Oracle Partitioning – everybody thought we're done

Hermann Bär
Oracle USA
Redwood Shores, CA USA

Keywords

Oracle Database, Oracle Database 12c Release 12.1.0.2, Partitioning, VLDB, data management

Introduction

Oracle Partitioning is one of the most commonly used options of the Oracle Database Enterprise Edition, enabling tens of thousands of customers, day in and out, to run their businesses successfully. Oracle Partitioning not only provides functionality to improve performance of many database operations, it also plays a crucial role in data management and maintenance.

Oracle Partitioning is also not only one of the most commonly used options, it is one of the "oldest" one with more then two decades of continuous and focused development.[1] After such a rather long time the question begs what else is there to develop for Oracle Partitioning? While unfortunately we are not able to give you an answer to this question, we nevertheless want to use this article as opportunity to fill you in about some of the exciting new aspects of partitioning that got introduced in Oracle Database 12c Release 1 to refresh your knowledge about the latest enhancements in the currently available database release and to give you a taste of what might be there to come in the future – which will be equally exciting.

Oracle Partitioning – an overview

Large database systems tend to follow and often exceed the Pareto principle, often referred to as 80:20 rule, and store even way more than 80% of its crucial data in way less than 20% of its database objects, and these objects tend to become large. Very large. Dealing with these objects in the most efficient manner is key to success.

Since day one of its development, the goals of partitioning were therefore to improve the performance, manageability, and availability of exactly these large tables (and indexes). The core principle of partitioning is to subdivide these large objects into smaller, more manageable pieces without sacrificing the logical shape of the object. While the smaller pieces – the partitions or subpartitions – represent individually manageable objects for the administrator, the whole object looks just like a non-partitioned table. The information about the subdivision – the partitioning metadata – is used internally for more optimized processing and used by the administrator for data management and maintenance. This simple yet powerful concept makes Oracle Partitioning so successful.

[1] Oracle Partitioning got introduced in 1997 with Oracle Database 8.0.

Figure 1 gives you an overview of the enhancements over the last two decades with a bucketing of newly introduced functionality. The bucketing gives you a rough classification of functionality, which is not mutually exclusive. For example, 'fast partition splits' not only improve the performance of split operations by orders of magnitudes (by elimination of any data movement), it also improves the manageability of data significantly, due to the enormous speed improvement and the reduction of resources required.

	Core functionality	Performance	Manageability
Oracle 8.0	Range partitioning Local and global Range indexing	Static partition pruning	Basic maintenance: ADD, DROP, EXCHANGE
Oracle 8*i*	Hash partitioning Range-Hash partitioning	Partition-wise joins Dynamic partition pruning	Expanded maintenance: MERGE
Oracle 9*i*	List partitioning		Global index maintenance
Oracle 9*i* R2	Range-List partitioning	Fast partition SPLIT	
Oracle 10*g*	Global Hash indexing		Local Index maintenance
Oracle 10*g* R2	1M partitions per table	Multi-dimensional pruning	Fast DROP TABLE
Oracle 11g	Virtual column based partitioning More composite choices Reference partitioning		Interval partitioning Partition Advisor Incremental stats mgmt
Oracle 11g R2	Hash-* partitioning Expanded Reference partitioning	"AND" pruning	Multi-branch execution (aka table or-expansion)
Oracle 12c R1	Interval-Reference partitioning	Partition Maintenance on multiple partitions Asynchronous global index maintenance	Online partition MOVE Cascading TRUNCATE Partial indexing

Figure 1: Oracle Partitioning since Oracle Database 8.0

What's new in Oracle Database 12c Release 1?

The days of having the luxury of data maintenance windows with no user interaction are over for most of our customers. And if you still one of the lucky ones who still have one, how often is the expected time of the necessary data maintenance exceeding your window? Improving the data management and maintenance was the biggest focus of partitioning in the latest release of the database, to make these operations faster, more resource efficient, and making the impact on concurrent ongoing work as minimal as possible – zero if possible. The following article will focus on the top three data maintenance enhancements you should put on your radar when upgrading to Oracle Database 12c.

