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 Frťala
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

Invited Papers

Regular Papers

Relation between the Temporal Database Environment and Disc Block Size

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

***Abstract*— Temporal databases are used to monitor states in the object, attribute, or group granularity. Performance of the data retrieval is a critical requirement, consisting of relevant data block identification, loading, and tuple extraction. Instance memory loading must always take the whole block. This paper deals with the impact of block size on the performance of temporal databases, by pointing to block fragmentation, expansion, and shrinking. It also points to the definition of block migration, if the tuple after the Update operation does not fit the original block.**

Keywords—temporal database, block, fragmentation, shrinking, retrieval, performance

I. INTRODUCTION

The IT industry has changed significantly over the decades. Originally, there was only a small amount of data to be handled, commonly representing current environment properties by the conventional approach. It meant original data were physically replaced by newer versions. Later, the amount of data started to rise. Nowadays, data management is identified by the dynamics and complexity. It is necessary to store and reflect the whole evolution forming a temporal environment. Thus, instead of replacing the original state physically, the new version is created and stored, delimited by the validity time frame. From the architectural point of view, relational databases are still the most often used due to a security perspective. The relational paradigm is associated with data consistency and a strict data model supervised by the integrity constraints managed by the transactions. Thus, any data portion loaded into the database must pass all the integrity constraints to be accepted [5], focusing on threat defense [14]. Such checking is done either immediately after the statement itself or at the end of the transaction. Whereas the data complexity and demands are still rising, the relational paradigm was extended by object-oriented attributes, XML, or JSON definitions [5].

The data management is operated on the block-level structure physically by locating a particular block and treating data. Thus, the block size and its complex management are critical, whereas the loading and evaluation are done on the block granularity.

This paper aims at database block management and performance impacts to optimize the processing. In the first part, temporal database principles are summarized stating the study environment, followed by the data retrieval process and evaluation. It relates to the block size impacts, emphasizing the data fragmentation. During the computational study, a temporal environment is used. Temporal systems are rather dynamic and storage demands evolve rapidly. Thus, each Update and Insert operation brings a new data object version. On the other hand, to ensure performance and relevant information value, historical data are continuously aggregated and migrated to the archive repositories, data warehouses, or marts consequencing in block fragmentation. Moreover, there can be data corrections (e.g. getting higher precision, getting the delayed data from the sensor or satellite) forming two-dimensional space for the temporality – not only validity is monitored, by transaction correctness, as well. If the data do not fit the originally associated block space, migration is present. To reach complexity and performance, individual object states are grouped together into the same neighbor blocks simplifying the evaluation and loading perspectives. The aim is to optimize physical block architecture and infrastructure, reflected by the air transport system usage, by which the proper and accurate data retrieval process must be ensured.

This paper is structured as follows. Chapter 2 deals with the temporal database definition forming state-of-the-art. Chapter 3 deals with the table indexes as the core elements ensuring the performance by effective block location. Chapter 4 focuses on fragmentation management, migration handling, and block shrinking by proposing own techniques for optimizing physical structure. Chapter 5 evaluates the impacts of the block size by emphasizing the computational study and performance impacts.

Performance evaluation for the block size impact identification using the temporal environment has been performed in *Oracle Database 19c Enterprise Edition (Release 19.0.0.0.0 – Production)* system. Server parameters are: *processing unit*: Intel Xeon E5620; 2,4GHz (8 cores), *operation memory*: 48GB, *SSD* drive: 1 000GB.

A spatio-temporal data locating and identifying airplane objects by the occurrence time, and GPS position was used, delimited by the speed, destination, current airspace association (entry and exit time), planned route vs. current route, as well as the weather conditions influencing the flight. There were 20 attributes, enclosed by the validity time frame. The data set consists of 1 808 390 data tuples in total. Examples of the data are in fig. 1:

```
"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. 1. Solution – Shrinking space module architecture

II. TEMPORAL DATABASES

Temporal databases arizes from the conventional principles by extending the state definition by the Date and Time borders, mostly defining the validity. Generally, there are three principles related to the granularity levels – object, attribute, and group granularity.

