DE GRUYTER OPEN

Print ISSN: 1312-2622; Online ISSN: 2367-5357
DOI: 10.1515/itc-2018-0008

Large Data Storing and Processing Approach

K. Shvertner

Key Words: *Increasing DB response time; In-Memory Databases; Data Base machines; Exadata DB machine; Hana DB; Oracle In-Memory DB.*

Abstract. *The growth of data volumes and the need for fast analytical data processing lead to significant efforts to answer effectively these major challenges. It is known that Data Processing plays key role and is the leading motivation for the research and development in software and hardware technologies. Now there are new trends in databases technologies: large data processing and in-memory option in the database functionality. The In-Memory functionality leads to new approaches in the data processing algorithms and database architecture. The paper observes also big improvements in the hardware: complex engineered data base machines (Exadata, Exalytics) and even special enhancements in the processor architecture (SPARC M7). Also for first time a new database was designed and successfully implemented in Europe – the SAP HANA.*

Introduction

Most data collected by organizations used to be transaction data that could easily fit into rows and columns of relational database management systems (DBMS). From other hand the traditional tool for analyzing corporate data for the past two decades has been the data warehouse. A data warehouse is a database that stores current and historical data of potential interest to decision makers throughout the company. The volume of these data is very large.

In the contemporary Data Processing another types of processing arts become also quite common. One is the CRM (Customer Relationship Management – needs high speed retrieval of customer data). The second are high access speed operational data used by fire services, police departments etc. (high speed retrieval of massive data chunks). The third are control and military systems where the reaction time plays vital role. These challenges lead to new approaches in data processing.

One way is to use database machines with autonomous intelligence oriented to do fast data retrieval (part 3). Another way are the in-memory options of the databases. Both approaches combine columnar storage of the table data, advanced columnar data compression (part 4, 5). It is important to mention that the some of the contemporary computer processors have functionalities that allows search in enormous big data chunks if the data is stored sequential in the RAM.

In the past the data repository resides on hard disk devices. The new way of facilitating large data analysis is to use in-memory computing, which relies primarily on a computer's main memory (RAM) for data storage. The applications programs access data stored in system primary memory, thereby eliminating bottlenecks from retrieving and reading data in a traditional, disk-based database and dramatically shortening query response times. In-Memory processing makes it possible for very large sets of data, amounting to the size of a data mart or small data warehouse, to reside entirely in memory. Complex business calculations that used to take hours or days are able to be completed within seconds, and this can even be accomplished using handheld devices.

Now there are advances in contemporary computer hardware technology that make In-Memory processing possible, such as powerful high-speed processors, multicore processing, and falling computer memory prices. These technologies help companies optimize the use of memory and accelerate processing performance while lowering costs.

Leading commercial products for In-Memory computing include SAP HANA and Oracle In-Memory option. Each provides a set of integrated software components, including In-Memory database software and specialized analytic software, run on hardware optimized for In-Memory computing work.

The article is an introduction review of the new architecture and new functionalities of the most complex systems in the IT – the Data Bases. In next article we will report more technical data, observations and comparisons of the different approaches.

1. Basic Concepts of In-Memory Technology

In-Memory means that the data (or the most frequently accessed data) reside in the main memory (RAM) of the database server. The basic idea is that memory is much faster than disk, actually it is times faster. A 2014 CPU has a memory bandwidth of 10 GByte/sec and higher, a single disk around 150 MByte/sec – difference by factor 70. If the program is using the same memory frequently, it is cached inside the CPU's L1 or L2 cache, speeding up the memory bandwidth by another factor of 10. On the contrary the disk speed of 150 MB/sec is for sequential read access only, random access is times worse for a disk system whereas has no negative impact on the RAM.

The downside of memory is the costs of the memory chip itself (7 USD/GByte for RAM compared to 0.05 USD/GByte for disks as of 2014) and the hardware platform you need, in order to cope with more memory, is getting increasingly more expensive also.