Online partition move

Moving partitions (or subpartitions) is one of the most widely used **p**artition **m**aintenance **op**eration (PMOP). Even without changing the metadata of a partitioned table there are a variety of reasons to manage data:

- Over time, data in a partition becomes more static and it is time to compress and condense the data. Moving a partition can change the storage format of a partition from being formerly uncompressed to a compressed format.

- Over time, data in a partition becomes less business critical and should be placed on a less performant and cost effective storage platform. Moving a partition can change the physical location of a data segment and move it from one tablespace to another. Tablespaces as logical storage containers of an Oracle database can reside on different storage devices.

- Over time, DML activities on a table slow down and you consider to do some 'clean up' work. If you are familiar with Oracle's logical storage structures[2], you know that especially for volatile tables with concurrent DML activities you can waste space over time, or introduce row chaining, and not having all records stored as dense as it would be possible. Moving a partition recreates the whole data segment and stores all data records in the densest and optimal way possible.
-

Prior to Oracle Database 12c, moving a partition takes an exclusive DML lock on the partition that you are working on (there are other DDL locks taken for metadata integrity, which are not relevant for this discussion). Concurrent DML attempts simply wait until the move is finalized and are then processed. This is simply to ensure data integrity since the actual movement is done as a bulk operation – think of it as an undercover create-table-as-select - and the database had no way to track changes on individual records that are 'in flight'.

While this behavior by itself does not cause any problems or violate the data integrity, some applications cannot cope with this behavior. For example, think of a transactional system that expects DML operations to succeed in sub-seconds and now all of a sudden 'stops' for multiple seconds or even minutes because the underlying data segment is moved. The wait and subsequent cascading of blocked operations can be catastrophic. Application developers better code around this.

The good news is that with Oracle Database 12c, moving a partition does not require any DML lock anymore. As Figure 2 shows, DML operations continue to work seamlessly while a partition move operation is in flight, without any interruption for concurrently ongoing operations.

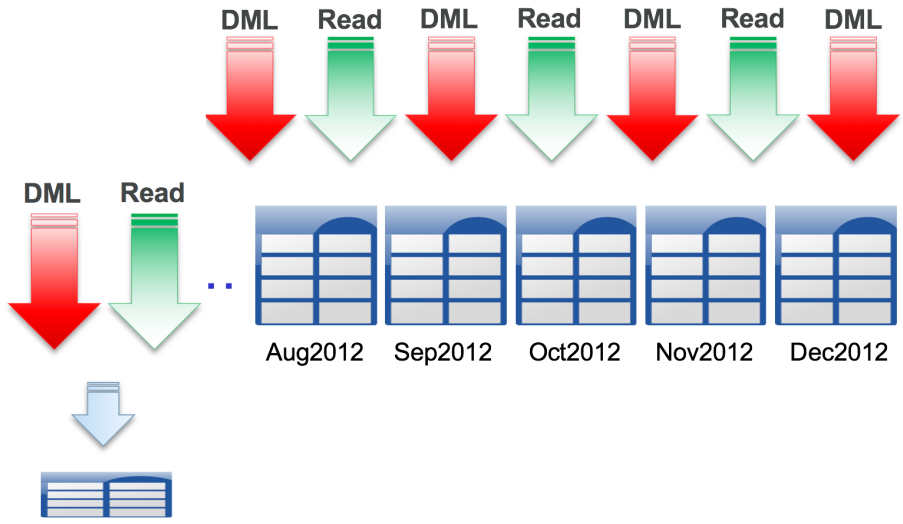

Figure 2: Online partition move compress with concurrently ongoing DML and read operations

The new behavior has not become the default behavior, so a minimal code change is required: adding the new keyword `ONLINE` to any `ALTER TABLE MOVE [SUB]PARTITION` command enables this new functionality and you're set.

