documentum

DISCLAIMER: The information contained in this document is for INTERNAL use only. Do NOT send this document to customers.

Server 5.2 UpdateStats Job

Roger Taranto
June 2003

Information on the 5.2 dmUpdateStats Job

All modern relational databases use a cost-based optimizer which determines the most efficient query plans by checking statistics on the likelihood of values occurring in database tables, that is, the histogram statistics. If the histogram statistics are not correct, then the database optimizer may not generate an efficient query plan.

Because of this, Documentum has created the dmUpdateStats job which is enabled to run by default when the Documentum Content Server is installed. The purpose of this job is to update database histogram statistics on a weekly basis so that SQL queries sent to the database use optimal query plans in order to access data in the most efficient manner.

One important thing to consider is that query optimization is still an evolving science and many times a database optimizer may pick the wrong query plan even when the best query plan is obvious – to a human. As we perform performance testing, we find queries that do not perform well and attempt to 'tune' the database optimizer by adjusting how statistics are generated. This will always be on-going work as queries change and database companies modify their optimizer code.

In the 5.2 Content Server release, the dmUpdateStats job was rewritten. This document describes how database statistics are gathered in the 5.2 Content Server release. While some of the code has not changed from previous releases, most of it is significantly different and the descriptions here cannot be used for previous releases.

For each database, the dmUpdateStats job recalculates the database statistics for all of the columns of every table owned by the docbase owner. And, depending on the options set in the method arguments, either reports on fragmented tables or reports on and fixes fragmented tables. The dmUpdateStats job does not do any of this database work directly. Instead, it calls database commands that tell the database to do this work. This work can be CPU and disk intensive and should only be scheduled during off hours, for example, late Saturday night.

Prior to 5.2, the dmUpdateStats job would generate a script file and send this file to a database-supplied command interpreter to be executed. In 5.2, all of the commands are executed through a Documentum API connection to the Content Server which then passes appropriate commands to the database.

Because each database does this work differently, I will describe the work in separate sections for each of the four databases that we support.

An important method argument to understand is the `-dbreindex` argument. It can be either `-dbreindex READ` or `-dbreindex FIX`. If the argument is set to `-dbreindex READ`, then the job will report on the

fragmentation level of all of the tables but will not modify them. If the argument is set to `–dbreindex FIX`, then the job will report on all of the fragmented tables and then fix the ones that are fragmented beyond a certain level. The `–dbreindex FIX` option must be used with care because it is very resource intensive. It should not be enabled by default and should only be run when the Documentum administrator looks at the output from `–dbreindex READ` and determines that a reorganization of tables is necessary. Once the reorganization has been completed, the method argument should be set back to `–dbreindex READ`.

Oracle:

The job creates a table called `dm_update_stats_commands` which contains commands of the form:

analyze table *tablename* compute statistics

This table contains commands for all of the tables owned by the docbase owner, and the commands tell the database to calculate statistics for all rows for every column in each table. The job then creates a second docbase connection, and for each entry retrieved from the `dm_update_stats_commands` table it runs the command using the execsql Documentum API in the second connection. For example, for the dm_folder_s table, it runs the command:

execsql,s1,analyze table dm_folder_s compute statistics

After all of these commands have been run, the job then looks for a file called "custom_oracle_stat.sql" which was copied to the $DOCUMENTUM/dba/config/*docbase_name* directory when the docbase was first created. This file contains custom commands intended to tweak the Oracle optimizer into generating better query plans than if it only used the default statistics. Customers may add commands to this file based on the specific needs of their data, but changes should be made carefully because they could have a drastic effect on performance. To add a command to the file, you may use multiple lines, but each command must end with a semi-colon (;). You may not put comments into the file because each line is treated as part of a command. Currently, the file distributed by Documentum contains custom commands that should improve the performance of all docbases.

The `–dbreindex READ` and `–dbreindex FIX` commands are ignored on Oracle because there is no table reorganization command in Oracle. A qualified Oracle DBA may want to look at the Oracle table information using Oracle tools on a regular basis to determine whether there is specific maintenance that needs to be performed.

