TECHNICAL CO-SPONSORS

IEEE Region 8

IEEE Croatia Section, Communications Chapter, Computer Chapter, Systems Man and Cybernetics Chapter, Reliability Chapter

Ministry of Science and Education

J. J. Strossmayer University of Osijek

International Council on Large Electric Systems CIGRÉ, Croatian National Committee

Croatian Academy of Engineering

International Conference on Smart Systems and Technologies (SST 2022)

PROCEEDINGS OF

International Conference on Smart Systems and Technologies (SST 2022)

FERIT

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,
Josip Juraj Strossmayer University of Osijek
Osijek, Croatia

**October 19 -21
2022**

ISBN: 978-1-6654-8214-1
The IEEE Catalog number: CFP22G03-USB

PROCEEDINGS OF

International Conference on Smart Systems and Technologies (SST 2022)

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,
Josip Juraj Strossmayer University of Osijek
Osijek, Croatia

October 19-21
2022

ISBN: 978-1-6654-8214-1
The IEEE catalog number: CFP22G03-USB

COPYRIGHT INFORMATION

Published by: Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

Edited by: Emmanuel Karlo Nyarko, Tomislav Matić, Robert Cupec, Mario Vranješ

ISBN: 978-1-6654-8214-1

IEEE Catalog number: CFP22G03-USB

2022 International Conference on Smart Systems and Technologies (SST 2022)

IEEE Catalog Number: CFP22G03-USB
ISBN: 978-1-6654-8214-1

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

MESSAGE FROM THE SST 2022 GENERAL CHAIR AND PROGRAM CO-CHAIRS

Welcome to Osijek, Croatia. Two years after having four very successful IEEE International Conferences on Smart Systems and Technologies, it is our great pleasure to welcome you to the 5th IEEE International Conference on Smart Systems and Technologies 2022 (SST 2022), organized and hosted by the Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, Josip Juraj Strossmayer University of Osijek.

The conference provides an international platform for researchers and practitioners interested in the theory and practice of smart systems and technologies related to electrical engineering, communications, computer science and engineering, control systems, robotics, as well as interdisciplinary research and applications. SST 2022 enables participants to present their own research results and exchange experience in all aspects of smart systems and technologies.

Conference presentations are divided into nine technical sessions and two special sessions focusing on power generation, renewable energy sources, smart power systems, electric vehicles, software and systems engineering, electronics, control systems, sensors, internet of things, image and video processing and machine learning applications, as well as a workshop related to smart technologies for sustainable agriculture. The program has been developed around 54 reviewed and accepted papers by 174 authors from 17 countries. The technical program resulted from numerous months of hard work by 24 Steering Committee members, 52 Program Committee members, 85 reviewers, 23 Organizing Committee members, conference Chairs and many other volunteers.

The program also features a plenary talk by Prof. Julius Georgiou on smart systems for an improved quality of life, a keynote speech by Prof. Hrvoje Pandžić on the necessity of distribution-level markets and Prof. Georges Kaddoum on industrial-internet-of-things for the next-generation of smart grids and an invited lecture by Prof. Boris Dumnić on an innovative approach to a more sustainable and reliable energy society as well as the Awards Ceremony.

We would like to thank all the authors, plenary, keynote and invited speakers for their contributions; all enthusiastic colleagues involved in the reviewing process, conference committees, our technical co-sponsors, supporting scientific, professional and other organizations, institutions, companies and volunteers. We owe you a debt of gratitude for everything you have done.

In addition to the technical program of the conference, we strived that this conference provides not only opportunities for your scientific development but also nice memories of Osijek and Eastern Croatia.

We thank you all for your support and attendance and wish you a pleasant stay in Osijek and Croatia or successful virtual online participation.

