TECHNICAL NOTE

GATHERING AGGREGATED COST-BASED OPTIMISER STATISTICS ON PARTITIONED OBJECTS IN ORACLE 11GR2

Prepared By David Kurtz, Go-Faster Consultancy Ltd.
Technical Note
Version 0.01
Friday 29 November 2013
(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)
File: Partition.Statistics.11gR2.docx, 29 November 2013

Contents

Introduction 2
Correcting Statistics 2
Collecting Statistics 3
Collected Statistics 10
Implementation of New Approach 16
Installation Instructions 20
Implementation Instructions 20
Source Code 24

Introduction

This document follows on from 'Gathering Aggregated Cost-Based Optimiser Statistics On Partitioned Objects In Oracle 10g'. It investigates whether the alternative approach that I proposed for collecting statistics on partitioned objects in Oracle 10g is necessary on 11g[1].

Correcting Statistics

I encountered a range of problems in 10g. There is no doubt that there have been a significant number of corrections and enhancements to the way that statistics are collected in Oracle.

However, in 11g I have found one outstanding issue. For composite partitioned tables that are either list or hash subpartitioned there is still a problem with the number of distinct values (NDV) calculated by DBMS_STATS (see page 13).

In general, a minimum value of NDV can be calculated as the maximum of NDV of constituent partitions or sub-partition, but if we are calculating a value for NDV for the partitioning column, then we can sum the NDV across the partitions[2].

This error can be corrected by the *agg_stats* PL/SQL package in the appendix to this document. To that limited extent, it is still necessary to correct the statistics collected by *dbms_stats*.

[1] The tests in this document have been run against Oracle 11.2.0.2.0 on Windows 64-bit.

[2] I have not looked at any case where a table is partitioned on a combination of columns. I would guess that this rule only works for the first partitioning column.

©GO-FASTER CONSULTANCY LTD.

Collecting Statistics

I will repeat the same tests that I performed on 10g, on the same table with the same data set[3]. All the necessary scripts are included so that the test can be reproduced.

1. I'll start by creating a subset of PS_GP_RSLT_ACUM, one of the Global Payroll result table. The subset table is called GFC_GP_RSLT_ACUM. This will have only three of the range partitions, each of which will only have 3 list partitions.

```
DROP TABLE sysadm.gfc_gp_rslt_acum
/
CREATE TABLE sysadm.gfc_gp_rslt_acum
(emplid VARCHAR2(11) NOT NULL
,cal_run_id VARCHAR2(18) NOT NULL
,empl_rcd SMALLINT NOT NULL
,gp_paygroup VARCHAR2(10) NOT NULL
,cal_id VARCHAR2(18) NOT NULL
,orig_cal_run_id VARCHAR2(18) NOT NULL
,rslt_seg_num SMALLINT NOT NULL
,pin_num INTEGER NOT NULL
,empl_rcd_acum SMALLINT NOT NULL
,acm_from_dt DATE
,acm_thru_dt DATE
,slice_bgn_dt DATE
,slice_end_dt DATE
,seq_num8 INTEGER NOT NULL
,user_key1 VARCHAR2(25) NOT NULL
,user_key2 VARCHAR2(25) NOT NULL
,user_key3 VARCHAR2(25) NOT NULL
,user_key4 VARCHAR2(25) NOT NULL
,user_key5 VARCHAR2(25) NOT NULL
,user_key6 VARCHAR2(25) NOT NULL
,country VARCHAR2(3) NOT NULL
,acm_type VARCHAR2(1) NOT NULL
,acm_prd_optn VARCHAR2(1) NOT NULL
,calc_rslt_val DECIMAL(18,6) NOT NULL
,calc_val DECIMAL(18,6) NOT NULL
,user_adj_val DECIMAL(18,6) NOT NULL
,pin_parent_num INTEGER NOT NULL
,corr_rto_ind VARCHAR2(1) NOT NULL
,valid_in_seg_ind VARCHAR2(1) NOT NULL
,called_in_seg_ind VARCHAR2(1) NOT NULL)
TABLESPACE GPAPP PCTUSED 95 PCTFREE 1
PARTITION BY RANGE(EMPLID)
SUBPARTITION BY LIST (CAL_RUN_ID)
SUBPARTITION TEMPLATE
(SUBPARTITION x2010M08 VALUES
('2010UM08','2010UW29','2010UW30','2010UW31','2010UW32','2010AM08','2010AW29','2010AW30','2010AW31','2010AW32')
 TABLESPACE GP2010M08TAB
,SUBPARTITION x2010M09 VALUES
```

[3] The data values in this example have been obfuscated to maintain client confidentiality.

```
('2010UM09','2010Uw33','2010Uw34','2010Uw35','2010Uw36','2010AM09','2010Aw33','2010Aw34','2010Aw35','2010Aw36')
 TABLESPACE GP2010M09TAB
,SUBPARTITION x2010M10 VALUES
('2010UM10','2010Uw37','2010Uw38','2010Uw39','2010Uw40','2010AM10','2010Aw37','2010Aw38','2010Aw39','2010Aw40')
TABLESPACE GP2010M10TAB)
(PARTITION gp_rslt_acum_001 VALUES LESS THAN ('12942100') TABLESPACE GPSTRM01TAB
,PARTITION gp_rslt_acum_007 VALUES LESS THAN ('30331100') TABLESPACE GPSTRM07TAB
,PARTITION gp_rslt_acum_032 VALUES LESS THAN (MAXVALUE)   TABLESPACE GPSTRM32TAB)
/
truncate table gfc_gp_rslt_acum;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_001_X2010M08);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_007_X2010M08);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_032_X2010M08);
commit;

INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_001_X2010M09);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_007_X2010M09);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_032_X2010M09);
commit;

INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_001_X2010M10);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_007_X2010M10);
commit;
INSERT /*+APPEND NOLOGGING*/ INTO gfc_gp_rslt_acum SELECT * FROM ps_gp_rslt_acum SUBPARTITION (GP_RSLT_ACUM_032_X2010M10);
commit;
```

2. I will create a 'statistics table' that I have called GFC_STAT_TABLE into which I can export statistics. Note that I have compressed the primary key index that is automatically created with the table. I have created an additional index which improves the performance of some the queries that correct the statistics.

```
EXECUTE sys.dbms_stats.create_stat_table('SYSADM','GFC_STAT_TABLE');
ALTER INDEX sysadm.gfc_stat_table REBUILD COMPRESS PARALLEL;
CREATE INDEX sysadm.gfc_stat_table2 ON sysadm.gfc_stat_table (type, c5, c1, c2, c3, c4) COMPRESS PARALLEL;
```

3. I will delete and regather the statistics in a number of ways, and will export each set of statistics. Note that I have only collected histograms on certain columns.

    a. I have used table preferences (introduced in 11g) to specify parameters that are common to all tests.

```
--set table preferences so we dont have to set parameters in dbms_stats
BEGIN
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'CASCADE'
      ,pvalue  => 'TRUE'
      );
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'METHOD_OPT'
      ,pvalue  => 'FOR ALL COLUMNS SIZE 1, FOR COLUMNS SIZE 254 CAL_RUN_ID, CAL_ID, ORIG_CAL_RUN_ID'
      );
END;
/
TRUNCATE TABLE sysadm.gfc_stat_table;
```

b. First I will gather the statistics on my table at all levels. I have disabled incremental statistics. These are exported with the ID of 'ALL'. This is the method I used on 10g before implementing aggregated statistics.

   i. This took 147s on my laptop. No coincidence that it took nearly 3 times as long as the other options, because it is scanning the table three times: once for sub-partition statistics, once for partition statistics, and once for table level statistics.

```
Set timi on serveroutput on
--create statistics the old wrong way
BEGIN
    sys.dbms_stats.delete_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      );
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'INCREMENTAL'
      ,pvalue  => 'FALSE'
      );
END;
/
BEGIN
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': started');
    sys.dbms_stats.gather_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,granularity => 'ALL'
      );
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': finished');
END;
/
BEGIN
    DELETE FROM sysadm.gfc_stat_table WHERE statid = 'ALL';
    sys.dbms_stats.export_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'ALL'
      );
end;
/
```

 ©GO-FASTER CONSULTANCY LTD.

c. Next I will clear all the statistics, at all levels, on the table and then only gather statistics as I propose we should be doing, i.e. at sub-partition level but with incremental statistics disabled. I will also export them to the table with the ID of 'SUBPART'.

i. This took just 50s.

```
--create statistics the old way
begin
    sys.dbms_stats.delete_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      );
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'INCREMENTAL'
      ,pvalue  => 'FALSE'
      );
END;
/
BEGIN
    FOR i IN (SELECT * FROM dba_tab_subpartitions
              WHERE table_owner = 'SYSADM' and table_name = 'GFC_GP_RSLT_ACUM'
    ) LOOP[4]
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': '||i.subpartition_name);
    sys.dbms_stats.gather_table_Stats
      (ownname =>i.table_owner
      ,tabname =>i.table_name
      ,partname=>i.subpartition_name
      ,granularity => 'SUBPARTITION'
      );
    END LOOP;
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': finished');
END;
/
BEGIN
    DELETE FROM sysadm.gfc_stat_table WHERE statid = 'SUBPART';
    sys.dbms_stats.export_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'SUBPART'
      );
end;
/
```

[4] I could have collected the statistics with a single call to *dbms_stats*, but I am using a loop to collect them on each physical partition to emphasise that the statistics are built for a number of independent calls.

d. Next I will repeat the previous test, but with with incremental statistics enabled. I will export statistics to the table with the ID of 'SUBPINC'.

   i. This also took 50s, so there doesn't appear to be an overhead for incremental stats.

```
--create statistics the old way but with incremental stats enabled
begin
    sys.dbms_stats.delete_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      );
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'INCREMENTAL'
      ,pvalue  => 'TRUE'
      );
END;
/
BEGIN
    FOR i IN (SELECT * FROM dba_tab_subpartitions
              WHERE table_owner = 'SYSADM' and table_name = 'GFC_GP_RSLT_ACUM'
    ) LOOP
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': '||i.subpartition_name);
    sys.dbms_stats.gather_table_Stats
      (ownname =>i.table_owner
      ,tabname =>i.table_name
      ,partname=>i.subpartition_name
      ,granularity => 'SUBPARTITION'
      );
    END LOOP;
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': finished');
END;
/
BEGIN
    DELETE FROM sysadm.gfc_stat_table WHERE statid = 'SUBPINC';
    sys.dbms_stats.export_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'SUBPINC'
      );
end;
/
```


©GO-FASTER CONSULTANCY LTD.

e. Just for comparison, I will also collect statistics using the new granularity introduced in 11g; APPROX_GLOBAL AND PARTITION

```
begin
    sys.dbms_stats.delete_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      );
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'INCREMENTAL'
      ,pvalue  => 'TRUE'
      );
END;
/
BEGIN
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': started');
    sys.dbms_stats.gather_table_Stats
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,granularity => 'APPROX_GLOBAL AND PARTITION'
      );
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss')||': finished');
END;
/
BEGIN
    DELETE FROM sysadm.gfc_stat_table WHERE statid = 'AGPINC';
    sys.dbms_stats.export_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'AGPINC'
      );
end;
/
```

Now, I can compare the sets of statistics.

