Oracle Database 11g
Best Practices for using Partitioning in HA Environments and VLDBs

Session# S307725

Ami Aharonovich
Oracle Certified Professional
Independent Oracle DBA Consultant & Instructor
aaharon@gmail.com
About Myself

• Oracle Certified Professional (OCP) DBA
• Over nine years of expertise as an Oracle database developer and administrator
• Working as an independent Oracle DBA consultant and instructor
• Specializing in training Oracle University official courses
• Expertise and hands-on experience using core Oracle Database technologies in versions 8, 8i, 9i, 10g, and 11g
Agenda

• Partitioning Techniques
• Oracle11g New Partitioning Features
• Demonstration:
  – Interval Partitioning
  – Interval Partitioning with Virtual Column
  – Reference Partitioning
  – Reference Partitioning with Hash Sub-Partitioning
  – System Partitioning
  – Interval Partitioning with Hash Sub-Partitioning (real-life example)
• Oracle11g Miscellaneous New Features (if we have time...)
## Oracle Database – 10 Years of Partitioning Development

<table>
<thead>
<tr>
<th></th>
<th>Core functionality</th>
<th>Performance</th>
<th>Manageability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle8</strong></td>
<td>Range partitioning</td>
<td>Static partition pruning</td>
<td>Basic maintenance operations: add, drop, exchange</td>
</tr>
<tr>
<td></td>
<td>Global range indexes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Oracle8i</strong></td>
<td>Hash and composite range-hash partitioning</td>
<td>Partitionwise joins Dynamic pruning</td>
<td>Merge operation</td>
</tr>
<tr>
<td><strong>Oracle9i</strong></td>
<td>List partitioning</td>
<td></td>
<td>Global index maintenance</td>
</tr>
<tr>
<td><strong>Oracle9i R2</strong></td>
<td>Composite range-list partitioning</td>
<td>Fast partition split</td>
<td></td>
</tr>
<tr>
<td><strong>Oracle10g</strong></td>
<td>Global hash indexes</td>
<td></td>
<td>Local Index maintenance</td>
</tr>
<tr>
<td><strong>Oracle10g R2</strong></td>
<td>1M partitions per table</td>
<td>Multidimensional pruning</td>
<td>Fast drop table</td>
</tr>
<tr>
<td><strong>Oracle Database 11g</strong></td>
<td>More composite choices REF Partitioning Virtual Column Partitioning</td>
<td></td>
<td>Interval Partitioning Partition Advisor</td>
</tr>
</tbody>
</table>
Partitioned Tables and Indexes

• Large tables and indexes can be partitioned into smaller, more manageable pieces. Each piece is a different segment
Partitioned Tables and Indexes

• Data is broken down into smaller, more manageable pieces called partitions, or even sub-partitions
• Each partition can be managed individually and function independently of others, providing a better tuned structure for availability and performance
• Transparent to existing applications
• Each partition or sub-partition is a segment
• Different partitions that belong to the same table/index can:
  – Reside in different tablespaces
  – Have distinct storage clauses
  – Be maintained by granular commands
Partitioning Strategies – Single Level

- **Range** (Oracle8): maps data to partitions based on ranges of partition key values for each partition
- **Hash** (Oracle8i): maps data to partitions by using a hashing algorithm applied to a partitioning key
- **List** (Oracle9i): maps data to partitions by using a list of discrete values for the partitioning column
- **Interval** (Oracle11g R1): maps data to partitions of ranges that are automatically created following a specified interval
- **System** (Oracle11g R1): allows the application to explicitly map rows to arbitrary partitions
# Composite Partitioning Techniques

<table>
<thead>
<tr>
<th>1st Level Partitioning</th>
<th>2nd Level Partitioning</th>
<th>Database Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Hash</td>
<td>Oracle8i</td>
</tr>
<tr>
<td></td>
<td>List</td>
<td>Oracle9i</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td>List</td>
<td>Range</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td></td>
<td>List</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td></td>
<td>Hash</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td>Interval</td>
<td>Range</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td></td>
<td>List</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td></td>
<td>Hash</td>
<td>Oracle11g R1</td>
</tr>
<tr>
<td>Hash</td>
<td>Hash</td>
<td>Oracle11g R2</td>
</tr>
</tbody>
</table>
Benefits of Partitioning

• Classic “Divide and Conquer” technique
• Maintenance commands can be applied at the partition level
• More granular storage allocation options including online, offline, rebuild, reorganize and object level backup/restore options
• Optimizer eliminates (prunes) partitions that do not need to be scanned (Partition Pruning)
• Join operations can be optimized to join “by the partition” (Partition-Wise Joins)
• Partitions can be load-balanced across physical devices
Manageability Benefits:
Partition-Level Management

