2022 IEEE 16th International Scientific Conference on Informatics

INFORMATICS 2022

November 23–25, 2022, Poprad, Slovakia

PROCEEDINGS

Editors
William Steingartner
Štefan Korečko
Anikó Szakál

Organized by

Slovak Society for Applied Cybernetics and Informatics (SSAKI), affiliated branch at Department of Computers and Informatics FEEI TUKE

Faculty of Electrical Engineering and Informatics, Technical University of Košice

IEEE SMCS Technical Committee on Computational Cybernetics

IEEE Computational Intelligence (CI) Society Chapter of Czechoslovakia Section

ISBN 979-8-3503-1032-0
IEEE catalog number CFP22E80-PRT

2022 IEEE 16th International Scientific Conference on Informatics

IEEE Catalog Number CFP22E80-PRT
ISBN 979-8-3503-1032-0

Additional copies of this publication are available from
Curran Associates, Inc.
57 Morehouse Lane
Red Hook, NY 12571 USA
+1 845 758 0400
+1 845 758 2633 (FAX)
email: curran@proceedings.com

Preface

It is a great privilege and a pleasure for us to present the proceedings of the 2022 IEEE 16th International Scientific Conference on Informatics to the authors and delegates of the event. We hope that all readers will find it useful, exciting and inspiring. Its success results from our improvement efforts to publish with higher standards in various areas of computer science and related fields. The ever-changing scope and rapid development of computer science and technologies create new problems and questions, resulting in the real need for sharing brilliant ideas and stimulating good awareness of this important research field. Within this, our conference has become an important international forum for academic scientists, engineers, researchers and young IT experts to exchange and share their experiences and research results on most aspects of science and social research, discuss practical challenges and solutions. adopt and bring new ideas.

This conference proceedings collection includes papers covering the research work submitted and accepted for the 16th edition of the International Scientific Conference on Informatics.

The topics of this conference cover theoretical and practical results, along with methods for transferring these research results into real-life domains, by scientists and experts working in computer science and computing-related fields. The role of the conference is also to provide an opportunity for young researchers to demonstrate their achievements and to discuss their results at an international scientific forum.

The main topics of the conference are the following:

- Applied Computer Science
- Artificial Intelligence
- Computer Networks and Telecommunication
- Data and Knowledge Bases
- Education and Learning Analytics
- Human-Computer Interaction
- New Trends in IT Security
- Programming Languages and Programming Paradigms
- Smart Technologies
- Software Engineering
- Theoretical Computer Science
- Virtual Reality and Computer Games

This year we also have a pleasure to host the "Workshop on Head-mounted VR-based BCI" that focuses on utilization of brain-computer interfaces, virtual reality, artificial intelligence and results from other related fields for neurorehabilitation.

The conference is co-organized by

- Faculty of Electrical Engineering and Informatics, Technical University of Košice,
- Slovak Society for Applied Cybernetics and Informatics (SSAKI), affiliated branch at Department of Computers and Informatics FEEI TUKE,
- IEEE SMCS Technical Committee on Computational Cybernetics and
- IEEE Computational Intelligence (CI) Society Chapter of Czechoslovakia Section.

The conference takes place on November 23rd-25th, 2022, in Poprad, the largest city of the historical region of Spiš, situated right in its center. Poprad is also an entrance to the beautiful High Tatras mountain range.

In total, around 80 articles were submitted for the conference by the authors. All submitted papers have been peer-reviewed by independent external referees, and acceptance is based on the quality and relevance of the research. In the end, 60 papers were accepted and recommended for presentation.

In this year's conference, three invited keynote speakers will present an overview of their successful scientific results to date. All invited lecturers are renowned scientific personalities in their fields of research.

We would like to thank the organization team, the members of the program committees and reviewers. They have worked very hard to review to papers, providing their opinions and making valuable suggestions for the authors to improve their works and helped us maintain the high quality of the manuscripts included in the proceedings published by IEEE. We also would like to express our gratitude to the external reviewers, for providing extra help in the review process, and the authors for contributing their research result to the conference. Finally, we would also like to extend our thanks to all the authors and keynote speakers, who contributed and guaranteed the high and professional standard of this conference.

We also thank the sponsors

- IEEE Hungary Section,
- IEEE SMC Chapter, Hungary,
- IEEE Joint IES/RAS Chapter, Hungary

and the technical co-sponsor

- IEEE SMC Society.

Let us wish that all the participants of 2022 IEEE 16th International Scientific Conference on Informatics will have a wonderful and fruitful time at the conference, and that our guests will enjoy their stay in Poprad and in High Tatras in this beautiful November. We look forward to seeing you all at the next conference event.

Poprad, November 2022

On behalf of the Program and Organizing Committees
William Steingartner

Committees

General Chair

Liberios Vokorokos, *Dean of Faculty of Electrical Engineering and Informatics, Technical University of Košice (SK)*

Honorary Chair

Mikuláš Alexík, *University of Žilina (SK)*

Honorary Program Chair

Valerie Novitzká, *Technical University of Košice (SK)*

Program Chair

William Steingartner, *Technical University of Košice (SK)*

Program Committee

Miklós Bartha, *Memorial University of Newfoundland (CA)*
Fatma Bozyiğit, *University of Antwerp (BE)*
Jan Čapek, *University of Pardubice (CZ)*
Zbigniew Domański, *Częstochowa University of Technology (PL)*
Erik Duval, *Katholieke Universiteit Leuven (BE)*
Lucía de Espona Pernas, *University of Applied Sciences and Arts Northwestern Switzerland (CH)*
Dimitar Filev, *Ohio State University (US)*
Zoltán Fülöp, *University of Szeged (HU)*
Gianina Gábor, *University of Oradea (RO)*
Darko Galinec, *Zagreb Universtity of Applied Sciences (HR)*
Ján Genči, *Technical University of Košice (SK)*
Andrzej Grzybowski, *Częstochowa University of Technology (PL)*
Tamás Haidegger, *Óbuda University, Budapest (HU)*
Zdeněk Havlice, *Technical University of Košice (SK)*
Pedro Rangel Henriques, *University of Minho, Braga (PT)*
Pavel Herout, *University of West Bohemia, Pilsen (CZ)*
Ladislav Hluchý, *Slovak Academy of Sciences, Bratislava (SK)*
Elke Hochmüller, *Carinthia University of Applied Sciences (AT)*
László Horváth, *Óbuda University, Budapest (HU)*
Zoltán Horváth, *Lóránd Eötvös University, Budapest (HU)*
Ladislav Huraj, *University of Ss. Cyril and Methodius in Trnava (SK)*
Péter Kádár, *Óbuda University, Budapest (HU)*
Waldemar W. Koczkodaj, *Laurentian University (CA)*
Štefan Korečko, *Technical University of Košice (SK)*
Levente Kovács, *Óbuda University, Budapest (HU)*
Michal Kvet, *University of Žilina (SK)*
Sandra Lovrenčić, *University of Zagreb, Varaždin* (HR)
Ivan Luković, *University of Belgrade (RS)*
Dragan Mašulović, *University of Novi Sad (RS)*
Karol Matiaško, *University of Žilina (SK)*
Marjan Mernik, *University of Maribor (SI)*
Jurij Mihelič, *University of Ljubljana (SI)*
Hanspeter Mössenböck, *Johannes Kepler University Linz (AT)*
Günter Müller, *Albert-Ludwigs-Universität Freiburg (DE)*
Hiroshi Nakano, *Kumamoto University (JP)*
Mykola S. Nikitchenko, *National University of Taras Shevchenko, Kyiv (UA)*
Lucia Pomello, *University of Milano-Bicocca (IT)*
Herbert Prähoffer, *Johannes Kepler University Linz (AT)*
Horia F. Pop, *Babes Bolyai University, Cluj (RO)*
Stanislav Racek, *University of West Bohemia, Pilsen (CZ)*
Mihály Réger, *Óbuda University, Budapest (HU)*
Sonja Ristić, *University of Novi Sad (RS)*