On the other hand, if the company need 1TB of RAM that would be 7000 USD. While this is much money compared to a single 100 USD disk, it is not much in terms of absolute numbers.

So from expenses point of view the hard disk memory is preferable, but from the data processing speed point of view the RAM based data have preference. Nothing new –

this is known fact in the computer science! The only news is that nowadays the big amounts of RAM are affordable for the shops.

2. Early and Alternative Variants of In-Memory

The system administrators and designers always has tried to use the RAM for keeping of data (caching).

All UNIX-like systems analyze the unused portion of the RAM and store data about size and location of the free RAM extents. If such free extents exist the OS are placing there temporarily the most frequently accessed blocks from the disks without to distinguish blocks needed for the OS and blocks needed for the applications. So the number of the physical read and write operations decrease. But if there are need of RAM this area is emptied and the blocks are stored back to the disk without any possibility to manage this process. Nevertheless the RAM is used in the whole extent and the physical access to the disks decreases which increases the speed of the data processing.

Another approach used in the databases is to keep the frequently used data files in the RAM (if there are enough free extents). The association of the files to the applications (the so called table spaces) allows to distinguish the files belonging to different applications and open the possibility to place some files in the RAM instead on disks.

On table level of the databases it is possible the frequently used tables to be placed in the RAM (in Oracle this is the SGA – System Global Area). Additionally in the RAM is possible to keep the results of the data processing (in Oracle – Result Sets). So if the data in the tables after the last processing are not changed the application is provided with the last result of the data processing instead of repeating the slow and expensive process of data retrieval.

3. New Challenges in Data Processing, New Ways

Small business works with small volume of data, the big business – big volume of data, the continental or global business – an ocean of data and gigantic data processing. To be compliant the needs of the big businesses the supplier of the data bases are trying to develop adequate software tools, in some cases also with hardware appliances in which are united all hardware and network components for the data processing.

These appliances comprises processors, RAM, disk memory with enhanced advanced architecture, some network circles (one for superfast inter component communication, another for the applications access), integrated storage chips (cache) with very fast access (approaching the speed of the RAM) for caching the most frequently used disk blocks. In these appliances the notion disk is replaced with the notion intelligent cell – a set of 12 classical disk devices, which works as one unit and poses intelligence provided by two powerful Intel Xeon processors. The intelligence has many functions (automatic compression of the disc data, automatic indexing of part of the tables and automated support of different indexes over different parts of a single table, automatic caching in the in the cache memory the most frequently read disk data blocks).

The most important advantage of these appliances is that the data processing is done in the disc cells from the mentioned processors. So is avoided the transfer of the data via the network the disk blocks to the RAM and main processors. Instead a special selective protocol is used (known as iSQL) and the data base is provided with retrieved data and results. This significantly lowers the data transfer by the processing (from one hand because the work of additional processors in the disc cell, from other hand because the parallel usage all the discs in one cell). This technology is implemented in Oracle Exadata, but it is used from other hardware providers. The big disadvantage of such appliances is the big price and that it is hard to expand such systems.

Another approach are the In-Memory databases. The basic ideas here are many, but some of them are important. The In-Memory databases give the possibility to place in the RAM whole tables (and in some cases part of the columns of some tables) and to leave other tables (in some cases other table columns) on the discs. This choice is a new challenge to the system administrators because they need to know in details the application, the data structure and the data processing. From other side they need to know in details the hardware landscape.

The paper presents some features of the In-Memory database HANA of the German Company SAP and Oracle. Other providers like IBM и Microsoft have also In-Memory Data Bases but SAP HANA and Oracle are the most common.

4. SAP HANA In-Memory Database

The SAP HANA database supports two types of table: those that store data either column-wise (column tables) or row-wise (row tables). SAP HANA is optimized for **column storage**.