- A partition can be moved from one tablespace to another
- A partition can be divided at a user-defined value
- Partitioning can isolate subsets of rows that must be treated individually
- A partition can be dropped, added, or truncated
- SELECT, UPDATE, INSERT and DELETE operations can be applied on a partition level instead of a table level
Oracle11g Partitioning New Features

• **Interval Partitioning**
  – Automates the creation of range partitions
  – Extends the capabilities of the range method to define equal-partitioned ranges using an interval definition
  – Oracle will create any partition automatically as-needed whenever data for a partition is inserted for the very first time
  – Greatly improves the manageability of a ranged partitioned table
  – Available techniques are interval, interval-list, interval-hash and interval-range
  – You must specify at least one range partition
  – Partitioning key column must be of NUMBER or DATE type
  – Cannot be used for index-organized tables
  – Not supported at the sub-partition level
Oracle11g R1 Interval Partitioning

```
CREATE TABLE INTERVAL_PARTITIONING_DEMO
(serial_num NUMBER,
 name VARCHAR2(32))
PARTITION BY RANGE (serial_num) INTERVAL (10000)
( PARTITION p1 VALUES LESS THAN (10000),
  PARTITION p2 VALUES LESS THAN (20000),
  PARTITION p3 VALUES LESS THAN (30000));
```

Automatically created when inserting data

P1  P2  P3  Pi1  ...  Pin

Range section  Interval section  Transition point

INTERVAL.sql
Oracle11g Partitioning New Features

- **Virtual Column-Based Partitioning:**
  - Allows the partitioning key to be defined by an expression, using one or more existing columns of a table and storing the expression as metadata only
  - Enables a more comprehensive match of the business requirements
  - Supported with all basic partitioning strategies
  - Can also be used with interval partitioning as well as the partitioning key for REF partitioned tables
  - Virtual columns are treated as real columns except no DML operations are allowed
  - One of the most useful innovations in Oracle11g R1
CREATE TABLE SALES
(
  PROD_ID NUMBER NOT NULL,
  CUST_ID NUMBER NOT NULL,
  TIME_ID DATE NOT NULL,
  CHANNEL_ID NUMBER NOT NULL,
  PROMO_ID NUMBER NOT NULL,
  QUANTITY_SOLD NUMBER(10,2) NOT NULL,
  AMOUNT_SOLD NUMBER(10,2) NOT NULL,
  PROD_TYPE NUMBER(1) AS (TO_NUMBER(SUBSTR(TO_CHAR(PROD_ID), 1, 1)))
)

TABLESPACE USERS
PARTITION BY RANGE (PROD_TYPE) INTERVAL (1)
(PARTITION p1 VALUES LESS THAN (1));

INTERVAL_VIRTUAL.sql
Oracle11g Partitioning New Features

- **REF Partitioning:**
  - Allows to partition a table by leveraging an existing parent-child relationship
  - The partitioning strategy of the parent table is inherited to its child table without the necessity to store the parent’s partitioning key column in the child table
  - Transparently inherits all partition maintenance operations that change the logical shape of a table from the parent table to the child table (for example when you drop/add/split partitions)
  - Automatically enables partition-wise joins for the equal-partitions of the parent and child table
  - Perfect for star schemas in data warehouses; partition the fact table according to the dimension tables
Oracle 11g R1 REF Partitioning

Without using reference partitioning

Table ORDERS

Table ORDER_ITEMS

Redundant storage/maintenance of ORDER_DATE

Range(ORDER_DATE)
Primary key (ORDER_ID)

Reference partitioning

Range(ORDER_DATE)
Foreign key (ORDER_ID)

Partition key inherited through PK/FK relationship

Without using reference partitioning...

Reference partitioning...

Table ORDERS

Table ORDER_ITEMS

Range(ORDER_DATE)
Foreign key (ORDER_ID)

Partition key inherited through PK/FK relationship

REF.sql
Oracle11g Partitioning New Features

• **System Partitioning:**
  – Enables application-controlled partitioning
  – Allows the application to explicitly map rows to arbitrary partitions
  – Provides the ability to break down a table into meaningless partitions
  – All aspects of partitioning are controlled by the application
  – Usual performance benefits of partitioned tables are not available (do not have a partitioning key)
  – No support for traditional partition pruning, partition wise joins, and so on
SQL Access Advisor – Overview

• Part of the manageability improvements in Oracle10g and Oracle11g
• Advices on defining appropriate access structures to optimize SQL queries
• Identifies and helps resolve performance problems that relate to the execution of SQL statements
• Recommends which indexes, materialized views or partitions to create, drop, or retain
• Can be run using Database Control or PL/SQL procedures
## SQL Access Advisor – Recommendations

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Comprehensive</th>
<th>Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add new <em>(partitioned)</em> index on table or materialized view</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Drop an unused index</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Modify an existing index by changing the index type</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Modify an existing index by adding columns at the end</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Add a new <em>(partitioned)</em> materialized view</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Drop an unused materialized view <em>(log)</em></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add a new materialized view <em>(log)</em></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Modify an existing materialized view <em>(log)</em> to add new columns or clauses</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Partition an existing unpartitioned table or index</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Partition Advisor

• Beginning with Oracle Database 11g Release 2, the SQL Access Advisor has been enhanced to generate partitioning recommendations.