Imre J. Rudas, *Óbuda University, Budapest (HU)*
Gábor Sági, *Hungarian Academy of Sciences, Budapest (HU)*
Wolfgang Schreiner, *RISC, Johannes Kepler University Linz (AT)*
Elena Somova, *University of Plovdiv (BG)*
Jiří Šafařík, *University of West Bohemia, Pilsen (CZ)*
Petr Šaloun, *VŠB – Technical University of Ostrava (CZ)*
Jarmila Škrinárová, *Matej Bel University, Banská Bystrica (SK)*
Michal Štepanovský, *Czech Technical University in Prague (CZ)*
József K. Tar, *Óbuda University, Budapest (HU)*
Tsuyoshi Usagawa, *Kumamoto University (JP)*
Valentino Vranić, *Slovak University of Technology, Bratislava (SK)*
Neven Vrček, *University of Zagreb (HR)*
František Zbořil, *Brno University of Technology (CZ)*
Jaroslav Zendulka, *Brno University of Technology (CZ)*
Marianna Zichar, *University of Debrecen (HU)*
Viktor Zhukovskyy, *National University of Water and Environmental Engineering, Rivne (UA)*
Doina Zmaranda, *University of Oradea (RO)*

Organizing Committee

Milan Šujanský, *Technical University of Košice (SK)* (chair)
Anikó Szakál, *Óbuda University, Budapest (HU)* (financial chair)
William Steingartner, *Technical University of Košice (SK)* (general manager)
Sergej Chodarev, *Technical University of Košice (SK)*
Štefan Korečko, *Technical University of Košice (SK)*
Ján Perháč, *Technical University of Košice (SK)*

Acknowledgement to Reviewers

To the many reviewers of papers submitted to 2022 IEEE 16th International Scientific Conference on Informatics, we are grateful not only as Editors and Organizers, but on behalf of authors of the papers. Preparing a professional and valuable review is not an easy process and requires insight and broad knowledge in the given field. The reviewers in their reviews provided the authors with many valuable recommendations, suggestions, and many times very positively appreciated the authors' efforts to contribute new knowledge and information in their field. The reviews have almost invariably been of great value to the authors and thus in future to the readership of the conference proceedings and to the general community of computer science. Without the high standards set by our reviewers (and Associate Editors), our conference would surely be unable to maintain its present position. The following reviewers have been of great help during the preparation of the conference:

Norbert Ádám
Rudolf Andoga
Gabriela Andrejkova
Lubomir Antoni
František Babič
Anton Baláž
Igor Bandurič
Fatma Bozyiğit
Peter Butka
Hana Bučková
Sergej Chodarev
Grigoreta Sofia Cojocar
Jasmin Cosic
Lucia De Espona Pernas
Tomasz Derda
Remi Desmartin
Jiří Dostál
Adam Dudáš
Dorota Dąbrowska
Norbert Ferenčík
Tomáš Frt'ala
Darko Galinec
Daniel Gecášek
Ján Genči
Peter Gnip
Dejana Herceg
Ladislav Huraj
Ján Hurtuk
Alexandru Iovanovici
Marián Jenčik
Erik Kajáti
Štefan Korečko
Pawel Kossecki
Mariusz Kubanek
Roland Kunkli
Michal Kvet
Attila Lovas
Alen Lovrenčić
Ivan Luković
Branislav Madoš
Ján Magyar
Marko Maliković
Miriama Mattová
Dragan Mašulović
Miroslav Melicherčík
Alzbeta Michalikova
Jurij Mihelič
Bálint Molnár
Gabriela Nečasová
Valerie Novitzka
Peter Csaba Ölveczky
Flavius Opritoiu
Norbert Pataki
Ján Perháč
Ema Pietriková
Matúš Pleva
Horia Pop
Zoltan Porkolab
Lucian Prodan
Piotr Puchała
Elzbieta Pustulka
Davorka Radaković
Danijel Radosevic
Sonja Ristic
Gabor Sagi
Wolfgang Schreiner
Olga Siedlecka-Lamch
Vladimír Siládi
Slavomír Šimoňák
Jarmila Škrinárová
Branislav Sobota
William Steingartner
Michal Stepanovsky
Matúš Sulír
Csaba Szabó
Sabina Szymoniak
Petr Šaloun
Michal Vagac
Ján Vaščák
Petr Veigend
Juraj Vincúr
Patrik Voštinár
Neven Vrcek
Jacek Wachowicz
Konrad Zdanowski
Viktor Zhukovskyy

TABLE OF CONTENTS

Invited Papers

Regular Papers

Impact of Disc Types on Database Performance

Michal Kvet
Department of Informatics, Faculty of Management Science and Informatics
University of Žilina
Žilina, Slovakia
Michal.Kvet@fri.uniza.sk

***Abstract*— Databases are continuously migrated to the cloud environment. On the other hand, there are still many requirements to locate the data in a local repository due to legislative and security reasons. Database files are block oriented forming the storage. This paper aims to analyze the impact of disc storage on performance in an on-premise world, pointing to the data themselves, as well as indexes to split the workload into multiple disc-type environments. It provides the methodology for dealing with various storage types and access routes.**

Keywords—database performance, disc storage, NVMe, physical infrastructure

I. INTRODUCTION

Database forms the central data repository of almost any information system by separating data and application layers. Databases are robust if they ensure overall consistency, defined by the data model, user, and global constraints. Most of the data systems currently offer transaction support ensuring atomicity, consistency, isolation, and durability. Transaction as a unit of database system work ensure transfer from one consistent state to another by committing the changes. A significant aspect of the transaction is just the durability in terms of data reconstruction opportunity after the crash from any reason, like electricity failure, network failure, up to the disc, and hardware crashes. An inseparable part of the administration relates to security, threats [19], failures, SQL injection, etc. Thus, the aim is to provide robust architecture, by which the data can be reconstructed and the system can be recovered without any data loss. During the evolution, several database system architectures have been introduced. The database system itself is formed by the instance delimited by the background processes and memory structures and the database itself, formed primarily by the data files holding the data. Each data file of the relational system is block oriented, the whole data activity is done in the memory after loading into the memory Buffer cache, which is also block-oriented forming the matrix [1].

