scripts;
keep script;
run;
proc sql noprint;
select script into :execute_parallel separated by ' '
from execution_script;
```

```
quit;
&execute_parallel.
waitfor _all_ &task.;
signoff _all_;
%do j=1 %to &cnt_part.;
%let var_tgt = %scan(&partition.,&j.,' ');
proc sql;
connect to teradata( authdomain=&tgt_authdomain Server=&tgt_server );
execute(
insert into &t_schm..&tg.
select &var_tgt..*
from &t_schm..&var_tgt.
)by teradata;
execute (commit)by teradata;
quit;
%end;
%if &syserr NE 0 %then %do;
proc sql;
update &aud_schema..job_control_tbl
set load_flg = 'N'
where lnd_tab = "&tgt";
quit;
%end;
%else %do;
proc sql;
update &aud_schema..job_control_tbl
set load_flg = 'Y'
where lnd_tab = "&tgt";
quit;
%end;
%END;
%MEND;
%BigDataLoader(Metadata_repository_path);
```

4. CONCLUSION

Cost Benefit Analysis:

4.1. Traditional Approach:

- Time taken to load 1 table – 90 hours.
- Time taken to prepare job for one table – 30 mins
- Time taken to prepare 1000 jobs for 1000 tables – 30000 mins
- Time taken to load 1000 tables – 90 * 1000 = 90000 hours ~ 3750 days ~ 10.5 years

- Total efforts ~ 11 years

4.2. %Big Data Loader Approach:

- Time taken to load 1 table having 32 partitions – 90/32 = 2.8 hours.
- Time taken to load 1000 tables – 2.8 * 1000 = 2800 hours ~ 116 days ~ 3 months
- Time take to built 1 script for 1000 tables = 20 mins
- Total efforts ~ 3 months

Total Savings: 99%

ACKNOWLEDGMENTS

Wipro Technologies for giving me the environment and tool so that I can implement the logic.

Mayuresh Digamber Tendulkar, Mitul Pulak Diwanji and Mandar Thakar for their constant support and motivation.

REFERENCES

[1] “SAS Programming Essentials Part 1” by SAS.

[2] “SAS Programming Essentials Part 2” by SAS.

[3] SAS Support www.support.sas.com.