In object-oriented granularity, the whole state is extended by the validity frame. Thus, if any change occurs, a

completely new state image of the object is created. As a consequence, any state identification is straightforward and can be obtained easily taking the validity range. On the other hand, there can be severe limitations related to storage capacity. Therefore, to ensure performance, it is necessary to synchronize changes, otherwise, many duplicate values are present, and identification of real change can be complicated and demanding [1] [3]. The opposite solution is delimited by the attribute granularity extending each data value (attribute) by a Date and Time reflection. The complex state itself is then composed of individual attribute states [7]. The interlayer is defined by the synchronization groups processing not only attributes separately but they can be grouped if the change occurs at the same time [7]. The synchronization group is applied dynamically and is also temporal. It provides the universal solution, even conventional attributes with no time perspective can be covered. From the definition point of view, the table can be static (no changes can be present), conventional (data are not monitored over time), or temporal. The architecture consists of the Group detector, Group manager, and Synchronization layer responsible for the group forming and dynamic drops. Data themselves are part of three layers reflecting the validity – current valid states, non-actual data (future and historical), operated by the temporal management [7]. For the evaluation study, group granularity is used. Data are provided from the air transport systems and sensors.

Fig. 2 shows the architecture of the group-level temporal system. Data flow is taken to the temporal management layer, interconnected with current and inactual data representations. Future and historical data are attribute-oriented, current valid states use object granularity. The reason is based on the possibility of using conventional applications and their interconnection to the temporal architecture, whereas they just require current valid states in object granularity to form the result set. Besides the main temporal management, data synchronization is continuously monitored by the Group detector layer. If the group is detected, the Group manager creates new synchronization by notifying the Synchronization layer. The whole activity is supervised by temporal management, which stores the synchronization groups in a temporal architecture, as well.

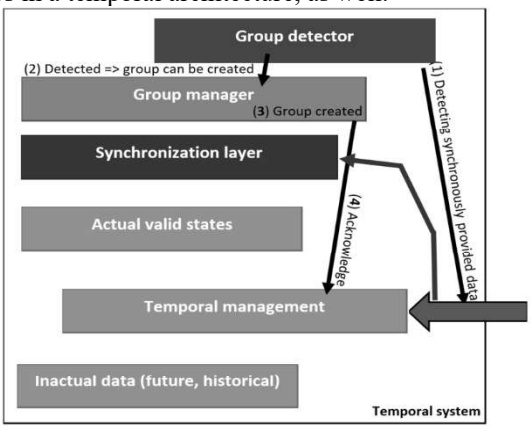


Fig. 2. Temporal database architecture

III. Retrieval & Indexing

The SQL language is a non-procedural language by specifying the data to be obtained with no step-by-step user construction. The evaluation and result set composition is the responsibility of the database system manager. The database system itself is formed by the instance consisting of the memory structures and background processes and the database delimited by the physical file storage, either in the local server or Cloud environment. A physical database is represented by the storage, covering data files. They are block oriented with a fixed size. To get the data from the database, relevant data blocks must be identified and transferred into the memory for consecutive evaluation. Thus, the smallest data portion to be operated is a block itself. However, data blocks are not created and associated separately, due to the high costs of the operation. Therefore, each table is formed by the extents, from the physical perspective. In fact, the extent is an array of data blocks (using a linked list), physically allocated in the continuous disc space. Thus, each block is part of the extent, which are interconnected forming a linear linked list. The table is formed by the specification and delimited by the data segment. It takes a chain of extents associated with the table (fig. 3). The last block of the table is pointed by the High Water Mark (HWM). Thus, by scanning the blocks sequentially, the upper limit is reflected just by the HWM. During the processing, fragmentation occurs as a consequence of the block and tuple size, updates, and deletes of the states. The main problem is just the empty block, which is loaded into the memory Buffer cache for evaluation during the sequential scanning. Such an operation is named the Table Access Full (TAF) method.