[2] For details see http://docs.oracle.com/database/121/CNCPT/logical.htm#CNCPT004

Without going into the details of all the internals that take place, what basically happens under the covers is that Oracle now keeps track of concurrent DML operations under the covers to ensure data integrity and not to loose any DML change. Consequently, the runtime for the partition move command now depends on the amount of concurrent DML operations, and so is the internal space consumption for tracking the changes. Although the change tracking is as minimal as possible, it needs to be done.

Existing applications should consider enhancing its code and take advantage of the new online move capabilities. This is mostly to prepare these application for future enhancements that might benefit from this new transparent behavior.

New applications should consider always using the new online move capabilities. However, you should not consider this as a blanket check to not think about the application workflow anymore: while the behavior has become transparent, the operation itself still requires resources – CPU, memory, space – to process the partition move. It is also obvious that the change tracking adds more work to the internal processing. Planning for these operations to happen in non-peak hours is still highly recommended.

Partition maintenance operations on multiple partitions

Whether you merge, split, or drop partitions, all of these operations are changing the metadata of a partitioned table. For example, monthly partitions get merged into a quarterly partition after data becomes older than one year to reflect the future data usage pattern and data maintenance granularity.

Prior to Oracle Database 12c, partition maintenance operations dealt with a maximum of two partitions: you could either merge two partitions into one partition, or you could split a single partition into two new ones (all other PMOPs impact only one partition). This represents a "limitation" for numerous business use cases that you currently have to work around with either multiple consecutive PMOPs or with application code that works around this. Limitation in this context means that there is not a simple single command to address the business requirement. Considering the previous example, merging three monthly partitions into a single quarterly partition could be implemented in two ways:

- Two consecutive merge partition commands: First, you merge month one and two into an intermediate partition and then you merge this intermediate partition with month three, resulting in the final quarterly partition.

- Create an intermediate nonpartitioned table using a create-table-as-select from the three monthly partitions, drop partitions for month one and two, and then to exchange partition three (which now spawns the whole quarter) with the intermediate nonpartitioned table.

As you can imagine, such approaches require more resources and more code than a possibly native implementation.

Oracle Database 12c enhances PMOPs to work on more than two partitions; in fact, there is no limitation how many partitions can be worked on. You now can merge, drop, split, and truncate as many partitions as you want in a single PMOP[3]. Coming back to our previous example to merge three

[3] Exchange partition does not support the exchange of multiple partitions with multiple tables in a single DDL command. If you see any business reason for such a command then please get in touch with hermann.baer@oracle.com with a description of your business justification.

monthly partitions into a single new quarterly partition, you now use a single atomic DDL operation to make this happen:

```
ALTER TABLE orders
MERGE PARTITIONS Jan2009, Feb2009, Mar2009
INTO PARTITION Quarter1_2009 COMPRESS FOR ARCHIVE HIGH;
```

