A New Way in Business Intelligence

Why and how Oracle's In-Memory database could have the ability to change the Business Intelligence based decision support systems?

Author: Patrik Szpisják, Co-author: Dr. Levente Rádai

Óbuda University, Alba Regia Technical Faculty, Székesfehérvár, Hungary
Óbuda University, Alba Regia Technical Faculty, Székesfehérvár, Hungary
E-mail: szpisjak.patrik@hok.uni-obuda.hu, radai.levente@amk.uni-obuda.hu

Abstract
Nowadays, almost every company use various IT technics to improve their decision making systems. The most common is to use analytic technics like OLAP. After the analytic process, we have to represent the data, that's where BI comes in place. While OLAP is a single high performance analytic tool, BI is a description of all the processes, tools and methodologies about how to access the information. It contains reporting, OLAP, dashboards, score carding, DWH and other products to extract the information. Currently, BI is called the most modern way to maintain the success of a company. But is this enough?

I. Introduction

Today, it's the era of Big Data. Since the 1980s, the world's technological per-capita capacity to store information has roughly doubled every 40 months. At 2012, 2.5 Exabyte of data was created each day. That is a huge amount of information. We have to work with more and more data each day, therefore we need to change the technology in order to keep up with the expanding needs. An easy way is to change the hardware, not the amount but the technology in order to improve the performance. This was where the In memory databases came in mind [1].

II. In memory databases

In memory databases (IMDBs) are generally designed to be fast but until now, DRAMs were so expensive that it was unaffordable to move the whole database into memory for almost any company. Today, terabytes of data can be stored in memory in a cost of some thousand dollars. With the data in memory, access times has been highly reduced and performance has been greatly improved. Besides the advantages there are some complexities with this architecture. Recoverability is critical in all databases. If the machine goes down, you must not to lose any of the data. Therefore you have to write the data to the disk as well as placing it in memory. It is cost effective but doing I/O operations on disk is slow and you can hardly write fast enough to keep up with the data that is being putted into memory. The other problem with this technique is that you risk the data consistency. Since the I/O operations on disk is slow, after some time these operations will fall behind and you may lose some data when the in memory database shuts down. Mirroring could be a solution. Duplicating it on two different machines keeps the database consistent. The safety level is also higher but if the two machines go down at the same time, you lose all the data. Using the combination of these two is complex and costly. There are other ways to handle the problem like Cloud Computing but each one of them only offers partial solution [2].

III. Oracle In-Memory Database

Oracle's In-Memory Database is part of the latest released database, Oracle Database 12C. It is not a database itself, more like an extension. Oracle In-Memory Database allows you to store the data in disk and in memory at the same time without becoming inconsistent. When an insert commits, the row will be inserted traditionally to the disk. Meanwhile, if the parameter has been set on the appropriate table where the data is being inserted, another thread will pump the data into the memory. This thread only wakes up every 2 minutes in order to keep the data consistent [3].

While the data on the disk is stored in traditionally row format, In-memory it is stored in column format in order to compress it. The compression level depends on the data compress algorithm we set. Oracle offer 3 major choices. The basic compress is no compress where the data costs as much space as on the disk. Another one is for query performance which can be separated into 2 parts: for query

low and for query high. The “for query low” option makes some compression while preferring optimizing for queries. “For query high” do less compression in order to maximize the queries performance. Depending on the data itself, these two options can make a 2-10 time compression ratio.
The third one is for capacity reduction. As before, you can choose among two sub categories: for capacity low and for capacity high. The “for capacity low” makes much compression but still cares with the queries performance. “For capacity high” does the most compression of all, almost totally neglecting performance. The ratio is really high here, depending on the data, it could even reach 100-1000 times compression ratio. Since the ratio is so high, in order to use queries on the appropriate object, first the database has to decompress it. Another mentionable thing about compression is that you can not only apply it on tables but also on subset of tables. This allows you to use for capacity high on rarely needed columns and for query high on often used columns.
Oracle built in an adviser called compression adviser. This feature estimates the compression ratio of the given object by virtually compressing it.

