Partitioning: Tips and Tricks

Arup Nanda

Longtime Oracle DBA
Agenda

• Partitioning primer
• Choosing a partition strategy
• Choosing a partition key
• Solutions to common problems using partitioning
• Potential issues to watch out for
• Creative solutions in partitioning
Local Indexes

- Index is partitioned exactly as the table
- Index entries of each part are found in the corresponding partition in index only
- When table partition is dropped, so is the index partition
- Example
  
  create index in_mytab on mytab (col1) local
Global Indexes

- Entries for all parts of the table are found all over the index.
- Usually used for unique indexes
- Index may be optionally partitioned
- When table part is dropped, the index needs to be rebuilt.
- Example
  
  ```
  CREATE INDEX PK_MYTAB ON MYTAB (COL2) GLOBAL;
  ```
Global-vs-Local Index

- Whenever possible, use local index

- In Primary Key (or Unique) Indexes:
  - If part column is a part of the PK – local is possible and should be used
  - E.g. TXN table. PK – (TXN_DT, TXN_ID) and part key is (TXN_DT)

- If not, try to include the column in PKs
  - E.g. if TXN_ID was the PK of TXN, can you make it (TXN_DT, TXN_ID)?

- Ask some hard design questions
  - Do you really need a PK constraint in the DW?
Global indexes can be partitioned

- The global indexes can themselves be partitioned in any manner, different from the table partitioning scheme

```sql
create table mytab
(
    col1 number,
    col2 date,
    col3 varchar2,
    ... and so on for other columns ...
)
partition by range (col1)
(
    partition p1 values less than (101),
    partition p2 values less than (201),
    partition p3 values less than (301)
)
```

```sql
create index pk_mytab
on mytab (col2)
global
partition by hash
partitions 4;
```

Global index is hash partitioned while table is range partitioned, on different columns.
Different Range Partitioning

create table mytab
  (...)
partition by range (col1)
  (partition p1 values less than (101),
   partition p2 values less than (201),
   partition p2 values less than (301))

create index IN1 on MYTAB (col4)
global
partition by range (col4)
  (partition p1 values less than (100),
   partition p2 values less than (maxvalue))

create index IN1 on MYTAB (col2)
global
partition by range (col4)

Will fail with ORA-14038: GLOBAL
partitioned index must be prefixed

create index IN1 on MYTAB (col4, col2) global
partition by range (col4)
Sub-Partitioning

- **Range-Hash**
  - Sales Date and Sales Trans ID
- **Range-List**
  - Sales Date and Product Code
- **Range-range**
  - 2 date columns
- **List-range**
  - Product code and then sales date
- **List-list**
  - Product code and geographic territory
- **List-Hash**
  - Product code and transaction id
Global Index Maintenance

• Global Indexes maintained with the partition operation
  alter table mypart drop partition p1 update indexes;

• Or, only global indexes:
  alter table mypart drop partition p1 update global indexes;
You want to partition CUSTOMERS on ACC_REP column.

The column is not present on child tables.

Earlier option: add the column to all tables and update it — Difficult and error-prone.

11g has referential partitioning.
Referential Partitioning

Partition CUSTOMERS as usual
create table SALES (  
    SALES_ID number not null,  
    CUST_ID number not null,  
    TOT_AMT number  
constraint fk_sales_01  
    foreign key (cust_id)  
    references customers)  
partition by reference  
(fk_sales_01);

Partitions of SALES are created with data from CUSTOMERS.

Partitioning Tips and Tricks
Addressing Ref Partitions

• USER_PART_TABLES view has info
  – partitioning_type – "REFERENCE"
  – ref_ptn_constraint_name – the FK name

• The partitions will also bear the same name as the parent
INTERVAL Partitioning

• **SALES** table partitioned on **SALES_DT**
  – Partitions defined until SEP 2008. Before Oct starts, you have to create the partition
  – If you don't create the part, the **INSERT** will fail on Oct 1st.