Collected Statistics

4. Lets take a look at the stats we collected and exported using the following query

```
set lines 150 pages 999
break on c4 skip 1 on statid skip 1
--break on c4 skip 1 on statid skip 1 on segment skip 1
column seq format 999
column statid format a7
column segment format a25
column c1 format a18 Heading 'C1|Table'
column c2 format a16 heading 'C2|Partition'
column c3 format a25 heading 'C3|Subpartition'
column c4 format a10 heading 'C4|Column'
column n1 format 999,999 heading 'N1|NDV'
column n2 heading 'N2|Density'
column n3 format 999,999
column r1 format a16 heading 'R1|High Value'
column r2 format a16 heading 'R2|Low Value'
clear screen
spool dmk

WITH x AS (
select c4, statid, c2, c3
, COALESCE(s.c3, s.c2, s.c1) segment
, row_number() over (partition by statid, c1, c2, c3, c4, c5 order by n10) as seq
, s.n1, s.n2, s.n3
, s.n10
, s.n11
, sysadm.agg_stats.display_raw(s.r1, c.data_type) r1
, sysadm.agg_stats.display_raw(s.r2, c.data_type) r2
FROM sysadm.gfc_stat_table s
        inner join all_tab_columns c
        on c.owner=s.c5
        and c.table_name = s.c1
        and c.column_name = s.c4
WHERE s.c1 = 'GFC_GP_RSLT_ACUM'
--And s.c2 = 'GP_RSLT_ACUM_001'
and s.c5 = 'SYSADM'
--and s.statid = 'SUBPART'
and s.c4 IN('EMPLID','CAL_RUN_ID','CAL_ID')
ORDER BY c4, statid, c1, c2, c3, n10
)
SELECT DISTINCT c4, statid, segment, n1, 1/n2, n3, r1, r2
FROM X
ORDER BY c4 DESC, statid, segment desc
/
spool off
```


©GO-FASTER CONSULTANCY LTD.

I have used footnotes to indicate to exactly what the comment refers.

a. Column: EMPLID.

C4			N1		R1	R2
Column	STATID	SEGMENT	NDV	1/N2	N3 High Value	Low Value
EMPLID	AGPINC[5]	GFC_GP_RSLT_ACUM	11,928[6]	11928	11,928 10000100	99999000
		GP_RSLT_ACUM_001	4,204	4204	4,204 10000100	12942000
		GP_RSLT_ACUM_007	3,857	3857	3,857 27522100	30331000
		GP_RSLT_ACUM_032	3,867	3867	3,867 97300500	99999000
	ALL	GFC_GP_RSLT_ACUM	11,928	11928	11,928 10000100	99999000
		GP_RSLT_ACUM_001	4,204[7]	4204	4,204 10000100	12942000[8]
		GP_RSLT_ACUM_001_X2010M08	4,058	4058	4,058 10000100	12942000
		GP_RSLT_ACUM_001_X2010M09	4,120	4120	4,120 10000100	12942000
		GP_RSLT_ACUM_001_X2010M10	4,114	4114	4,114 10000100	12942000
		GP_RSLT_ACUM_007	3,857[9]	3857	3,857 27522100	30331000
		GP_RSLT_ACUM_007_X2010M08	3,706	3706	3,706 27522100	30331000
		GP_RSLT_ACUM_007_X2010M09	3,767	3767	3,767 27522100	30331000
		GP_RSLT_ACUM_007_X2010M10	3,761	3761	3,761 27522100	30331000
		GP_RSLT_ACUM_032	3,867	3867	3,867 97300500	99999000
		GP_RSLT_ACUM_032_X2010M08	3,729	3729	3,729 97300500	99999000
		GP_RSLT_ACUM_032_X2010M09	3,775	3775	3,775 97300500	99999000
		GP_RSLT_ACUM_032_X2010M10	3,751	3751	3,751 97300500	99999000

[5] Note that APPROX_GLOBAL AND PARTITION granularity does not collect sub-partition statistics.

[6] The actual number of distinct employee IDs is 11928. Oracle 11g statistics contain the correct answer, although statistics were only collected with the default automatic sample size. In Oracle 10g the number of distinct values was only 8394.

[7] As in 10g, the number of distinct employee IDs on the partition is greater than that on any sub-partition, and this is as expected.

[8] In 10g, the high value on the global statistics on partition GP_RSLT_ACUM_001 was less than the maximum high value of the constituent sub-partitions. This is not the case in 11g without incremental statistics.

[9] In 10g we saw that the number of distinct values on a partition could be less than on a constituent sub-partition. This is clearly impossible, but was another effect of sampling. This problem has not been observed in 11g.

SUBPART	GFC_GP_RSLT_ACUM	11,662[10]	11662	0	10000100	99999000
	GP_RSLT_ACUM_001	4,120[11]	4120	0	10000100	12942000
	GP_RSLT_ACUM_001_X2010M08	4,058	4058	4,058	10000100	12942000
	GP_RSLT_ACUM_001_X2010M09	4,120	4120	4,120	10000100	12942000
	GP_RSLT_ACUM_001_X2010M10	4,114	4114	4,114	10000100	12942000
	GP_RSLT_ACUM_007	3,767	3767	0	27522100	30331000
	GP_RSLT_ACUM_007_X2010M08	3,706	3706	3,706	27522100	30331000
	GP_RSLT_ACUM_007_X2010M09	3,767	3767	3,767	27522100	30331000
	GP_RSLT_ACUM_007_X2010M10	3,761	3761	3,761	27522100	30331000
	GP_RSLT_ACUM_032	3,775	3775	0	97300500	99999000
	GP_RSLT_ACUM_032_X2010M08	3,729	3729	3,729	97300500	99999000
	GP_RSLT_ACUM_032_X2010M09	3,775	3775	3,775	97300500	99999000
	GP_RSLT_ACUM_032_X2010M10	3,751	3751	3,751	97300500	99999000
SUBPINC[12]	GFC_GP_RSLT_ACUM	11,662	11662	0	10000100	99999000
	GP_RSLT_ACUM_001	4,120	4120	0	10000100	12942000
	GP_RSLT_ACUM_001_X2010M08	4,058	4058	4,058	10000100	12942000
	GP_RSLT_ACUM_001_X2010M09	4,120	4120	4,120	10000100	12942000
	GP_RSLT_ACUM_001_X2010M10	4,114	4114	4,114	10000100	12942000
	GP_RSLT_ACUM_007	3,767	3767	0	27522100	30331000
	GP_RSLT_ACUM_007_X2010M08	3,706	3706	3,706	27522100	30331000
	GP_RSLT_ACUM_007_X2010M09	3,767	3767	3,767	27522100	30331000
	GP_RSLT_ACUM_007_X2010M10	3,761	3761	3,761	27522100	30331000
	GP_RSLT_ACUM_032	3,775	3775	0	97300500	99999000
	GP_RSLT_ACUM_032_X2010M08	3,729	3729	3,729	97300500	99999000
	GP_RSLT_ACUM_032_X2010M09	3,775	3775	3,775	97300500	99999000
	GP_RSLT_ACUM_032_X2010M10	3,751	3751	3,751	97300500	99999000

[10] For the column upon which the table is range partitioned, *dbms_stats* correctly adds up the number of distinct values in the range partitions to produce the number of distinct values in the table. It knows it can do this because the low and high values do not overlap. This figure was also correct in 11g.

[11] In 10g, the number of distinct values on the partition could be less that the largest number of distinct values of the constituent sub-partitions. This is not possible, it must be at the least the maximum value of NDV on the sub-partitions, and may, as we see in the global statistics, be higher. This problem does has not been observered in 11g.

[12] Enabling incremental statistics has no effect on the aggregated statistics created by *dbms_stats*. Synonpses are created when incremental statistics are enabled, but there is only a single hash. Whereas if statistics are collected with granularity APPROX_GLOBAL AND PARTITION full synopses are created.

 ©GO-FASTER CONSULTANCY LTD.

b. CAL_RUN_ID. This table is list sub-partitoned on CAL_RUN_ID andwe can see that while the global statistics are reasonable, both are wrong, but in different ways.

Each of the list sub-partitions within a range partition contains a distinct list of CAL_RUN_IDs. So the number of distinct CAL_RUN_IDs for the partition should be the sum of the number of distinct values on each constituent sub-partitions.