The critical database operation is just the data block management, relevant block identification, and location, followed by the memory loading and complex treatment. In principle, during the data retrieval process, relevant blocks must be processed to compose the result set. It can be done by sequential scanning or using an index [8] [9]. One way or another, there is always necessary to load blocks. Blocks are part of the data files physically stored on the disc storage. This paper aims at physical architecture. Several disc types currently exist and are used for storage, from conventional discs and SSD up to the newest memory cards using NVMe technology. This study analyses the impact of physical architecture on the performance of data block identification and loading. Behind the scene, many systems are consecutively migrated to the cloud environment, however, there are still many situations, which require private solutions, on-premise world, and local systems. The most significant reason is related to the security rules, application domain, and critical data, which cannot be stored outside of the particular country, like many European Union data cannot be handled outside the region. Therefore some vendors are planning to spread the cloud repositories to apply those rules, like Oracle Cloud building separate Cloud repositories for the European Union [22].

The paper is structured as follows. Section 2 deals with the database architecture summary, pointing to the instance and database mapping. Section 3 summarizes data access methods delimited by the sequential scanning and index access paths. Section 4 emphasizes disc types used for the evaluation study, which is part of section 6. State of the art is summarized in section 5 focusing on the existing research related to the discs and databases.

II. DATABASE ARCHITECTURES

The general database system architecture is shown in fig. 1. Database server is formed by the database and the instance is available to the users via the listener, which interconnects the user process and created server process. After the handshaking, the data flow is made directly from the user to the server processes and vice versa. A server process is extended by the Private Global Area (PGA) storing local data associated with the session, while other memory structures are shared forming an SGA structure set [8].

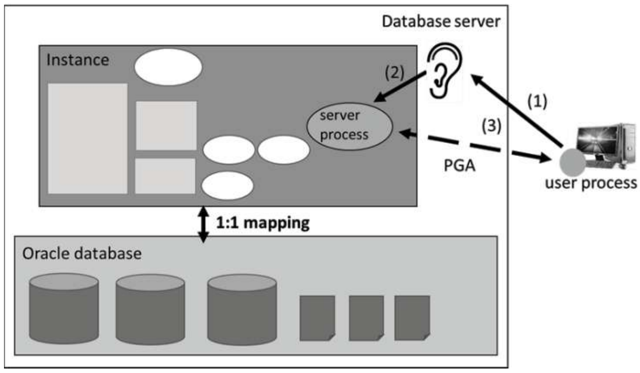

Fig. 1. Single-tenant database architecture generalization

As can be seen, the database server has an instance and database reflected by the permanent storage. Users cannot access the database directly, all the operations are managed by the background processes. The reference between the instance and database, however, does not need to be 1:1, dynamic database loading and unloading can be present in the container style. This section delimits individual architectures focusing on the instance and database mapping.

Single-tenant database (non-container database) consists of the data files, control files consisting of the data file description and paths, transaction logs, parameter files, and metadata. For that database, one instance is created.

Single-tenant RAC database uses a RAC listener and multiple instances are present for one database to balance the workload across the instances by building a reliable solution prone to instance collapse. Each instance has its listener, covered by the Single Client Access Name (SCAN) RAC listener supervision.

Multi-tenant container database uses a root container consisting of the control files, log files, parameter files, and metadata file definitions. However, there are no application data present there, they are separated from the pluggable databases, which can be attached and detached dynamically. Pluggable databases store only data, thus they need a container to be applicable. The generalization of the containerization is delimited by the Multi-tenant RAC database consisting of multiple instances. The database layer has one root container database, to which multiple pluggable databases (PDB) can be routed. Fig. 2 shows the architecture of the Multi-tenant RAC database. This architecture is used for the evaluation study in this paper.

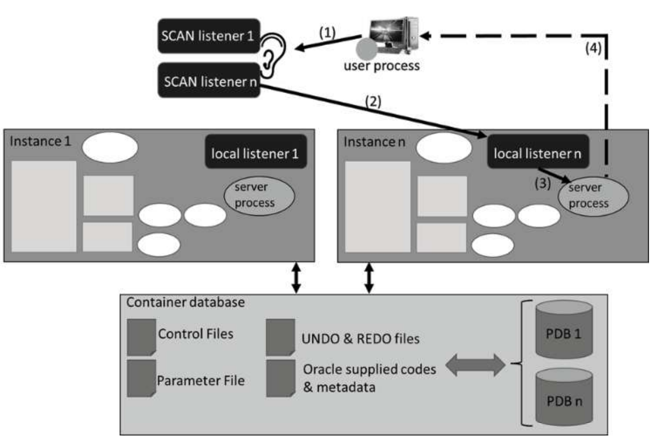

Fig. 2. Multi-tenant RAC database

III. INDEX AND ACCESS METHODS

One of the strongest optimization techniques for data access management is the index. It is primarily used for the data location using pointers to the physical database but can be used to ensure constraint passing. Commonly, relational database systems highlight a B+tree index structure consisting of the root note, internal nodes, and leaf nodes dealing with the data row addresses – ROWIDs. ROWID is defined by the 10 bytes and consists of the object identifier, data file definition, block assignment, and position of the row inside the index. It is managed and visualized by the DBMS_ROWID package [2] [6] [10] [14] [18].

The traverse path is delimited by the index keys – attributes or deterministic function results can be part of the key. The advantage of the B+tree index tree is its spread to the width rather than the height, so the performance does not degrade with the significant data number growth, e.g. for 200 million rows, only 4 layers are present. The limitation of the B+tree index is the undefined value management, if the index key is totally undefined, the particular row is not indexed [11] [15]. As a result, if there is a possibility the result set contains also rows with undefined index keys, a particular index cannot be used, at all. There are several index access methods to be used. The first category is formed by the Index Unique and Index range scan, defined by the direct conditions to be evaluated. The output of such methods is a list of ROWIDs to be consecutively processed by loading a particular block to the memory Buffer cache for the evaluation (provided by the Table Access by Index ROWID method). The second category is covered by the scanning full index. Index Full Scan operates the reading of all entries in a given index using the fact, that the leaf data layer is sorted based on the index key. Index Fast Full scan method uses the fact the index contains all the required data. So, it uses a sequential scanning operation of the whole index, which is significantly smaller compared to the table. Moreover, the structure is optimized, regarding the free blocks and fragmentations [12] [14]. A specific method is related to the Skip scanning introduced by the DBS Oracle. It can be used for multi-attribute keys by skipping a leading attribute of it, based on the selectivity expressing the ratio between the number of distinct index keys and the total number of records.

