Top Ten Tips for Partitioning

Hermann Bär
Oracle USA
Redwood Shores, CA USA

Keywords:

Oracle Partitioning, Data Partitioning, Performance, Data Management, Oracle Database

Introduction

Oracle Partitioning is probably the most adopted option of Oracle Database Enterprise Edition, helping tens of thousands of customers day in and out to address their business requirements. As functionality that was introduced in Oracle Database 8.0 it is also one of the most mature parts of the Oracle database, providing benefits for a system's performance and data management

Despite the fact that such a large user population uses Oracle Partitioning, there are specific topics that customers are inquiring on – or simply getting wrong. This session/paper is trying to address some of the most common questions Product Management and Support is getting asked when it comes to Partitioning, both from a functional as well as from a how-to perspective. It is by no means the document that makes you the best expert on the world, nor does it claim to only discuss topics that you might have never heard about. However, there are hopefully titbits of information that enable the reader to make an even better use of Oracle Partitioning than what they're doing today.

It is never too late for Oracle Partitioning

Not everybody starts off with a system and table design that is set in stone for eternity; changes are inevitable. As the business grows, so does the data. Core tables start to grow and grow, and at some point in time the increase in size shows its ugly face, impacting the performance of a system. Performance starts to degrade and data maintenance not only becomes slow but impossible. Using Oracle Partitioning could alleviate those problems, but how to get there? What to do next? Downtime is not an option, and the tables keep growing … .

Thankfully Oracle Database 12c Release 2 introduced the functionality to convert any nonpartitioned table into a partitioned table in a completely online fashion while converting and updating indexes – and all of this with a single DDL command (this functionality is complementary to the existing online redefinition package, completely lock free, and single purpose-built). All you have to do is to specify the target partitioning scheme and how to convert indexes. The latter one might not even necessary since Oracle introduced a set of comprehensive conversion rules, having minimal to zero performance impact in mind:

- All prefixed indexes will be converted to local partitioned indexes
- Global partitioned indexes will retain the original partitioning shape unless prefixed
- Non-prefixed indexes will become global nonpartitioned indexes
- Bitmap indexes will become local partitioned indexes

Assuming those default rules are applicable, the conversion of a nonpartitioned table SALES into a list partitioned table could look as simple as the following:

```
ALTER TABLE sales MODIFY
PARTITION BY LIST (region)
  (PARTITION p1 VALUES ('USA'),
   PARTITION p2 VALUES ('Germany'),
   PARTITION p3 VALUES ('Japan'),
   PARTITION p4 VALUES (DEFAULT))
UPDATE INDEXES ONLINE;
```

With this single command, the table will be converted completely online, and all existing indexes will be converted according to the rules listed above. Job done.

Partition independence for tables and indexes

As secondary access structures, indexes can either have a coupled 1:1 relationship with table partitions or exist independent of the table's partitioning structure.

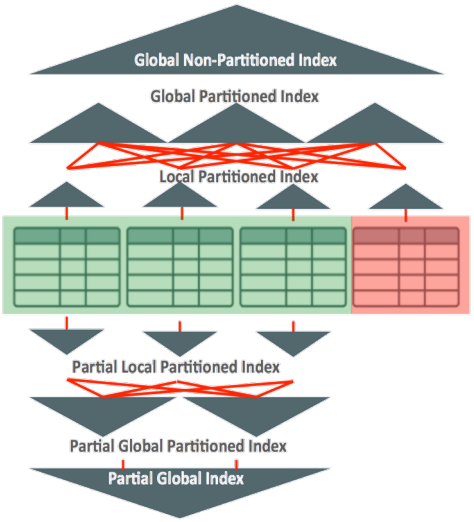

Illustration. 1: Indexing of a partitioned table

Coupled indexes are called **local indexes**: the index inherits the partitioning strategy of the table, and the existence of an index segment is tied to the existence of a table partition. If a partition is added, a new index segment is created. If a partition is dropped or truncated, the exact same operation takes place for the index segment. **Global indexes** are not coupled to the partitioning strategy of the table; such indexes can be partitioned using a different partitioning strategy than the table or can be nonpartitioned at all.