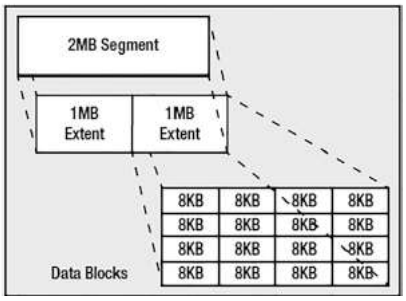


Fig. 3. Structure definition – segment, extent, and block structures [5]

When dealing with data retrieval, evaluation and relevant data extraction passing all the conditions is a staged process. Fig. 4 shows the data retrieval process by parsing the query and selecting the best suitable access path followed by providing the row sources and building the result set.

The TAF method is characterized by sequential data block scanning. There is no option to identify the fill ratio of the blocks, nor to detect completely empty blocks. Instead, each block content is sequentially copied to the memory Buffer cache for evaluation using I/O operation [5] [6]. Buffer cache memory structure is formed as a matrix of the blocks, which can be empty (newer used or directly prepared for the loading), clean (hold data, which can be, however, directly rewritten, whereas they do not hold any extra data difference in comparison with the database storage reflection) or dirty (locked blocks, which hold new data, which must be stored in the database before rewriting and consecutive processing) [6]. Thus, for the TAF, memory space for the loading must be identified, followed by the I/O operations, which are time and resource-consuming.

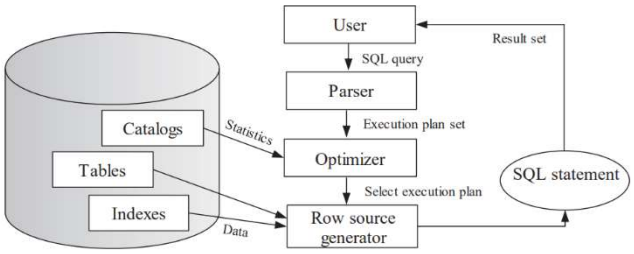


Fig. 4. Data retrieval process definition [5]

Another option for data source identification is based on using indexes. An index is an optional structure associated with the table, which can be used for the data location by using the index key, which must be compatible with the query. There are many types of indexes (B+tree, bitmap, hash indexes, clusters, etc.) with various enhancements (reverse, compression, function-based, virtual column indexing, etc.) [2] [4] [8]. The default strategy used in an object-relational database environment is associated with the B+tree consisting of the root nodes, and internal nodes by traversing using the index keys and leaf nodes referencing the data using the ROWID pointers. ROWID address is a direct data locator formed by 10 bytes, consisting of the identifier of the data file, particular data block, and position of the row inside it. Thus, it is the fastest access to the data. For data retrieval, the index can have significant performance impacts. Firstly, the traversing index is far faster in comparison with sequential loading and scanning. Namely, there is no non-relevant block loading necessity. Secondly, the leaf node of the index provides a direct address, so the accessibility is straightforward. Thirdly, the index is rather wide than deep, so the amount of data path in the index is strictly limited. Note, that despite the fact, ROWID points to the data position inside the block, the loading operation always refers to the block itself, thus, always, the whole block is loaded.

Index definition and usage can have many negative aspects to be highlighted. The index must cover the current situation. Therefore, it must be updated during the Insert, Update or Delete operations increasing the time processing demands of such operations. The second problem is related to the undefined (NULL) values, which cannot be mathematically compared and are not part of the index. Finally, there is a limitation caused by the data migrations, by which the state after the change does not fit the original data block and must be shifted to another free one. The consequence is associated with the existing index set, which does not cover the change, and the leaf layer ROWID address still points to the original block, where the data are not located. Instead, there is just another pointer present there, forcing the system to load a different block. In general, such a row chaining can be deep resulting in performance degradation. In [7], the solution is based on the ROWlog layer replacing physical ROWID addresses with logical pointers in a separate layer. Multiple indexes can reference that layer used as a physical mapper. Therefore, migration is limited by creating new references in the additional layer – done only once for all indexes. Other solutions described in [9] [10] are related to the undefined value coverage by the index delimited by the specific extension, which does not use the original index key, but the unique row identification is used (either delimited by the physical address, primary key or hash value can be used).

These techniques and approaches form state of art by emphasizing data retrieval and limiting migrated rows. Next section deals with the proposed contribution dealing with the de-fragmentation and space shrinking, focusing on the block size impacts.