SQL Server

The job creates a view based on the `sysobjects` system table in order to find all of the tables owned by the docbase owner. The job then iterates through the rows in this view and in a second docbase connection, uses the execsql API to execute commands of the form:

execsql,s1,update statistics *tablename* with fullscan, all

The update statistics command tells the database to calculate statistics for all rows for every column in each table. For example, for the dm_folder_s table, it runs the command:

execsql,s1,update statistics dm_folder_s with fullscan, all

The dmUpdateStats job for SQL Server does not use a custom statistics file. We may implement this functionality in the future, but for now, there is no custom file.

The next step for SQL Server is to re-run the query against the `sysobjects` view in order to go back through the list of tables. This time, the job checks on the fragmentation of the tables using an enhanced version of the EXEC_SQL apply method. The SQL statement that is sent to SQL Server for each table is:

dbcc showcontig("*tablename*")

This command is sent to SQL Server using the EXEC_SQL API invoked as:

apply,s1,,EXEC_SQL,WITH_INFO,B,T,QUERY,S, dbcc showcontig("*tablename*")

The "WITH_INFO" boolean parameter was added in the 5.2 release in order to return extra information generated by the database for specific commands. It is important to note that the extra information returned when the WITH_INFO parameter is set to true (the default is false) is NOT query results. Instead, it is extra information generated by the database as a side-effect of the command that was run. In this case, when you run this command, you get a collection object back containing two attributes. The first attribute is "result". This attribute is unchanged and is present whether or not WITH_INFO is specified. It contains T or F and describes whether the command was successful. The second attribute is called "info" and is only present when the WITH_INFO parameter is set to true. Sample contents of the info attribute are:

```
[Microsoft][ODBC SQL Server Driver][SQL Server]DBCC SHOWCONTIG scanning
'dm_sysobject_s' table...

[Microsoft][ODBC SQL Server Driver][SQL Server]Table: 'dm_sysobject_s'
(981578535); index ID: 1, database ID: 5

[Microsoft][ODBC SQL Server Driver][SQL Server]TABLE level scan performed.

[Microsoft][ODBC SQL Server Driver][SQL Server]- Pages
Scanned.............................: 565

[Microsoft][ODBC SQL Server Driver][SQL Server]- Extents
Scanned.............................: 78

[Microsoft][ODBC SQL Server Driver][SQL Server]- Extent
Switches............................: 116

[Microsoft][ODBC SQL Server Driver][SQL Server]- Avg. Pages per
Extent.....................: 7.2

[Microsoft][ODBC SQL Server Driver][SQL Server]- Scan Density [Best
Count:Actual Count].......: 60.68% [71:117]
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]- Logical Scan Fragmentation
..................: 7.08%

[Microsoft][ODBC SQL Server Driver][SQL Server]- Extent Scan Fragmentation
..................: 83.33%

[Microsoft][ODBC SQL Server Driver][SQL Server]- Avg. Bytes Free per
Page.....................: 2726.0

[Microsoft][ODBC SQL Server Driver][SQL Server]- Avg. Page Density
(full).....................: 66.32%

[Microsoft][ODBC SQL Server Driver][SQL Server]DBCC execution completed. If
DBCC printed error messages, contact your system administrator.
```

The above information is added to the job output per table if the `-dbreindex` job argument is set to READ or FIX.

If the `-dbreindex` argument is set to FIX, then as the dmUpdateStats job retrieves the above output for each table, if the line containing the words "Scan Density" does not show 100% for the table, then the table is considered fragmented and the job run the following command on the table:

execsql,s1,DBCC DBREINDEX(*tablename*)

Please note that the above dbreindex command is only run on a table if (a) the `-dbreindex` parameter is set to FIX and (b) the table's scan density is less than 100%. This can be an expensive operation on larger tables and should only be run after careful consideration as to whether the table is sufficiently fragmented to warrant a reindexing.