C4			N1		R1	R2
Column	STATID	SEGMENT	NDV	1/N2	N3	High Value	Low Value
CAL_RUN_ID	AGPINC	GFC_GP_RSLT_ACUM	18	24782210	18	2010GL08	2010UL10
		GP_RSLT_ACUM_001	18	8507804.96	18	2010GL08	2010UL10
		GP_RSLT_ACUM_007	18	7866520.24	18	2010GL08	2010UL10
		GP_RSLT_ACUM_032	18	8306537.03	18	2010GL08	2010UL10
	ALL	GFC_GP_RSLT_ACUM	18	24993985.2	18	2010GL08	2010UL10
		GP_RSLT_ACUM_001	18	8694255.93	18	2010GL08	2010UL10
		GP_RSLT_ACUM_001_X2010M08	6	2694370.83	6	2010GL08	2010UL08
		GP_RSLT_ACUM_001_X2010M09	6	2894468.98	6	2010GL09	2010UL09
		GP_RSLT_ACUM_001_X2010M10	6	3036732.7	6	2010GL10	2010UL10
		GP_RSLT_ACUM_007	18	8157605.99	18	2010GL08	2010UL10
		GP_RSLT_ACUM_007_X2010M08	6	2529399.79	6	2010GL08	2010UL08
		GP_RSLT_ACUM_007_X2010M09	6	2632673.43	6	2010GL09	2010UL09
		GP_RSLT_ACUM_007_X2010M10	6	2772372.62	6	2010GL10	2010UL10
		GP_RSLT_ACUM_032	18	8324513.31	18	2010GL08	2010UL10
		GP_RSLT_ACUM_032_X2010M08	6	2574526.23	6	2010GL08	2010UL08
		GP_RSLT_ACUM_032_X2010M09	6	2821980.5	6	2010GL09	2010UL09
		GP_RSLT_ACUM_032_X2010M10	6	2908695.03	6	2010GL10	2010UL10
	SUBPART	GFC_GP_RSLT_ACUM	6	6	0	2010GL08	2010UL10
		GP_RSLT_ACUM_001	6[13]	6	0	2010GL08	2010UL10
		GP_RSLT_ACUM_001_X2010M08	6	2721954.88	6	2010GL08	2010UL08
		GP_RSLT_ACUM_001_X2010M09	6	2804718	6	2010GL09	2010UL09
		GP_RSLT_ACUM_001_X2010M10	6	3015740.18	6	2010GL10	2010UL10
		GP_RSLT_ACUM_007	6	6	0	2010GL08	2010UL10
		GP_RSLT_ACUM_007_X2010M08	6	2485922.19	6	2010GL08	2010UL08
		GP_RSLT_ACUM_007_X2010M09	6	2621160.28	6	2010GL09	2010UL09
		GP_RSLT_ACUM_007_X2010M10	6	2736275.34	6	2010GL10	2010UL10
		GP_RSLT_ACUM_032	6	6	0	2010GL08	2010UL10
		GP_RSLT_ACUM_032_X2010M08	6	2556240.58	6	2010GL08	2010UL08
		GP_RSLT_ACUM_032_X2010M09	6	2805665.53	6	2010GL09	2010UL09
		GP_RSLT_ACUM_032_X2010M10	6	2877016.68	6	2010GL10	2010UL10

[13] The number of distinct CAL_RUN_ID values in the partitions should be 18. Oracle 11g has made the same mistake as 10g. This problem occurs with composite partitioned tables that are list and hash subpartitioned. It does not occur on range subpartitioned tables.

```
SUBPINC GFC_GP_RSLT_ACUM                6          6        0 2010GL08          2010UL10
        GP_RSLT_ACUM_001                6[14]          6        0 2010GL08          2010UL10
        GP_RSLT_ACUM_001_X2010M08       6 2690430.25        6 2010GL08          2010UL08
        GP_RSLT_ACUM_001_X2010M09       6 2864204.11        6 2010GL09          2010UL09
        GP_RSLT_ACUM_001_X2010M10       6 3064906.86        6 2010GL10          2010UL10
        GP_RSLT_ACUM_007                6          6        0 2010GL08          2010UL10
        GP_RSLT_ACUM_007_X2010M08       6 2520341.96        6 2010GL08          2010UL08
        GP_RSLT_ACUM_007_X2010M09       6 2623079.13        6 2010GL09          2010UL09
        GP_RSLT_ACUM_007_X2010M10       6 2781524.04        6 2010GL10          2010UL10
        GP_RSLT_ACUM_032                6          6        0 2010GL08          2010UL10
        GP_RSLT_ACUM_032_X2010M08       6 2578277.14        6 2010GL08          2010UL08
        GP_RSLT_ACUM_032_X2010M09       6 2752132.02        6 2010GL09          2010UL09
        GP_RSLT_ACUM_032_X2010M10       6 2821969.07        6 2010GL10          2010UL10
```

[14] The same error described in footnote 1313 occurs even if INCREMENTAL statistics were enabled.

 ©GO-FASTER CONSULTANCY LTD.

c. CAL_ID. This column has histograms, though I have not shown the individual buckets.

C4			N1		R1	R2	
Column	STATID	SEGMENT	NDV	1/N2	N3	High Value	Low Value
CAL_ID	AGPINC	GFC_GP_RSLT_ACUM	166	24782210	166	ABCDEF 2009AW45	UKPAY 2010UW25
		GP_RSLT_ACUM_001	150	8507804.96	150	ABCDEF 2009AW52	UKPAY 2010M12
		GP_RSLT_ACUM_007	141	7866520.24	141	ABCDEF 2009AW52	UKPAY 2010UW25
		GP_RSLT_ACUM_032	155	8306537.03	155	ABCDEF 2009AW45	UKPAY 2010M10
	ALL	GFC_GP_RSLT_ACUM	166	24993985.2	166	ABCDEF 2009AW45	UKPAY 2010UW25
		GP_RSLT_ACUM_001	150	8694255.93	150	ABCDEF 2009AW52	UKPAY 2010M12
		GP_RSLT_ACUM_001_X2010M08	87	2694370.83	87	ABCDEF 2010AW01	UKPAY 2010M08
		GP_RSLT_ACUM_001_X2010M09	122	2894468.98	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_001_X2010M10	85	3036732.7	85	ABCDEF 2010AW01	UKPAY 2010M12
		GP_RSLT_ACUM_007	141	8157605.99	141	ABCDEF 2009AW52	UKPAY 2010UW25
		GP_RSLT_ACUM_007_X2010M08	54	2529399.79	54	ABCDEF 2010AW24	UKPAY 2010M08
		GP_RSLT_ACUM_007_X2010M09	122	2632673.43	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_007_X2010M10	56	2772372.62	56	ABCDEF 2010AW36	UKPAY 2010UW25
		GP_RSLT_ACUM_032	155	8324513.31	155	ABCDEF 2009AW45	UKPAY 2010M10
		GP_RSLT_ACUM_032_X2010M08	102	2574526.23	102	ABCDEF 2010AW02	UKPAY 2010M08
		GP_RSLT_ACUM_032_X2010M09	136	2821980.5	136	ABCDEF 2009AW45	UKPAY 2010M09
		GP_RSLT_ACUM_032_X2010M10	128	2908695.03	128	ABCDEF 2009AW49	UKPAY 2010M10
	SUBPART	GFC_GP_RSLT_ACUM	136	136	0	ABCDEF 2009AW45	UKPAY 2010UW25
		GP_RSLT_ACUM_001	122	122	0	ABCDEF 2009AW52	UKPAY 2010M12
		GP_RSLT_ACUM_001_X2010M08	87	2721954.88	87	ABCDEF 2010AW01	UKPAY 2010M08
		GP_RSLT_ACUM_001_X2010M09	122	2804718	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_001_X2010M10	85	3015740.18	85	ABCDEF 2010AW01	UKPAY 2010M12
		GP_RSLT_ACUM_007	122	122	0	ABCDEF 2009AW52	UKPAY 2010UW25
		GP_RSLT_ACUM_007_X2010M08	54	2485922.19	54	ABCDEF 2010AW24	UKPAY 2010M08
		GP_RSLT_ACUM_007_X2010M09	122	2621160.28	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_007_X2010M10	56	2736275.34	56	ABCDEF 2010AW36	UKPAY 2010UW25
		GP_RSLT_ACUM_032	136	136	0	ABCDEF 2009AW45	UKPAY 2010M10
		GP_RSLT_ACUM_032_X2010M08	102	2556240.58	102	ABCDEF 2010AW02	UKPAY 2010M08
		GP_RSLT_ACUM_032_X2010M09	136	2805665.53	136	ABCDEF 2009AW45	UKPAY 2010M09
		GP_RSLT_ACUM_032_X2010M10	128	2877016.68	128	ABCDEF 2009AW49	UKPAY 2010M10
	SUBPINC	GFC_GP_RSLT_ACUM	136	136	0	ABCDEF 2009AW45	UKPAY 2010UW25
		GP_RSLT_ACUM_001	122	122	0	ABCDEF 2009AW52	UKPAY 2010M12
		GP_RSLT_ACUM_001_X2010M08	87	2690430.25	87	ABCDEF 2010AW01	UKPAY 2010M08
		GP_RSLT_ACUM_001_X2010M09	122	2864204.11	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_001_X2010M10	85	3064906.86	85	ABCDEF 2010AW01	UKPAY 2010M12
		GP_RSLT_ACUM_007	122	122	0	ABCDEF 2009AW52	UKPAY 2010UW25
		GP_RSLT_ACUM_007_X2010M08	54	2520341.96	54	ABCDEF 2010AW24	UKPAY 2010M08
		GP_RSLT_ACUM_007_X2010M09	122	2623079.13	122	ABCDEF 2009AW52	UKPAY 2010M09
		GP_RSLT_ACUM_007_X2010M10	56	2781524.04	56	ABCDEF 2010AW36	UKPAY 2010UW25
		GP_RSLT_ACUM_032	136	136	0	ABCDEF 2009AW45	UKPAY 2010M10
		GP_RSLT_ACUM_032_X2010M08	102	2578277.14	102	ABCDEF 2010AW02	UKPAY 2010M08
		GP_RSLT_ACUM_032_X2010M09	136	2752132.02	136	ABCDEF 2009AW45	UKPAY 2010M09
		GP_RSLT_ACUM_032_X2010M10	128	2821969.07	128	ABCDEF 2009AW49	UKPAY 2010M10

Implementation of New Approach

The aggregated statistics will be created and managed with the *agg_stats* packaged procedure. The package has been slightly modified for 11g to remove the parameters that can be specified via a table preference.

The demonstration in this section uses the script on page 5, that imulates the way the statistics are currently collected, to set up the statistics as a starting point.

5. Although it is possible to pass METHOD_OPT as a parameter to the *agg_stats* package though to *dbms_stats*, I recommend instead setting table preferences to determine the behaviour of *dbms_stats*.

```
BEGIN
    sys.dbms_stats.set_table_prefs
      (ownname => 'SYSADM'
      ,tabname => 'GFC_GP_RSLT_ACUM'
      ,pname   => 'METHOD_OPT'
      ,pvalue  => 'FOR ALL COLUMNS SIZE 1, FOR COLUMNS SIZE 254 CAL_RUN_ID, CAL_ID,
ORIG_CAL_RUN_ID'
      );
END;
/
```

6. Gathering, fixing and reimporting the statistics in one step.

   The *agg_stats* package contains a single procedure, *gather_fix_stats*, which will collect and fix stats on a named table. This procedure can then be called from a scheduled database job.

```
BEGIN
 agg_stats.gather_fix_stats
 (p_tabname=>'GFC_GP_RSLT_ACUM'
 ,p_statid=>'SUBPART',p_force=>TRUE
 );
END;
/
```

However, the constituent operations can also be run one by one, and I will demonstrate that in the following sections.