SST 2022 General Chair:
Emmanuel Karlo Nyarko

SST 2022 Program Co-Chairs:
Tomislav Matić
Mario Vranješ

CO-ORGANIZERS

Faculty of Electrical Engineering and Computing, University of Zagreb

Mechanical Engineering Faculty in Slavonski Brod, JJ Strossmayer University of Osijek

Faculty of Organization and Informatics Varaždin, University of Zagreb

CONFERENCE CHAIRS

General Chair

Emmanuel Karlo Nyarko

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 495 403

e-mail: karlo.nyarko@ferit.hr

Program Co-Chair

Tomislav Matić

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 224 602

e-mail: tomislav.matic@ferit.hr

Program Co-Chair

Mario Vranješ

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 224 670

e-mail: mario.vranjes@ferit.hr

Registration Chair

Marinko Barukčić

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 224 685

e-mail: marinko.barukcic@ferit.hr

Publication Chair

Robert Cupec

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 495 427

e-mail: robert.cupec@ferit.hr

Publicity/Web Chair

Danijel Topić

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 224 614

e-mail: danijel.topic@ferit.hr

Publicity/Web Chair

Josip Balen

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek,

Josip Juraj Strossmayer University of Osijek,

Kneza Trpimira 2b, Osijek, Croatia
+385 31 224 759

e-mail: josip.balen@ferit.hr

PROGRAM COMMITTEE

Danilo Babin
Ghent University, Belgium

Mato Baotić
University of Zagreb, Croatia

Adrijan Baric
University of Zagreb, Croatia

Ertuğrul Başar
Koç University, Turkey

Tin Benšić
J.J. Strossmayer University of Osijek, Croatia

Vlatko Cingoski
Goce Delčev University of Štip, North Macedonia

Zlatko Čović
Subotica Tech – College of Applied Sciences, Serbia

Mahdiyeh Eslami
Islamic Azad University – Kerman Branch, Iran

Damir Filko
J.J. Strossmayer University of Osijek, Croatia

Irena Galić
J.J. Strossmayer University of Osijek, Croatia

Sonja Grgic
University of Zagreb, Croatia

Francesco Grimaccia
Politecnico di Milano, Italy

Andrej Gubina
University of Ljubljana, Slovenia

Daniel Hofman
University of Zagreb, Croatia

Georges Kaddoum
University of Quebec, Canada

Tilda Karkour Akiki
Holy Spirit University of Kaslik, Lebanon

Tomislav Keser
J.J. Strossmayer University of Osijek, Croatia

Tatjana Konjić
University of Sarajevo, Bosnia and Herzegovina

Dražan Kozak
University of Slavonski Brod, Croatia

Dieter Kraus
Hochschule Bremen City University of Applied Sciences

Pawan Kumar
Thapar Institute of Engineering & Technology, India

Ramya Kuppusamy
Sri Sairam College of Engineering, India

Mario Kusek
University of Zagreb, Croatia

Edin Lakic
University of Ljubljana, Slovenia

Ivan Magdalenić
University of Zagreb, Croatia

Vlado Majstorović
University of Mostar, Bosnia and Herzegovina

Ivan Markovic
University of Zagreb, Croatia

Goran Martinovic
J.J. Strossmayer University of Osijek, Croatia

Tomi Medved
University of Ljubljana, Slovenia

Henry Muccini
University of L'Aquila, Italy

Gregor Papa
Jožef Stefan Institute, Slovenia

Marcin Paprzycki
Systems Research Institute of the Polish Academy of Sciences, Poland

Petra Pejić
J.J. Strossmayer University of Osijek, Croatia

Rene Prenc
University of Rijeka, Croatia

Kiran Ravulakolla
Woxsen University, India

Snježana Rimac-Drlje
J.J. Strossmayer University of Osijek, Croatia

Slavko Rupčić
J.J. Strossmayer University of Osijek, Croatia

Vasilija Sarac
Goce Delčev University of Štip, North Macedonia

Marija Seder
University of Zagreb, Croatia

Sebastijan Seme
University of Maribor, Slovenia

Dejan Skvorc
University of Zagreb, Faculty of Electrical Engineering and Computing