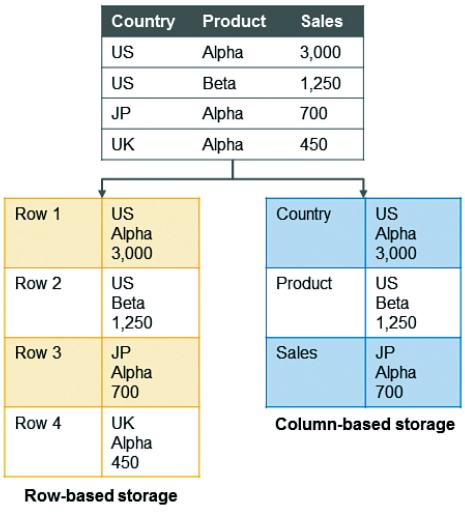

Figure 1. Principle of Row- and Column-Based Storage for a Table

Conceptually, a database table is a two dimensional data structure with cells organized in rows and columns. Computer memory however is organized as a linear sequence. For storing a table in linear memory, two options can be chosen as shown below. A row store stores a sequence of records that contains the fields of one row in the table. In a column store, the entries of a column are stored in contiguous memory locations.

In the SAP HANA database, tables that are organized in columns are optimized for high-performing read operations while still providing good performance for write operations. Efficient data compression is applied to save memory and speed up searches and calculations. Furthermore, some features of the SAP HANA database, such as partitioning, are available only for column tables. Column-based storage is typically suitable for big tables with bulk updates. However, update and insert performance is better on row tables. Row-based storage is typically suitable for small tables with frequent single updates.

The approach to highlight the criteria to be used to decide whether a table should be a colon or a sequence is as follows:

• Column type storage – when to use: Calculations are typically executed on individual or a small number of columns; The table is searched based on the values of a few columns; The table has a large number of columns; The table has a large number of rows and columnar operations are required (aggregate, scan, and so on High compression rates can be achieved because the majority of the columns contain only a few distinct values (compared to the number of rows).

• Row type storage – when to use: The application needs to process only one single record at one time (many selects and/or updates of single records); The application typically needs to access the complete record; The columns contain mainly distinct values so compression rate would be low; Neither aggregations nor fast searching are required; The table has a small number of rows (for example, configuration tables).

It is important to note, that the SAP HANA database allows row tables to be joined with column tables. However, it is more efficient to join tables of the same storage type. It is possible to change an existing table from one storage type to the other (ALTER TABLE ALTER TYPE).

SAP HANA supports **history tables** which allow queries on historical data (also known as time-based queries).

History tables are special database tables that only allow inserts. Write operations on history tables do not physically overwrite existing records. Instead, write operations always insert new versions of the data record into the database. The most recent versions in history tables are called current data. All other versions of the same data object contain historical data. Each row in a history table has timestamp-like system attributes that indicate the time period when the record version in this row was the current one. Historical data can be read by requesting the execution of a query against a historical view of the database (SELECT ... AS OF time).

Memory management – the column store is the part of SAP HANA database that manages data organized in columns in memory. Tables created as column tables are stored here.

The column store is optimized for read operations but also provides good performance for write operations. This is achieved through 2 data structures: main storage and delta storage.

The main storage contains the main part of the data. Here, efficient data compression is applied to save memory and speed up searches and calculations. Write operations on compressed data in the main storage would however be costly. Therefore, write operations do not directly modify compressed data in the main storage. Instead, all changes are written to a separate data structure called the delta storage. The delta storage uses only basic compression and is optimized for write access. Read operations are performed on both structures, while write operations only affect the delta.