You can also control the data content of an index and define whether all of the table data should be indexes or only some partitions; by default all data of a table is controlled by an index, the so-called **full indexing**. **Partial indexing**, as the name suggests, has an index created only on a subset of the partitions of a table; whether or not a table partition is indexed by a partial index is driven by the metadata of a partition.

Whether an index is local or global, full or partial, any index can be maintained online as part of a partition maintenance operation. For the longest time the biggest difference touted between local and global indexes was around its behavior for partition maintenance: local indexes were considered superior due to the fact that specific partition maintenance operations were optimized by coupling the index maintenance with the table partition maintenance. Drop and truncate partition operations on a table translated directly into the equivalent drop and truncate index partition operation for index maintenance. No data had to be touched for keeping an index valid.

While this was true prior to Oracle Database 12c, global indexes have caught up and are now on par with local indexes in terms of optimized partition maintenance operations. Global index maintenance for drop and truncate partition no longer has to touch any data for staying valid. Any real data maintenance is decoupled from the origin partition maintenance operation and can be delayed to a later point in time. This functionality will be discussed briefly in the next section of this article.

With local and global indexes on par (in terms of optimizations for partition maintenance), the appropriate indexing strategy should always be chosen based on the business requirements and access patterns, making partitioning well suited to support any kind of application.

Asynchronous global index maintenance

One of the most significant enhancements in Oracle Database 12c for data maintenance operations is the capability to decouple global index maintenance from certain partition maintenance operations, namely drop and truncate partition. Without any index maintenance global indexes remain valid, and the partition maintenance operation becomes a pure metadata operation.

As mentioned earlier, this brings local and global index maintenance on par: all partition maintenance operations that remove data are now metadata-only operations. There's only one difference: after such an operation global indexes have orphaned data entries that should be cleaned up at some point in time.

At design time of this functionality the decision was made to implement a 'deep cleaning' of an index, with the expectation in mind that people will in fact decouple the partition maintenance operation from the actual index cleanup. Using an alter index coalesce command Oracle would go over the index blocks and not only cleanup all orphaned entries but also coalesce index blocks.

However, old habits die hard. Oracle learned that while customers love this functionality, the are not really decoupling the processing. Most often you see people replacing the old partition maintenance operation that included the index cleanup with the new meatadata-only partition maintenance followed immediately by the index cleanup call. The initial implementation is too heavy-weight for such a processing, so Oracle enhanced this functionality by adding a second cleanup method that is more or less identical to the classical index maintenance (without block coalescing); the second method also becomes the default to reflect the most common use case of this new functionality.

All existing customers can benefit from this functionality by applying patch for "bug" 24515918, which is more a feature than a bug, but that should not be said too loud ;-)

Read-only partitions versus read-only tablespaces

Available with Oracle Database 12c Release 2, read-only [sub]partitions provide the functionality to set an individual [sub]partition to ready only. Prior to this release, setting this attribute semantically was only possible on a table-level, which made this an all-or-nothing setting. Details of this new functionality are discussed in various places like this blog. The emphasis of this paper is on the differences to a much older functionality that is somewhat similar, yet different: read only tablespaces.