Column format also allows the CPU to use vector processing. This format is specifically designed to maximize the column entries number that can be evaluated in a single CPU instruction. This enables to scan billions of rows per second per core instead of the capacity of the buffer cache, which can scan only millions of rows per second per core [3].

You can set yourself which object you would like to put into memory. It is possible to set this option on tablespaces, tables, columns and even parts of the columns, example on partitions too. Therefore it is highly customizable for the queries. Setting the in memory size allows you to decide what amount of data would you like to store in memory. In case the memory is full, population to the memory stops and when some space frees up in memory, it will be pumped in.

Oracle offers to use priority levels on in memory objects. The possible 5 levels are none, low, medium, high and critical. The critical data is on the highest level and the priority decreases as we go backwards. The goal is to always keep the critical data in memory, so when we “startup” the instance, these data will be populated first followed by high priorities and so on until we populate all or as many data as we can. It is important that this option cannot be used on subset of tables [4].
Recoverability did not change. Since you have all the data on disk, after shutting down and starting up, everything will recover on your disk and the appropriate ones will be populated back into the memory.
Besides reporting and ad-hoc queries, there is also an improvement in the Online Transaction Processes, called OLTPs. To increase the performance, the designer had to avoid the excessive use of indexes. The cost of maintaining indexes are high, but since we can highly customise how and what we want to store in memory, it is possible to drop the analytic indexes on the tables that is stored on the disk. Of course, it is necessary to keep the primary and foreign keys.

IV. Business Intelligence nowadays

In order to effectively grant fast access to data, we use various technics and technologies under the name of Business Intelligence. Using OLTP technics to pump the generated data to the database is fast and effective. This data is inserted on the disk in row format therefore it can be extracted slowly because the slow I/O operations of the disk. On the other hand, it is not well structured for applying analytic queries. The best way is to use an Extract Transform Load (ETL) process. During this phase, the data is being extracted from the original database, being transformed to the required format – which can be a column store format too by pivoting – and then being loaded into another database which is technically a data warehouse (DWH). A typical ETL based DWH uses multiple layers: staging, data integration and access layers usually called data marts. The staging layer stores all the data extracted from the disparate data source systems. The data integration layer transforms the data from the staging layers into data sets. These data sets are usually divided by operational systems like sales, financial and marketing. Finally this integrated data is arranged into hierarchical groups often called dimensions and into facts and aggregated facts in a typical form of star form. This is the layer from which we extract the data with various reporting tools [5].

Since each layer could be a database itself, building a BI system could cost much money by buying database licenses, support, various tools, hiring ETL specialists and database administrators. Besides money, it also costs much time. Developing a well optimized and structured BI system for reporting could take up to two years or even more. Another disadvantage is if the data warehouse shuts down, no data can be accessed which may cause huge delays in important decisions. Looking at performance, by transforming the data into the required format roughly decreases the generating time of reports, dashboards, etc. There is a real possibility of having to wait for minutes until it finally generates. This is caused by the slow I/O operations of the disk [6].

V. Oracle In-Memory Database as a real-time analytic tool

The importance of a BI system and the need of various reports, dashboards and other analytics is unquestionable. In order to keep up with the market trend changes, it is a must to improve the decision making systems by using the new IT technologies. Using Oracle In-Memory feature on the data warehouse could roughly improve the performance or in some cases it can even eliminate the need for a data warehouse which was required to achieve fast analytic execution. Yet the real power is reflected upon using it on the live transactional data while also accelerating OLTP [7]. The ETL process which loads the data warehouse runs at nights when the OLTP system is less busy in order not to affect the transactional system's performance. Therefore, the analytic queries does not represent the current state. With this feature activated, real-time ad-hoc analytics is possible, enabling leaders to get on-the-fly answers. Real-time analytics is the answer when the timeliness of the insights is important or when you need a very large influx of live data. Applying this technology on an organisation, it could detect the errors instantly, after changing business strategy changes can be noticed immediately, fraud can be detected, keeping up with customer trends becomes possible, can improve service and gives better sales insights which could lead to additional revenue [8].