IV. Fragmentation, Shrinking

Before introducing our own methods and contribution, let's reflect on the physical block management during the Insert and Update operation. A user defines a statement to be processed. After parsing, and checking privileges, integrity, and all the constraints, the row is to be stored physically in the database, formed by the extents and blocks associated with the table, with HWM as an upper limit. Thus, the task is to locate the appropriately suitable block. Over history, four approaches can be identified. The oldest and simplest solution was based on block set traversing, loading, and evaluating, whether it can cover a new row or not [5] [11]. That technique was too demanding in dynamic systems, reducing the parallelism. Thus, it can be, generally, said, that the Insert and Update operation was enclosed by the Select statement providing a suitable block. The scanning was done sequentially. By reaching the HWM, it was evident, that a new extent must be allocated. The first-fit method was used to extend the problem of fragmentation – even a small tuple can be mapped to large block space.

The second solution removed the scanning and a new tuple was always stored in a new block. From the storage perspective, each tuple was stored in a separate block using 1:1 mapping, which negatively impacted the storage demands. Although the block size was reduced, such an approach was confusing and quickly rejected [12].

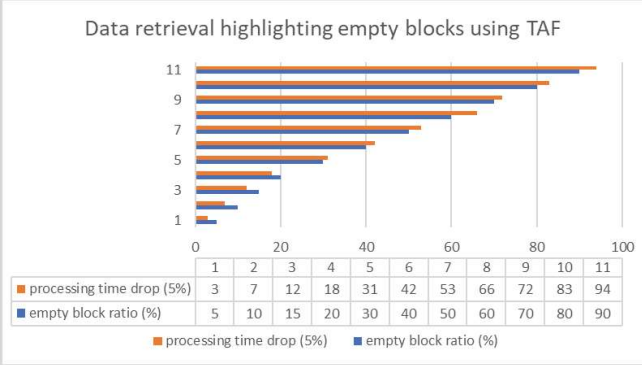
The third and fourth solutions are based on free space block categorization into sets. The first category covers the blocks, which are filled from 0 to 25% of the capacity. The second category ranges from 25% to 50%, the third delimits the range from 50% to 75%, and finally, the last category deals with ranging from 75% to 100%. The difference between solutions 3 and 4 is based on the data information repository, which can be in the data dictionary or part of the object structure segment. Based on [5], the data dictionary forms the bottleneck due to the high workload and locking, so the best suitable solution is operated by table segment categorization.

The algorithm is based on getting the size of the row to be processed, followed by identifying the best suitable block based on the categorization. Thanks to that, the best suitable blocks are used as a priority to reduce fragmentation. That profile, however, results in one very important consequence. Since the filling of the block is optimized, free or too thinly filled blocks are present, but cannot be identified, and are no further processed in the current solution. The proposed solution uses a fulness ratio and usability prediction soon based on the change flow frequency.

A. *Proposed solutions – tuple management*

Categorization of block usage is a relevant strategy to minimize fragmentation. Empty (or almost free) blocks are then used only if others cannot serve and cover them. Therefore, such blocks must be identified to exclude them from the TAF processing. SOLUTION 1 is based on extending the categorization strategy by dividing the first category reflecting the empty blocks with no tuples. It forces the system to use 5 categories. The total storage demands are

extended by 0.1% for the whole segment, irrespective of the number of extents. On the other hand, the TAF method can significantly benefit by lowering I/O operations. Fig. 5 shows the impacts of the TAF method related to the free block ratio. The X-axis represents the percentage of the free blocks and the percentage of the time processing demands cuts. The Y-axis delimits the scenarios. It represents an almost linear dependence type. If the ratio of empty blocks is more than 28%, the results are almost the same. It is because the loading necessity is limited, which is one of the most demanding operations during the processing.



