ORACLE DATABASE 12*c* IN-MEMORY OPTION

The Top Tier of a Multi-tiered Database Architecture

There is this famous character, called Mr. Jourdain, in The Bourgeois Gentleman, a piece written by Molière almost 350 years ago. When he learns that the word "prose" means all speech that is not verse, he suddenly realizes that he has spoken prose for more than 40 years.

In a way, in-memory is prose, too. Many people today claim that the in-memory database is a revolutionary architecture. It turns out, however, that it is surprisingly difficult to tell what exactly makes this architecture revolutionary:

- The name suggests considering the fact *that data to be read or manipulated are kept in memory*. However, this is true for all databases. Data may be stored on disk to guarantee persistence, but whenever a user wants to read or update existing data they are loaded into a database cache and kept there as long as possible. In well-tuned Oracle systems, 95% or more of all data requests find that the data they need are already available in memory. Which means: That is „almost in-memory", and a difference of 5% is certainly not a valid reason to speak of a revolution
- "Traditional" databases store data as rows, whereas *in-memory databases use a columnar format*. But again, that is nothing new. For many years columnar databases have been available from several vendors
- Finally, certain vendors provide the option to create *stored procedures*, which allow developers to push down application functionality to the database in order to reduce the data traffic between client (or application server) and database server. This is certainly a good idea, but it is nothing new, nor is it an in-memory-specific feature. The Oracle Database introduced stored procedures more than 20 years ago, and state-of-the-art applications have always made use of them

What is really new about in-memory databases?

All this does not mean that there is *nothing* new at all. It just means that the answer is neither to be found in the term „in-memory database" nor in the majority of the marketing claims. Not one single of the technologies used in in-memory databases is new. What *is* new, however, is the idea to combine several technologies which so far were only available in separate products.

In the past, there were "disk-based" OLTP databases and „in-memory" Decision Support (DSS) databases. Customers who wanted to have both had to buy two separate products, to build two separate systems and most probably also to figure out how to feed data generated by users of the OLTP system into the DSS system.

What is really new about in-memory databases is the idea to get rid of this separation and to combine both technologies in one single product. Or at least: this is the vendor's perspective. From the customer's perspective, "one single product" means: one single system (instead of two), less integration and administration effort, possibly new types of applications and real-time decision support.

Gravity

We are able to build rockets flying to the moon. However, this does not mean that we got rid of gravity. We need to overcome it. Similarly, there are two fundamental laws in the world of database architectures, which everybody who wants to fly to a pure in-memory architecture will experience as a kind of gravity:

- OLTP transactions, which operate on few rows, but many columns, work best on the row format, whereas analytics (DSS), accessing few columns of many rows, works best on the column format
- Data in memory are volatile. In order to guarantee not only performance, but persistence as well, a non-volatile (disk-based) storage must be available as a secondary storage tier

The first law actually explains, why the project to combine the disk- and row-based database technology with the memory- and column-based technology in one single product is worth the effort. If one technology were able to handle all kinds of transactions optimally, the combination would not make any sense. Here "gravity" means: If the goal is the consolidation of OLTP and DSS in one single product/system, then a columnar-only architecture is not possible.

The second law explains, why even the latest in-memory database systems must run on traditional computers and make use of disk storage. This is the second meaning of "gravity".

The marketing teams of the different vendors put this in different ways. Some talk primarily about in-memory columnar computing, while silently adding support for row format and disk storage. Others describe in-memory computing as an extension of traditional, disk-based computing. In all cases, however, they really say: We want to fly, but we cannot neglect gravity.

Figure 1: Oracle 12c– dual format in-memory database

Oracle Database 12*c* In-Memory Architecture

Oracle is and wants customers to be aware of the two fundamental laws just described. Nevertheless, Oracle wants to help customers fly as well. Therefore the Oracle Database 12*c* In-Memory Option is based on a *dual-format data store*:

- Data are persistently stored on disk, and they are *stored in row format only*
- Whenever data are requested for read/write operations (data manipulations), they are loaded into the traditional *Row Store (Buffer Cache)*
- Whenever data are requested for read-only operations, they are populated into a new *In-Memory Column Store*. This population, of course, includes a transformation from row to columnar format
- Whenever a transaction that includes inserts, updates, or deletes is committed, the new data will immediately and simultaneously appear in both the row store and the in-memory column store. Therefore both stores are *transactionally consistent*

Note that this approach does not necessarily require more memory. There is no need to populate the same data in both stores. If they are required for OLTP only, they will not be populated into the column store, and if they are used for DSS only, they will not be kept in the row store. In addition (as we will see shortly) it is possible to restrict the data populated into the in-memory column store to subsets of the table data.

