

## Tipps & Tricks: April 2017

Bereich:	DBA	Erstellung:	04/2017 CK
Versionsinfo:	11g, 12c	Letzte Überarbeitung:	04/2017 CK

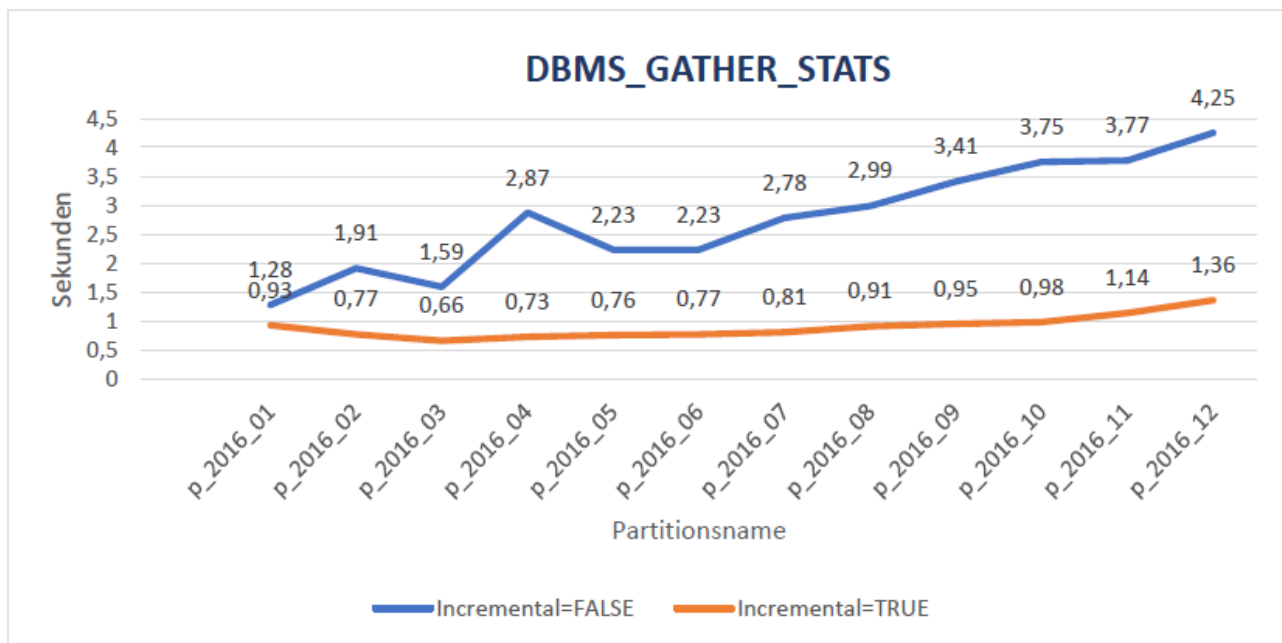
### Oracle Statistikverwaltung für partitionierte Tabellen

Sie laden regelmäßig Daten in eine oder mehrere Partitionen?

Die Erstellung bzw. Verwaltung der **globalen** Statistiken führt bei grossen, partitionierten Tabellen zu einem Ressourcen- bzw. Zeitproblem?

Abhilfe kann hier das Verwenden von **inkrementeller** Statistikerstellung sein.

Hier sehen Sie die Laufzeiten der Statistiksammlung **ohne** inkrementelle und **mit** inkrementeller Statistikerstellung der Tabelle PART\_RANGE im Schema SCOTT.



Hinweis:

Sie benötigen für die Verwendung von partitionierten Tabellen die Oracle Enterprise Edition mit der Option Partitioning.