• To mitigate the risk, you created the **PMAX** partition. **Undesirable**

• When you finally add the **OCT08** partition, you will need to split the **PMAX** – **highly undesirable**
Interval Partitions

create table SALES ( sales_id number, sales_dt date )
partition by range (sales_dt) interval (numtoyminterval(1,'MONTH'))
    store in (TS1,TS2,TS3)
( partition SEP08 values less than (to_date('2008-10-01','yyyy-mm-dd')) );

Creates a partition automatically when a new row comes in.

Specifies one partition per month

This is the first partition. The subsequent partition names are system generated.
Addressing Interval Partitions

• **USER_PART_TABLES view:**
  - partitioning_type – "INTERVAL"

• **USER_TAB_PARTITIONS view:**
  - high_value shows the upper bound of partition

• To address a specific partition:
  
  ```sql
  select * from SALES partition for
  (to_date('22-sep-2008','dd-mon-yyyy'));
  ```
Non-Interval Process

• To add partitions automatically:

• To drop partitions automatically:
alter table drop partition part3
update global indexes;

A scheduler job
pmo_deferred_gidx_maint_job cleans up

Column
ORPHANED_ENTRIES in
USER_INDEXES view
giptab_test1.sql
SQL> alter table ptab1 modify partition p1 indexing on;

SQL> alter table ptab1 modify partition p2 indexing off;

SQL> create index in_g2_ptab1 on ptab1 (c1) global indexing partial;

partindex_test1.sql
Watchout!
Date Partition-keys

• Clear definition helps

• This will not choose the partition at compile time

\[
\text{where sales\_date between '1\text{-}jan\text{-}09' and '31\text{-}jan\text{-}09'};
\]

• This will:

\[
\text{where sales\_date between TO\_DATE('2009\text{-}01\text{-}01 00:00:00', 'SYYYY\text{-}MM\text{-}DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN')}
\]
\[
\text{and TO\_DATE('2009\text{-}01\text{-}31 00:00:00', 'SYYYY\text{-}MM\text{-}DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN') }
\]
Partition-wise Joins

- Works for range partitioned tables
- Not for hash partitioned
- Works only for equality operators; not ranges
### Multicolumn

```sql
CREATE TABLE mcptab1
(
    col1 NUMBER(10),
    col2 NUMBER(10)
)
PARTITION BY RANGE (col1, col2)
(
    PARTITION p1 VALUES LESS THAN (101, 101),
    PARTITION p2 VALUES LESS THAN (201, 201)
)
```

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>102</td>
</tr>
<tr>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>201</td>
</tr>
<tr>
<td>100</td>
<td>202</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
</tr>
</tbody>
</table>

### mcpart_test1.sql

```sql```
```
```
Multi-part key Determination

Consider 1st column

< boundary value?

Yes

Place in Partition 1

No

Consider 2nd column

= boundary value?

Yes

Place in Partition 1

No

Place in Partition 2

2nd column is considered only when 1st column is equal to the boundary, not less or not more

Partitioning Tips and Tricks
Subpart Stats Collection

• The normal method to collect stats

begin
    dbms_stats.gather_table_stats (
        ownname=> user, tabname=>'PTEST2');
end;

• Problem:
  – This populates the partition stats but not subpartition stats
  – To collect the subpartition stats, you must use granularity parameter. It has to be either ALL or SUBPARTITION

begin
    dbms_stats.gather_table_stats (
        ownname=> user, tabname=>'PTEST2', granularity=>'SUBPARTITION');
end;
Partition stats collection

- The granularity parameter controls the scope for the stats
- Possible Values
  1. AUTO – determined by Oracle
  2. GLOBAL AND PARTITION – global stats and partition-level stats (subpartition level stats are not collected)
  3. SUBPARTITION – down to subpartition level
  4. GLOBAL – only global stats
  5. ALL – global, part and subpart level
  6. APPROX_GLOBAL AND PARTITION – new in 11g. Global stats are not calculated; but derived from partition stats

<table>
<thead>
<tr>
<th>GRANULARITY</th>
<th>Table Global</th>
<th>Partition Global</th>
<th>Partition Statistics</th>
<th>Subpartition Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>PARTITION</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>SUBPARTITION</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ALL</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
Stats for a specific partition only

• To collect stats for a specific partition (or subpartition)

• Use the partname parameter