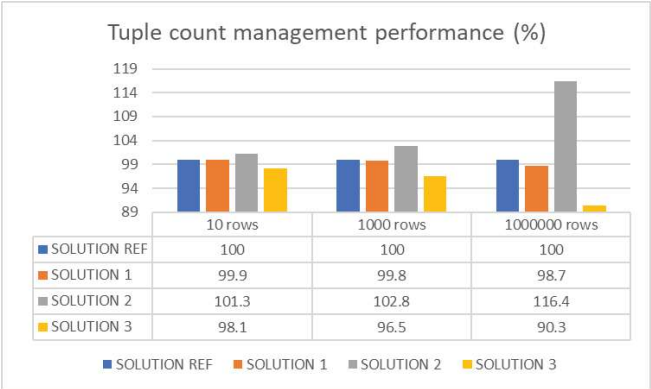
	1	2	3	4	5	6	7	8	9	10	11
processing time drop (5%)	3	7	12	18	31	42	53	66	72	83	94
empty block ratio (%)	5	10	15	20	30	40	50	60	70	80	90

Fig. 5. Data retrieval highlighting empty blocks ratio using TAF

SOLUTION 2 is based on using tuple count categorization instead of block fill ratio. By using such an approach, empty blocks can be very easily identified. On the other hand, it brought additional demands for the Insert and Update operation, whereas the count itself does not relate to the tuple size. And if the structure of the tuple was dynamic from a size perspective, even a block with one tuple cannot hold additional rows, whereas the original one fits almost the whole block.

Therefore, SOLUTION 3 extends it by marking usability. Once the block is loaded for the potential new tuple assigning, it can be marked as full by excluding it from further evaluation. This mark is then automatically reevaluated during any change on the block. Fig. 5 shows the results expressing processing time costs for loading 10, 1 000, and 1 000 000 rows to the existing structure dealing with 10 000 000 rows. It takes the original solution categorizing the block fill ratio and solutions 2 and 3. It is expressed in the percentage. The reference model (SOLUTION REF) is the existing approach used by default currently. The aim of the proposed solutions is the maximum number of tuples covered by the block, compared to the existing approach maximizing the full rate. By the diversity of the data structures and tuple types, such a difference can have significant consequences.

Reflecting the reached outputs, pure tuple count (SOLUTION 2) does not bring additional power, whereas the size of the tuple is not emphasized, thus the large tuples are preferred to map a new row, but the size demands cannot cover it. On the other hand, SOLUTION 3 uses marking by limiting such a negative aspect. As evident from the results, it more properly points to block optimization. Namely, for the evaluation, if the block holds more data, there is a natural assumption, that the probability it holds particular data is increased.



	10 rows	1000 rows	1000000 rows
SOLUTION REF	100	100	100
SOLUTION 1	99.9	99.8	98.7
SOLUTION 2	101.3	102.8	116.4
SOLUTION 3	98.1	96.5	90.3

Fig. 6. Data retrieval highlighting empty blocks ratio using TAF

When dealing with the tuple count, the crucial task is to identify the number of covered rows. In principle, if the new structure is created, any operation can change the assigned value. Tuple count is associated with each data block and can be located in a data dictionary, separate structure, or part of the block itself. The data dictionary does not provide sufficient power due to the locking forming the bottleneck. Locating the tuple count directly inside the block is irrelevant, to obtain such a value, it would be necessary to load it into the memory, so the benefit is removed. The specific structure is the most relevant, which can be sorted based on suitability using a linear linked list or B+tree (key index value is tuple count). Fig. 7 shows the results. It takes tuple count definition during the Insert statement to identify a suitable data block. In the first phase, the tuple block is taken, then the selection is made based on the block fill to ensure coverage. The reference model is extracting tuple count into a separate structure organized randomly. Data dictionary (is limited by the locking and parallelism) and block management (is limited by the I/O operations to get the value) reach almost the same results. The best solution is based on the B+tree lowering the demands up to 40%. By applying the extended fill block management, migrated rows are limited, resulting in lowering the processing time demands by 49%.

Fig. 7 shows the performance drop related to various architectures dealing with the tuple count for the block.