 ©GO-FASTER CONSULTANCY LTD.

a. Delete old Global Statistics on any logical segments. Previously global statistics had been collected on all segments in the table, but we need to get rid of them to make *dbms_stats* do the aggregation for us.

```
set serveroutput on
DECLARE
 l_stats_changed BOOLEAN;
BEGIN
 agg_stats.delete_global_stats(p_tabname=>'GFC_GP_RSLT_ACUM' ,p_force=>TRUE
                              ,p_stats_changed=>l_stats_changed);
END;
/

07:40:58 11.07.2012:delete_global_stats(p_ownname=>SYSADM, p_tabname=>GFC_GP_RSLT_ACUM, p_force=>TRUE,
p_stats_changed=>FALSE)
07:40:58 11.07.2012:Deleting Table Level Statistics on SYSADM.GFC_GP_RSLT_ACUM
07:40:59 11.07.2012:Deleting Partition Statistics on SYSADM.GFC_GP_RSLT_ACUM(GP_RSLT_ACUM_001)
07:41:00 11.07.2012:Deleting Partition Statistics on SYSADM.GFC_GP_RSLT_ACUM(GP_RSLT_ACUM_007)
07:41:00 11.07.2012:Deleting Partition Statistics on SYSADM.GFC_GP_RSLT_ACUM(GP_RSLT_ACUM_032)
07:41:00 11.07.2012:Deleted Statistics on 4 logical partitons

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.59
```

b. Now we need to collect statistics to create aggregated statistics, so this procedure will collects stats on any physical partitions whose stats are missing or stale. Then it will collect stats on the last sub-partition in each partition where the partition doesn't have stats, and the last partition in the table where the table doesn't have stats. This will make *dbms_stats* do the aggregation.

   i. I have specified *p_allparts=>TRUE* to force collection of statistics whether stale or note. This is only necessary on once occasion when transitioning to a new way of collecting statistics, such as a restricted list of histograms.

```
set serveroutput on
DECLARE
 l_stats_changed BOOLEAN;
BEGIN
 agg_stats.gather_table_stats(p_tabname=>'GFC_GP_RSLT_ACUM', p_allparts=>TRUE
                              ,p_force=>TRUE,p_stats_changed=>l_stats_changed);
END;
/

08:00:24 11.07.2012:gather_table_stats(p_ownname=>SYSADM, p_tabname=>GFC_GP_RSLT_ACUM, p_allparts=>TRUE,
p_force=>TRUE, p_stats_changed=>FALSE)
08:00:24 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_001_X2010L09)
08:00:30 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_032_X2010L10)
08:00:36 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_032_X2010L09)
08:00:42 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_007_X2010L08)
08:00:47 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_032_X2010L08)
08:00:52 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_001_X2010L08)
08:00:58 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_001_X2010L10)
08:01:04 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_007_X2010L10)
08:01:10 11.07.2012:Gathering statistics on SYSADM.GFC_GP_RSLT_ACUM physical subpartition
(GP_RSLT_ACUM_007_X2010L09)
08:01:15 11.07.2012:0 jobs outstanding (0 running, 0 scheduled)
08:01:15 11.07.2012:Gathered Statistics on 9 physical partitons
PL/SQL procedure successfully completed.
```

c. Now we need to fix stats the generated stats when NDV is obviously too low. First we export the statistics to the statistics table, and do sub-partition aggregation to partition first, and that rolls up into the partition to table aggregation.

```
DECLARE
 l_stats_changed BOOLEAN;
BEGIN
 agg_stats.fix_stats(p_tabname=>'GFC_GP_RSLT_ACUM',p_statid=>'SUBPART',p_stats_changed=>l_stats_changed);
```

```
END;
/
08:05:02 11.07.2012:fix_stats(p_ownname=>SYSADM, p_tabname=>GFC_GP_RSLT_ACUM, p_statid=>SUBPART,
p_stats_changed=>FALSE)
08:05:02 11.07.2012:C:partition:SYSADM.GFC_GP_RSLT_ACUM.GP_RSLT_ACUM_001:col CAL_RUN_ID:NDV 6->18
08:05:02 11.07.2012:C:partition:SYSADM.GFC_GP_RSLT_ACUM.GP_RSLT_ACUM_007:col CAL_RUN_ID:NDV 6->18
08:05:02 11.07.2012:C:partition:SYSADM.GFC_GP_RSLT_ACUM.GP_RSLT_ACUM_032:col CAL_RUN_ID:NDV 6->18
08:05:02 11.07.2012:C:table SYSADM.GFC_GP_RSLT_ACUM:col CAL_RUN_ID:NDV 6->18
```

So far, only the contents of the statistics table have been updated.

d. Finally, the corrected statistics are reimported from the statistics table into the data dictionary.

```
BEGIN
    sys.dbms_stats.import_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'SUBPART'
      ,cascade=>TRUE
      );
END;
/
```

Installation Instructions

The installation instructions are unchanged since 10g.

7. It would not be unreasonable to increase the value of the JOB_QUEUE_PROCESSES parameter from the default setting of 10 to be equal to the value of CPU_COUNT.

8. The *agg_stats* package requires certain privileges to be directly granted by SYS (not via a role)

```
GRANT EXECUTE ON dbms_lock TO sysadm;
GRANT EXECUTE ON dbms_scheduler TO sysadm;
GRANT CREATE JOB TO sysadm;
```

9. The script *agg_stats.sql* (see page 24) creates the statistics table GFC_STAT_TABLE and a package of procedures

Implementation Instructions

10. As a one-off action for each table to use the new approach, I suggest that the existing statistics shoud be exported to the statistics table so that they can be restored later if necessary. This will also be useful during testing to switch back and forth at will between the two sets of statistics as necessary.

```
BEGIN
    sys.dbms_stats.export_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'ORIGINAL'
      ,cascade=>TRUE
      );
END;
/
```


©GO-FASTER CONSULTANCY LTD.

11. Although you can pass the METHOD_OPT parameter into AGG_STATS, I recommend that table preferences be set on the table. Note that statistics on the table will be locked so that the default schema wide jobs do not pick the table up.

```
BEGIN
  sys.dbms_stats.set_table_prefs
  (ownname => user
  ,tabname => i.table_name
  ,pname => 'CASCADE'
  ,pvalue => 'TRUE'
  );
  sys.dbms_stats.set_table_prefs
  (ownname => user
  ,tabname => i.table_name
  ,pname => 'METHOD_OPT'
  ,pvalue => 'FOR ALL COLUMNS SIZE 1, FOR COLUMNS SIZE 254 CAL_RUN_ID, CAL_ID, ORIG_CAL_RUN_ID'
  );
  sys.dbms_stats.lock_table_stats
  (ownname => user
  ,tabname => i.table_name
  );
END;
/
```

12. Collecting and correcting statistics on a given table can be performed with a single command.

    a. If you are changing the way you collect statistics on the physical partitions or sub-partitions, then you may want to collect statistics on all partitions whether or not they are stale. For example, if you change the number of buckets in histograms on certain columns.

       p_allparts parameters is set to true to force collection on all sub-partitions. Otherwise the parameter defaults to false, and the package only collects statistics that are considered to be stale.

    b. Collecting sub-partition statistics on a whole table does not use database parallelism well. Therefore, I have enhanced the package to optionally collect statistics for all physical partitions or sub-partitions whether they are stale or otherwise, and also to optionally submit requests to collect statistics to the database scheduler, so that multiple requests can run in parallel. The first time

       If the *p_sched_job* parameter is set to true, a job will be submitted to the Oracle job scheduler for each partition or sub-partitions. *agg_stats.gather_fix_stats* will wait for all the jobs to complete before it returns. Otherwise, the parameter defaults to false, and the package sequentially executes calls to *dbms_stats.gather_table_stats* for each partition that needs to be analysed.

```
Spool dmk
Set serveroutput on lines 200 pages 0
BEGIN
 agg_stats.gather_fix_stats
 (p_tabname=>'GFC_GP_RSLT_ACUM'
```

```
 ,p_allparts=>TRUE /*force collection on all partitions regardless*/
 ,p_statid=>'SUBPART'
 ,p_force=>TRUE
 ,p_sched_job=>TRUE /*collect stats in scheduled jobs*/
 );
END;
/
Spool off
```

c. Subsequently, it is only necessary to gather stale statistics, so the following commands should be run from database job. Currently we only collect statistics once per week on Saturday mornings. However, with the new approach stale statistics could be refreshed daily.

   i. In the case of the payroll tables suggest the following commands be run daily before the payroll batch processes.

```
BEGIN
 agg_stats.gather_fix_stats
 (p_tabname=>'GFC_GP_RSLT_ACUM'
 ,p_allparts=>FALSE /*force collection on all partitions regardless*/
 ,p_statid=>'SUBPART'
 ,p_force=>TRUE
 ,p_sched_job=>TRUE /*collect stats in scheduled jobs*/
 );
END;
/
```

13. Done. So what have we got in the statistics table?

```
Column owner format a8
Column table_name format a18
Column statid format a10
break on owner skip 1 on table_name skip 1
select c5 owner
, c1 table_name
, statid
, count(*)
from gfc_stat_table
group by statid, c5, c1
order by 1,2,3
/
```

```
OWNER    TABLE_NAME         STATID       COUNT(*)
-------- ------------------ ---------- ----------
SYSADM   GFC_GP_RSLT_ACUM   AGPINC            855
                            ALL              2117
                            SUBPART          1436
                            SUBPINC          1428
```

 ©GO-FASTER CONSULTANCY LTD.