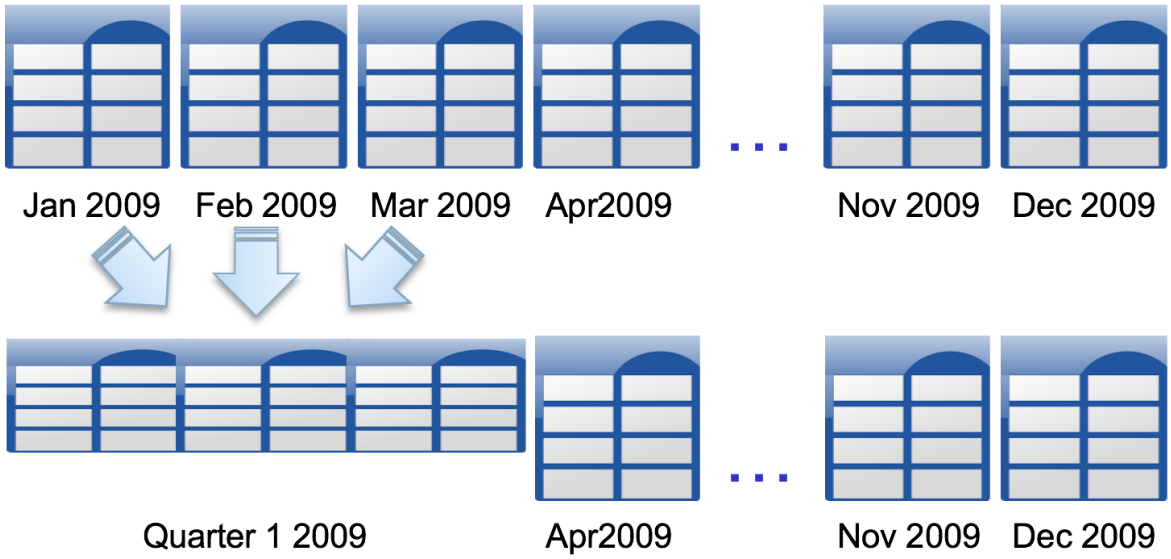

Figure 3: Merging multiple monthly partitions into a single quarterly partition

As you can see, partition maintenance operations on multiple partitions simplifies the coding for many business scenarios. Furthermore, and this is an often overlooked fact, reducing the number of partition maintenance operations means reducing the number of internal metadata operations, both for the dictionary and the library cache. Especially on systems with large partitioned objects and potentially highly concurrent usage of these objects (DML and DDL), this enhancement can make a significant difference.

You should consider enhancing your applications with this new functionality whenever possible. The code simplification and reduction of metadata operations is worth the investment of revisiting your code.

Asynchronous global index maintenance

Last but not least, probably the most significant enhancement for data maintenance operations with Partitioning is the kernel's capability to decouple global index maintenance from certain partition maintenance operations without sacrificing the validity of the index[4]. PMOPs that remove data from a partitioned table – namely drop and truncate [sub]partition - no longer need the instantaneous index maintenance as part of the PMOP.

[4] Partial indexing, another Oracle Database 12c enhancement, can achieve something similar.

No, this optimization does not make index maintenance void – at some point in time it hast to be done – but the decoupling provides a number of obvious business benefits:

- Significantly faster partition maintenance operations drop and truncate. Whenever any of these operations is embedded into a more complex sequential business workflow, the workflow will become faster and provide more time for other operations in the workflow.

- Drop and truncate will not only be faster. They're reduced to metadata-only operations. Irrespective of the data volume that is dropped or truncated, you need almost zero resources for this operation and continue to have valid global indexes. When developing applications you do not have to worry about the runtime or resource requirement of these operations anymore.

- As mentioned previously you still have to do the index maintenance at some point in time. But you do this maintenance when it's best for the system, meaning in off-peak hours and possibly only once after multiple drop and truncate operations. You can rely on the automated cleanup job that is set up by the system or do it manually using `ALTER INDEX COALESCE` or `REBUILD`.

Asynchronous global index maintenance for drop and truncate partition is implemented as completely transparent functionality. If you want to take advantage of this new functionality you only have to upgrade to Oracle Database 12c. If you do not want to rely on the automated cleanup job then the only task for you is to decide how to control the index cleanup. That's it.

Summary

Partitioning in Oracle Database 12c contains a wealth of enhancements beyond the previously discussed functionality. Since its first introduction in Oracle 8.0 in 1997, Oracle continuously enhanced the functionality of Oracle Partitioning and the latest release is no different.

Partitioning is for everybody. Partitioning can greatly enhance the manageability, performance, and availability of almost any database application. If you have not looked closely into Partitioning, now is the time. Sooner or later it will provide benefits for your application.

Contact information:

Hermann Bär, Oracle USA
400 Oracle Parkway MS 4op7
Redwood Shores, CA 94065 USA

Telefon: +1 (650) 506.6833
Fax: +1 (650) 506.6833
E-Mail Hermann.baer@oracle.com
Internet: www.oracle.com