In my research, I would like to measure the performance difference between a single Oracle Database and an Oracle In-Memory Database in two fields: BI and real-time analytics. On the BI field, I will use two different queries on a DWH, both with in memory disabled and with in-memory enabled. In real-time analytics, I will use two queries again to measure the performance difference on the unstructured data and the number of columns selected, as it is suggested in [9]. The compression algorithm was set to for query low during all tests.

VI. Research, testing

The first test runs on a traditional BI system which is based on a data warehouse. In order to show the difference, I used two types of analytic queries. The first query selects three columns, one from the fact table and two from the joined two dimension tables. The fact table has 13 columns, the dimension tables have 3-6 columns. The selected columns are: department name, sales price and country name. From these, sales price will be aggregated.

A. First query:

select sum(a.price), b.department_name, c.country_name from sales a, dep b, country c where a.department_id=b.department_id and a.country_id=c.country_id group by b.department_name, c.country_name order by c.country_name ASC;

On the first test, dep table contains 27 rows, country table contains 4 rows and the sales table contains about 15 million rows.

Results:

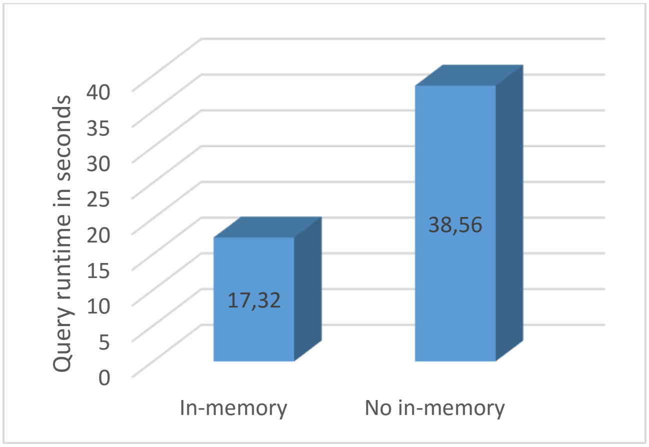

Figure 1. Using the first query: sales table has about 15 million rows

Executing the query a getting the results from the traditional database took 38.56 seconds. After enabling in-memory feature and populating the data into memory, the runtime of the same query was only 15.32 seconds. The runtime when in-memory enables is two times less when disabled.

Doubling the number of rows, the result looks like this:

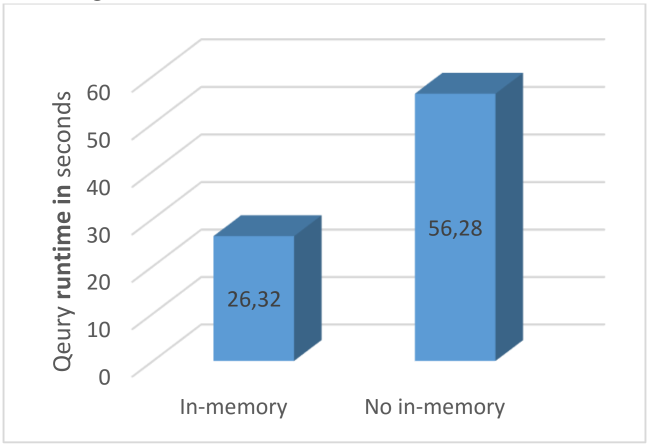

Figure 2. Using the first query: sales table has nearly 35 million rows

In this case the sales table contains nearly 35 million rows. As we see, it took more time to get the results than before. The number of rows increased the runtime, but did not double it. On the next performance test, I will increase the number of joins from two to five and the number of column from 3 to 6.

The second query selects six columns, one from the fact table and five from the joined five dimensional tables. The columns are: department name, country name, material type, payment type, sales price and region name. As before, sales price will be aggregated.