14. To restore the original statistics we merely need to reimport them from the statistics table.

```
BEGIN
    sys.dbms_stats.import_table_stats
      (ownname=>'SYSADM'
      ,tabname=>'GFC_GP_RSLT_ACUM'
      ,statown=>'SYSADM'
      ,stattab=>'GFC_STAT_TABLE'
      ,statid=>'ORIGINAL'
      ,cascade=>TRUE
      ,force=>TRUE
      );
END;
/
```

APPENDIX

Source Code

```
REM agg_stats11.sql
REM (c)Go-Faster Consultancy 2012-3
-------------------------------------------------------------------------------
REM documentation: http://www.go-faster.co.uk/Partition.Statistics.11gR2.pdf
-------------------------------------------------------------------------------
REM see also http://www.centrexcc.com/SQL%20Tuning%20with%20Statistics.ppt.pdf
REM see also http://skdba.blogspot.com/2006/06/keeping-history-of-cbo-stats.html
REM see also http://oracledoug.com/serendipity/index.php?/archives/1590-Statistics-on-Partitioned-Tables-Contents.html
-------------------------------------------------------------------------------

spool agg_stats
set serveroutput on timi on echo on

REM This package can submit requests to job scheduler, consider increasing JOB_PROCESSES to the same value as CPU_COUNT

REM requires the following grants to be made explicitly by SYS
GRANT EXECUTE ON dbms_lock TO sysadm;
GRANT EXECUTE ON dbms_scheduler TO sysadm;
GRANT CREATE JOB TO sysadm;

ROLLBACK;

--NB this package does not handle global partitioned indexes

--The Stat Table must be created before the package which references it
--EXECUTE sys.dbms_stats.drop_stat_table('SYSADM', 'GFC_STAT_TABLE');
EXECUTE sys.dbms_stats.create_stat_table('SYSADM','GFC_STAT_TABLE');
--ALTER INDEX sysadm.gfc_stat_table REBUILD COMPRESS PARALLEL;
CREATE INDEX sysadm.gfc_stat_table2 ON sysadm.gfc_stat_table (type, c5, c1, c2, c3, c4) COMPRESS PARALLEL;

-------------------------------------------------------------------------------
CREATE OR REPLACE PACKAGE sysadm.agg_stats AS
-------------------------------------------------------------------------------
--documentation: http://www.go-faster.co.uk/Partition.Statistics.11gR2.pdf
-------------------------------------------------------------------------------
--10.05.2012 - initial release
--30.10.2012 - corrections to all_tab_statistics queries
-------------------------------------------------------------------------------
--This procedure deletes statistics on any logical segments that have global statistics
-------------------------------------------------------------------------------
  PROCEDURE delete_global_stats
    (p_ownname       VARCHAR2 DEFAULT 'SYSADM'
    ,p_tabname       VARCHAR2
    ,p_force         BOOLEAN DEFAULT FALSE
    ,p_stats_changed IN OUT BOOLEAN
    ,p_testmode      BOOLEAN DEFAULT FALSE
    );
```


©GO-FASTER CONSULTANCY LTD.

```
--------------------------------------------------------------------------------
--Function to decode raw values in R1 and R2 statistics columns.  Not used elsehwere in this package
--stolen from http://structureddata.org/2007/10/16/how-to-display-high_valuelow_value-columns-from-user_tab_col_statistics/
--------------------------------------------------------------------------------
  FUNCTION display_raw
    (p_rawval RAW
    ,p_type   VARCHAR2
    ) RETURN VARCHAR2;
--------------------------------------------------------------------------------
--This procedure corrects the NDV and density column statistics that have been
--exported to the stat table when the value is lower than theoretically possible.
--------------------------------------------------------------------------------
  PROCEDURE fix_stats
    (p_ownname       VARCHAR2 DEFAULT 'SYSADM'
    ,p_tabname       VARCHAR2
    ,p_statid        VARCHAR2
    ,p_stats_changed IN OUT BOOLEAN
    ,p_testmode      BOOLEAN DEFAULT FALSE
    );
--------------------------------------------------------------------------------
--This procedure gathers statistics on physical partitions or sub-partitions whose statistics
--are either missing  or stale, or on the last partition or sub-partition in a table or partition
--that does not have aggregated statistics
--------------------------------------------------------------------------------
  PROCEDURE gather_table_stats
    (p_ownname       VARCHAR2 DEFAULT 'SYSADM'
    ,p_tabname       VARCHAR2
    ,p_method_opt    VARCHAR2 /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
    ,p_allparts      BOOLEAN DEFAULT FALSE
    ,p_force         BOOLEAN DEFAULT FALSE
    ,p_sched_job     BOOLEAN DEFAULT FALSE
    ,p_stats_changed IN OUT BOOLEAN
    ,p_testmode      BOOLEAN DEFAULT FALSE
    );
--------------------------------------------------------------------------------
--This procedure provides performs all the steps involved in gathering, exporting,
--correcting and reimporting statistics on a partitioned table
--------------------------------------------------------------------------------
  PROCEDURE gather_fix_stats
    (p_ownname    VARCHAR2 DEFAULT 'SYSADM'
    ,p_tabname    VARCHAR2
    ,p_method_opt VARCHAR2 DEFAULT NULL /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
    ,p_allparts   BOOLEAN DEFAULT FALSE
    ,p_statid     VARCHAR2
    ,p_force      BOOLEAN DEFAULT FALSE
    ,p_sched_job  BOOLEAN DEFAULT FALSE
    ,p_testmode   BOOLEAN DEFAULT FALSE
    );
--------------------------------------------------------------------------------
--This procedure calls gather_fix_stats for all partitioned tables whose stats are locked.
--------------------------------------------------------------------------------
  PROCEDURE gather_all_locked_part_tables
    (p_ownname    VARCHAR2 DEFAULT 'SYSADM'
    ,p_method_opt VARCHAR2 DEFAULT NULL /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
    ,p_allparts   BOOLEAN DEFAULT FALSE
```

```
    ,p_statid     VARCHAR2
    ,p_testmode   BOOLEAN DEFAULT FALSE
    );
--------------------------------------------------------------------------------
END agg_stats;
/
show errors
pause
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
CREATE OR REPLACE PACKAGE BODY sysadm.agg_stats AS
--------------------------------------------------------------------------------
k_ownname CONSTANT VARCHAR2(8)  := 'SYSADM';
k_stattab CONSTANT VARCHAR2(16) := 'GFC_STAT_TABLE';
k_module  CONSTANT VARCHAR2(48) := 'AGG_STATS.';
--------------------------------------------------------------------------------
--Procedure to emit messages
--------------------------------------------------------------------------------
FUNCTION display_bool
(p_bool BOOLEAN
) RETURN VARCHAR2 IS
BEGIN
  IF p_bool THEN
    RETURN 'TRUE';
  ELSE
    RETURN 'FALSE';
  END IF;
END display_bool;
--------------------------------------------------------------------------------
PROCEDURE msg
(p_msg VARCHAR2
,p_testmode BOOLEAN DEFAULT FALSE
) IS
BEGIN
  IF p_testmode THEN
    dbms_output.put_line('Test: '||TO_CHAR(SYSDATE,'hh24:mi:ss dd.mm.yyyy')||':'||p_msg);
  ELSE
    dbms_output.put_line(TO_CHAR(SYSDATE,'hh24:mi:ss dd.mm.yyyy')||':'||p_msg);
  END IF;
END msg;
--------------------------------------------------------------------------------
PROCEDURE exec_sql
(p_sql VARCHAR2
,p_testmode BOOLEAN DEFAULT FALSE) IS
BEGIN
  IF p_testmode THEN NULL;
    msg('Test SQL: '||p_sql);
  ELSE
    msg(p_sql);
    EXECUTE IMMEDIATE p_sql;
  END IF;
END exec_sql;
--------------------------------------------------------------------------------
--This procedure deletes global statistics on logical segments
--------------------------------------------------------------------------------
```

```
PROCEDURE delete_global_stats
(p_ownname       VARCHAR2 DEFAULT 'SYSADM'
,p_tabname       VARCHAR2
,p_force         BOOLEAN DEFAULT FALSE
,p_stats_changed IN OUT BOOLEAN
,p_testmode      BOOLEAN DEFAULT FALSE
) IS
  l_counter INTEGER := 0;
  l_module VARCHAR2(48);
  l_action VARCHAR2(32);
BEGIN
  dbms_application_info.read_module(module_name=>l_module, action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module||'DELETE_GLOBAL_STATS', action_name=>p_tabname);
  msg('delete_global_stats(p_ownname=>'||p_ownname||', p_tabname=>'||p_tabname||', p_force=>'||display_bool(p_force)||',
p_stats_changed=>'||display_bool(p_stats_changed)||')',p_testmode);
  FOR i IN ( --delete global table level stats where partition exists
          SELECT t.*
          FROM       all_tab_statistics t
          WHERE      t.owner = p_ownname
          AND        t.table_name = p_tabname
          AND        t.object_type = 'TABLE'
          AND        t.global_stats = 'YES'
          AND EXISTS(
                     SELECT 'x'
                     FROM       all_tab_partitions p
                     WHERE      p.table_owner = p_ownname
                     AND    p.table_name = p_tabname)
          ) LOOP
    msg('Deleting Table Level Statistics on '||i.owner||'.'||i.table_name,p_testmode);
    IF NOT p_testmode THEN
      sys.dbms_stats.delete_table_stats
        (ownname=>i.owner
        ,tabname=>i.table_name
        ,cascade_parts=>FALSE
        ,force=>p_force
        );
    END IF;
    l_counter := l_counter + 1;
  END LOOP;

  FOR i IN ( --delete partition level global stats where subpartition exists
          SELECT t.*
          FROM       all_tab_statistics t
          WHERE      t.owner = p_ownname
          AND        t.table_name = p_tabname
          AND        t.object_type = 'PARTITION'
          AND        t.global_stats = 'YES'
          AND EXISTS(
                     SELECT 'x'
                     FROM       all_tab_subpartitions p
                     WHERE      p.table_owner = p_ownname
                     AND        p.table_name = p_tabname
                     AND        p.partition_name = t.partition_name)
          ) LOOP
    msg('Deleting Partition Statistics on '||i.owner||'.'||i.table_name||'('||i.partition_name||')',p_testmode);
```

```
    IF NOT p_testmode THEN
      sys.dbms_stats.delete_table_stats
        (ownname=>i.owner
        ,tabname=>i.table_name
        ,partname=>i.partition_name
        ,cascade_parts=>FALSE --do not cascade to child partiton
        ,force=>p_force
        );
      l_counter := l_counter + 1;
    END IF;
  END LOOP;
  msg('Deleted Statistics on '||l_counter||' logical partitons',p_testmode);
  IF l_counter > 0 THEN
    p_stats_changed := TRUE;
  END IF;
  dbms_application_info.set_module(module_name=>l_module, action_name=>l_action);

END delete_global_stats;
--------------------------------------------------------------------------------
--Function to decode raw values in R1 and R2 statistics columns.  Not used elsehwere in this package
--stolen from http://structureddata.org/2007/10/16/how-to-display-high_valuelow_value-columns-from-user_tab_col_statistics/
--------------------------------------------------------------------------------
FUNCTION display_raw
(p_rawval RAW
,p_type   VARCHAR2
) RETURN VARCHAR2 IS
  cn  NUMBER;
  cv  VARCHAR2(32);
  cd  DATE;
  cnv NVARCHAR2(32);
  cr  ROWID;
  cc  CHAR(32);
BEGIN
  IF (p_type = 'NUMBER') THEN
    dbms_stats.convert_raw_value(p_rawval, cn);
    RETURN TO_CHAR(cn);
  ELSIF (p_type = 'VARCHAR2') THEN
    dbms_stats.convert_raw_value(p_rawval, cv);
    RETURN TO_CHAR(cv);
  ELSIF (p_type = 'DATE') THEN
    dbms_stats.convert_raw_value(p_rawval, cd);
    RETURN TO_CHAR(cd);
  ELSIF (p_type = 'NVARCHAR2') THEN
    dbms_stats.convert_raw_value(p_rawval, cnv);
    RETURN TO_CHAR(cnv);
  ELSIF (p_type = 'ROWID') THEN
    dbms_stats.convert_raw_value(p_rawval, cr);
    RETURN TO_CHAR(cnv);
  ELSIF (p_type = 'CHAR') THEN
    dbms_stats.convert_raw_value(p_rawval, cc);
    RETURN TO_CHAR(cc);
  ELSE
    RETURN 'UNKNOWN DATATYPE';
  END IF;
END display_raw;
```

```
--------------------------------------------------------------------------------
--This procedure corrects some column statistics that have been exported to the stat table
--------------------------------------------------------------------------------
PROCEDURE fix_stats
(p_ownname       VARCHAR2 DEFAULT 'SYSADM'
,p_tabname       VARCHAR2
,p_statid        VARCHAR2
,p_stats_changed IN OUT BOOLEAN
,p_testmode      BOOLEAN DEFAULT FALSE
) IS
  l_srec DBMS_STATS.STATREC;
  l_module VARCHAR2(48);
  l_action VARCHAR2(32);
BEGIN
  dbms_application_info.read_module(module_name=>l_module, action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module||'FIX_STATS', action_name=>p_tabname);
  msg('fix_stats(p_ownname=>'||p_ownname||', p_tabname=>'||p_tabname||', p_statid=>'||p_statid
           ||', p_stats_changed=>'||display_bool(p_stats_changed)||')',p_testmode);
  FOR i IN ( --this query corrects ndv and density statistics on partitions from subpartitions
with subpart as ( --aggregates stats on columns with histograms so only have one row per segment per column
          SELECT    statid, type, c5, c1, c4, c2, c3
          ,         MAX(n1) n1
          ,         COUNT(*) num_histograms
--        ,         MIN(r1) r1
--        ,         MAX(r2) r2
          FROM      sysadm.gfc_stat_table
          WHERE     c2 IS NOT NULL --partition
          AND       c3 IS NOT NULL --subpartition
          GROUP BY statid, type, c5, c1, c4, c2, c3
), agg as ( --for each column we calculate max and sum of NDV
          SELECT    statid, type, c5, c1, c4, c2
          ,         MAX(n1) max_n1
          ,         SUM(n1) sum_n1
          ,         COUNT(*) num_rows
--        ,         MIN(r1) r1
--        ,         MAX(r2) r2
          FROM      subpart
          GROUP BY statid, type, c5, c1, c4, c2
), part as ( --get current stats for partition
          SELECT    *
          FROM      sysadm.gfc_stat_table
          WHERE     c2 IS NOT NULL
          AND       c3 IS NULL
), cf as ( --compare aggregated stats with actual stats
SELECT    agg.statid, agg.type, agg.c5, agg.c4, agg.c1, agg.c2
,         part.n1
,         CASE WHEN k.column_position IS NOT NULL THEN sum_n1 --sum NDV across partitions on partitioning column
               ELSE max_n1 --otherwise take maximum NDV across columns
          END as new_n1
,         part.n2
,         part.n3
--,       n1*n2 x
--,       part.r1, part.r2
FROM      agg
          LEFT OUTER JOIN all_subpart_key_columns k --list of subpartition key columns
```

```
          ON k.owner = agg.c5
          AND k.object_type = 'TABLE'
          AND k.name = agg.c1
          AND k.column_name = agg.c4
,         part
WHERE     agg.statid = part.statid
AND     agg.type = part.type
and       agg.c5 = part.c5
and       agg.c1 = part.c1
and       agg.c4 = part.c4
and       agg.c2 = part.c2
ORDER BY agg.c5, agg.c1, agg.c2
)
SELECT    *
FROM      cf
WHERE     statid = p_statid
AND       c5 = p_ownname
AND       c1 = p_tabname
AND       new_n1 > n1 --only if calculated NDV is higher
AND     type = 'C'
ORDER BY c4, c1, c2
  ) LOOP

    --update stats via Oracle API
    msg(i.type||':partition:'||i.c5||'.'||i.c1||'.'||i.c2||':col '||i.c4||':NDV '||i.n1||'->'||i.new_n1,p_testmode);
    p_stats_changed := TRUE;

    /*not updating stats via Oracle API because it blanks values not set.  Easier to update table
    sys.dbms_stats.set_column_stats
    (ownname  => p_ownname
    ,tabname  => p_tabname
    ,colname  => i.c4
    ,partname => i.c2
    ,statown  => k_ownname
    ,stattab  => k_stattab
    ,statid   => p_statid
    ,distcnt  => i.new_n1
    ,density  => 1/NULLIF(i.new_n1,0) --density always reciprocal of NDV because never have stats
    );
    */

    IF NOT p_testmode THEN
      UPDATE sysadm.gfc_stat_table
      SET    d1 = SYSDATE --update last analysed time because set_column_stats does not
      ,      n1 = i.new_n1 --number of distinct values
      ,      n2 = 1/NULLIF(i.new_n1,0) --density
--    ,      r1 = i.r1  --set min column value
--    ,      r2 = i.r2  --set max column value
      WHERE  c5 = i.c5  --owner
      AND    c1 = i.c1  --table
      AND    c2 = i.c2  --partition
      AND    c3 IS NULL --subpartition
      AND    c4 = i.c4  --column
      AND    type = i.type
      AND    statid = p_statid
```

