m_utl_hash_size.sas File Reference

Utilities

Utility macro to determine the size of a hash table in memory

---

Description

The macro can be used anywhere in a SAS program including within a SAS procedure or SAS data step. If the table does not exist, it returns -1. If the table exists, but cannot opened it returns 0. Different parameter names are allowed. This macro is based on the hashsize.sas macro program from SAS Support Samples (http://support.sas.com/kb/34/193.html).

Note

The SHOW_ERR parameter shows or suppresses possible warnings or errors in the log. The default for SHOW_ERR is value is N.

Note

In case of encrypted SAS datasets, the ENCRYPTKEY= parameter must be provided as part of the CREDS credentials string.

Note

The output argument RESULT should be set to: %local result; in the program that calls the m_utl_hash_size.sas macro.

Autors

Paul Alexander Canals y Trocha (paul.canals@gmail.com)

Date

2020-02-02 00:00:00

Version

20.1.02

Link

https://github.com/paul-canals/toolbox

Parameters

Input	help	Parameter, if set (or ?) to print the Help information in the log. In all other cases this parameter should be left out from the macro call.
Input	table	Full LIBNAME.TABLENAME name of the or SAS dataset to be loaded into memory. The parameter can contain a combination of SAS data step style statement between brackets like where, keep, drop or rename. The default value for TABLE is: _NONE_
Input	dataset	Alias of the _table_ parameter.
Input	data	Alias of the _table_ parameter.
Input	creds	Optional. Specifies the ENCRYPTKEY= parameter value if there is an encrypted table involved.
Input	kcols	Optional. Specifies key variables for the hash object. Key variables are separated by a blank character. If there are no key variables set, the first table column is set as key variable.
Input	dcols	Optional. Specifies data variables for the hash object. Data variables are separated by a blank character. If there are no data variables set, all table columns except the first column are set as data variable.
Input	nrecs	Optional. Specifies the number of records to calculate the table size. If the NRECS parameter is not provided, the macro obtains the number of records using the m_utl_nlobs.sas macro.
Input	format	Boolean [Y|N] parameter to specify if the result size is to be formyatted by the SIZEKMG format. See examples 4 and 5. The default value is: N.
Input	show_err	Boolean [Y|N] parameter to show or hide warnings or errors in the log. The default value is Y.
Output	result	Contains the macro variable name in which the result output is to be loaded. The default value for RESULT is: _SIZE_.
Input	debug	Boolean [Y|N] parameter to provide verbose mode information. The default value is: N.

Returns

- The table size of a hash table in memory

Calls

- m_utl_nlobs.sas
- m_utl_print_message.sas
- m_utl_quote_elems.sas

Usage

Example 1: Show help information:

```
%m_utl_hash_size(?)
```

Example 2: Calculate the hash table size from an encrypted SAS dataset:

```
data WORK.class(encrypt=aes encryptkey=aespasskey);
   set SASHELP.class;
run;

proc print data=WORK.class(encryptkey=aespasskey);
   where Sex='F';
run;

%let hash_size=;

%m_utl_hash_size(
   data   = WORK.class(where=(Sex='F'))
 , creds  = %str(encryptkey=aespasskey)
 , result = hash_size
 , debug  = Y
   );

%put Hash table size of WORK.class is: &hash_size. bytes;

data WORK.size;
   table = 'WORK.class';
   sex   = 'F';
   size  = &hash_size.;
run;

proc print data=WORK.size noobs;
run;
```

Example 3: Calculate the hash table size from a SAS dataset with where statement:

```
proc print data=SASHELP.class(drop=Sex);
   where Age > 13;
run;

%let hash_size=;

%m_utl_hash_size(
   data   = SASHELP.class(where=(Age > 13))
 , kcols  = Name
 , dcols  = Age Weight Height
 , result = hash_size
 , debug  = Y
   );

%put Hash table size of SASHELP.class is: &hash_size. bytes;

data WORK.size;
   table = 'SASHELP.class';
   size  = &hash_size.;
run;

proc print data=WORK.size noobs;
run;
```

Example 4: Calculate the formattted hash table size from a SAS dataset with where statement:

```
proc print data=SASHELP.class;
   where Age > 13;
run;

%let hash_size=;

%m_utl_hash_size(
   data   = SASHELP.class(where=(Age > 13))
 , format = Y
 , result = hash_size
 , debug  = Y
   );

%put Hash table size of SASHELP.class is: &hash_size.;

data WORK.size;
   table = 'SASHELP.class';
   size = "&hash_size.";
run;

proc print data=WORK.size noobs;
run;
```

Example 5: Calculate the table size for a table with 5.000.000 records:

```
%let hash_size=;

%m_utl_hash_size(
   data   = SASHELP.class
 , nrecs  = 5000000
 , format = Y
 , result = hash_size
 , debug  = Y
   );

%put Hash table size for SASHELP.class with 5M records is: &hash_size.;
```

Copyright

Copyright 2008-2020 Paul Alexander Canals y Trocha.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.