When dealing with indexing, a critical task is related to index suitability. If the index key is not relevant for the query processing and data access generally, a particular index is excluded from the potential set of access methods. If no suitable index is present, the table must be sequentially scanned, block by block using the Table Access Full (TAF) access method. This method is the most demanding, each block must be memory transferred and evaluated, up to the upper border, pointed by the High Water Mark (HWM). Whereas the blocks are not created and deallocated separately but are covered by the extent forming a set of blocks to be allocated at once, even empty blocks can be present. Also, after the data Updates, Deletes, and shift to the archive repository, analytical environment, or data warehouses, empty blocks can be present. Another problem related to the performance is just the data block fragmentation. The Master index discussed in [11] can be used to access only blocks with some data tuples. Thus, rather than holding ROWIDs inside the block, the Master index uses just the compressed list of blocks holding the data.

Among the B+tree, there are several other index structures and enhancements, like bitmap indexes commonly used by the data warehouses and marts [1] or hash indexes [6].

There are also other optimization techniques related to data block management and physical storage, like partitioning [4], compression [15], and data and index distribution [6] [16] [17]. However, whatever technique is used, the important element is just the block holding the data.

In the next section, we will analyze existing technical types and hardware equipment serving the data in a block structure.

IV. DISC TYPES

Over the decades, several disc types have been used, and many of them are still present in database systems, mostly in a test environment, but there are also many equipment pieces even powered by physical hard drives. In this section, we will summarize individual types used for the consecutive evaluation study to highlight the performance impacts and benefits caused by the shift to newer and better technology and to highlight the processing improvements. Several studies comparing various disc types are present, but database processing and block management differ from the common office or gaming world, therefore in this paper, a study related to database systems is present.

Hard drives (HDDs) are traditional storage devices with the various number of spinning platters holding the data. Each platter is covered by a ferromagnetic coating. The direction of the magnetization represents individual bits. To access a particular segment, rotation must be done, and the disc head arm identifies it, extracts the data, and shift them to the memory for evaluation. Similar step order is used for the writing process. This disc type was primarily used on personal computers and servers until the 2000s, characterized by 7200 rotations per minute for 3.5 inches. Smaller size HDDs used 5400 rotations and were used in mobile devices and laptops. Vice versa, some solutions used discs with 10000 rotations per minute. Since the HDDs are completely mechanical, the hard drive is the slowest and most fragile component of the whole equipment. On the other hand, HDDs are cheap and have a longer lifespan compared to SSD.

Solid state discs (SSDs) are faster, quieter, and smaller with no mechanical parts inside making them more durable. Moreover, they consume less energy and are shock-resistant. On the other hand, SSDs are more expensive and offer a smaller capacity compared to HDDs. The usage of SSD and its lifetime is more limited. The data recovery after the SSD drive crash is more complicated and demanding.

The speed difference between the HDD and SSD types is significant. Many studies have been performed, focusing on the sequential read/write operations of large files, like video movies, by which the copying process was less than 10 seconds, compared to the two minutes required for HDDs. When dealing with small files, the speed ranges from 0.1 to 2 MB/s, while SSDs operate ranging from 50 up to more than 250 MB/s for the newest NVMe memory types. The performance comparison was mostly operated on large files or covered by the gaming environment. The situation in database systems is, however, significantly different. The files are commonly large, but the operational level is done on the block granularity, which can, however, differ from the operating system block. In the case of using a database index, individual blocks are spread across the disc, when using sequential scanning, individual extent can group physical blocks in the neighborhood.

Non-Volatile Memory Express (NVMe) is a communication interface and driver defining the feature and command set for PCIe-based SSD drives. It communicates between the storage interface and the System CPU using high-speed PCIe sockets by lowering the latency and increasing the queue number and commands per queue. It does not communicate with the SATA controller, instead, direct system CPU communication is done.

Except for the physical disc types directly connected to the server motherboard, during the performance evaluation study, we will also highlight external drives and individual connection types, using the USB 2.0, USB 3.1, and USB-C standards. A specific approach is covered by the flash discs, which can easily extend the storage capacity. Although they are primarily used for data file transfer and local file backups, during the evaluation study, we will evaluate their usability in a database system for holding data.

V. State of art & Research perspectives

Solid-state discs have been hugely used in database technology. In [21], a discussion about the HDD and SSD disc comparison is present. Whereas the disc capacity of the SSD discs is still limited, it is necessary to build a robust solution combining both technologies. The proposed solution of that paper is based on a priority-based data placement method considering an integrated mechanism and migration rules between HDD and SSD discs.

Another limitation of SSD drives relates to their life. In [7], SSD internal implementations are highlighted, by demonstrating how a database engine can be optimized for a particular device by learning its hidden parameters. Although it does not have a direct impact on the performance, it can potentially increase the lifetime of the disc storage. In that paper, multiple rules to analyze the I/O approach and transfer techniques are proposed, focusing on the size and data location profiles.

Relation to the astronomical database is discussed in [3], focusing on the query planning and optimization algorithms. The evaluation is based on the single instance PostgreSQL databases using Quad Tree Cube indexes. In comparison with sequential techniques, the proposed solution of the paper reaches a 28.40 improvement factor by speeding up ratios to 3.7 observing CPU and disc access using intensive computations.

Currently, in-memory databases are becoming spread providing robust performance and direct access. On the other hand, such services lack the complexity of durability and recovery management. SSDs can provide the high performance these services require. In [13], performance profiling information is analyzed to shift the in-memory options to the high-performance SSD drives by tracking the I/O request granularity. Fig. 3 shows the data transfer perspectives related to the SQL command, related to the paper [13].

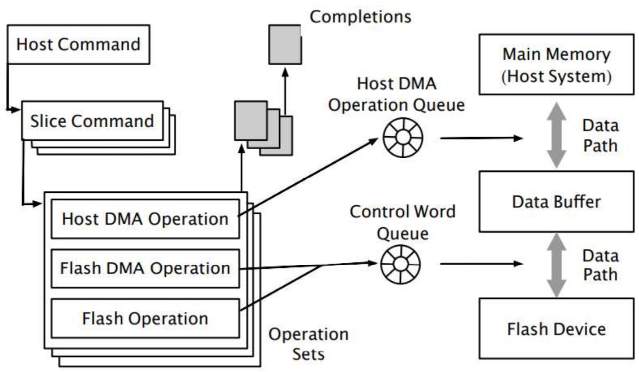

Fig. 3. Data management access during the query [13]

The storage subsystem has undergone huge innovations over the last years to cover the data amount demand increase. NVMe type provides a strong performance by pointing to the latency and peak bandwidth. Based on the performance study covered in [20], they should be preferred for data-intensive applications with a high rate of I/O operations. That paper provides in/depth performance analysis of NVMe discs. It relates to the individual parameters and impacts on NoSQL databases.