Sybase

The job creates a view based on the `sysobjects` system table in order to find all of the tables owned by the docbase owner. The job then iterates through the rows in this view and in a second docbase connection, uses the execsql API to execute commands of the form:

execsql,s1,update all statistics *tablename* using 100 values

The update statistics command tells the database to calculate statistics for all rows for every column in each table using 100 "steps" in the histogram table. Through experimentation, Documentum determined that the default Sybase value of 20 steps was not sufficient and that 100 steps produces much better histogram data for most Documentum database data. For example, for the dm_folder_s table, it runs the command:

execsql,s1,update all statistics dm_folder_s using 100 values

After all of these commands have been run, the job then looks for a file called "custom_sybase_stat.sql" which was copied to the $DOCUMENTUM/dba/config/*docbase_name* directory when the docbase was first created. This file contains custom commands intended to tweak the Oracle optimizer into generating better query plans than if it only used the default statistics. Customers may add commands to this file based on the specific needs of their data, but changes should be made carefully because they could have a drastic effect on performance. To add a command to the file, you may use multiple lines, but each command must end with a line containing only the word "go" at the

beginning of the line. You may not put comments into the file because each line is treated as part of a command. Currently, the file distributed by Documentum contains custom commands that should improve the performance of all docbases.

The `-dbreindex` argument for Sybase works different than on SQL Server. On Sybase, the READ parameter is not available because Sybase does not supply a command to provide table fragmentation via SQL. If the `-dbreindex` argument is set to "FIX", then the dmUpdateStats job re-issues the query against the `sysobjects` view in order to re-process all of the tables owned by the docbase owner. For each table, the job executes the command:

execsql,s1,reorg compact *tablename*

using a second docbase connection. The reorg compact command tells Sybase to both reclaim space and undo row forwarding in the table specified.

DB2

The job creates a view based on the `syscat.tables` system table in order to find all of the tables owned by the docbase owner. The job then iterates through the rows in this view and in a second docbase connection, uses the new UPDATE_STATISTICS apply method to execute commands of the form:

apply,s1,,UPDATE_STATISTICS,TABLE_NAME,S,*tablename*

(Complete details of all of the arguments to the UPDATE_STATISTICS command are documented below.)

The UPDATE_STATISTICS command tells the database to calculate statistics for all rows for every column in each table. For example, for the dm_folder_s table, it runs the command:

apply,s1,,UPDATE_STATISTICS,TABLE_NAME,S,dm_folder_s

The dmUpdateStats job for DB2 does not use a custom statistics file. We may implement this functionality in the future, but for now, there is no custom file.

The `-dbreindex` argument on DB2 needs to create a custom view in order to retrieve table fragmentation data from the database. The view calculates table fragmentation using three different formulas. If the `-dbreindex` argument is set to "READ", then the table fragmentation data is calculated and the results from the view are placed into the job results file. The output looks like:

```
      Tablename                Organization

DMI_DD_TYPE_INF                        -*-

DMI_DIST_COMP_R                        ---
```

```
DMI_DIST_COMP_R                         ---

DMI_DUMP_OBJECT                         ---

DMI_EXPR_CODE_S                         ---

    DMI_INDEX_R                         **-

    DMI_INDEX_S                         --*

DMI_LINKRECORD_                         ---

DMI_LINKRECORD_                         ---

DMI_LOAD_OBJECT                         ---

DMI_OBJECT_TYPE                         ---

DMI_OTHERFILE_S                         ---

  DMI_PACKAGE_R                         -**

  DMI_PACKAGE_S                         ---
```

The "Organization" column displays the results of the three fragmentation formulas. A dash ('-') means that the formula does not think the table is fragmented, and an asterisk ('*') means that the formula thinks that it is. In the above data, formulas 1 and 2 think that the DMI_INDEX_R is fragmented, and formulas 2 and three think that the DMI_PACKAGE_R table is fragmented.

If the `-dbreindex` argument is set to "FIX", the above fragmentation information is also printed. However, as each table is processed, if a table has two or three asterisks, then the table is reorganized using the REORGANIZE_TABLE apply method in a separate docbase connection, for example,

apply,s1,,REORGANIZE_TABLE,TABLE_NAME,S,*tablename*

(The REORGANIZE_TABLE apply method is documented below.) After a table is reorganized, the UPDATE_STATISTICS apply method is re-run against the table because tables must have their statistics regenerated if they are reorganized.

The REORGANIZE_TABLE method can be an expensive operation on larger tables and should only be run after careful consideration as to whether the table is sufficiently fragmented to warrant reorganization.

UPDATE_STATISTICS

The UPDATE_STATISTICS apply method was mainly added in order to update table statistics for DB2 since there was no way to do the same thing via execsql. The apply method was implemented for all four databases, but the dmUpdateStats job only uses it for DB2 since execsql works for the other databases.