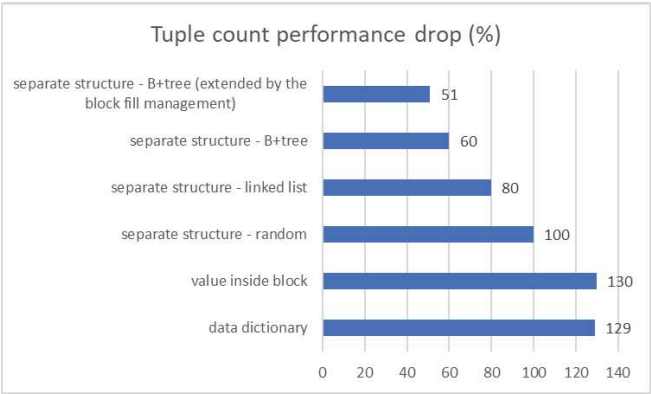


Fig. 7. Performance of the Insert statement using tuple count architecture

B. *Proposed solutions – rescanning*

In the preceding system, there was an assumption, that tuple management is added to the empty system. Thus, there are no extents initially and each extent to be allocated is initialized by getting a zero value for each block (tuple

count). However, how to apply the rules, if the tuple count management needs to be implemented on an existing system, by which the table is formed to multiple extents? How to calculate tuple count values without sequential block loading into the memory, whereas such activity would be too demanding from the time processing, as well as storage demands? The Buffer cache structure should be, there, emphasized, which is a destination for the block during the loading and evaluation. First, a free block must be identified there, if there is no one, loading hangs, during which some blocks from the Buffer cache are swapped into the disc, so the number of I/O is increased. Therefore, it is important to find another way to get the tuple count by minimizing I/O operations and Buffer cache usage.

Our proposed solution for the existing system arises from the Master indexing, which was first defined in 2017 [7]. One of the table indexes covering all the data (NULL values are not part of the index) is marked as Master and is used in case there is no suitable index for the evaluation. Instead of using the TAF method, the Master index is used as a data locator. It optimizes access by limiting fragmentation and empty block loading. In the proposed solution, it is used for calculating tuple count by evaluating ticks for the block inside the index. Namely, the Master index is always smaller in comparison with the whole tuple, so the number of blocks is reduced. Moreover, the index structure is better optimized from the storage perspective and consecutive loading. Finally, the Master index is often used during the data retrieval, so it is at least partially pre-loaded into the Buffer cache. Such blocks can be directly used without I/O operation necessity (loading).

The algorithm for the tuple count calculation for the existing system starts with the Master index location. If it exists, it is used for the calculation by using the block-hit ratio. If the Master index is not defined for the table, then the system evaluates existing indexes based on the size, pre-loading, and assumed usage to select one of them to be declared as Master. The only condition is that it covers all the data tuples to ensure tuple count consistency. Using the Master index, tuple count calculation can be done.

In case there is no index for the table – there is no primary key, nor secondary indexes present, tuple count is just assumed using the current statistics, block number, and row count. Such an estimate can have, however, significant performance impacts, if the size of the rows varies dynamically for the particular table.

C. *Shrinking space and index impacts*

Free space categorization and tuple count management are useful for the data change operation, mostly reflected by the Insert statement. Update operation can be considered as a Delete operation followed by the Insert statement, typically placed to the same data block, if possible. Database systems do not shrink space across the data blocks automatically due to several reasons [5]. First, it is assumed, that such free blocks will be used soon, therefore there is no pressure for the structure optimization, where it would be valid only temporarily, and later, new extent allocation would be demanding. Moreover, data blocks cannot be deallocated separately, the management granularity is always the whole extent (formed by several data blocks). Finally, data blocks are pointed by the indexes, thus the shrinking would cause migrated row problems [11].

On the other hand, there are many occasions, where the structure optimization would be beneficial – after the data archiving [5], moving to the data warehouse, lake, or mart [12], or after the reconsolidation [9], where blocks are free or by fixing the data structure and content [13]. By these means, if the blocks repository is not optimized, additional blocks must be loaded with no relevant content during the TAF method. Similarly, index access techniques could be degraded by the migrated rows.