This approach has many important benefits for customers:

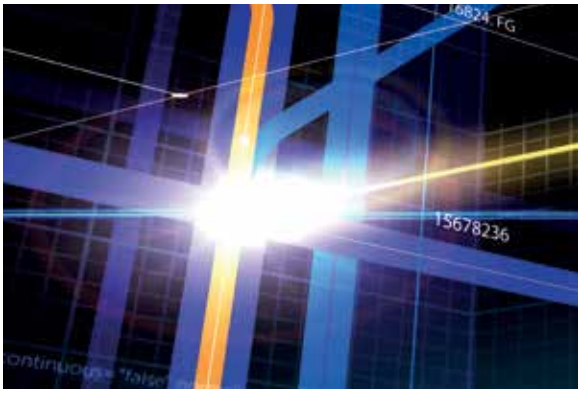
- There is no need to modify applications. All existing applications run unchanged in the new architecture
- There is no need to modify the database. The Oracle Database 12*c* In-Memory Option can be implemented without a database migration or a table reorganization
- There are no limits for database or table sizes. The Oracle Database 12*c* In-Memory Option can be used with databases and systems of any size
- Therefore there is no need to change the infrastructure. The new in-memory feature can be implemented on the existing hardware

- Oracle's In-Memory Option is fully compatible with other optional Oracle Database features such as table compression, table encryption, and table partitioning. It is also compatible with the scale-out architecture provided by Real Application Clusters (RAC) and with all existing high availability technologies (such as Data Guard). These features work exactly the same way with and without the In-Memory Option
- Customers can decide how they want to implement the new architecture. If customers want a *revolution*: if they want to buy completely new hardware and store as many tables as possible in the in-memory column store – that can be done. If, on the other hand, customers prefer an *evolution*: if they want to keep the existing hardware, to keep just a few tables in the in-memory column store and see how it works – that can be done as well

An easy start …

There is yet another benefit: Oracle has made it very easy to get started with the In-Memory Option. Here is what needs to be done:

- Assign a value to the new initialization parameter *inmemory_size* in order to define the size of the in-memory column store
- Select the table(s) that you want to be available in the in-memory column store:
  ALTER TABLE T1 INMEMORY;



And that's it!

… and fine-grained control

An easy start based on intelligent defaults for typical situations – this is what Oracle customers expect. In addition, however, Oracle customers expect mechanisms, which allow for fine-grained control and tuning in special cases. The Oracle Database 12*c* In-Memory Option provides such mechanisms. Let us look at three examples.

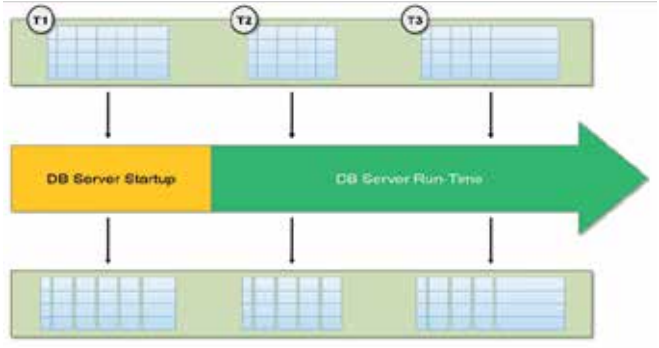


Figure 2: Population of the In-Memory Column Store

(1) Even if you use the Oracle Database 12*c* In-Memory Option, all data is stored in row format on disk. The ALTER TABLE statement does not change this, nor does it populate the table data into the in-memory column store. It only tells the database that you want the table data to be available in the in-memory column store at a certain point in time.

But at which point in time? Two basic options are available as an answer to this question: (a) on demand or (b) during database system startup. Option (a) is the default: Table data are populated into the in-memory column store, when they are first referenced by a query (see figure 2, tables T2 and T3). Whenever this behavior is not appropriate, the database administrator can specify that this job should be executed during database server startup (see figure 2, table T1). By using and combining these options, *the population work can be distributed in time*.

(2) Tables can contain historical (in ILM terminology: "cold") data, which are neither updated anymore nor accessed by queries. If those tables are very large, it would be a waste of memory to keep them completely in the in-memory column store. Therefore administrators may want to restrict the population process to the data really needed by DSS queries.

Table partitioning allows them to make this happen, because the INMEMORY attribute can not only be specified on the table level, but on the partition or subpartition level as well (see figure 3, left-hand side). If the table is partitioned in a useful way (e.g. by month), this internal structure can be used to *define a horizontal subset of the table data* to be kept in the in-memory column store.