You must be a superuser to execute this method. It is available via API and DQL.

The method ignores the id parameter and takes three arguments:

TABLE_NAME: This is the name of the table for which you wish to update statistics. The parameter is the same on all databases.

COUNT: This integer parameter is optional for all databases and means different things in different databases:

Oracle: The number of buckets to use when calculating histogram statistics. The Oracle default is 75 if no value is specified. The range is 1 to 254. A value of 0 means that you want to delete all histogram statistics for the table.

SQL Server: The sampling percentage to use when determining how many rows to look at when calculating statistics. For example, specifying 50 means that half of the rows will be used to generate statistics. The default is 100 and it means that all rows will be used to generate statistics.

Sybase: The number of steps to use when calculating histogram statistics. The Sybase default is 20.

DB2: This attribute is currently unimplemented for DB2. In the future, this parameter will determine the number of frequency values to use when calculating table statistics in DB2.

EXTRA_DATA: This string parameter is optional for all databases and means different things in different databases:

Oracle: This parameter contains a comma-separated list of columns to be analyzed. The columns must exist in the table specified in the TABLE_NAME parameter, and will be the only columns analyzed for statistics. If the columns are not specified using this parameter then all of the columns in the table will be analyzed. One important note: because this is a comma-separated list, it must be enclosed in single quotes when passed as a parameter to the apply API. For example,

apply,c,,UPDATE_STATISTICS,TABLE_NAME,dm_sysobject_s,EXTRA_DATA,'keywords,authors'

SQL Server: This parameter specifies an index on the specified table for which you wish to generate statistics. Unlike Oracle and Sybase, this parameter is NOT a comma-separated list but instead must refer to only one index. If this parameter is not used, then the whole table and all of its indexes will be analyzed.

Sybase: This parameter contains a comma-separated list of columns to be analyzed. The columns must exist in the table specified in the TABLE_NAME parameter, and will be the only columns analyzed for statistics. If the columns are not specified using this parameter then all of the columns in the table will be analyzed. One important note: because this is a comma-separated list, it must be enclosed in single quotes when passed as a parameter to the apply API. For example,

apply,c,,UPDATE_STATISTICS,TABLE_NAME,dm_sysobject_s,EXTRA_DATA,'keywords,authors'

It can be expensive to specify specific columns for Sybase because if you use update statistics to generate statistics for a column or list of columns, update statistics must scan the table and perform a sort.

DB2: This parameter specifies an index on the specified table for which you wish to generate statistics. Unlike Oracle and Sybase, this parameter is NOT a comma-separated list but instead must refer to only one index. If this parameter is not used, then the whole table and all of its indexes will be analyzed.

REORGANIZE_TABLE

The REORGANIZE_TABLE apply method was mainly added in order to reorganize tables in DB2 since there was no way to do the same thing via execsql. The apply method was implemented for all four databases, but the dmUpdateStats job only uses it for DB2 since execsql works for the other databases.

You must be a superuser to execute this method. It is available via API and DQL.

The method ignores the id parameter and takes two arguments:

TABLE_NAME: This is the name of the table for which you wish to update statistics. The parameter is the same on all databases.

INDEX: This parameter is required on Oracle and optional on the other databases.

Oracle: This parameter is required on Oracle. There is no command to rebuild tables in Oracle. However, there is a command to rebuild indexes, and that is what this method does. You must specify one and only one index for the specified table that you want to rebuild.

SQL Server: You may only specify one index using this parameter. If you specify an index (on the table specified in the TABLE_NAME parameter), then only that indexed is re-indexed. If you do not specify an index with this parameter, then all of the table's indexes will be re-indexed.

Sybase: This parameter is ignored.

DB2: This is an EXTREMELY powerful parameter for DB2. You may specify one and only one index on the table specified using the TABLE_NAME parameter. The index specified in this parameter is used to order the rows in the table; that is, the table is organized using the specified index. **This is a powerful feature that must be used wisely.** If you reorganize the table on the wrong index, performance could suffer greatly. The parameter is not required, so if you are not sure about which index to use to rebuild the table, do not use this parameter. For most Documentum tables, especially the repeating attribute tables, the r_object_id index is probably the best choice for this parameter.