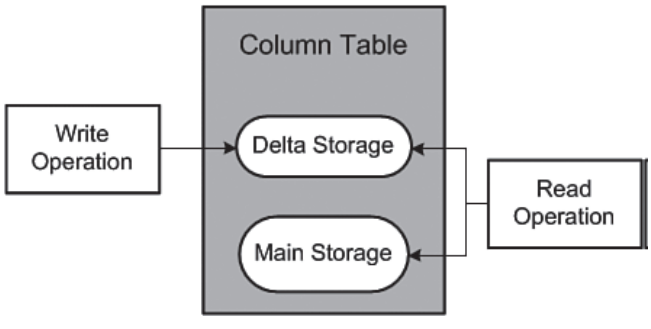

Figure 2. Main Storage and Delta Storage

The purpose of the delta merge operation is to move changes collected in the delta storage to the read-optimized main storage. After the delta merge operation, the content of the main storage is persisted to disk and its compression recalculated and optimized if necessary.

A further result of the delta merge operation is truncation of the delta log. The delta storage structure itself exists only in memory and is not persisted to disk. The column store creates its logical redo log entries for all operations executed on the delta storage. This log is called the delta log. In the event of a system restart, the delta log entries are replayed to rebuild the in-memory delta storages. After the changes in the delta storage have been merged into the main storage, the delta log file is truncated by removing those entries that were written before the merge operation. As only data in memory is relevant, the load status of tables is significant. A table can have one of the following load statuses:

• Unloaded, that is, none of the data in the table is loaded to main memory.

• Partly loaded, that is, some of the data in the table is loaded to main memory, for example, a few columns recently used in a query.

• Fully loaded, that is, all the data in the table is loaded into main memory.

The SAP HANA database aims to keep all relevant data in memory. Standard row tables are loaded into memory when the database is started and remain there as long as it is running. They are not unloaded. Column tables, on the other hand, are loaded on demand, column by column

when they are first accessed. This is sometimes called lazy loading. This means that columns that are never used are not loaded and memory waste is avoided.

This is the default behavior of column tables. In the metadata of the table, it is possible to specify that individual columns or the entire table are loaded into memory when the database is started.

The database may actively unload tables or individual columns from memory, for example, if a query or other processes in the database require more memory than is currently available. It does this based on a least recently used algorithm.

The Delta Merge operation – Write operations are only performed on the delta storage. In order to transform the data into a format that is optimized in terms of memory consumption and read performance, it must be transferred to the main storage. This is accomplished by the delta merge operation.

Figure 3 shows the different steps in the merge process, which objects are involved, and how they are accessed.

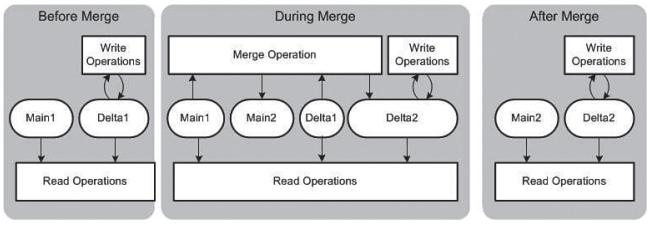

Figure 3. The Delta Merge Process

Before the merge operation, all write operations go to Delta 1 storage and all read operations read from Main 1 and Delta 1 storages.

While the merge operation is running, the following happens:

• All write operations go to the second delta storage, Delta 2.

• Read operations read from the original main storage, Main 1, and from both delta storages, Delta 1 and Delta 2.

• Uncommitted changes in Delta1 are copied to Delta 2.

• The content of Main 1 and the committed entries in Delta 1 are merged into the new main storage, Main 2.

After the merge operation has completed, the following happens:

• Main 1 and Delta 1 storages are deleted.

• The compression of the new main storage (Main 2) is reevaluated and optimized. If necessary, this operation reorders rows and adjust compression parameters. If compression has changed, columns are immediately reloaded into memory.

• The content of the complete main storage is persisted to disk.

With this double buffer concept, the table only needs to be locked for a short time: at the beginning of the process when open transactions are moved to Delta2, and at the end of the process when the storages are "switched".