Our proposed technique dealing with the shrinking space arises from the logical ROWID management by mapping the logical definition part of the index into physical pointers [5]. The original approach takes a new migration pointer, thus, after the block loading, data are not present in that particular block. Our proposed solution references the migration mapper layer by replacing the data pointer to the new block, where the data are moved. Thanks to that, the original block is freed, and removed from the evaluation, whereas no data are there. And mostly, existing indexes reference a new position after the processing, thus there is no migration present. Fig. 7 shows the proposed solution. Each index is connected to the migration mapper, extended by the shrinking space module, consisting of the blocks and their real size. By shrinking the space (operated by the Shrinker background process), the Migration mapper is reflected, covering only one Update operation, compared to the common migration, which must dereference each block separately. Fig. 8 also shows the data flow and block interconnection.

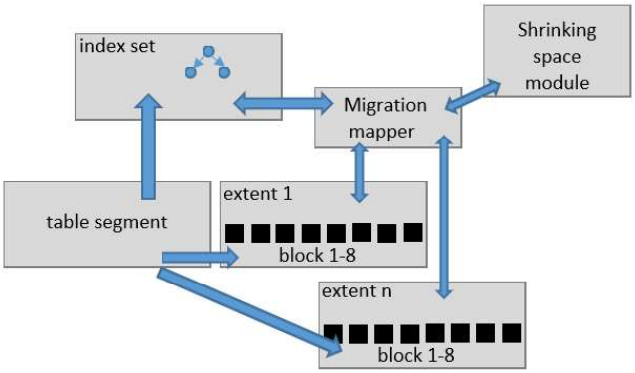


Fig. 8. Solution – Shrinking space module architecture

V. BLOCK SIZE IMPACTS

The database is formed by the files holding the data. Each data file is formed by blocks, in which the data resist. System instance is formed by the memory structures and background processes. The interlayer between them is the tablespace characterizing the physical infrastructure – files, blocks, etc. Each database system must contain at least one tablespace (like *SYSAUX* and *SYSTEM* tablespaces in DBS Oracle). The aim is to group data files with the same characteristics together based on the application domain. One of the significant parameters is just the block size, which is then mapped to the Buffer cache extension – *DB_cache*. If the block size is not specified explicitly for a tablespace, the default value is used, set for the whole database. Such a value can be obtained by referencing the *db_block_size* parameter, the default value is 8kB and can range from 2kB to 32kB.

show parameter db_block_size

```
NAME                           TYPE                VALUE
------------------------------ ------------------- ------------
db_block_size                  integer             8192
```

The proper value of the database block set can have various consequences, like contention reduction, reduced row chaining (if the row cannot fit one block), faster scans, and lowering the number of loading [5] [6] [7].

The size of the block for the tablespace can be set using the blocksize clause:

create tablespace <tablespace_name>

datafile <location_and_name>

blocksize {2 | 4 | 8 | 16 | 32}K;

However, before creating a tablespace with a non-default block value, the DB_cache memory structure for a particular block size matrix must be allocated, using the following command:

alter system

set db_{2 | 4 | 8 | 16 | 32}k_cache_size=<value> {M | G};

The evaluation study analyzed CPU costs (absolute values) and processing time (in [mi:ss:ff]). It highlights the demands for the table structure separately, and extended solution with a primary key index with various block sizes. Tab. 1 shows the results of taking 30% fragmentation by getting 10 000 rows. It takes an almost uniform definition. With the rise of the block size, I/O operations and waiting times are lowered. It shows, that the main aspect of the data retrieval processing is associated with memory loading and evaluation.