All the above papers provide sufficient study for data management across the disc-type portfolio, focusing on the data in general. This paper aims to highlight various architectures in terms of connection details and parameters, delimited by the temporal architecture, by which the data can be fragmented across the data blocks. It does not rely only on the data themselves, the emphasis is done on the indexing and size aspects.

VI. EVALUATION STUDY

Evaluation of the disc types related to the database system performance was performed in *Oracle Database 19c Enterprise Edition (Release 19.0.0.0.0 – Production)* system. It used multi-tenant RAC container database architecture.

Server parameters are:

- Processor: AMD Ryzen 5 Pro 5650 2.3GHz
- Operating memory: 64 GB, DDR4 3200MHz,
- Disc storage for the root container: 2TB PCIe Gen3 x4 NVMe v1.4, reading 3500 MB/s.

The server used Windows Server 2019 Standard Edition operating system, which is stored in a separate NVMe disc. The whole performance evaluation study is divided into multiple parts. The first part deals with the tablespace definition. The second evaluated part relates to the processing time of table definition and loading it using a conventional Insert statement. The third category deals with the costs of the database index definitions. The fourth part emphasizes the data retrieval process, followed by the Update and Delete statements.

The interconnection between the server and disc storage used an external interface – USB 3.1 or USB 3c. The reason for the separation is based on the disc arrays, which are commonly used. Moreover, existing systems generally do not offer new technologies, which can be placed directly inside and the separation of the hardware is also beneficial.

A spatio-temporal database structure was used [5] [6], dealing with air transport. A table consisted of five attributes describing the flight – unique identifier of the flight (NUMBER(38)), sequence_number of the obtained value related to the flight (NUMBER(38)), airspace assignment and reference (VARCHAR(10)), entry and exit time of the particular airspace using second precision (DATE data type).

The total number of data for the table was 4 948 094, characterized by the flight positions across airspace regions in Europe. Examples of the data are in fig. 4:

```
"ECTRL ID","Sequence Number","AUA ID","Entry Time","Exit Time"
"186858226","1","EGGXOCA","01-06-2015 04:55:00","01-06-2015 05:57:51"
"186858226","2","EISNCTA","01-06-2015 05:57:51","01-06-2015 06:28:00"
"186858226","3","EGTTCTA","01-06-2015 06:28:00","01-06-2015 07:00:44"
"186858226","4","EGTTTCTA","01-06-2015 07:00:44","01-06-2015 07:11:45"
"186858226","5","EGTTICTA","01-06-2015 07:11:45","01-06-2015 07:15:55"
```

Fig. 4. Solution – Shrinking space module architecture

The most common disc types used in the database servers were used for the evaluation, focusing on the overall impacts, processing time, and costs:

- WD 4 TB 3.5, 7200 rps, NAS optimized (**D1**),
- WD 500 GB 2.5, 5400 rps, server optimized using internal buffer (**D2**),
- WD 256 GB 2.5, SSD (**D3**),
- WD 5 TB, HDD, 2.5 external, 5400 rps, USB *3.1* (**D4**),
- WD 5 TB, HDD, 2.5 external, 540 0rps, USB *3c* (**D5**),
- WD 128 GB, NVMe internal, PCIe *Gen1* (**D6**),
- WD 256 GB, NVMe internal, PCIe *Gen3* x4 NVMe v1.4 (**D7**).

The used block size was 8kB (letting the system use the default option). For the study reference, notations D1 – D7 were used.

A. Tablespace definition

Each data file is part of the tablespace used for the reference, as well as other parameters delimiting the structure. Disc storage capacity was extended by the Autoextend option, the original size of the file was 1 MB and 100 MB. Tab. 1 shows the results expressed in processing time (s).

TABLE I. RESULTS – TABLESPACE DEFINITION – PROCESSING TIME DEMANDS

	D1	D2	D3	D4	D5	D6	D7
1 MB	0.389	0.796	0.598	0.740	0.663	0.370	0.271
100 MB	1.029	1.640	1.031	1.596	1.466	1.512	0.983

The results can be categorized into three architectures and storage principles. The first category covers the conventional discs, by which the NAS technology can be highlighted. By switching from a 7200 rotation speed to 5400, which reflects 75%, the processing time ratio for the tablespace definition is 103,86%. 2.5 SSD disc requires 0.796s for 1 MB and 1.640s for 100 MB. The difference between conventional and SSD 2.5 relates to 24.84% or 43.35% respectively, preferring SSD. Switching from the USB 3.1 to USB 3c connection interface brings 10.42% for 1MB and 9.29% for 100MB. Thus, it is evident, that the connection definition is significant for the tablespace definition, related to the maximum transfer speed and operation. Generations of NVMe cards are also important. NVMe generally does not need to bring performance improvements, compared to the NAS rotational discs. On the other hand, Generation 3 lowers the processing time demands up to 4.96% (reference D1) primarily used for NAS. However, compared to the 2.5 discs, processing time can be lowered up to 40.62%. SSD vs. 3rd generation of NVMe reflects 5.32% improvements.

B. Table definition + Insert operations

This evaluation study highlights the table definition, extended by the loading process using conventional Insert statements. No secondary indexes were defined, the integrity refers just to the data modeling – domain and column integrity. Tab. 2 shows the results. It is evident the NVMe discs do not bring sufficient performance shift, whereas they are mostly optimized for the data retrieval process. On the other hand, the size of the rotational disc can strongly affect the performance, namely, D2 requires more than 35 seconds for the processing, whereas D1 reflects just 22.7 seconds, expressing 64.51% improvements. Interesting results were obtained by the D3. It was assumed, that the results should be better, compared to the rotational discs, whereas it is SSD type. After a deeper analysis, it can be concluded, that the NAS discs use additional internal data buffers in SSD style, which reduces the differences. Moreover, the table and data definition are not devoted just to the data transfer, transaction management, logging, consistency, and integrity must be covered. However, all these data are stored in a root container, not the pluggable disc.

TABLE II. RESULTS – INSERT STATEMENT

	D1	D2	D3	D4	D5	D6	D7
processing time (s)	21.673	35.654	28.522	24.440	20.903	20.845	18.248

C. Index definition

Before dealing with the data retrieval process, index definition is emphasized, whereas it covers one of the milestones of the performance of the whole system, reducing the data block number to be treated. By using the index, sequential scanning is limited, replaced by precise block identification using an index. Five indexes were created, the first is based on the sequential number only, and the second index used sequential number and temporal attributes in the second and third positions. The third index is similar to the second, but the order of attributes is changed, preferring temporal elements. The fourth index is function-based, and takes an hour in a 24-hour format of the entry time, followed by the exit time and sequential number. The fifth index is also function-based, the structure is similar, however, a 12-hour format is used, instead. All indexes used the pre-created tablespace in the same storage type as the table data themselves. They were B+tree-oriented.

