

Database performance boost using table partitioning

Helsinki Metropolia University of Applied Sciences

Degree: Bachelor of Engineering

Degree Programme: Information Technology

Thesis

Date: 10-January-2015

|  |  |  |
| --- | --- | --- |
| Author(s)  Title  Number of Pages  Date | Database performance boost using table partitioning | |
| Degree | Bachelor of Engineering | |
| Degree Programme | Information Technology | |
| Specialisation option | Software | |
| Instructor(s) | Patrick Ausderau, Principal Lecturer(Advisor) | |
| The purpose of this thesis is to show how a performance gain can be boosted using different table partitioning techniques in oracle database. In this thesis, the comparison between oracle cloud database and traditional database regarding privacy issues will be discussed.  The techniques and applications used to support this thesis are as follows:  On the client side, a personal computer with oracle database 11g installed and an application named Oracle SQL Developer, an Integrated development environment for working with SQL in Oracle databases, will be used to send and retreive data.  SQL\*PLUS, an interactive and batch query tool that is installed with every Oracle Database Server or Client installation, is also another tool that is used in this thesis.  Different tables from different sources are created manually and with the help of PL/SQL, (Procedural Language/Structured Query Language, coding techniques in order to show the performance of the database before and after table partioning is applied. | | |
| Keywords | | SQL\*PLUS, SQL DEVELOPER, PL/SQL, 12c |

**Contents**

1 Introduction 1

2 Chapter Heading 1

2.1 Subheading 1

2.2 Subheading 1

2.2.1 Subheading 1

2.2.2 Subheading 1

References 1

Appendices

Appendix 1. Title of the Appendix

Appendix 2. Title of the Appendix

# Introduction

Oracle Partitioning is one of the most commonly used and successful functionalities of the Oracle database. Oracle Partitioning enables large tables and indexes to be subdivided in smaller pieces, improving the performance, manageability, and availability for tens of thousands of applications. Queries and maintenance operations are improved by a large number of magnitude for mission critical systems of any shape, data warehousing, or mixed workloads and any size from hundreds of Gigabytes to Petabytes. Tables and partitions are broken down into smaller zones, allowing for almost unlimited data pruning.

Partitioning 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 extreme high availability requirements. Moreover, partitioning can greatly reduce the total cost of data ownership, using a “tiered archiving” approach of keeping older relevant information still online, in the most optimal compressed format and on low cost storage devices.

Maintenance of large tables and indexes can become very time and resource consuming. At the same time, data access performance can reduce drastically for these objects. Partitioning of tables and indexes can benefit the performance and maintenance in several ways.

In this thesis we will see the different types of partitioning in oracle and also try to see the pluses and minuses of performing partitioning in oracle 12c in regard to security issues as all the tables we partition will be stored in the cloud.

**1.1 Advantages of partitioning**

* Partitioning helps operations like data loads, index creation and rebuilding, backup and recovery which are considered as data management operations at the granularity of partition level rather than at the entire table level.
* Partitioning helps boost query performance. Often, we may query a subset of partitions rather than the entire table and this will provide order-of-magnitude gains in terms of performance which is also known as partition pruning. Because you only access the piece of the table that contains the data needed by the query, partition pruning can also dramatically reduce the amount of data retrieved from disk and reduce processing time, especially for I/O bound databases with small ram data buffers (db\_cache\_size).
* Partitioning can significantly reduce the impact of scheduled downtime for maintenance operations. Partition independence for partition maintenance operations lets you perform concurrent maintenance operations on different partitions of the same table or index. You can also run concurrent SELECT and DML operations against partitions that are unaffected by maintenance operations.
* Partitioning increases the availability of mission-critical databases if critical tables and indexes are divided into partitions to reduce the maintenance windows, recovery times, and impact of failures.
* Partitioning can be implemented without requiring any modifications to your applications. For example, you could convert a no partitioned table to a partitioned table without needing to modify any of the SELECT statements or DML statements which access that table. You do not need to rewrite your application code to take advantage of partitioning.

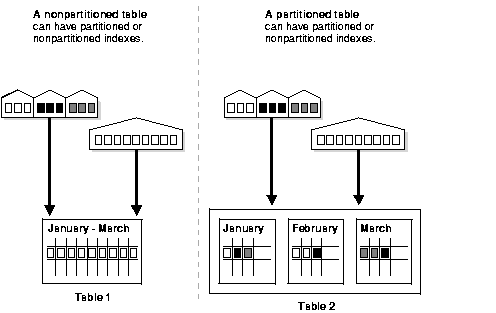


Fig 1.1 View of partitioned tables

## **When to partition a table**

It is always a good practice to understand the timing on which tables has to be partitioned. Here are some of the conditions on which one has to consider partitioning a table.

Tables containing historical data, in which new data is added into the newest partition. A typical example is a historical table where only the current month's data is updatable and the other 11 months are read only.

When the contents of a table need to be distributed across different types of storage devices.

Tables greater than 2 GB should always be considered as candidates for partitioning.

**2.0 Partitioning fundamentals**

**2.1 partitioning for performance**

When you place a given row into a schema, its placement is determined by its value of the partitioning key. How the data of a table is subdivided across the partitions is stored as partitioning metadata of a table or index. The metadata is used to determine for each SQL operation like DML, partition maintenance operations. When this operations happen, automatically the operation touches data which are relevant to this which are portions of a table or even a partition. So performance is boosted by limiting the amount of data to be examined or operated in a partition.

**2.2 Partition pruning (partition elimination)**

Pruning in partitioning is the most effective and simple way of improving performance. This method will most likely improve query performance by several orders of magnitude by leveraging the partitioning metadata to only perform SQL operation on the relevance data.

