Neat Things to Help Manage Partitioned Objects

NYOUG June 8, 2004

Michael S. Abbey

The PYTHIAN GROUP™

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

Enterprise -- online

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

DEFAULT	Global and partition
SUBPARTITION	Only subpartition
PARTITION	Only partition
GLOBAL	Global
ALL	Global, partition, and subpartition

DBMS_STATS

ESTIMATE_PERCENT

Hard-coded	Number from 1 to 100
AUTO_SAMPLE_SIZE	Decision made based on condition of data

estimate_percent=>dbms_stats.auto_sample_size

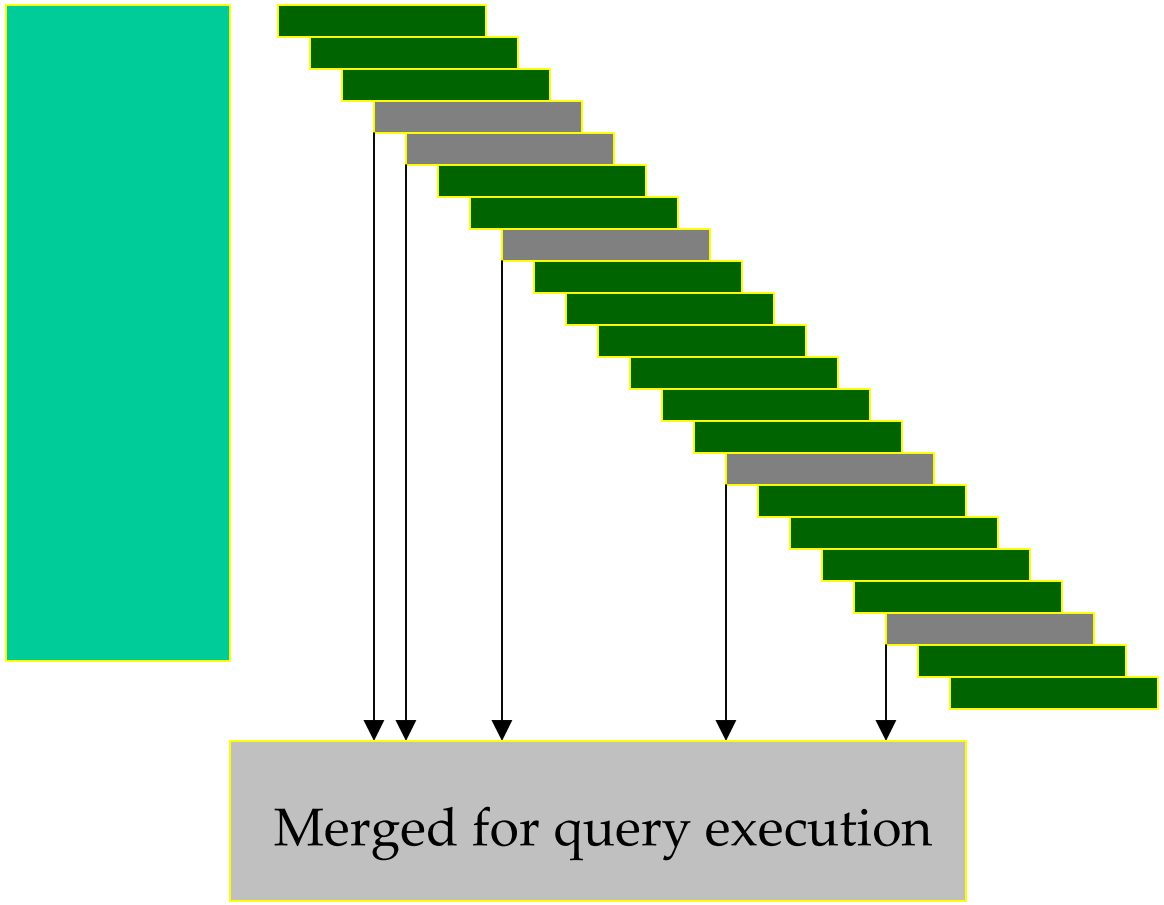
DBMS_STATS

DEGREE