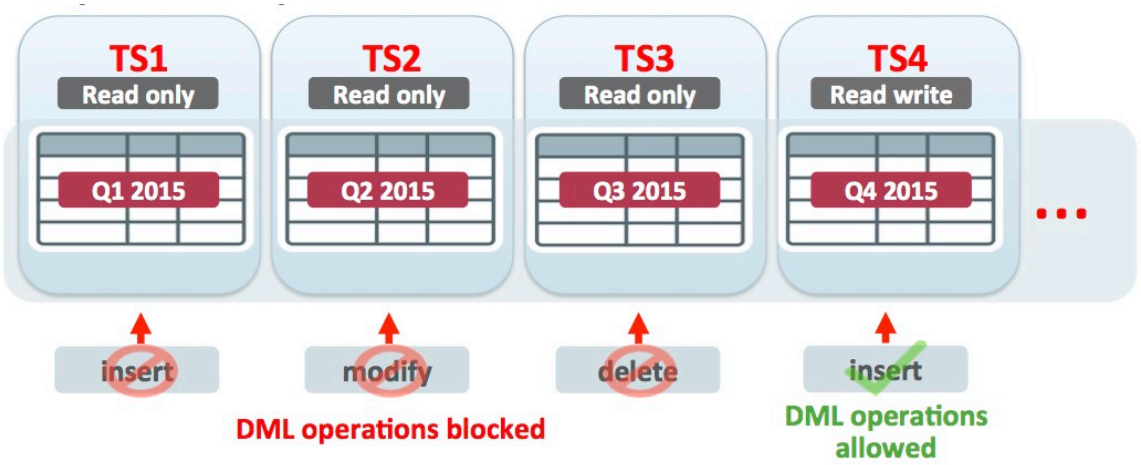

Illustration. 2: Partitioning using read only tablespaces

With read only tablespaces (that got introduced in what feels the dawn of time) you can set individual tablespaces – the physical containers storing logical objects – to disallow read write operations. On the physical container. That exactly is the difference. By protecting the physical container and disallowing read write operations you cannot guarantee data immutability. A lot

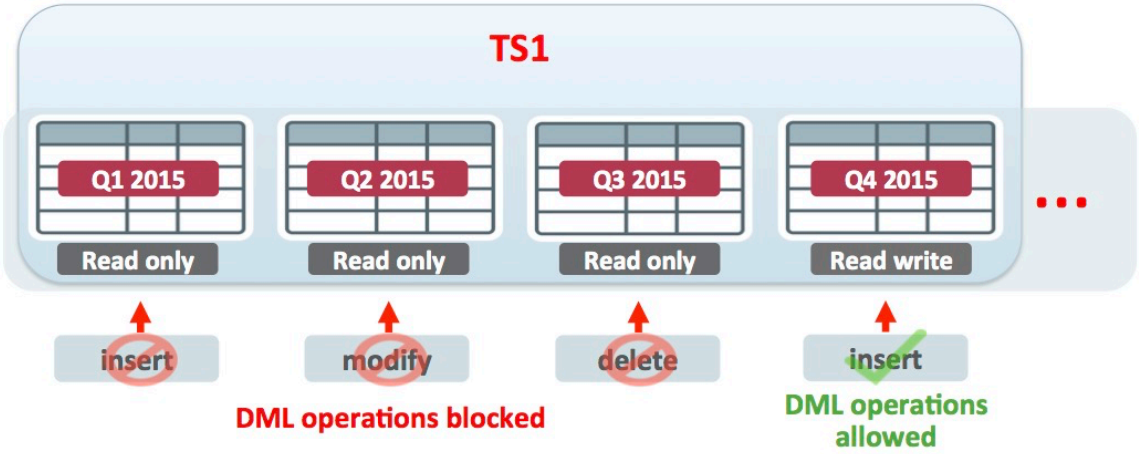

Illustration. 3: Partitioning using read only partitioning

Summary

Irrespective whether you are using Oracle Partitioning for a month or for decades, there is a lot to know about Partitioning, and often the "little things" are making the difference. Sometimes it is equally important to understand what functionality is not built for as opposed to knowing what it provides. Addressing some of these "little things" was the purpose of this article.

Partitioning in any Release contains a wealth of functionality and continually evolves, enhancing existing functionality and introducing new functionality. This is true for the current release of Oracle Database 12c as well as for the upcoming release of Oracle Database 18c, so keep on the good work and enhance your knowledge and your systems together with Oracle Partitioning.

Contact address:

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

Phone: +1 (650) 506.6833
Fax: +1 (650) 506.6833
Email hermann.baer@oracle.com
Internet: www.oracle.com