```
begin
    dbms_stats.gather_table_stats (
        ownname=> user, tabname=>'PTEST2', partname=>'SALES_Q1',
    );
end;
```

• In 11g, the global stats are automatically updated
Creative Solutions
Partition on Virtual Columns

- VC: not stored with the table
- Computed at runtime
- Can be indexed and partitioned
Partition on Invisible Columns

- Invisible columns are not visible
- Need not be entered
- Can be indexed and partitioned
Index Blocks Too Hot to Handle

- Consider an index on TRANS_ID – a monotonically increasing number
- It may make a handful of leaf blocks experience severe contention
- This hot area shifts as the access patterns change
- Solution: Reverse Key Index?
Solution: Hash Partitioned Index

- Index Can be hash-partitioned, regardless of the partitioning status of the table
- Table SALES is un-partitioned; while index is partitioned
- This creates multiple segments for the same index, forcing index blocks to be spread on many branches
- Can be rebuilt:
  ```sql
  alter index IN_SALES_01 rebuild partition <PartName>;
  ```
- Can be moved, renamed, etc.

create index IN_SALES_01 on SALES (SALES_TRANS_ID) global partition by hash (SALES_TRANS_ID) partitions 8
• **Overlap between Logical Modeling and Physical Design**

• **Last part of logical design and first part of physical design**

• **When should partitioning be used**
  – In almost all the time for large tables

• **There is no advantage in partitioning small tables, right?**
  – Wrong. In some cases small tables benefit too
How to Choose

Partitioning

Arup Nanda
Why? Common Reasons

• Easier Administration:
  – Smaller chunks are more manageable
  – Rebuilding indexes partition-by-partition
  – Data updates, does not need counters

• Performance:
  – Full table scans are actually partition scans
  – Partitions can be joined to other partitions
  – Latching
More Important Reasons

• Data Purging
  – DELETEs are expensive – REDO and UNDO
  – Partition drops are practically free
  – Local indexes need not be rebuilt

• Archival
  – Usual approach: insert into archival table select * from main table
  – Partition exchange
  – Local indexes need not be rebuilt
Materialized Views Refreshes

- Partition Exchange
  - Create a temp table
  - Create Indexes, etc.
  - When done, issue:
    ```sql
    alter table T1 exchange partition sp11 with table tmp1;
    ```
  - Data in TMP1 is available
Backup Efficiency

• When a tablespace is read-only, it does not change and needs only one backup
  – RMAN can skip it in backup
  – Very useful in DW databases
  – Reduces CPU cycles and disk space

• A tablespace can be read only when all partitions in them can be so

SQL> alter tablespace Y08M09 read only;
Data Transfer

• Traditional Approach
  insert into target select * from source@dblink

• Transportable Tablespace
  – Make it read only
  – Copy the file
  – "Plug in" the file as a new tablescape in the target database
  – Can also be cross-platform
Information Lifecycle Management

- When data is accessed less frequently, that can be moved to a slower and cheaper storage, e.g. from Fiber to SATA
- Two options:
  1. Create a tablespace ARC_TS on cheaper disks
     ```sql
     ALTER TABLE TableName MOVE PARTITION Y07M08 TABLESPACE ARC_TS;
     ```
     Reads will be allowed; but not writes
  2. ASM Approach
     ```sql
     ALTER DISKGROUP DROP DISK … ADD DISK …
     ```
     Fully available
How to Decide

• First, decide on the objectives of partitioning. Multiple objectives possible

• Objectives
  – Data Purging
  – Data Archival
  – Performance
  – Improving Backups
  – Data Movement
  – Ease of Administration
  – Different Type of Storage

Assign priorities to each of these objectives
Thank You!

My Blog: arup.blogspot.com
My Tweeter: arupnanda