Damir Sljivac
J.J. Strossmayer University of Osijek, Croatia

Vlado Sruk
University of Zagreb, Croatia

Marinko Stojkov
University of Slavonski Brod, Croatia

Yuvaraja Teekaraman
University of Sheffield, United Kingdom

Domagoj Tolic
Rochester Institute of Technology, Croatia

Massimo Villari
University of Messina, Italy

Tomislav Volaric
University of Mostar, Bosnia and Herzegovina

Denis Vranješ
J.J. Strossmayer University of Osijek, Croatia

Mario Vrazic
University of Zagreb, Croatia

Ivan Yatchev
Technical University of Sofia, Bulgaria

Bruno Zorić
J.J. Strossmayer University of Osijek, Croatia

STEERING COMMITTEE

Nina Begičević-Ređep
University of Zagreb, Croatia

Leo Budin
Croatian Academy of Sciences and Arts;
University of Zagreb, Croatia

Matjaž Colnarič
University of Maribor, Slovenia

Bojan Čukić
University of North Carolina at Charlotte, USA

Gordan Gledec
University of Zagreb, Croatia

Francisco Gonzalez-Longatt
Faculty of Technology,
University of South-Eastern Norway, Norway

Sven Gotovac
University of Split, Croatia

Mislav Grgić
University of Zagreb, Croatia

Norbert Herencsar
Brno University of Technology, Czech Republic

Željko Hocenski
J.J. Strossmayer University of Osijek, Croatia

Ignac Lovrek
University of Zagreb, Croatia

Francisco Martínez Álvarez
Universidad Pablo de Olavide, Seville, Spain

Ngoc Thanh Nguyen
Wroclaw University of Technology, Poland

Srete Nikolovski
J.J. Strossmayer University of Osijek, Croatia

Milos Oravec
Slovak University of Technology, Slovakia

Hrvoje Pandžić
University of Zagreb, Croatia

Davor Pavuna
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Nedjeljko Perić
University of Zagreb, Croatia

Ivan Petrović
University of Zagreb, Croatia

Sasikumar Punnekkat
Mälardalen University, Sweden

Ivan Samardžić
University of Slavonski Brod, Croatia

Andrea M. Tonello
Alpen-Adria-Universität Klagenfurt, Austria

Mladen Vouk
North Carolina State University, USA

Drago Žagar
J.J. Strossmayer University of Osijek, Croatia

ORGANIZING COMMITTEE

Davor Begić
J.J. Strossmayer University of Osijek, Croatia

Domagoj Bilandžija
J.J. Strossmayer University of Osijek, Croatia

Dragana Božić Lenard
J.J. Strossmayer University of Osijek, Croatia

Viktorija Dozan
J.J. Strossmayer University of Osijek, Croatia

Mirela Glavaš
J.J. Strossmayer University of Osijek, Croatia

Mirta Hanzer
J.J. Strossmayer University of Osijek, Croatia

Željko Jeršek
J.J. Strossmayer University of Osijek, Croatia

Ružica Kljajić
J.J. Strossmayer University of Osijek, Croatia

Zorislav Kraus
J.J. Strossmayer University of Osijek, Croatia

Iva Perić Kuzmanović
J.J. Strossmayer University of Osijek, Croatia

Jasminka Majdandžić
J.J. Strossmayer University of Osijek, Croatia

Marina Maslać Popadić
J.J. Strossmayer University of Osijek, Croatia

Mario Miloloža
J.J. Strossmayer University of Osijek, Croatia

Marina Peko
J.J. Strossmayer University of Osijek, Croatia

Maristela Primorac
J.J. Strossmayer University of Osijek, Croatia

Krešimir Romić
J.J. Strossmayer University of Osijek, Croatia

Marko Šarić
J.J. Strossmayer University of Osijek, Croatia

Marijana Širić
J.J. Strossmayer University of Osijek, Croatia

Robert Šojo
J.J. Strossmayer University of Osijek, Croatia

Ana Šokčević
J.J. Strossmayer University of Osijek, Croatia

Tihana Vajnand
J.J. Strossmayer University of Osijek, Croatia

Davor Vrandečić
J.J. Strossmayer University of Osijek, Croatia

Stephen Ward
J.J. Strossmayer University of Osijek, Croatia

TECHNICAL CO-SPONSORS

region 8 IEEE
IEEE CROATIA SECTION

Ministry of Science and Education of the Republic of Croatia

J. J. Strossmayer University of Osijek

International Council on Large Electric Systems CIGRÉ, Croatian National Committee