B. Second query:

select sum(a.price), b.department_name, c.country_name, d.material_type, e.payment_type, f.region_name from sales a, dep b, country c, mat d, paym e, region f where a.department_id=b.department_id and a.country_id=c.country_id and a.material_id=d.material_id and a.paym_id=e.paym_id and a.region_id=f.region_id group by b.department_name, c.country_name, d.material_type, e.payment_type, f.region_name order by c.country_name ASC;

In this test, dep table has 27 rows, country table has 4 rows, mat table has 14 rows, paym table has 6 rows, region table has 9 rows and the sales table has about 15 million rows.

Results:

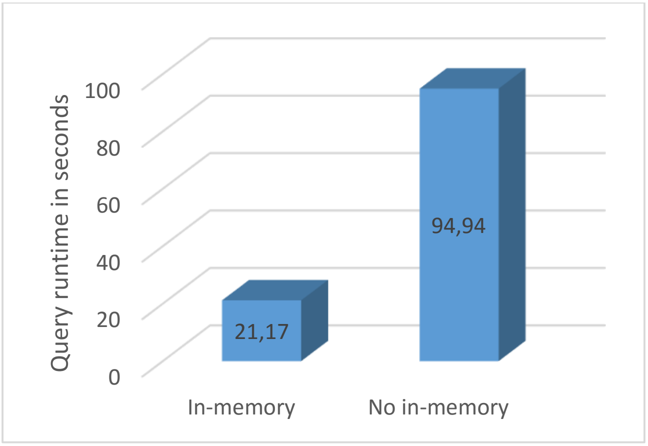

Figure 3. Using the second query: sales table has about 15 million rows

In this test, we can see that by joining more dimensional tables to the fact table, the query runtime significantly increased. This time, the query ran about 4-5 times faster when the in-memory feature was enabled and the data was populated into the memory.

We have seen before, that doubling the rows affected the performance, however joining more dimension tables increased the runtime way more.

Double the number of rows and see the results again:

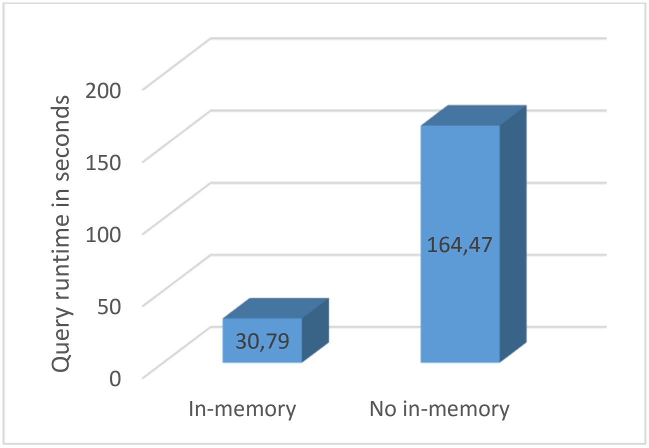

Figure 4. Using the second query: sales table has nearly 35 million rows

Generally speaking, using more joins significantly decreases the performance, while using the query on a bigger table decreases performance firmly less. On average, 2 dimensional queries are 2x faster while 5 dimensional queries are 6x faster with the in-memory feature enabled. The more complex the query, selecting the data from in-memory becomes more effective.

We can clearly say, that using data warehouses with in-memory option roughly increases performance and reduces time needed to generate various reports. In the era of Big Data, it is a powerful tool which helps in day by day decisions. It can speed up the preparing of business processes which could lead to market benefits.

The second test is represents a real-time analytics. The most important difference is that in the OLTP systems, the data is not structured well and is not transformed to the required format. Therefore, using analytic queries can cause overloads, can shut down systems which could lead to market loss. Using a powerful analytic tool which does not affect the OLTP performance is a key to real-time analytics.

The first test showed how in-memory benefits after we insert more data and do more joins. In this test, I will use two queries which will concentrate on the performance issues caused by the unstructured data and the number of columns selected. The query selects six columns, one from the fact table and five from dimension tables. The fact table has nearly 70 columns, the dimension tables have 3-11 columns. The selected columns are the same as on the other test: department name, country name, material type, payment type, sales price and region name. Like before, sales price is aggregated again. There are some differences on the table and column names, but the query is the same as in the second.