*create index **I1** on flights(sequence_number);*
*create index **I2***
on flights(sequence_number, entry_time, exit_time);
*create index **I3***
on flights(entry_time, exit_time, sequence_number)
*create index **I4***
on flights(to_char(entry_time, 'HH24'),
exit_time, sequence_number);
*create index **I5***
on flights(to_char(entry_time, 'HH'),
exit_time, sequence_number);

Processing time for the index definitions is shown in tab. 3.

TABLE III. RESULTS – INDEX MANAGEMENT

	D1	D2	D3	D4	D5	D6	D7
I1	17.196	21.077	15.301	18.544	18.008	15.785	10.068
I2	22.051	31.119	20.273	24.583	22.381	19.517	12.948
I3	22.412	31.057	20.449	24.352	22.635	19.012	17.042
I4	16.845	20.541	14.321	17.862	17.262	15.359	9.772
I5	16.652	20.388	14.024	17.511	17.012	14.895	9.209

The ratio between the index and table definition is analogous. I1 refers to a single attribute, while I2 is extended by the time perspectives, giving the additional demands of 20 to 30%. D7 uses specific data structure optimization, related to the 3rd NVMe generation, taking 22.23%.

The order of attributes inside the index does not play any role and the processing time differences are negligible, as expressed in fig. 5.

Similar results were obtained in the function-based index. I4 takes a 24-hour format, the whole 12-hour is used for I5. The processing time for the definition is almost the same. Fig. 6 shows the results in a graphical representation.

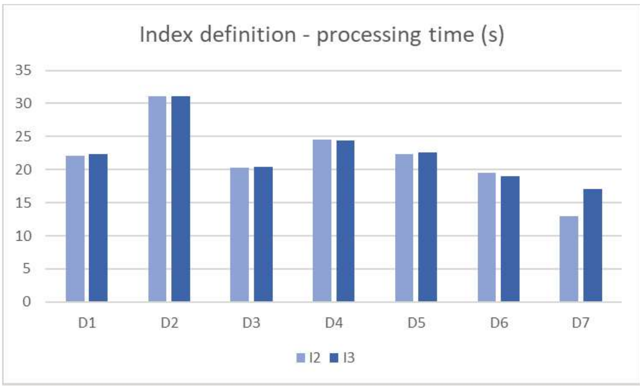

Fig. 5. Index definition – processing time

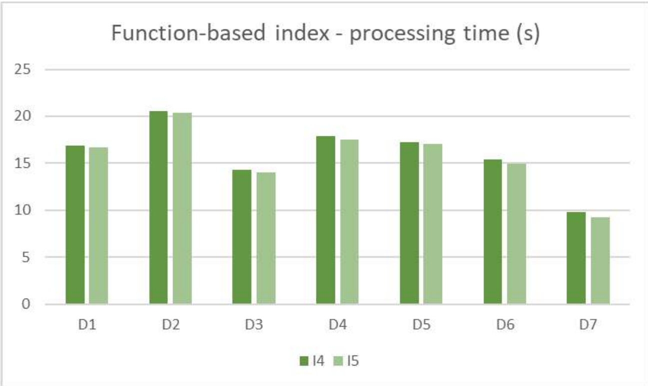

Fig. 6. Index definition – processing time

D. Data retrieval process

Query evaluation and result set building is a complex staged process, based on the execution plan, which selects the best suitable access path. During the performance evaluation, processing time was reflected, focusing on index usage. Sequential scanning of the block set regardless of the created indexes is delimited by the Table Access Full method (TAF). By using the index, Range Scan (IRS) is generally used, based on the conditions of the query. For the evaluation, Index Fast Full Scan (IFFS) method was also used.

During the evaluation, the following Select statements were used, covering the aggregate functions (S1), selection with no projection (S2), selection with Date spectrum projection (S3), and condition-based Date and Time management (S4):

S1: *select count(*) from flights where sequence_number=0;*
S2: *select * from flights where sequence_number=0;*
S3: *select entry_time, exit_time from flights*
where sequence_number=0;
S4: *select * from flights*
where sequence_number=0
and to_char(entry_time, 'HH24')=5;

For the S3, various index sets were created and evaluated. Each experiment consisted of just one index, highlighting its usage and data access method. Indexes and Data tables were part of the same tablespace:

*create index **I1** on flights(sequence_number);*
*create index **I2** on flights(sequence_number, entry_time,*
exit_time);

*create index **I3** on flights(entry_time, exit_time, sequence_number);*

Individual indexes highlight the order of attributes covered by them. The optimal index order for the query starts with the Where condition and JOIN criteria, followed by the Select clause attributes or function-based usage.

For S4 evaluation, the following function-based indexes were created, referencing the Date and Time perspective. Note the hour extraction format for the indexes.

*create index **X1** on flights(entry_time, exit_time, sequence_number);*
*create index **X2** on flights(to_char(entry_time, 'HH24'), exit_time, sequence_number);*

*create index **X3** on flights(to_char(entry_time, 'HH'), exit_time, sequence_number);*
*create index **X4** on flights(to_char(entry_time, 'HH24'), sequence_number, exit_time);*

TABLE IV. DATA RETRIEVAL PERFORMANCE

Query	Index	Used method	D1	D2	D3	D4	D5	D6	D7
S1	None	TAF	02.36	03.12	02.12	03:02	02:51	00.82	00.51
S2	None	TAF	01.10	01.93	01.01	00.36	00.34	00.43	00.31
S3	None	TAF	01.24	01.78	01.03	00.39	00.30	00.44	00.30
S3	I1	TAF	01.22	01.80	01.05	00.39	00.34	00.46	00.27
S3	I2	IRS	00.73	00.99	00.70	00.31	00.27	00.37	00.25
S3	I3	IFFS	01.02	01.36	00.83	00.33	00.30	00.43	00.26
S4	X1	TAF	00.99	01.38	00.86	00.37	00.34	00.41	00.26
S4	X2	TAF	01.13	01.44	00.85	00.30	00.34	00.43	00.27
S4	X3	TAF	01.06	01.42	00.84	00.33	00.35	00.45	00.24
S4	X4	TAF	01.08	01.41	00.92	00.33	00.36	00.44	00.27

The results are summarized in Tab. 4. The additional required demands comparing 3.5 and 2.5 discs reference range from 26% to 44%. A special case is covered by the S2 statement, which brings additional processing time demands using 68.75%. The reason is based on getting the whole attribute set, not just the tuple tick used in S1. Generally, SSD discs lower the processing time demands from 31% to 41%, whereas the block can be faster located and accessed, compared to the rotation waiting time in conventional discs. The best solution is done by the NVMe discs. Any generation of such discs brings significant performance benefits. When dealing with the whole block set management in S2, the total processing time demands are lowered from 1.12s to 0.29, expressing only 25.89%.