 ©GO-FASTER CONSULTANCY LTD.

```
      ;
    END IF;

  END LOOP;

  FOR i IN ( --this query corrects ndv and density statistics on tables from partitions
with part as ( --this query aggregates stats on columns with histograms so we only have one row per segment per column
          SELECT    statid, type, c5, c1, c4, c2
          ,         MAX(n1) n1
          ,         COUNT(*) num_histograms
--        ,         MIN(r1) r1
--        ,         MAX(r2) r2
          FROM      sysadm.gfc_stat_table
          WHERE     c2 IS NOT NULL --partition
          AND       c3 IS NULL --subpartition
          GROUP BY statid, type, c5, c1, c4, c2
), agg as ( --for each column we calculate max and sum of NDV
          SELECT    statid, type, c5, c1, c4
          ,         MAX(n1) max_n1
          ,         SUM(n1) sum_n1
          ,         COUNT(*) num_rows
--        ,         MIN(r1) r1
--        ,         MAX(r2) r2
          FROM      part
          GROUP BY statid, type, c5, c1, c4
) , tab as ( --get current stats for table
          SELECT    *
          FROM      sysadm.gfc_stat_table
          WHERE     c2 IS NULL
          AND       c3 IS NULL
) , cf as ( --compare aggregated stats with actual stats
SELECT    agg.statid, agg.type, agg.c5, agg.c4, agg.c1
,         tab.n1
,         CASE WHEN k.column_position IS NOT NULL THEN sum_n1 --sum NDV across partitions on partitioning column
               ELSE max_n1 --otherwise take maximum NDV across columns
          END as new_n1
,         tab.n2
,         tab.n3
--,       n1*n2 x
--,       tab.r1, tab.r2
FROM      agg
          LEFT OUTER JOIN all_part_key_columns k --list of partition key columns
          ON k.owner = agg.c5
          AND k.object_type = 'TABLE'
          AND k.name = agg.c1
          AND k.column_name = agg.c4
,         tab
WHERE     agg.c5 = tab.c5
and       agg.c1 = tab.c1
and       agg.c4 = tab.c4
AND     agg.type = tab.type
AND       agg.statid = tab.statid
ORDER BY agg.c5, agg.c1
)
SELECT    *
```

```
FROM      cf
WHERE     statid = p_statid
AND       c5 = p_ownname
AND       c1 = p_tabname
AND       new_n1 > n1 --only if calculated NDV is higher
AND     type = 'C'
  ) LOOP

    msg(i.type||':table '||i.c5||'.'||i.c1||':col '||i.c4||':NDV '||i.n1||'->'||i.new_n1,p_testmode);
    p_stats_changed := TRUE;

    /*not updating stats via Oracle API because it blanks values not set.  Easier to update table
    sys.dbms_stats.set_column_stats
    (ownname  => p_ownname
    ,tabname  => p_tabname
    ,colname  => i.c4
    ,partname => NULL
    ,statown  => k_ownname
    ,stattab  => k_stattab
    ,statid   => p_statid
    ,distcnt  => i.new_n1
    ,density  => 1/NULLIF(i.new_n1,0) --density always reciprocal of NDV because never have stats
    );*/

    IF NOT p_testmode THEN
      UPDATE sysadm.gfc_stat_table
      SET    d1 = SYSDATE --update last analysed time because set_column_stats does not
      ,      n1 = i.new_n1 --number of distinct values
      ,      n2 = 1/NULLIF(i.new_n1,0) --density
--    ,      r1 = i.r1  --set min column value
--    ,      r2 = i.r2  --set max column value
      WHERE  c5 = i.c5  --owner
      AND    c1 = i.c1  --table
      AND    c2 IS NULL --partition
      AND    c3 IS NULL --subpartition
      AND    c4 = i.c4  --column
      AND    type = i.type
      AND    statid = p_statid
      ;
    END IF;

  END LOOP;
  dbms_application_info.set_module(module_name=>l_module, action_name=>l_action);

END fix_stats;
--------------------------------------------------------------------------------
--This procedure waits for database jobs submitted to collect statistics.
--The time it waits is determined by the estimate to complete, but at least every minute
--------------------------------------------------------------------------------
PROCEDURE wait_for_jobs
(p_ownname       VARCHAR2 DEFAULT 'SYSADM'
,p_tabname       VARCHAR2
,p_job_submitted INTEGER
) IS
  l_rindex     BINARY_INTEGER; --longops index
```

```
  l_slno       BINARY_INTEGER;
  l_obj        BINARY_INTEGER;

  l_sleep      INTEGER; --seconds to sleep in wait loop
  l_startdttm  DATE; --timestamp when start to submit jobs

  l_num_jsched NUMBER; --number of scheduled jobs
  l_num_jrun   NUMBER; --number of running jobs
  l_num_jobs   NUMBER; --total number of jobs
BEGIN
  l_rindex := dbms_application_info.set_session_longops_nohint;
  l_startdttm := SYSDATE;

  WHILE TRUE LOOP --wait here if stats jobs scheduled
    with X as (
             select t.owner, t.table_name, p.subpartition_name segment_name
             from   all_tab_subpartitions p
             ,      all_part_tables t
             where  t.owner = p.table_owner
             and    t.table_name = p.table_name
             and    t.subpartitioning_type != 'NONE'
             union all
             select t.owner, t.table_name, p.partition_name segment_name
             from   all_tab_partitions p
             ,      all_part_tables t
             where  t.owner = p.table_owner
             and    t.table_name = p.table_name
             and    t.partitioning_type != 'NONE'
             and    t.subpartitioning_type = 'NONE'
             )
    select COUNT(*) jobs
    ,      NVL(SUM(CASE WHEN j.state = 'RUNNING'   THEN 1 END),0) running
    ,      NVL(SUM(CASE WHEN j.state = 'SCHEDULED' THEN 1 END),0) scheduled
    INTO   l_num_jobs, l_num_jrun, l_num_jsched
    from   x
    ,      ALL_SCHEDULER_JOBS j
    where  x.owner = j.owner
    and    x.segment_name = j.job_name
    and    x.table_name = p_tabname
    and    x.owner = p_ownname
    ;

    msg(''||l_num_jobs||' jobs outstanding ('||l_num_jrun||' running, '||l_num_jsched||' scheduled)');
    dbms_application_info.set_session_longops(l_rindex, l_slno,
          p_ownname||'.'||p_tabname, l_obj, 0, p_job_submitted-l_num_jobs, p_job_submitted, 'partition', 'partitions');
    IF l_num_jobs = 0 THEN --break out of loop if no jobs left
      EXIT;
    ELSE --30.10.2012 sleep duration calculation simplified
      l_sleep := (SYSDATE-l_startdttm)*86400*l_num_jobs/NULLIF(p_job_submitted-l_num_jobs,0)/2; --sleep for half estimated
time to complete
      IF l_sleep > 60 THEN --do not sleep for more than 1 minute
        l_sleep := 60;
      ELSIF l_sleep IS NULL OR l_sleep < 3 THEN --do not sleep for less than 3 seconds
        l_sleep := 3;
      END IF;
```

```
      sys.dbms_lock.sleep(l_sleep);
    END IF;
  END LOOP;
END wait_for_jobs;
--------------------------------------------------------------------------------
--This procedure gathers statistics on physical partitions or sub-partitions whose statistics
--are either missing  or stale, or on the last partition or sub-partition in a table or partition
--that does not have aggregated statistics
--------------------------------------------------------------------------------
PROCEDURE gather_table_stats
(p_ownname       VARCHAR2 DEFAULT 'SYSADM'
,p_tabname       VARCHAR2
,p_method_opt    VARCHAR2 /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
,p_allparts      BOOLEAN DEFAULT FALSE
,p_force         BOOLEAN DEFAULT FALSE
,p_sched_job     BOOLEAN DEFAULT FALSE
,p_stats_changed IN OUT BOOLEAN
,p_testmode      BOOLEAN DEFAULT FALSE
) IS
  l_counter    INTEGER := 0; --number of partitions stats updated
  l_module     VARCHAR2(48);
  l_action     VARCHAR2(32);
  l_allparts   VARCHAR2(1) := 'N'; --if y then updated all partitions even if not stale states
  l_message    VARCHAR2(100); --string to hold part of message
  l_method_opt VARCHAR2(100) := '' ;--string to hold method opt clause
BEGIN
  dbms_application_info.read_module(module_name=>l_module, action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module||'GATHER_TABLE_STATS', action_name=>p_tabname);
  msg('gather_table_stats(p_ownname=>'||p_ownname||', p_tabname=>'||p_tabname||', p_method_opt=>'||p_method_opt
     ||', p_allparts=>'||display_bool(p_allparts)||', p_force=>'||display_bool(p_force)||',
p_stats_changed=>'||display_bool(p_stats_changed)||')',p_testmode);

  IF p_allparts THEN
    l_allparts := 'Y';
  END IF;

  IF p_method_opt IS NOT NULL THEN
    l_method_opt := ', method_opt=>'''||p_method_opt||'''';
  END IF;

  FOR i IN ( --partitioned table stale/missing stats
          SELECT  p.owner, p.table_name, p.partition_name
          ,         CASE WHEN p.num_rows IS NULL THEN ' missing'
                         WHEN p.stale_stats = 'YES' THEN ' stale'
                         ELSE ''
                    END as reason
          from      all_tab_statistics p
          ,         all_part_tables t
          WHERE     p.owner = p_ownname
          AND       p.table_name = p_tabname
          AND       (   p.num_rows IS NULL
                 or l_allparts = 'Y'
                 or p.stale_stats = 'YES')
      and           t.owner = p.owner
          AND       t.table_name = p.table_name
```