Depending upon the actual SQL statement, Oracle Database may use static or dynamic pruning. Static pruning occurs at compile-time, with the information about the partitions accessed beforehand. Dynamic pruning occurs at run-time, meaning that the exact partitions to be accessed by a statement are not known beforehand. A sample scenario for static pruning is a SQL statement containing a WHERE condition with a constant literal on the partition key column. An example of dynamic pruning is the use of operators or functions in the WHERE condition. Partition pruning affects the statistics of the objects where pruning occurs and also affects the execution plan of a statement.

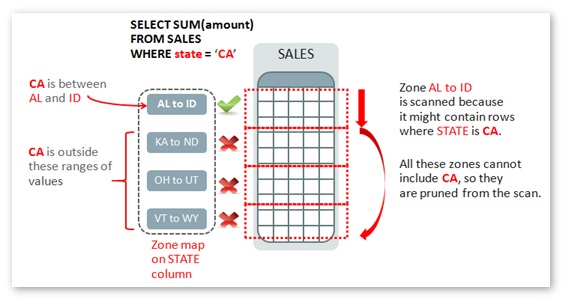
2.2.1 Zone maps

In oracle 12c, a partition pruning is advanced to a new and efficient way called zone maps. Zone maps provides enhanced pruning abilities like better performance with less source consumption and less time-to-information.

You can build zone maps independently for accessing a table. Based on predicates of the table columns, zone maps enable you to prune disk blocks of a table and partitions of a partitioned table. Because zone maps does not have correlation to the partition key columns of a portioned table, statements on partitioned tables with zone maps can prune partitions on non-partition key columns.

Partition pruning with zone maps works well when the zone map column values correlate with partition key column values.

Consider a query that filters a sales table by (North American) state; in this case “CA”. A zone map on the STATE column will record the minimum and maximum values for this column for each zone in the table. This makes it possible to skip the zones that we can be certain won’t contain rows for “CA”.

Fig 2. Table scans with Zones maps

**2.3 Partition for Manageability**

Partitioning for manageable is obtained by partitioning tables and indexes into smaller and more manageable units. Oracle has many set of SQL commands for managing partition tables which are commands for dropping, splitting, moving, adding new partitions, merging, truncating, and exchanging partitions.

When performing maintenance operations in a partitioned table, maintenance can be done on a particular portion instead of the whole table which will degrade the performance of the database if the table is a very large table. Let’s say we have a table portioned by years. One can compress a single partition containing like say 2014 of a table rather than compressing the whole table. As part of the compression operation, the partition can be moved to a lower cost storage tier on oracle 12c version which will reduce the cost of ownership for the stored data. In oracle 12c version, this type of partition maintenance task can be handled completely online allowing DML and queries to occur while the data maintenance operation is happening.

In oracle cloud database, different partitions can be handled as a single atomic operation. For instance, if we have a partition based on a year we can merge part of these partitions like ‘June 2014’, ‘July 2014’, ‘August 2014’, and ‘September 2014’ into a single partition with a name ‘Y1 2014’ with a single merge partition operation.

Another common usage of partitioning for manageability is to support a scenario in data warehouse called rolling window. Here, if we have a data warehouse which stores the most recent 12 months of a sales data, a new partition can be added to the sales table for each new month and an old partition can be removed from the sales table which in this case we will be able to maintain the 12 months of data in the data warehouse as intended.

When we perform delete operations against a very large table, unless the table is partitioned we will encounter poor performance against the system as a whole. So removing data in a very efficient and elegant manner is necessary and this is acquired with the help of partitioning. To purge data from a partitioned table, one can simply drop or truncate the needed partitions instead of issuing a similar delete command which will use lots of resources and every row there is until it gets the one to be deleted. I oracle 12c removing data with a partition maintenance operations like drop or truncate is optimized because this operations don not require any immediate index maintenance to keep all index valid, making it fast metadata-only operations.

**2.4 Partition for Availability**

Partition availability is a key future of partitioning. This provides a partition independence which is a high availability strategy. If there is a condition by which one or more part of a partition is unavailable, the remaining parts of the partitions should be accessible and online. Unless the database operations need the unavailable part of the partition, the application will continue to execute queries and transactions against the available partitions independently.

Storing each partition in a separate table space and having these table spaces stored on different storage tier allows us to easy do backup and recovery operations on each individual partitions independent of the other partitions in the table. This approach will help us perform database operations like the backup and recovery mentioned above on some part the partitions and DML operations on the active data. This is helpful in a sense that more relevant data will be available in the shortest amount of time regardless of the size of the database.

Partitioning can reduce downtime and performance obtained by partitioning may enable us to complete maintenance operations on a very large sized databases in relatively small batch windows.

**3.0 Partition Strategies**

Partition strategies are dependent upon the fundamental data distribution methods that can be used for either single (one-level) or composite (two-level) partitioned tables. There are also many partitioning extensions which will increase the flexibility for the partitioning key selection, providing automated partition as needed, sharing partition strategies across groups of locally connected tables through parent-child relationships, and giving advises for those that don’t have partitions.

**References**

1. Oracle database 12c Partitioning. [Online]

<http://www.oracle.com/technetwork/database/options/partitioning/index.html>

Accessed February 3, 2015.

1. The Data Warehouse Insider

Technical details, ideas and news on data warehouse and big data from the Oracle team. [Online]

<https://blogs.oracle.com/datawarehousing/entry/optimizing_table_scans_with_zone>. Accessed February 6, 2015

1. VLDB and Partitioning Guide. [Online]

<http://docs.oracle.com/cd/E11882_01/server.112/e25523/part_avail.htm>. Accessed February 8, 2015