Reflecting the index definition and usage, range scanning brings expected results since only the blocks relevant to the given query are loaded. An inverse relationship can be identified, as the disk quality and access to blocks increase, the overall benefit decreases, ranging from 11.54 for NVMe up to 45.25% for rotational discs. IFFS can be considered as an interlayer solution between TAF and IRS.

S4 deals with the hour extraction from the Date value attribute ENTRY_TIME. Associated indexes deal with the hour extraction in 12 and 24-hour format. Reflecting the results, the database optimizer does not use any index, whereas all data attributes need to be extracted from the result set.

During this study, the impact of high-performance mobile USB sticks was analyzed. The selection was based on speed. SSD hybrid drive was required. Two drives were used – Samsung 3.1 Flash drive – 256 GB and Kingson HyperX 256 GB, reading speed – 350 MB/s, writing speed 250 MB/s. Comparing the results with NVMe discs, analogous performance was reached, however, just for the data retrieval process. Although the access and transfer speed of the NVMe discs is far higher, whereas the retrieval is based on transferring small blocks, overall speed and technology cannot be properly used. On the other hand, other operations changing the data do not provide sufficient results in terms of performance. Thus, such sticks can be alternatively used just for the static data, which do not evolve and change values over time.

Concluding the data retrieval process study, it is evident, that the physical storage layer brings significant performance improvements. Moving from the conventional rotational discs to fast NVMe cards is the best solution by lowering access delays and block location necessity. On the other hand, to reach a stable and robust performance, a suitable index set must be developed and handled, referencing the used query types. Although the amount of data based on the Where condition is reduced, necessary function calls and extraction brings additional demands. In the end, although the number of blocks is reduced, almost the same performance expressed by the processing time is reached, because for each record it is necessary to evaluate the fulfillment of the condition through a function call.

E. Update and Delete operations

Update operation requires particular block identification, followed by the content change. For this evaluation study, the block change description is copied to the physical repository by reaching commit. The Update statements were performed. The first type adds one hour to the original value of the entry and end time. Although the original values are changed, the size of the row remains the same, so there is no problem with data migration.

A more complicated case is done by the air space reference. After the Update statement, the whole string is filled, 10 bytes. The original size characteristics are in tab. 5. Update statement refers just to the first reference of the particular flight by using the condition limiting the sequential number to value zero.

TABLE V. STORAGE DEMAND SIZE CATEGORIES

length	count
4	8336
7	4241018
8	698740

Tab. 6 shows the results for the following Update statements.

update flights **-- *U1***
set entry_time=entry_time + 1/24
where sequence_number=0;
update flights **-- *U2***
set fir_id=dbms_random.string('a', 10)
where sequence_number=0;

TABLE VI. RESULTS – UPDATE STATEMENT

	D1	D2	D3	D4	D5	D6	D7
U1	17.198	21.089	15.303	18.539	18.032	15.786	10.064
U2	22.055	31.122	20.264	24.565	22.365	19.513	12.937

Comparing conventional discs pointing to the rotation speed, 7200 rps lowers the processing time demands in U1 using 3.887s, which relates to 22.61%. For U2, it reflects 9.061s (41.06%). The best performance of the rotational disc provides D1. However, if we compare it to the best NVMe drive solution (D7), the processing time costs will be reduced by 7.133s (29.12%) for U1 and 9.113s (41.32%).
The most significant difference ratio between U1 and U2 is done for D2. The reason is associated with the migrated rows by extending the structure and storage demands.

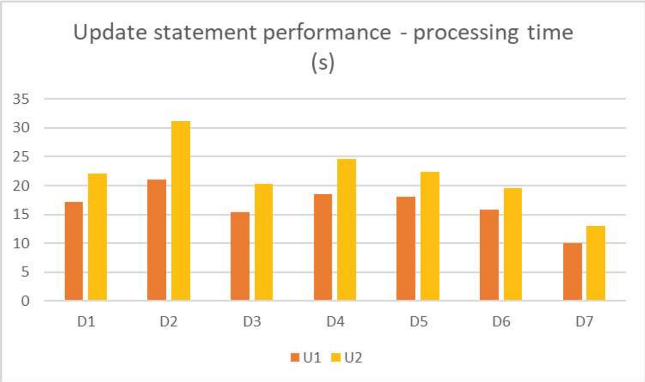

Fig. 7. Update statement performance – processing time

The last evaluation study deals with the Delete statements. It was primarily assumed, that there should be a strong relation to the Update operation, however, the results indicate not so strong a correlation. In general, it is caused by the block management of the operating system, which does not physically replace the values, they are just marked as free. Thus, there is no necessity to load and rewrite data blocks physically. Tab. 7 shows the results. For the Delete operation, the condition relates to the sequence number:

delete from flights where sequence_number=0;

TABLE VII. RESULTS – DELETE THE STATEMENT

	D1	D2	D3	D4	D5	D6	D7
D	2.369	2.377	2.054	2.273	2.031	2.001	1.953

A graphical representation of the Delete operation is in fig. 8.

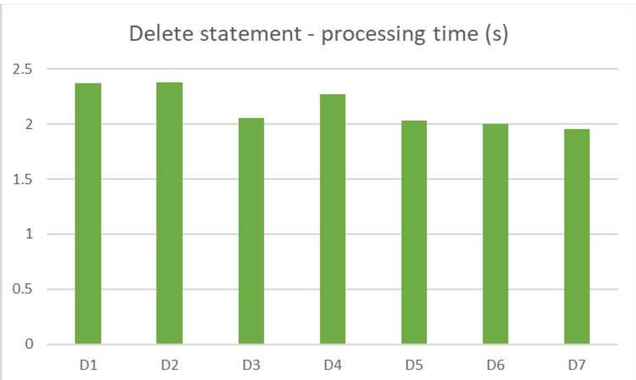

Fig. 8. Delete statement performance – processing time

VII. CONCLUSIONS

A database environment is formed by a complex system. Each database is formed by the physical storage located on the discs, which can have various characteristics, interfaces, speed, and block location principles. In this paper, we have focused on the various disc storage orientation, from conventional rotational discs through solid-state drives, up to the current trend of NVMe discs, which are commonly used in Cloud environments. The computational study was performed using a traditional server, focusing on HDDs, SSDs and NVMe drive with multiple configurations and access interfaces (USB 3.1 and USB 3c). During the evaluation, several streams were reflected – data table modeling, Insert, Update, Delete and Select operations, supervised by the index treatment. The most valuable aspect is related to the Delete operation, by which the results are not highly dependent on the used disc. Namely, individual blocks are not accessed, instead, they are just marked as free in the operating system manner. Thus, the results across the platforms are almost the same.

