

Partitioning in Oracle Database 10g Release 2

An Oracle White Paper
May 2005

EXECUTIVE OVERVIEW

Oracle Partitioning will enhance the manageability, performance, and availability of a wide variety of applications. Partitioning allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity. Oracle provides a rich variety of partitioning schemes to address every business requirement. Moreover, since it is entirely transparent in SQL statements, partitioning can be applied to almost any application.

BENEFITS OF PARTITIONING

Partitioning can provide tremendous benefits to a wide variety of applications by improving manageability, performance, and availability. It is not unusual for partitioning to improve the performance of certain queries or maintenance operations by an order of magnitude. Moreover, partitioning can greatly simplify common administration tasks.

Partitioning also enables database designers and administrators to tackle some of the toughest problems posed by cutting-edge applications. Partitioning is a key tool for building multi-terabyte systems or systems with extremely high availability requirements.

Basics of Partitioning

Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerably flexibility in managing partitioned object. However, from the perspective of the application, a partitioned table is identical to a non-partitioned table; no modifications are necessary when accessing a partitioned table using SQL DML commands.

Tables are partitioning using a 'partitioning key', a set of columns which determine in which partition a given row will reside. Oracle Database 10g Release 2 provides six techniques for partitioning tables:

- Range Partitioning

Each partition is specified by a range of values of the partitioning key (for a table with a date column as the partitioning key, the 'January-2005' partition contains rows with the partitioning-key values from '01-JAN-2005" - '31-JAN-2005')

- List Partitioning

Each partition is specified by a list of values of the partitioning key (for a table with a region column as the partitioning key, the 'North America' partition may contain values 'Canada', 'USA', and 'Mexico')

- Hash Partitioning

A hash algorithm is applied to the partitioning key to determine the partition for a given row

- Composite Range-Hash Partitioning

A combination of Range and Hash partitioning techniques, whereby a table is first range-partitioned, and then each individual range-partition is further sub-partitioned using the hash partitioning technique.

- Composite Range-List Partitioning

A combination of Range and List partitioning techniques, whereby a table is first range-partitioned, and then each individual range-partition is further sub-partitioned using the list partitioning technique.

- Index-organized tables can be partitioned by range, list, or hash..

Oracle Database 10g Release 2 also provides three types of partitioned indexes:

- Local Indexes

A local index is an index on a partitioned table that is partitioned in the exact same manner as the underlying partitioned table. Each partition of a local index corresponds to one and only one partition of the underlying table.

- Global Partitioned Indexes

A global partitioned index is an index on a partitioned or non-partitioned table that is partitioned using a different partitioning-key from the table. Global-partitioned indexes can be partitioned using range or hash partitioning. For example, a table could be range-partitioned by month and have twelve partitions, while an index on that table could be range-partitioned using a different partitioning key and have a different number of partitions.

- **Global Non-Partitioned Indexes**

A global non-partitioned index is essentially identical to an index on a non-partitioned table. The index structure is not partitioned.

Oracle provides a robust set of techniques for partitioning tables, indexes, and index-organized tables, so that partitioning can be optimally applied to any application in any business environment.

Oracle additionally provides a complete set of SQL commands for managing partitioning tables. These include commands for adding new partitions, dropping partitions, splitting partitioning, and merging partitions.

Partitioning for Manageability

Oracle Partitioning allows tables and indexes to be partitioned into smaller, more manageable units, providing database administrators with the ability to pursue a "divide and conquer" approach to data management.

With partitioning, maintenance operations can be focused on particular portions of tables. For example, a database administrator could back up a single partition of a table, rather than backing up the entire table. For maintenance operations across an entire database object, it is possible to perform these operations on a per-partition basis, thus dividing the maintenance process into more manageable chunks.

A typical usage of partitioning for manageability is to support a 'rolling window' load process in a data warehouse. Suppose that a DBA loads new data into a table on weekly basis. That table could be range-partitioned so that each partition contains one week of data. The load process is simply the addition of a new partition. Adding a single partition is much more efficient than modifying the entire table, since the DBA does not need to modify any other partitions.

Partitioning for Performance

By limiting the amount of data to be examined or operated on, and by enabling parallel execution, Oracle Partitioning provides a number of performance benefits. These features include:

Partitioning Pruning

Partitioning pruning is the simplest and also the most substantial means to improve performance using partitioning. Partition pruning can often improve query performance by several orders of magnitude. For example, suppose an application contains an Orders table containing a historical record of orders, and that this table has been partitioned by week. A query requesting orders for a single week would only access a single partition of the Orders table. If the Orders tables had 2 years of historical data, this query would access one partition instead of 104 partitions. This query could potentially execute 100x faster simply

because of partition pruning. Partition pruning works with all of Oracle's other performance features. Oracle will utilize partition pruning in conjunction with any indexing technique, join technique, or parallel access method.

Partition-wise Joins

Partitioning can also improve the performance of multi-table joins, by using a technique known as partition-wise joins. Partition-wise joins can be applied when two tables are being joined together, and both of these tables are partitioned on the join key. Partition-wise joins break a large join into smaller joins that occur between each of the partitions, completing the overall join in less time. This offers significant performance benefits both for serial and parallel execution.

Partitioning for Availability

Partitioned database objects provide partition independence. This characteristic of partition independence can be an important part of a high-availability strategy. For example, if one partition of a partitioned table is unavailable, all of the other partitions of the table remain online and available. The application can continue to execute queries and transactions against this partitioned table, and these database operations will run successfully if they do not need to access the unavailable partition.

The database administrator can specify that each partition be stored in a separate tablespace; this would allow the administrator to do backup and recovery operations on each individual partition, independent of the other partitions in the table.

Moreover, partitioning can reduce scheduled downtime. The performance gains provided by partitioning may enable database administrators to complete maintenance operations on large database objects in relatively small batch windows.

Statement of Direction

In every major release since the introduction of its partitioning technology, Oracle has added new partitioning methods. Oracle8 introduced range partitioning, Oracle8i introduced hash and composite range-hash partitioning, Oracle9i introduced list partitioning. Oracle Database 10g enhanced the partitioning strategies for index-organized tables and for global partitioned indexes and extended its concurrent index maintenance capabilities for all partition maintenance operations. The latest release, Oracle Database 10g Release 2, enables a fast split operation for partitioned index organized tables and offer enhanced pruning capabilities; it also increases the maximum number of partitions for a single object to one million. Oracle plans to continue to add

new partitioning techniques to ensure that an optimal partitioning technique is available for every business requirement.

CONCLUSION

Oracle Database 10g Release 2 with Oracle Partitioning can greatly enhance the manageability, performance, and availability of almost any database application. Partitioning can be applied to cutting-edge applications and indeed partitioning can be a crucial technology ingredient to ensure these applications' success. However, partitioning can also be applied to more commonplace database applications in order to simplify the administration and costs of managing such applications.



Partitioning in Oracle Database 10g Release 2

May 2005

Author: Hermann Baer

Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

oracle.com

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.