• Recommendations will show the anticipated performance gains that will result if they are implemented.

• Generated scripts can either be implemented manually or submitted onto a queue within Oracle Enterprise Manager.

• Part of Oracle’s Tuning Pack, an extra licensable option.

• Can be used from within Enterprise Manager or via a command line interface.
Next-Generation Application Performance Management

“A New Technology Platform that Redefines End User Experience Management”

• The Aternity Frontline Performance Intelligence (FPI) Platform uniquely monitors, aggregates, analyzes, and correlates the three components that dynamically interact, define, and constantly impact end user IT experience – in real-time: application performance, real and virtual desktop performance, and end user productivity

• “Aternity provides in-depth visibility into how applications are performing from the perspective of the end-users on the business frontline, with the information and metrics needed to rapidly identify and address performance challenges”

(Moshe Horev, Vice President and Managing Director, Oracle)
• **Preemptive Problem Detection**
  – Automatically identifies impacted users
  – Pinpoints probable cause
  – Dramatically reduces business disruptions
• **360° View of End User Experience**
  – How users consume IT services
  – Correlation of real/virtual desktop performance and Applications usage
  – End-to-end response times
  – Optimization of IT infrastructure/software licensing costs
• **Right-Time Decision Support**
  – Analyzes, correlates, transforms data into actionable intelligence
  – Supports effective business/IT decisions
Aternity Historical Data Tables

- Real-time user activities across all desktop applications (office, ERP, CRM etc)
- Diverse data usage patterns:
  - Workflow of a user for the last 30 minutes
  - Application performance trends for the last 30 days
  - Drill-down to slowest performing users across the organization in the last week
- Optimal performance for all queries:
  - Short timeframes need granular data
  - Long timeframes need to go through huge amounts of data
Aternity Historical Data Tables

- Schema includes 3 main tables:
  - HISTORICAL_DATA_1 – 1st aggregation level, daily partitions
  - HISTORICAL_DATA_2 – 2nd aggregation level, weekly partitions
  - HISTORICAL_DATA_3 – 3rd aggregation level, monthly partitions
- Queries based on TIMEFRAME (date and time) and MA_ID (basic unit of monitoring; response time, latency etc)
- Range partitioned on TIMEFRAME
- Hash sub-partitioned on MA_ID
- Number of hash sub-partitions depends on customer implementation
Example Script – HISTORICAL_DATA_1 Table

CREATE TABLE HISTORICAL_DATA_1
( TIMEFRAME DATE,
  MA_ID NUMBER(10),
  EP_COMBO_ID NUMBER(10),
  CTX_COMBO_ID NUMBER(10),
  ORIGINATING_PATH_INDEX NUMBER(2),
  VARIANCE NUMBER(37,10),
  AVERAGE NUMBER(37,10),
  VALUE_COUNT NUMBER(10,0))
TABLESPACE %TABLESPACE_HD%
PARTITION BY RANGE (TIMEFRAME)
SUBPARTITION BY HASH (MA_ID)
SUBPARTITIONS %SUBPARTITIONS_HIST%
(PARTITION HD_AL1_30012000 VALUES LESS THAN
(TO_DATE('01-02-2000','DD-MM-YYYY')));

HD.sql
Historical Data Procedure

- PL/SQL procedure designed to handle historical data tables
- Designed to work with all three aggregation levels
- Used for creating, managing and dropping table and index partitions based on customer implementation
- Creating new partitions is a must otherwise application fails with ORA-14400: inserted partition key does not map to any partition
- Called from within the Java application code
Historical Data Tables in Oracle11g

CREATE TABLE HISTORICAL_DATA_1...
PARTITION BY RANGE (TIMEFRAME)
INTERVAL(NUMTODSINTERVAL(1, 'DAY'))
SUBPARTITION BY HASH (MA_ID) SUBPARTITIONS %SUBPARTITIONS_HIST%

CREATE TABLE HISTORICAL_DATA_2...
PARTITION BY RANGE (TIMEFRAME)
INTERVAL(NUMTODSINTERVAL(7, 'DAY'))
SUBPARTITION BY HASH (MA_ID) SUBPARTITIONS %SUBPARTITIONS_HIST%

CREATE TABLE HISTORICAL_DATA_3...
PARTITION BY RANGE (TIMEFRAME)
INTERVAL(NUMTOYMINTERVAL(1, 'MONTH'))
SUBPARTITION BY HASH (MA_ID) SUBPARTITIONS %SUBPARTITIONS_HIST%
What About Statistics?