The minimum memory requirement for the delta merge operation includes the current size of main storage plus future size of main storage plus current size of delta storage plus some additional memory. It is important to understand that even if a column store table is unloaded or partly loaded, the whole table is loaded into memory to perform the delta merge. The performance of the delta merge depends on the size of the main storage. This size can be reduced by splitting the table into multiple partitions, each with its own main and delta storages.

The standard method for initiating a merge in SAP HANA is the **auto merge**. A system process called mergedog periodically checks the column store tables that are loaded locally and determines for each individual table (or single partition of a split table) whether or not a merge is necessary based on configurable criteria (for example, size of delta storage, available memory, time since last merge, and others).

If an application powered by SAP HANA requires more direct control over the merge process, SAP HANA supports a function that enables the application to request the system to check whether or not a delta merge makes sense now. This function is called **smart merge**. For example, if an application starts loading relatively large data volumes, a delta merge during the load may have a negative impact both on the load performance and on other system users. Therefore, the application can disable the auto merge for those tables being loaded and send a "hint" to the database to do a merge once the load has completed.

The database can trigger a **critical merge** in order to keep the system stable. For example, in a situation where auto merge has been disabled and no smart merge hints are sent to the system, the size of the delta storage could grow too large for a successful delta merge to be possible. The system initiates a critical merge automatically when a certain threshold is passed.

The delta merge operation for column tables is a potentially expensive operation and must be managed according to available resources and priority. This is the responsibility of the **merge monitor**.

The system uses cost functions to decide which table to merge, when, and in which order. There are also cost functions that control how many tables are merged at the same time and how many threads are used to merge a single table.

The merge monitor is responsible for controlling all merge requests for all column tables on a single host. In a distributed system, every index server has its own merge monitor.

Data compression in the column store – The column store allows for the efficient compression of data. This makes it less costly for the SAP HANA database to keep data in main memory. It also speeds up searches and calculations. Data in column tables can have a two-fold compression:

• Dictionary compression – This default method of compression is applied to all columns. It involves the mapping of distinct column values to consecutive numbers, so that instead of the actual value being stored, the typically much smaller consecutive number is stored.

• Advanced compression – Each column can be further compressed using different compression methods, namely

prefix encoding: run length encoding (RLE), cluster encoding, sparse encoding, and indirect encoding. The SAP HANA database uses compression algorithms to determine which type of compression is most appropriate for a column.

Advanced compression is applied only to the main storage of column tables. As the delta storage is optimized for write operations, it has only dictionary compression applied. Compression is automatically calculated and optimized as part of the delta merge operation.

Table partitioning – The partitioning feature of the SAP HANA database splits column-store tables horizontally into disjunctive sub-tables or partitions. In this way, large tables can be broken down into smaller, more manageable parts. Partitioning is typically used in multiple-host systems, but it may also be beneficial in single-host systems. When a table is partitioned, the split is done in such a way that each partition contains a different set of rows of the table. There are several alternatives available for specifying how the rows are assigned to the partitions of a table, for example, hash partitioning or partitioning by range. The following are the typical advantages of partitioning:

• Load balancing in a distributed system – Individual partitions can be distributed across multiple hosts. This means that a query on a table is not processed by a single server but by all the servers that host partitions.

• Overcoming the size limitation of column-store tables – A non-partitioned table cannot store more than 2 billion rows. It is possible to overcome this limit by distributing the rows across several partitions. Each partition must not contain more than 2 billion rows.

• Parallelism – Partitioning allows operations to be parallelized by using several execution threads for table.

• Partition pruning – Queries are analyzed to determine whether or not they match the given partitioning specification of a table. If a match is found, it is possible to determine the actual partitions that hold the data being queried. Using this method, the overall load on the system can be reduced, thus improving the response time.

• Improved performance of the delta merge operation which depends on the size of the main index. If data is only being modified on some partitions, fewer partitions will need to be delta merged and therefore performance will be better.

• Explicit partition handling - Applications may actively control partitions, for example, by adding partitions to store the data for an upcoming month.