C. Third query:

select sum(a.price), b.dep_name, c.country_name, d.mat_type, e.paym_type, f.reg_name from sales a, depart b, country c, mater d, paym e, region f where a.dep_id=b.dep_id and a.country_id=c.country_id and a.mat_id=d.mat_id and a.paym_id=e.paym_id and a.region_id=f.region_id group by b.dep_name, c.country_name, d.mat_type, e.paym_type, f.reg_name order by c.country_name ASC;

In this test, depart table has 27 rows, country table has 4 rows, mater table has 14 rows, paym table has 6 rows, region table has 9 rows and the sales table has more than 90 million rows, nearly 100 million. I chose this large number of rows, because this will surely show the difference.

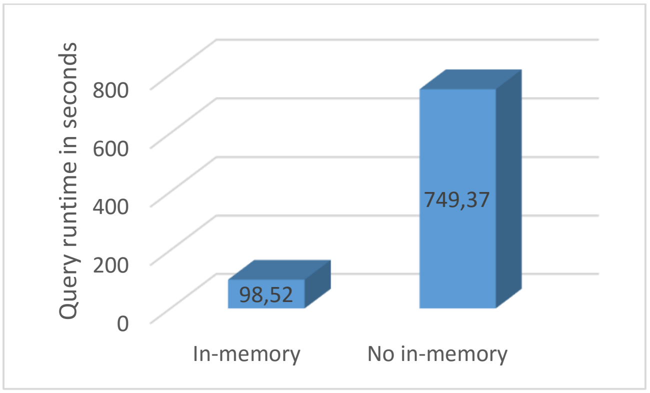

Figure 5: Using the third query on unstructured data: sales table has nearly 100 million rows, 6 columns selected

The difference is huge. By the advantage of vector processing and in-memory column store architecture, using in-memory on unstructured data improves the performance even more than before.

During the second test, I will use the same tables but I will select 18 columns instead of six.

D. Fourth query:

select sum(a.price), a.date, a.load_id, a.ref_number b.dep_name, b.dep_id, c.country_name, c.contry_id, d.mat_type, d.mat_id, d.mat_unit, d.seq_num, e.paym_type, e.paym_div, e.load_id, f.reg_name, f.code, f.id from sales a, depart b, country c, mater d, paym e, region f where a.dep_id=b.dep_id and a.country_id=c.country_id and a.mat_id=d.mat_id and a.paym_id=e.paym_id and a.region_id=f.region_id group by a.date, a.load_id, a.ref_number b.dep_name, b.dep_id, c.country_name, c.contry_id, d.mat_type, d.mat_id, d.mat_unit, d.seq_num, e.paym_type, e.paym_div, e.load_id, f.reg_name, f.code, f.id order by c.country_name ASC;

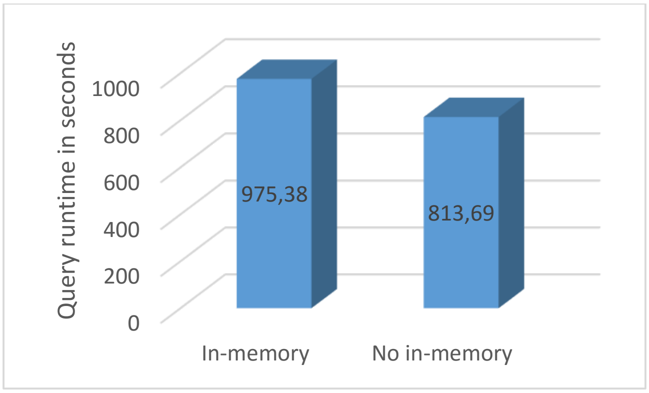

Figure 6: Using the fourth query on unstructured data: sales table has nearly 100 million rows, 18 columns selected

This is where in-memory column store architecture is not effective. Upon selecting a larger number of columns, the performance highly drops. This is caused by the in-memory column store architecture which was designed to select only a few columns.