• PL/SQL package designed to handle all schema statistics
• Three main procedures:
  – Gathering statistics on all “regular” tables and indexes
  – Gathering statistics on transient tables using a high-water mark
  – Gathering statistics on historical data tables and indexes partitions
• All statistics are gathered using the DBMS_STATS Oracle supplied package
• Procedures are executed automatically using jobs defined with DBMS_SCHEDULER Oracle supplied package
No Need to Gather – Just Copy

- `DBMS_STATS.COPY_TABLE_STATS` – used to copy statistics between different partitions of the same historical data table (available since Oracle 10g 10.2.0.4)
- `DBMS_STATS.GET_INDEX_STATS` and `SET_INDEX_STATS` – used to copy statistics between different index partitions of the same historical data table index
- Statistics are copied between tables and indexes partitions of the same aggregation level (between days, weeks and months)
- `DBMS_STATS.RESTORE_TABLE_STATS` is used to restore previous statistics for the transient tables
It’s The Little Things…..

• “It’s the little things in life that really make a difference”
• Here’s a little thing that will make your coding life a bit easier and the PL/SQL language a bit more complete:

```
CREATE SEQUENCE this_is_nice;
DECLARE
    n NUMBER;
BEGIN
    n := this_is_nice.NEXTVAL;
END;
/
```
Who’s Using the Default Password?

- Oracle Database 11g now offers a way to quickly identify users with default passwords, implemented in the rather ludicrously simple way of checking a single data dictionary view: DBA_USERS_WITH_DEFPWD
- You can simply identify these users by issuing:
  
  SELECT * FROM dba_users_with_defpwd;
Oracle11g Passwords are Case Sensitive

• Starting with Oracle11g, database passwords have become case sensitive
  SQL> ALTER USER SCOTT IDENTIFIED BY TIGER;
  User altered.
  SQL> CONN SCOTT/tiger
  ERROR:
  ORA-01017:invalid username/password; logon denied
  SQL> CONN SCOTT/TIGER
  Connected.

• Behavior can be changed using the system parameter:
  SEC_CASE_SENSITIVE_LOGON=FALSE
DBMS_COMPARISON

• New in Oracle11g, a PL/SQL supplied package which can be used as a fast and easy way to compare between tables, views and materialized views
• DBMS_COMPARISON.COMPARISON – performs a comparison
• DBMS_COMPARISON.CONVERGE – execute compensating DML to get the two objects to converge
Easy Recovery from Loss of SPFILE

• In Oracle Database 11g, the FROM MEMORY clause creates a pfile or spfile using the current system-wide parameter settings
• During startup, all parameters are logged to the alert.log file
• The alert.log parameter dump text is written in valid parameter syntax which facilitates cutting and pasting of parameters into a separate file, then used as a pfile
• The pfile or spfile name is written to the alert.log at startup and in cases when an unknown client-side pfile is used, the alert log indicates this as well
• To support this additional functionality, the COMPATIBLE initialization parameter must be set to 11.0.0.0 or higher
Easy Recovery from Loss of SPFILE

- The **FROM MEMORY** clause allows the creation of current system-wide parameter settings

```sql
CREATE PFILE [= 'pfile_name' ]
    FROM { { SPFILE [= 'spfile_name' ] } | MEMORY } ;
```

```sql
CREATE SPFILE [= 'spfile_name' ]
    FROM { { PFILE [= 'pfile_name' ] } | MEMORY } ;
```
Some Oracle11g R2 New Features

• Improved partitioning: virtual columns can be used as the primary key or the foreign key of a reference partition table

• Segment creation on demand: initial segment creation for non-partitioned tables and indexes can be delayed until data is first inserted into an object so that empty database objects do not consume any space, reducing the installation footprint and speeding up the installation

• E-mail notification: enables users to be notified of any job activity that is of interest to them without having to constantly monitor the job
Q&A

Ami Aharonovich
Oracle Certified Professional
Independent Oracle DBA Consultant & Instructor
aaharon@gmail.com
Thank You!

Ami Aharonovich
Oracle Certified Professional
Independent Oracle DBA Consultant & Instructor
aaharon@gmail.com