Disadvantages of SAP HANA – it has traditionally been a pure In-Memory database, which limited HANA to databases that could be fully loaded into memory. This meant that HANA could not support very large databases where the size exceeded the amount of memory available. This has been a source of a large number of complaints from HANA customers over the need for a smooth transition within HANA to non-volatile storage. SAP partially addressed this problem with the dynamic tiering option and support for "extended tables" released in late 2014. Extended tables are disk-based structures that allow HANA to access data on disk as well as in memory but currently there is no concept of partitioning, and extended tables are discrete objects from in-memory tables, which must be managed by applications.

More recently, their Data Lifecycle Management (DLM) tool can be used for moving data between memory, disk and Hadoop/SybaseIQ based on specified rules and policies. This solved the large database problem, but introduced other issues. Besides the issue that extended tables will be very slow to access, use of extended tables is also not transparent to the application. The logic to manage data tiering and access for extended tables may need to be added to every application that accesses data in a large database.

HANA does have support for row based tables, but the performance emphasis is on columnar based tables and this is the format that all of HANA's performance features operate on. This is a significant issue in HANA, as a table is either row based or it is column based; it cannot be both simultaneously.

So although row based tables are supported, HANA is primarily a columnar database and SAP emphasizes that most tables should be columnar based. This implies that a single data format is suitable for both OLTP and analytics.

Organizations can deploy HANA on premise after purchasing an SAP-certified hardware appliance, or on a public cloud from SAP, Amazon, Microsoft, IBM, and several other cloud providers. The processors used are Intel Xeon. The OS is Linux. So it is not possible to claim that SAP Hana is a multiplatform DB.

5. Oracle Database In-Memory

Oracle database In-Memory is available in Oracle database 12c Enterprise Edition as additionally paid option. This means that the database can be used without the In-Memory feature. The Oracle database 12c has variants for almost all industrial used operating systems and processor types and do not need special certified appliance to be run. The only exception is the option to use the so called SQL in Silicon: the SPARC M7 microprocessor, specifically engineered for optimal performance for the database In-Memory.

Dual-format architecture – Oracle database In-Memory optimizes analytics and mixed workload OLTP, delivering outstanding performance for transactions while simultaneously supporting real-time analytics, business intelligence, and reports.

This breakthrough capability is enabled by the dual-format architecture of Oracle database In-Memory. Up to now, databases have forced users to store data in either column or row format. Column format is highly efficient for analytics, but imposes very large overheads when used in OLTP environments. Similarly, row format enables extremely fast OLTP, but is less optimized for analytics. The only way to optimize for both OLTP and analytics has been to copy data from OLTP systems to analytic systems using complex ETL processes that add a great deal of expense and latency.

The dual-format architecture of Oracle database In-Memory eliminates this tradeoff by representing tables simultaneously using traditional row format and a new in-memory column format.

The Oracle SQL Optimizer automatically routes analytic queries to the column format and OLTP queries to the row format, transparently delivering best-of both-worlds performance. Oracle database 12c Release 2 (12.2) automatically maintains full transactional consistency between the row and the column formats, just as it maintains consistency between tables and indexes.

The new column format is a pure in-memory format. Tables are stored on disk using Oracle's existing row-based or hybrid columnar formats. Since there is no persistent columnar storage format, there are no additional storage costs or storage synchronization issues. Changes to the purely in-memory column format are very fast because they don't need expensive persistent logging.

Having both a column and a row-based in-memory representation does not double memory requirements. Oracle uses its highly optimized buffer cache management algorithms to keep only actively accessed row data in memory. Decades of experience has shown that caching a small percentage of data blocks in memory eliminates the vast majority of storage I/Os, and flash caching eliminates virtually all the rest. Therefore most of the memory capacity in a database server can be allocated to the column format.

Oracle's in-memory column format uses sophisticated compression to expand memory capacity and improve query performance. Compression ratios vary from 2X – 20X, depending on the option chosen and redundancy in the data. The compression method may be different across columns, partitions or tables. For example, some table partitions can be optimized for scan speed, others for memory footprint, while others may be optimized to efficiently handle frequent DML operations.