From the overall perspective, the most powerful disc type is NVMe, however, the emphasis should be done on the generation and other parameters, like transfer speed or read and write operation differentiation, whereas there are various standards to be applied. Drives, which are primarily used for NAS storage also provide a relevant solution, whereas they contain buffers for data transfer load balancing.

This paper deals with the various disc types for reference. Disc storage perspective and the hardware itself is, however, just one aspect of the performance. To reach the complexity, robustness, and transparency of the whole system, it is necessary to emphasize the database system architecture, instance parameters, memory size, and data access. Although this paper refers to the index definition, data distribution, partitioning, block size, and other capabilities must be handled. In the future, our focus will be on the table and index separation into multiple tablespaces, referenced by various block sizes and granularity. The emphasis will also be related to the pre-fetching and memory Buffer cache optimization.

ACKNOWLEDGMENT

This publication was realized with support of Operational Program Integrated Infrastructure 2014 - 2020 of the project: Intelligent operating and processing systems for UAVs, code ITMS 313011V422, co-financed by the European Regional Development Fund.

EUROPEAN UNION
European Regional Development Fund
OP Integrated Infrastructure 2014 – 2020
MINISTRY OF TRANSPORT AND CONSTRUCTION OF THE SLOVAK REPUBLIC

It was partially supported by the Erasmus+ projects:

- Project number: 2022-1-SK01-KA220-HED-000089149, Project title: Including EVERyone in GREEN Data Analysis.
- Project number: 2020-1-HR01-KA226-HE-094713, Project title: Cloud cOmputing for Digital Education Innovation.
- Project number: 2021-1-SI01-KA220-HED-000032218, Project title: Better Employability for Everyone with APEX.

REFERENCES

[1] J. Delplanque, A. Etien, N. Anquetil, and O. Auverlot, "Relational database schema evolution: An industrial case study," IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, Spain, 2018, pp. 635-644

[2] A. Dudáš, J. Škrinárová, and E. Vesel, "Optimization design for parallel coloring of a set of graphs in the High-Performance Computing, " Proceedings of 2019 IEEE 15th International Scientific Conference on Informatics. pp 93-99. ISBN 978-1-7281-3178-8.

[3] R. Gutiérrez-Sánchez, J. Segovia, B. Merín, F. Aguado-Agelet, “A parallel model for SQL astronomical databases based on solid state storage: Application to the Gaia Archive PostgreSQL database”, in Astronomy Journal, Volume 20, 2017. https://doi.org/10.1016/j.ascom.2017.03.006

[4] D. Jin, G. Chen, W. Hao, and L. Bin, “Whole Database Retrieval Method of General Relational Database Based on Lucene,” Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Computer Applications, ICAICA 2020. 1277–1279 (2020). https://doi.org/10.1109/ICAICA50127.2020.9182496

[5] T. Johnston. “Bi-temporal data – Theory and Practice”, Morgan Kaufmann, 2014.

[6] H. Kriegel, P., Kunath, M. Pfeifle, and M., Renz, “Acceleration of relational index structures based on statistics, “ 15th International Conference on Scientific and Statistical Database Management, 2003

[7] A. Kakaraparthy, J. Patal, K. Park, “Optimizing Databases by Learning Hidden Parameters of Solid State Drives”, VLDB, Volume 13, Number 4, 2019. https://doi.org/10.14778/3372716.3372724

[8] D. Kuhn and T. Kyte, Expert Oracle Database Architecture: Techniques and Solutions for High Performance and Producitivity. 2021, ISBN – 9781484274989

[9] D. Kuhn and T. Kyte, Oracle Database Transactions and Locking Revealed. Oracle Database Transactions and Locking Revealed. (2021). https://doi.org/10.1007/978-1-4842-6425-6

[10] M. Kvassay, E. Zaitseva, J. Kostolny, and V. Levashenko, “Importance analysis of multi-state systems based on integrated direct partial logic derivatives”, In 2015 International Conference on Information and Digital Technologies, 2015, pp. 183–195.

[11] M. Kvet, “Managing, locating and evaluating undefined values in relational databases”. 2020

[12] F. Lahouti, V. Kostina, and B. Hassibi, “How to Query An Oracle Efficient Strategies to Label Data,” IEEE Transactions on pattern Analysis and Machine Intelligence, vol. 7, 2021, pp. 1-25.

[13] A. Lerner, J. Kwak et al., “It Takes Two: Instrumenting the Interaction between In-Memory Databases and Solid-State Drives.”. In CIDR 2020, http://cidrdb.org/cidr2020/papers/p19-lerner-cidr20.pdf

[14] M. Lorenzini, W. Kim, and A.Ajoudani, “An Online Multi-Index Approach to Human Ergonomics Assessment in the Workplace,” IEEE Transactions on Human-Machine Systems. (2022). https://doi.org/10.1109/THMS.2021.3133807

[15] G. Mirza, “Null Value Conflict: Formal Definition and Resolution,” 13th International Conference on Frontiers of Information Technology (FIT), 2015.

[16] S. Pendse, V. Krishnaswamy, et al., “Oracle Database In-Memory on Active Data Guard: Real-time Analytics on a Standby Database”, 2020 IEEE 36th International Conference on Data Engineering (ICDE), 20-24 April 2020

[17] O. Rolik, K. Ulianytska, M. Khmeliuk, V. Khmeliuk, U. Kolomiiets, “Increase Efficiency of Relational Databases Using Instruments of Second Normal Form”, 221–225 (2022). https://doi.org/10.1109/ATIT54053.2021.9678605

[18] W. Schreiner, W., Steingartner, W. and V. Novitzká, V, “A Novel Categorical Approach to Semantics of Relational First-Order Logic”, Symmetry-Basel, Vol. 12, No. 10, MDPI, OCT 2020, doi: 10.3390/sym12101584

[19] W. Steingartner, D. Galinec, “Threat Defense: Cyber Deception Approach and Education for Resilience in Hybrid Threats Model,” In Symmetry-Basel, Volume 13, Issue 4, 2021.

[20] [Q. Xu, H. Siyamwala, et.al., “Performance analysis of NVMe SSDs and their implication on real world databases”, SYSTOR '15: Proceedings of the 8th ACM International Systems and Storage Conference, 2015, https://doi.org/10.1145/2757667.2757684

[21] Ch. Wu, Ch. Huang, Ch, Chang, “A priority-based data placement method for databases using solid-state drives”, RACS '18: Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, 2018, https://doi.org/10.1145/3264746.3264769

[22] https://www.oracle.com/cloud