Generally speaking, in order to benefit from Oracle In-Memory Database we have to keep in mind that selecting large number of columns and returning large number of rows can cause huge performance loss. Furthermore, using complex predicates and joining multiple large tables has bad impact on performance. Therefore it is reasonable to look into the details of tests and create a performance difference map based on the number of rows and columns.

VII. Summary

Oracle In-Memory Database has the ability to change the regular BI concept. Being able to store the required data in-memory for real-time ad-hoc analysing while also accelerating OLTP on disk is powerful. Performance benefits can be also seen at aggregation, at filtering and at BI type reporting which typically summarises a lot of data. Nevertheless, in most cases, if we need the data instantly from huge tables it does not enough, we still need aggregated tables. Therefore, if the organization creates and stores a huge amount of data using this feature instead of a data warehouse is not recommended. The real power shows upon using it on a single database for real-time ad-hoc querying. Depending on the settings and on the data, queries could be executed a hundred times faster or even more. At last, having the chance to use one product for multiple purposes offers great support possibility and prevents problems which can appear when integrating products from separate organizations.

REFERENCES

[1] István Orosz, Tamás Orosz, Company level Big Data Management, In: Anikó Szakál (szerk.), 9th IEEE International Symposium on Applied Computational Intelligence and Informatics ; SACI 2014. Conference place, date: Timisoara, Románia, 2014.05.15-2014.05.17. (IEEE), Timisoara: IEEE Hungary Section, 2014. pp. 209-303.

[2] A Selmeci, T Orosz, Usage of SOA and BPM changes the roles and the way of thinking in development, In: Szakál A (szerk.), 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012, Subotica, 2012, September, 20-22. Konferencia helye, ideje: Subotica, Szerbia, 2012.09.20-2012.09.22. Piscataway: IEEE, 2012. pp. 265-271.

[3] Oracle Database In-Memory, Oracle Database 12c, Oracle White Paper, Oracle Corporation, July 2015, URL: http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html, p: 29, Downloaded: 15th September, 2015.

[4] Oracle Database In-Memory, Powering the Real-Time Enterprise, Oracle Data Sheet, Oracle Database 12c, Oracle Corporation, URL: http://www.oracle.com/technetwork/database/options/database-in-memory-ds-2210927.pdf, p: 9, Downloaded: 15th September, 2015.

[5] Tamás Orosz, István Orosz, Optimization of Enterprise Business Processes with Big Data Management, BULETINUL STIINTIFIC AL UNIVERSITATII POLITEHNICA DIN TIMISOARA ROMANIA SERIA AUTOMATICA SI CALCULATORAE / SCIENTIFIC BULLETIN OF POLITECHNICA UNIVERSITY OF TIMISOARA TRANSACTIONS ON AUTOMATIC CONTROL AND COMPUTER SCIENCE 59:(2) pp. 105-110. (2014)

[6] A. Selmeci, I. Orosz, Gy. Györök, T. Orosz, Key Performance Indicators Used in ERP Performance Measurement Applications, In: Szakál A (szerk.), 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012, Subotica, 2012, September, 20-22. Konferencia helye, ideje: Subotica, Szerbia, 2012.09.20-2012.09.22. Piscataway: IEEE, 2012. pp. 43-48.

[7] L. Ellison and J. Loaiza, Oracle Database In-Memory - Powering the Real-Time Enterprise, Oracle YouTube Channel, URL: https://www.youtube.com/watch?v=CjkPUg1m6PA, Published: 13th June, 2014.

[8] C. Millsap, K. Osborne and T. Poder, Oracle Database InMemory in Action, URL: https://www.youtube.com/watch?v=Ze3umRJ_HBM, Published: 12th November, 2014.

[9] M. Rittman, In-Memory Analytics Speeds Business Intelligence, Oracle Magazine, January/February 2015, URL: http://www.oracle.com/technetwork/issue-archive/2015/15-jan/o15ba-2398995.html, Downloaded: 15th September, 2015.