©GO-FASTER CONSULTANCY LTD.

```
         AND       t.subpartitioning_type = 'NONE'
      AND   p.object_type = 'PARTITION' --added 30.10.2012
  ) LOOP
    l_message := i.reason||' statistics on '||i.owner||'.'||i.table_name||' physical partition ('||i.partition_name||')';
    IF NOT p_testmode THEN
      IF p_sched_job THEN
        msg('Gathering'||l_message,p_testmode);
        sys.dbms_scheduler.create_job
        (job_name   => i.partition_name
        ,job_type   => 'PLSQL_BLOCK'
        ,job_action => 'BEGIN sys.dbms_stats.gather_table_stats(ownname=>'''||i.owner
                              ||''', tabname=>'''||i.table_name
                              ||''', partname=>'''||i.partition_name
                              ||''', cascade=>TRUE'
                              ||l_method_opt
                              ||', force=>'||display_bool(p_force)
                              ||  ', granularity=>''PARTITION''); END;'
        ,start_date => SYSTIMESTAMP --run job immediately
        ,enabled    => TRUE --job is enabled
        ,auto_drop  => TRUE --request will be dropped when complete
        ,comments   => 'Gather stats on table partition '||i.owner||'.'||i.table_name||'.'||i.partition_name
        );
      ELSE
        msg('Submitting job to gather'||l_message,p_testmode);
        sys.dbms_stats.gather_table_stats
          (ownname     => i.owner
          ,tabname     => i.table_name
          ,partname    => i.partition_name
          ,cascade     => TRUE
          ,granularity => 'PARTITION'
          ,method_opt  => p_method_opt
          ,force       => p_force
          );
      END IF;
    END IF;
    l_counter := l_counter + 1;
 END LOOP;

  FOR i IN ( --subpartitioned table stale/missing stats
          SELECT    s.owner, s.table_name, s.subpartition_name
          ,         CASE WHEN s.num_rows IS NULL THEN ' missing'
                         WHEN s.stale_stats = 'YES' THEN ' stale'
                         ELSE ''
                    END as reason
          from      all_tab_statistics s
          WHERE     s.owner = p_ownname
          AND       s.table_name = p_tabname
          AND       s.object_type = 'SUBPARTITION'
          AND       (   s.num_rows IS NULL
                 or l_allparts = 'Y'
                 or s.stale_stats = 'YES')
  ) LOOP
    l_message := i.reason||' statistics on '||i.owner||'.'||i.table_name||' physical subpartition
('||i.subpartition_name||')';
    IF NOT p_testmode THEN
```

```
    IF p_sched_job THEN
      msg('Submitting job to gather'||l_message,p_testmode);
      sys.dbms_scheduler.create_job
      (job_name   => i.subpartition_name
      ,job_type   => 'PLSQL_BLOCK'
      ,job_action => 'BEGIN sys.dbms_stats.gather_table_stats(ownname=>'''||i.owner
                          ||''', tabname=>'''||i.table_name
                          ||''', partname=>'''||i.subpartition_name
                          ||''', cascade=>TRUE'
                          ||l_method_opt
                          ||', force=>'||display_bool(p_force)
                          ||  ', granularity=>''SUBPARTITION''); END;'
      ,start_date => SYSTIMESTAMP --run job immediately
      ,enabled    => TRUE --job is enabled
      ,auto_drop  => TRUE --request will be dropped when complete
      ,comments   => 'Gather stats on table sub_partition '||i.owner||'.'||i.table_name||'.'||i.subpartition_name
      );
    ELSE
      msg('Gathering'||l_message,p_testmode);
      sys.dbms_stats.gather_table_stats
        (ownname     => i.owner
        ,tabname     => i.table_name
        ,partname    => i.subpartition_name
        ,cascade     => TRUE
        ,granularity => 'SUBPARTITION'
        ,method_opt  => p_method_opt
        ,force       => p_force
        );
    END IF;
  END IF;
  l_counter := l_counter + 1;
END LOOP;

IF l_counter > 0 THEN
  wait_for_jobs(p_ownname, p_tabname, l_counter); --wait for database jobs
  msg('Gathered Statistics on '||l_counter||' physical partitons');
  p_stats_changed := TRUE;
  l_counter := 0;
END IF;

FOR i IN (  --last subpartition in partition without stats
        SELECT s.* from (
        SELECT    x.*
        ,         row_number() OVER (
                          PARTITION BY table_name, partition_name
                          ORDER BY stale_stats DESC, subpartition_position DESC) as ranking
        from      all_tab_statistics x
        where     object_type = 'SUBPARTITION'
        ) s
        ,         all_tab_statistics p
        WHERE     s.owner = p_ownname
        AND       s.table_name = p_tabname
        and       p.owner = s.owner
        AND       p.table_name = s.table_name
        and       p.partition_name = s.partition_name
```