Comprehensive In-Memory optimizations – Oracle implements state-of-the-art algorithms for in-memory scans, in-memory joins, and in-memory aggregation. Tables are logically split into sections, and minimum and maximum values of every column are maintained for every section of a table. This allows queries to quickly skip table sections that only contain data outside of the range of data needed by the query. Modern microprocessors support SIMD (Single Instruction for Multiple Data values) vector processing instructions to accelerate graphics and scientific computing. Oracle can use these SIMD vector instructions to process multiple column values in a single CPU clock cycle. In-Memory table joins take advantage of the new columnar compressed format by converting join conditions into filters applied during very fast data scans or by executing the join on the compressed values within the join columns. Analytic workloads typically spend a considerable amount of time on expression evaluation for each row returned by a query. In-Memory expressions enable frequently evaluated expressions to be materialized into the In-Memory column store. Once in the column store, all in-memory optimizations seamlessly apply to the expressions: vector processing, storage index pruning, etc. The expression columns are also maintained automatically as the underlying table columns are updated. In-Memory aggregation algorithms leverage the column format to speed up analytic queries and reports that aggregate large amounts of data.

Cost effective in-Memory processing for any database size – Oracle database In-Memory does not require all database data to fit in memory. Users can choose to keep only performance sensitive tables or partitions in memory. Less performance sensitive data can reside on much lower cost flash or disk. Queries execute transparently on data residing on all three tiers – memory, flash and disk – enabling Oracle database In-Memory to be used with databases of any size.

6. Practical Experience

Oracle Since the last months of 2017 we take part in migration for the data warehouses of Health Insurance companies from OLTP oriented Oracle instances to Hana DB running with SAP applications. The migration tool is provided by SAP and works fine and reliable. The response time of the applications on Hana DB reduces and the customers are satisfied. The operations run smoothly and without big problems. We faced two disadvantages of Hana DB. The first one is the high price of Hana and there are a lot of complains about this in the net. The second is the shortage of experienced DBA staff to operate Hana. In the same time there are articles confirming that at least one big company do migrations from Hana to Oracle In-memory.

Conclusion

In the second decade of this century the database got significant improvements in their functionality. Even more – there are new trends in the balance of the usage of RAM and hard disk memories. The attempt to place massive parts of the databases in the RAM mature and is now almost standard feature of the databases implemented in thousands of shops. These leads to new approaches in the design of the applications and rise new challenges for hardware, operating systems and database administrators. The architectural and functional changes are significant and need knowledge to operate them successful.

References

1. An Oracle White Paper, Analysis of SAP HANA High Availability Capabilities, 2014.
2. Mitschang, B. et al. (Hrsg.). Datenbanksysteme für Business, Technologie und Web (BTW 2017). Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, 2017, 545 SAP HANA – The Evolution of an In-Memory DBMS from Pure OLAP Processing towards Mixed Workloads, 2017.
3. Kyte, T. On Oracle Database In-Memory. – *Oracle Magazine*, January/February 2015.
4. All databases are in-memory now…aren't they, 2014.
5. Oracle Database In-Memory. Powering the Real-Time Enterprise, 2014.
6. SAP HANA Administration Guide, 2016.
7. Pilev, D. Digital Signing of Data in the Web-based Information Systems. – *Information Technologies and Control*, 14, 2016, No. 1.

Manuscript received on 05.02.2018

Prof. ***Krassimira Shvertner*** *(PhD) works in Sofia University. Her research is focused on databases, database-as-a-service, database security, database in-memory, engineered database appliances. She has published more than 100 papers.*

Contacts:
Faculty of Economics and Business Administration
Sofia University
125 Tsarigradsko chaussee, bl. 3,
1113 Sofia
e-mail: shvertner@feb.uni-sofia.bg