Vorbereitung:

Zur Demonstration und zum besseren Nachvollziehen legen wir im Schema SCOTT eine range partitionierte Tabelle PART\_RANGE an:

```
CREATE TABLE SCOTT.PART_RANGE
(
  DATUM DATE,
```

```

TEXT    VARCHAR2(400 BYTE)
)
LOGGING
PARTITION BY RANGE (DATUM)
(
  PARTITION P_2016_01 VALUES LESS THAN (TO_DATE(' 2016-02-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_02 VALUES LESS THAN (TO_DATE(' 2016-03-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_03 VALUES LESS THAN (TO_DATE(' 2016-04-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_04 VALUES LESS THAN (TO_DATE(' 2016-05-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_05 VALUES LESS THAN (TO_DATE(' 2016-06-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_06 VALUES LESS THAN (TO_DATE(' 2016-07-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_07 VALUES LESS THAN (TO_DATE(' 2016-08-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_08 VALUES LESS THAN (TO_DATE(' 2016-09-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_09 VALUES LESS THAN (TO_DATE(' 2016-10-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_10 VALUES LESS THAN (TO_DATE(' 2016-11-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_11 VALUES LESS THAN (TO_DATE(' 2016-12-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
  ,
  PARTITION P_2016_12 VALUES LESS THAN (TO_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD
HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
);

```

Die Tabelle PART\_RANGE wird mit 1 Mio. Zeilen befüllt (monatsweise v\_datum '01.01.2016', usw.):

```

DECLARE
v_value    VARCHAR2(400);
v_datum    DATE;

BEGIN

  FOR i IN 1..1000000 LOOP
    v_value := DBMS_RANDOM.STRING('A', 400);

```

```

        v_datum := TO_DATE('01.01.2016', 'dd.mm.yyyy') + DBMS_RANDOM.VALUE(0,30);

        INSERT /*+ APPEND */ into SCOTT.PART_RANGE (text, datum) VALUES (v_value,
v_datum);
        END LOOP;
        commit;

END;
/

```

Nach jedem Lauf wird die Statistik für die Tabelle PART\_RANGE erstellt:

```

set serveroutput on
set timing on
BEGIN
  DBMS_STATS.GATHER_TABLE_STATS (
    OwnName => 'SCOTT'
    ,TabName => 'PART_RANGE'
    ,Estimate_Percent => SYS.DBMS_STATS.AUTO_SAMPLE_SIZE
    ,Method_Opt => 'FOR ALL INDEXED COLUMNS SIZE AUTO'
    ,Degree => SYS.DBMS_STATS.AUTO_DEGREE
    ,granularity => 'AUTO'
    ,Cascade => TRUE
    ,No_Invalidate => DBMS_STATS.AUTO_INVALIDATE
  );
END;
/

```

Hinweis:

Die einzelnen **Laufzeiten** von GATHER\_TABLE\_STATS sind die **Grundlage** der oben dargestellten Grafik.

Nach dem letzten Ladelauf - hier Dezember 2016 - und der Erstellung der Statistik für die Tabelle PART\_RANGE sehen wir, dass der Zeitstempel **LAST\_ANALYZED** über alle Partitionen fast gleich ist:

```

SET LINESIZE 300
col table_name format A15
col partition_name format A20
col last_analyzed format A20
SELECT table_name,
       partition_name,
       global_stats,
       TO_CHAR (last_analyzed, 'DD.MM.YYYY HH24:MI:SS') last_analyzed,
       num_rows
  FROM dba_tab_partitions
 WHERE table_name = 'PART_RANGE' AND table_owner = 'SCOTT'
 ORDER BY 2
;

```

TABLE_NAME	PARTITION_NAME	GLOBAL_STATS	LAST_ANALYZED	NUM_ROWS
PART_RANGE	P_2016_01	YES	15.03.2017 10:59:46	1000000
PART_RANGE	P_2016_02	YES	15.03.2017 10:59:46	1000000

```

PART_RANGE      P_2016_03          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_04          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_05          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_06          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_07          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_08          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_09          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_10          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_11          YES          15.03.2017 10:59:47    1000000
PART_RANGE      P_2016_12          YES          15.03.2017 10:59:47    1000000

```

12 rows selected.