©GO-FASTER CONSULTANCY LTD.

```
        and       p.num_rows IS NULL
        and       p.object_type = 'PARTITION'
        and       s.ranking = 1
) LOOP
  msg('Gathering statistics on '||i.owner||'.'||i.table_name
    ||' physical subpartition ('||i.subpartition_name||') to generate incremental statistics',p_testmode);
  IF NOT p_testmode THEN
    IF p_sched_job THEN
      msg('Submitting job to gather'||l_message,p_testmode);
      sys.dbms_scheduler.create_job
      (job_name   => i.subpartition_name
      ,job_type   => 'PLSQL_BLOCK'
      ,job_action => 'BEGIN sys.dbms_stats.gather_table_stats(ownname=>'''||i.owner
                     ||''', tabname=>'''||i.table_name
                     ||''', partname=>'''||i.subpartition_name
                     ||''', cascade=>TRUE'
                     ||l_method_opt
                     ||', force=>'||display_bool(p_force)
                     ||  ', granularity=>''SUBPARTITION''); END;'
      ,start_date => SYSTIMESTAMP --run job immediately
      ,enabled    => TRUE --job is enabled
      ,auto_drop  => TRUE --request will be dropped when complete
      ,comments   => 'Gather stats on table sub_partition '||i.owner||'.'||i.table_name||'.'||i.subpartition_name
      );
    ELSE
      msg('Gathering'||l_message,p_testmode);
      sys.dbms_stats.gather_table_stats
        (ownname     => i.owner
        ,tabname     => i.table_name
        ,partname    => i.subpartition_name
        ,cascade     => TRUE
        ,granularity => 'SUBPARTITION'
        ,method_opt  => p_method_opt
        ,force       => p_force
        );
    END IF;
  END IF;
  l_counter := l_counter + 1;
END LOOP;

FOR i IN ( --last partition in table without stats
        SELECT s.* from (
        SELECT    x.*
        ,         row_number() OVER (
                          PARTITION BY table_name
                          ORDER BY stale_stats DESC, partition_position DESC) as ranking
        from      all_tab_statistics x
        where     object_type = 'PARTITION'
        ) s
        ,         all_tab_statistics p
        WHERE     s.owner = p_ownname
        AND       s.table_name = p_tabname
        and       p.owner = s.owner
        AND       p.table_name = s.table_name
        and       p.partition_name = s.partition_name
```

```
         and       p.num_rows IS NULL
         and       p.object_type = 'TABLE'
         and       s.ranking = 1
  ) LOOP
    msg('Gathering statistics on '||i.owner||'.'||i.table_name
      ||' physical partition ('||i.partition_name||') to generate incremental statistics',p_testmode);
    IF NOT p_testmode THEN
      IF p_sched_job THEN
        msg('Gathering'||l_message,p_testmode);
        sys.dbms_scheduler.create_job
        (job_name   => i.partition_name
        ,job_type   => 'PLSQL_BLOCK'
        ,job_action => 'BEGIN sys.dbms_stats.gather_table_stats(ownname=>'''||i.owner
                       ||''', tabname=>'''||i.table_name
                       ||''', partname=>'''||i.partition_name
                       ||''', cascade=>TRUE'
                       ||l_method_opt
                       ||', force=>'||display_bool(p_force)
                       ||  ', granularity=>''PARTITION''); END;'
        ,start_date => SYSTIMESTAMP --run job immediately
        ,enabled    => TRUE --job is enabled
        ,auto_drop  => TRUE --request will be dropped when complete
        ,comments   => 'Gather stats on table partition '||i.owner||'.'||i.table_name||'.'||i.partition_name
        );
      ELSE
        msg('Submitting job to gather'||l_message,p_testmode);
        sys.dbms_stats.gather_table_stats
          (ownname     => i.owner
          ,tabname     => i.table_name
          ,partname    => i.partition_name
          ,cascade     => TRUE
          ,granularity => 'PARTITION'
          ,method_opt  => p_method_opt
          ,force       => p_force
          );
      END IF;
    END IF;
    l_counter := l_counter + 1;
  END LOOP;

  IF l_counter > 0 THEN
    wait_for_jobs(p_ownname, p_tabname, l_counter); --wait for database jobs
    msg('Gathered Statistics on '||l_counter||' physical partitons');
    p_stats_changed := TRUE;
    l_counter:= 0;
  END IF;

  dbms_application_info.set_module(module_name=>l_module, action_name=>l_action);

END gather_table_stats;
--------------------------------------------------------------------------------
--This procedure provides performs all the steps involved in gathering, exporting,
--correcting and reimporting statistics on a partitioned table
--------------------------------------------------------------------------------
PROCEDURE gather_fix_stats
```

```
(p_ownname    VARCHAR2 DEFAULT 'SYSADM'
,p_tabname    VARCHAR2
,p_method_opt VARCHAR2 DEFAULT NULL /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
,p_allparts   BOOLEAN DEFAULT FALSE
,p_statid     VARCHAR2
,p_force      BOOLEAN DEFAULT FALSE
,p_sched_job  BOOLEAN DEFAULT FALSE
,p_testmode   BOOLEAN DEFAULT FALSE
) IS
  l_module VARCHAR2(48);
  l_action VARCHAR2(32);
  l_stats_changed BOOLEAN := FALSE;
BEGIN
  dbms_application_info.read_module(module_name=>l_module, action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module||'GATHER_FIX_STATS', action_name=>p_tabname);
  msg('gather_fix_stats(p_ownname=>'||p_ownname||', p_tabname=>'||p_tabname||', p_method_opt=>'||p_method_opt
      ||', p_statid=>'||p_statid||', p_allparts=>'||display_bool(p_allparts)||',
p_force=>'||display_bool(p_force)||')',p_testmode);

  agg_stats.delete_global_stats --delete global stats
  (p_ownname=>p_ownname
  ,p_tabname=>p_tabname
  ,p_force=>p_force
  ,p_stats_changed=>l_stats_changed
  ,p_testmode=>p_testmode
  );

  agg_stats.gather_table_stats --gather missing stale stats
  (p_ownname=>p_ownname
  ,p_tabname=>p_tabname
  ,p_method_opt=>p_method_opt
  ,p_allparts=>p_allparts
  ,p_force=>p_force
  ,p_sched_job=>p_sched_job
  ,p_stats_changed=>l_stats_changed
  ,p_testmode=>p_testmode
  );

  IF l_stats_changed THEN
    dbms_application_info.set_action(action_name=>SUBSTR('Delete export '||p_tabname,1,32));
    msg('Deleting previously exported statistics for table '||p_ownname||'.'||p_tabname,p_testmode);
    exec_sql('DELETE FROM sysadm.gfc_stat_table WHERE statid = '''||p_statid||''' AND c1 = '''
             ||p_tabname||''' AND c5 = '''||p_ownname||'''',p_testmode); --delete old export
    msg(''||SQL%ROWCOUNT||' rows deleted.');

    msg('Exporting statistics for table '||p_ownname||'.'||p_tabname,p_testmode);
    IF NOT p_testmode THEN
      dbms_application_info.set_action(action_name=>SUBSTR('Export '||p_tabname,1,32));
      sys.dbms_stats.export_table_stats --export stats
      (ownname=>p_ownname
      ,tabname=>p_tabname
      ,statown=>k_ownname
      ,stattab=>k_stattab
      ,statid=>p_statid
      ,cascade=>TRUE
```

```
      );
    END IF;
    dbms_application_info.set_action(action_name=>SUBSTR(p_tabname,1,32));
  END IF;

  agg_stats.fix_stats --fix exported stats
  (p_ownname=>p_ownname
  ,p_tabname=>p_tabname
  ,p_statid=>p_statid
  ,p_stats_changed=>l_stats_changed
  ,p_testmode=>p_testmode
  );

  IF l_stats_changed THEN

    msg('Importing corrected statistics for table '||p_ownname||'.'||p_tabname,p_testmode);
    IF NOT p_testmode THEN
      dbms_application_info.set_action(action_name=>SUBSTR('Import '||p_tabname,1,32));
      sys.dbms_stats.import_table_stats --reimport fixed statistics
      (ownname=>p_ownname
      ,tabname=>p_tabname
      ,statown=>k_ownname
      ,stattab=>k_stattab
      ,statid=>p_statid
      ,cascade=>TRUE
      ,force=>p_force
      );
      dbms_application_info.set_action(action_name=>SUBSTR(p_tabname,1,32));
    END IF;
  ELSE
    msg('No statistics have changed for table '||p_ownname||'.'||p_tabname,p_testmode);
  END IF;

  commit;
  msg('Finished processing statistics for table '||p_ownname||'.'||p_tabname,p_testmode);
  dbms_application_info.set_module(module_name=>l_module, action_name=>l_action);

END gather_fix_stats;
--------------------------------------------------------------------------------
--This procedure calls gather_fix_stats for all partitioned tables whose stats are locked.
--------------------------------------------------------------------------------
PROCEDURE gather_all_locked_part_tables
(p_ownname    VARCHAR2 DEFAULT 'SYSADM'
,p_method_opt VARCHAR2 DEFAULT NULL /*remove default in 11g--DEFAULT sys.dbms_stats.get_param('METHOD_OPT')*/
,p_allparts   BOOLEAN  DEFAULT FALSE
,p_statid     VARCHAR2
,p_testmode   BOOLEAN  DEFAULT FALSE
) IS
  l_module VARCHAR2(48);
  l_action VARCHAR2(32);
BEGIN
  dbms_application_info.read_module(module_name=>l_module, action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module||'GATHER_ALL_LOCKED_PART_STATS', action_name=>p_ownname);
  msg('gather_all_locked_part_tables(p_ownname=>'||p_ownname||', p_method_opt=>'||p_method_opt||', p_allparts=>'
     ||display_bool(p_allparts)||', p_statid=>'||p_statid||')',p_testmode);
```


©GO-FASTER CONSULTANCY LTD.

```
  FOR i IN(
    SELECT t.owner
    ,        t.table_name
    from   all_tab_statistics s
    ,      all_tables t
    where  t.owner = s.owner
    and    t.table_name = s.table_name
    and    s.object_type = 'TABLE' --added 30.10.2012
    and    s.partition_name is null
    and    s.stattype_locked != 'NO'
    and    t.partitioned = 'YES'
    and    (t.owner = p_ownname OR p_ownname IS NULL)
  ) LOOP
    gather_fix_stats
    (p_ownname=>i.owner
    ,p_tabname=>i.table_name
    ,p_method_opt=>p_method_opt
    ,p_allparts=>p_allparts
    ,p_statid=>p_statid
    ,p_force=>TRUE
    ,p_testmode=>p_testmode);
  END LOOP;
  dbms_application_info.set_module(module_name=>l_module, action_name=>l_action);
END gather_all_locked_part_tables;
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
END agg_stats;
/
show errors
--------------------------------------------------------------------------------
spool off
```