TABLE I. RESULTS – IMPACT OF BLOCK SIZE TO TABLE STRUCTURE

Block size	Costs	Processing time
2KB	12378K	06:27:01
4KB	6275K	03:55:35
8KB	3327K	01:58:92
16KB	1689K	01:01:77
32KB	912K	00:34:53

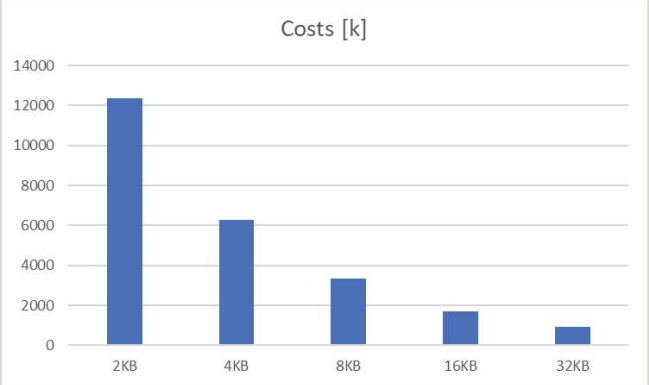


Fig. 9. Results – table size block => costs

Block migration reduction benefits from 5% fragmentation. Fig. 10 shows the results stating the original performance and solution dealing with the proposed technique of migration reduction using logical block addresses and mappers. Although the original solution uses migration pointers, it is always necessary to load a particular block and then, reference the pointed block. It uses 8KB block for processing. The results highlight CPU costs related to the fragmentation ratio.

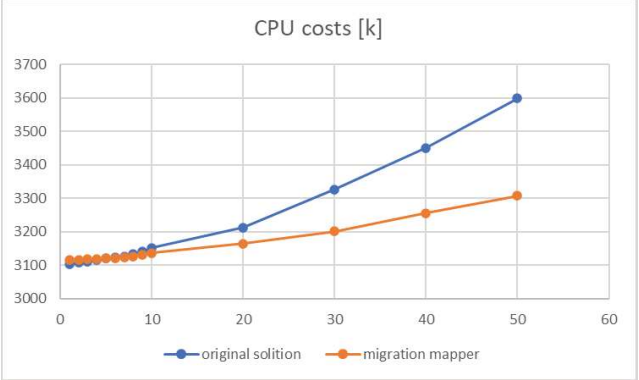


Fig. 10. Results fragmentation impacts => costs

VI. CONCLUSIONS

The data amount to be handled is still rising by emphasizing efficiency and overall performance. To ensure proper decision-making, temporal databases are used, dealing not only with the current valid data but also with historical states and future valid tuples are present.

This paper summarizes temporal architectures pointing to the granularity levels of the object, attribute, and synchronization group. Performance evaluation study takes spatio-temporal architecture of the synchronization group, dealing with the air traffic.

It emphasizes the physical database layer and infrastructure by pointing to the block size in terms of size, memory loading, as well as fill ratio, fragmentation, and shrinking.

Namely, tuple management has been discussed by listing the empty blocks excluded from the evaluation stream. The storage demands are extended by 0.1%. The fill ratio procedure is covered by another proposed solution, however, as discussed and evaluated, it does not bring a robust solution, whereas the tuple size can be dynamic. Thus, rather tuple count coverage by the block is preferred, lowering the performance demands of the retrieval to 90.3%.

The complexity of the proposed architecture highlights the data migration, by which the original index ROWID does not precisely locate the block with particular data and shrinking space module. In that architecture, each extent is rescanned by pointing to the empty blocks, and grouping them by the linked list array.

The second part of the paper deals with the block size demands. As evident from the results, total costs are almost linear reflecting the block size definition (30% data fragmentation is present). The main advantage is based on the fragmentation limitation, which is lowered with the rise of the block size. The default 8KB block requires 01.58:92 for the processing, while 2KB requires 06:27:01 and 32KB denotes 00:34:53. Thus, it is almost linear. On the other hand, if the block size increases, more memory space is also required for mapping. Therefore, the best solution is to use various block size levels, based on the data types and tuple structure.

In future research, we would like to evaluate the impact of tablespace definitions related to physical storage and data distribution to create a complex methodology of performance

evaluation in the temporal environment. There is an assumption the various environment and distribution characteristics can bring additional power by parallelizing the process of the retrieval by pointing to the migration mapper located in separate disc storage.

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.

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] 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

[4] 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

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

[6] 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

[7] M. Kvet, "Managing, locating and evaluating undefined values in relational databases". 2020

[8] 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.

[9] 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

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

[11] 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

[12] 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

[13] W. Steingartner, J. Eged, D. Radakovic, V. Novitzka, "Some innovations of teaching the course on Data structures and algorithms, " In 15th International Scientific Conference on Informatics, 2019.

[14] 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.