Neat Things to Help Manage Partitioned Objects

NYOUG June 8, 2004

Michael S. Abbey

abbey@pythian.com
Agenda

• General issues
• Monitoring partitioned indexes
• DBMS_STATS is a must
• Partitioning using materialized views
• List partition maintenance—the unknown frontier
Ripple affect

• Modifications required to properly detect problems
• Partition maintenance can invalidate index partitions/sub-partitions
• Object re-compilation required as part of maintenance
  ➢ impact scheduling
  ➢ communication required
Self-inflicted
Invalidating indexes

- Aborted direct path Loader sessions
- Direct path insert … /*+ append */
- Partition splitting when rows are in container being split
- Partition dropping when dropped partition contains rows
- Moving table partitions
Monitoring Partitioned Indexes

- PARTITIONED = 'YES' in `dba_indexes`
  - STATUS = 'N/A'
- PARTITIONED = 'NO' in `dba_indexes`
  - STATUS = 'VALID'
- COMPOSITE = 'YES' in `dba_ind_partitions`
  - STATUS = 'N/A'
- COMPOSITE = 'NO' in `dba_ind_partitions`
  - STATUS = 'USABLE'
Analyze
DBMS_STATS Highlights

- granularity—determines the type of statistics to gather for partitioned objects
- estimate_percent—user defined number or dynamic table-specific choice made by Oracle
- degree—parallelism
- dba_tab_modifications—held in memory and only forced to disk when DBMS_STATS runs
### DBMS_STATS

#### GRANULARITY

<table>
<thead>
<tr>
<th>DEFAULT</th>
<th>Global and partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBPARTITION</td>
<td>Only subpartition</td>
</tr>
<tr>
<td>PARTITION</td>
<td>Only partition</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>Global</td>
</tr>
<tr>
<td>ALL</td>
<td>Global, partition, and subpartition</td>
</tr>
</tbody>
</table>
### DBMS_STATS

#### ESTIMATE_PERCENT

<table>
<thead>
<tr>
<th>Hard-coded</th>
<th>Number from 1 to 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO_SAMPLE_SIZE</td>
<td>Decision made based on condition of data</td>
</tr>
</tbody>
</table>

estimate_percent=>dbms_stats.auto_sample_size
## DBMS_STATS

### DEGREE

<table>
<thead>
<tr>
<th>Hard-coded</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Uses table default</td>
</tr>
<tr>
<td>DEFAULT_DEGREE</td>
<td>Influenced by initialization values</td>
</tr>
</tbody>
</table>

\[\text{degree} \rightarrow \text{dbms_stats.default_degree}\]
Global / Partition-level

Merged for query execution
Global / Partition-level
Analyze DBMS STATS
Switch to partitioned

- No limit on size of table
  - number of rows
  - Mb, Gb, Tb
- Fat tables with fewer rows
- Real time with close to zero down time
- Application quiesced to accomplish switchover
- Keep non-partitioned as long as possible

Space permitting
Switch to partitioned

• Make partitioned replica of non-partitioned
• Place partitioned table in nologging mode
• Place a snapshot log on non-partitioned table
• Lay snapshot on top of prebuilt partitioned table
• Complete refresh on snapshot
Switch to partitioned

• Build primary key constraint on partitioned table
  • preferrably local index
  • strict implementation restrictions
• Create a refresh mechanism via the job queue
• Negotiate switchover time
The switchover

- Disable application access
- Monitor row count in snapshot log
- When zero, do the following
  - drop refresh job
  - drop snapshot
  - no rows export on non-partitioned table
  - name switching
  - build secondary indexes
  - import with ignore=y
The switchover—caveats

• Dropping the refresh job
  - check `dba_jobs_running` beforehand
  - `/*+ rule */` hint with `9i` and `10g`

• Building secondary indexes
  - names may clash with non-partitioned index names
  - perform in parallel
  - may require partition renames afterwards
The unknown frontier

```sql
SQL> create table voter (  
    2    name varchar2(30),  
    3    province varchar2(2))  
    4    partition by list (province)  
    5    (partition east values ('NF','PE','NS','NB'),  
    6    partition quebec_ontario values ('QC','ON'),  
    7    partition west values ('MB','SK','AB','BC'),  
    8    partition rest values (DEFAULT));
```

Table created.
List partitioning

- Values stored in HIGH_VALUE in USER_TAB_PARTITIONS
- REST partition is the equivalent of MAXVALUE'd partition in range-based partitioning
- REST value is pseudo-boolean DEFAULT
List partition splitting

SQL> alter table voter split partition quebec_ontario
    2    values ('QC')
    3    into
    4    (partition quebec, partition ontario);

Table altered.

• 'QC' values to store in new partition
• partition quebec is the new partition created by the split
Neat Things to Help Manage Partitioned Objects

abby@pythian.com