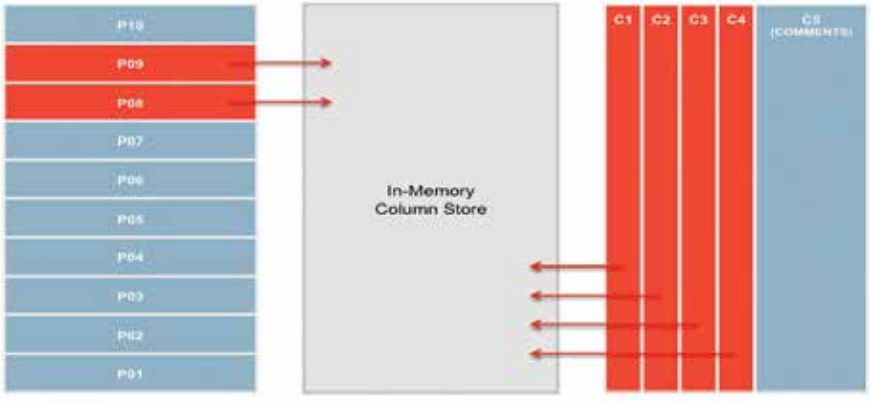


Figure 3: Defining horizontal and vertical subsets

(3) A CUSTOMERS table may contain the usual data (name, zip code, city, etc.), nicely divided into columns, and, in addition, a column which can be used by sales people to store customer-specific comments as text strings (see figure 3, right-hand side, column C5). The data in this column is unstructured, consists of unique values of very different lengths, and is most probably not relevant for DSS queries.

Again the database administrator may wish to restrict the data to be kept in the in-memory column store, but in this case the goal would be to define a vertical subset of the table data, i.e. to exclude one or more columns from the population process. And again it is possible to make this happen, because the Oracle In-Memory Option allows administrators to specify different in-memory characteristics for different table columns.

The first lesson learned

However brief our discussion of subset building may have been – it should already be clear that the Oracle Database 12*c* In-Memory Option is based on a certain concept, which is not necessarily the concept of all other in-memory database products. If „in-memory database" is supposed to mean that *all data* must be stored in memory and if the Oracle Database 12c In-Memory Option is seen as HANA for the poor, then all attempts to implement this new functionality will fail. Key to success is the insight into the fundamental difference between HANA and Oracle Database 12*c*.

There are two statements on which SAP and Oracle agree. The first one has already been discussed. It defines the goal of the current research and development efforts: It makes sense to combine row-based and column-based technologies in one single product, as this allows customers to combine OLTP and DSS in one single system and to develop completely new applications. The second statement is about hardware and the progress made during the last 10 years by hardware vendors: The hardware which is available today allows us to build combined OLTP/DSS systems. Such a combination was a pure dream 20 years ago, but today the hardware is powerful enough to support such an architecture.

But from this common ground emerged two very different approaches. SAP translates the statement about the progress made by hardware vendors as: Memory has become so cheap that whole databases can fit into memory. Oracle believes that this is an oversimplification and an unforced restriction of the options which are available today. Oracle's position can be summarized as: There is so much progress in so many different areas that it does not make sense to confine ourselves to one single technology.

The fundamental difference between HANA and Oracle Database 12c, then, is the database concept. Traditionally, a database was a set of data stored on disk. For SAP, a database is a set of data that should rather reside in memory. In other words: SAP proposes a different storage medium, but still relies on the traditional concept of a database as one single „thing", which must be stored here or there as a whole. For Oracle too, a database is a set of data, but this set can be divided into subsets, which are defined by certain characteristics such as age of the data or usage patterns. And Oracle believes that for those different subsets different technologies are most appropriate.

Figure 4: Oracle's multi-tiered database architecture

In other words: The Oracle Database 12*c* In-Memory Option is one of the building blocks of a multi-tiered database architecture (see figure 4). New ILM features (see article "Oracle Database 12*c*: Information Lifecycle Management", page 46 of this volume) help customers separate "hot", "warm", and "cold" data and store them on different types of disks. Flash support allows Oracle Database 12*c* to store "very hot" data in an intermediate layer between disk and memory. And memory is divided into a row store for OLTP processing and a column store for analytical queries.

Unlike other vendors, Oracle does not require all of the data to reside in memory in order to be able to take advantage of the new in-memory capabilities. A query can execute on data that resides in-memory, on flash and on disk completely transparently, and this is exactly what enables the Oracle Database 12c In-Memory option to be used with databases and systems of any size.

Within such a framework the idea to keep all or at least as many data as possible in the in-memory column store makes no sense at all. The in-memory column store is the appropriate place for all data that are frequently accessed by analytical queries, and therefore the goal is to define this subset as precisely as possible. Or, as the author of an Oracle white paper about the Exadata Smart Flash Cache puts it: "Knowing what not to cache is of great importance to realize the performance potential of the cache."