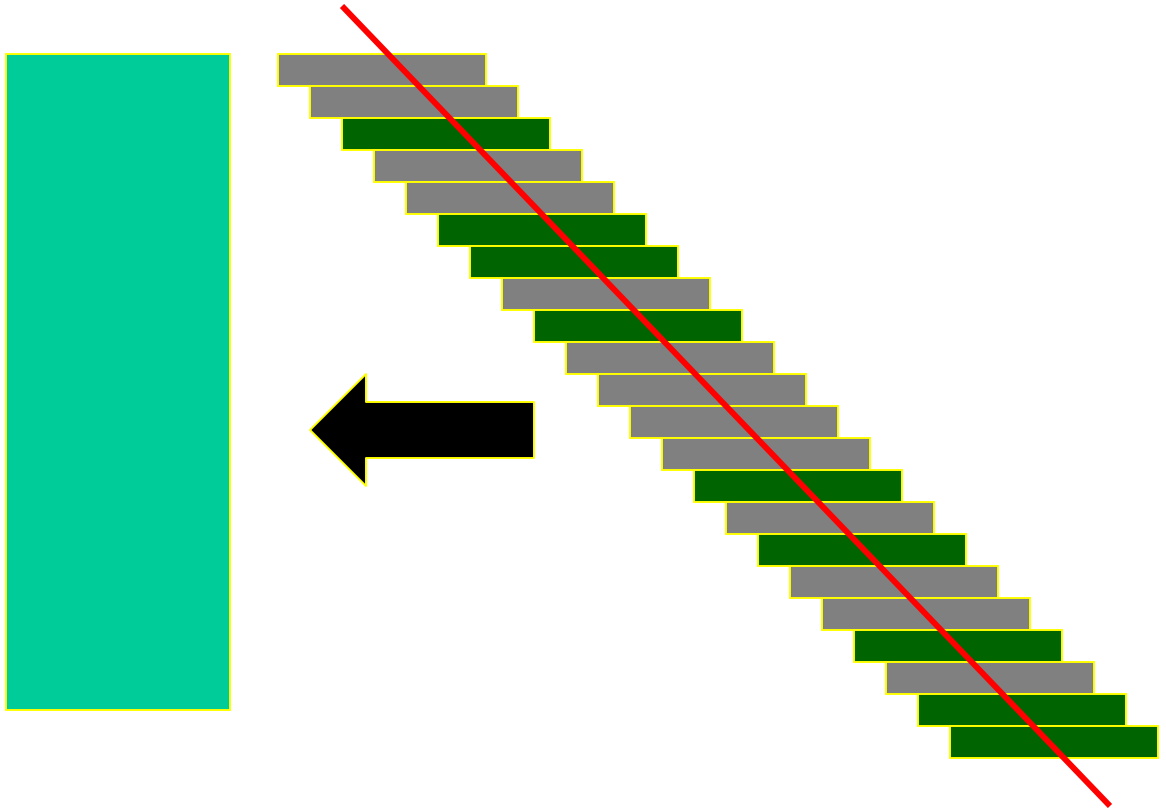
Hard-coded	Integer
Null	Uses table default
DEFAULT_DEGREE	Influenced by initialization values

degree=>dbms_stats.default_degree

Global / Partition-level

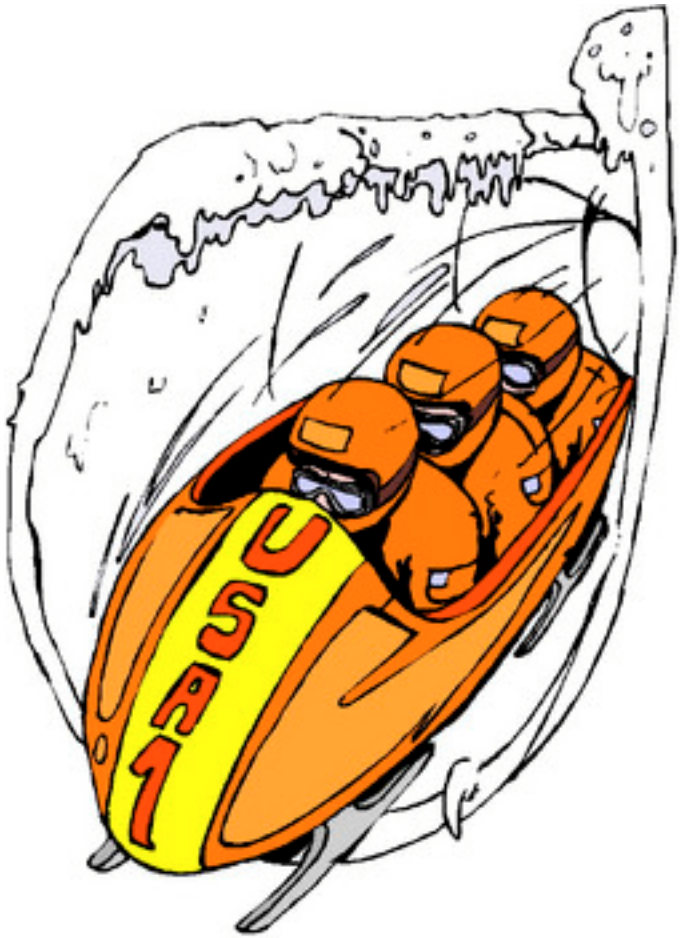


Global / Partition-level



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 applcation 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 9*i* and 10*g*
- Building secondary indexes
  - names may clash with non-partitioned index names
  - perform in parallel
  - may require partition renames afterwards

The unknown frontier

```
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