Die Lösung der beschriebenen Problemen ist:

Das Einschalten der **inkrementellen** Statistiksammlung, denn per default ist diese Funktion **nicht aktiviert**.

```

SQL> select DBMS_STATS.GET_PREFS('INCREMENTAL','SCOTT','PART_RANGE') INCREMENTAL
from dual;

INCREMENTAL
-----
FALSE

SQL> exec DBMS_STATS.set_table_prefs('SCOTT','PART_RANGE','INCREMENTAL','TRUE');

SQL> select DBMS_STATS.GET_PREFS('INCREMENTAL','SCOTT','PART_RANGE') INCREMENTAL
from dual;

INCREMENTAL
-----
TRUE

```

Wir löschen den Inhalt der Tabelle PART\_RANGE und die Statistiken:

```

SQL> TRUNCATE TABLE SCOTT.PART_RANGE;

SQL> exec DBMS_STATS.DELETE_TABLE_STATS('SCOTT','PART_RANGE');

```

Wir führen die Befüllung erneut durch; wie im vorherigen Test werden nach jedem Lauf die Statistiken gesammelt:

```

SET LINESIZE 300
col table_name format A15
col partition_name format A20
col last_analyzed format A20
SELECT table_name,
       partition_name,
       global_stats,

```

```

        TO_CHAR (last_analyzed, 'DD.MM.YYYY HH24:MI:SS') last_analyzed,
        num_rows
    FROM dba_tab_partitions
    WHERE table_name = 'PART_RANGE' AND table_owner = 'SCOTT'
ORDER BY 2
;

```

TABLE_NAME	PARTITION_NAME	GLOBAL_STATS	LAST_ANALYZED	NUM_ROWS
PART_RANGE	P_2016_01	YES	15.03.2017 11:58:57	1000000
PART_RANGE	P_2016_02	YES	15.03.2017 12:01:30	1000000
PART_RANGE	P_2016_03	YES	15.03.2017 12:04:03	1000000
PART_RANGE	P_2016_04	YES	15.03.2017 12:06:41	1000000
PART_RANGE	P_2016_05	YES	15.03.2017 12:09:19	1000000
PART_RANGE	P_2016_06	YES	15.03.2017 12:11:51	1000000
PART_RANGE	P_2016_07	YES	15.03.2017 12:14:28	1000000
PART_RANGE	P_2016_08	YES	15.03.2017 12:17:09	1000000
PART_RANGE	P_2016_09	YES	15.03.2017 12:19:42	1000000
PART_RANGE	P_2016_10	YES	15.03.2017 12:22:19	1000000
PART_RANGE	P_2016_11	YES	15.03.2017 12:24:53	1000000
PART_RANGE	P_2016_12	YES	15.03.2017 12:27:28	1000000

12 rows selected.

Wie wir sehen, sind die Zeitstempel von **LAST\_ANALYZED** nun unterschiedlich.

**Wie funktioniert die Statistiksammlung bei partitionierten Tabellen wenn incremental=FALSE gestellt ist?**

- Im ersten Scan der Tabelle erfolgt die globale Statistiksammlung
- Im zweiten Scan erfolgt die Statistiksammlung für alle Partitionen einzeln

**Wie funktioniert die Statistiksammlung bei partitionierten Tabellen wenn incremental=TRUE gestellt ist?**

Der Optimizer leitet die Informationen von den "Partition-Level-Statistiken" ab und verwendet die Informationen der "number of distinct values" NDV'S und muss daher keinen Full Table Scan mehr durchführen.

- Der Optimizer sammelt die Statistiken der neu geladenen Partition(en) und erstellt eine intern "**Übersicht**"(synopses).
- Der Optimizer **mischt** (mergt) alle Partitions-Übersichten in die **globale Übersicht**.

Fazit:

Bei großen partitionierten Tabellen (mehrere hundert Gigabyte) ist die inkrementelle Statistiksammlungen sehr effizient. Die inkrementelle Statistiksammlung kann auf **Tabellen-, Schema- und Datenbankebene** aktiviert werden.

Für weitere Unterstützung steht Ihnen unser [Consulting-Team](#) gerne zur Verfügung.