Croatian Academy of Engineering

TABLE OF CONTENTS

Identifying, Managing, and Accessing Undefined Tuple States in Relational Databases

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

***Abstract*— Current trends in relational databases are formed by storing whole state evolution with emphasis on temporality. Input data must be proper, obtained in an adequate time, but mostly consistent. It can result in managing multiple undefined states, commonly represented by the NULL notation, which cannot be mathematically compared. Its reference using the index is also limited. Nowadays, NULL values are transformed using the trigger or by replacing them by the default values. Both strategies bring additional storage demands. Our proposed solution remains the original data but extends the index structure using the External or Referencer modules to ensure identification and performance. Whereas the origin of the NULL value can differ, emphasis is done on the categorization supervised by the multi-indexing layer or bitmap category mapper.**

***Keywords*— bitmap, index, null categorization, reference, relational database, undefined value**

I. Introduction

Databases have become the basis of data processing over the decades. Almost no information system can do without database support. Data amount is still rising, and individual approaches and architectures must respond adequately to ensure performance [1].

Relational databases were created in the 60ties of the 20th century and are still used. They are formed by the entities and relationships between them, focused on the data normalization. Access and management themselves are supervised by the database optimizer dealing with the relational algebra transforming the query into individual operations to access the data and compose the result set [5]. The main advantage of relational database processing is associated with data modeling and integrity. Each data table (entity) is defined by attributes, delimited by the unique names, data types, and additional constraints.

The whole database integrity is formed by individual rules associated with the particular attributes, sets of attributes, or applied on the table or relationships. It deals with the following categories [5][6]:

- Column integrity deals with the property of uniqueness, or duplicate values are allowed. It also covers the management of undefined values modeled by the NULL notation.
- User integrity extends the existing data modeling techniques by adding additional requirements for the data based on the specific domain of the used system.
- Referential integrity deals with the primary and foreign keys denoted by the relationships between the entities. In principle, a foreign key can reference either a primary key or any unique index. It can also hold an undefined (NULL) value in case of using an optional relationship type of the non-identifying relationship.
- Entity integrity focuses on the unique table identification by the primary key, which consists of the set of attributes that are unique as a value set and minimal, so the references are effective.
- Domain integrity deals with the data types and other constraints limiting the available list of values to be used. It can be supervised by the check constraints at the data model layer or trigger associated with the table can be used to secure the operation value during the change (Insert or Update statements).

The integrity of the system is covered by the transactions, which transform the database from one consistent state to another by checking the integrity rules. Thus, after approving the transaction, all rules must be applied. Otherwise, the transaction is refused by rolling back all the changes executed inside. Transaction in a relational database is maintained by logging in the UNDO and REDO structures, mostly highlighting the data change vectors. Therefore, the transaction can be reverted by such data. Moreover, building a consistent image is relevant, as well. Whereas the REDO is stored physically, after the instance failure, it is possible to reconstruct the database easily [6] [10] [13].

The data amount to be covered in the database layer has grown dynamically, and the progress is still enhanced. It results in several facts to be stated. Firstly, there are several architectures splitting the workload to multiple nodes by applying parallelism. Such techniques are also used to ensure availability. Therefore, there are numerous instances, usually physically dislocated to server the workload dynamically or by applying the stand-by mode.

Oracle database, which is also used for performance evaluation, uses single and multitenant architectures. The single-tenant approach was introduced in 1988 and was used until 2012. Oracle 12c introduced a multi-tenant container database containing Control file, Parameter files, Logs, and Metadata. Pluggable database is mounted dynamically on demand. It consists of the data files [5].

In 2017, sharded database architecture was proposed by managing connection pools by the shard directions. Sharded database is formed by multiple database servers – instance, as well as the database. It provides robust scalability, fault tolerance by geographic data distribution opportunity [11] [16].

The mission of this paper is to deal with the performance of the system, focusing on the undefined tuple states or the individual attributes.

As evident, performance and availability are part of multiple spheres, starting with the physical architecture, partitioning, and data distribution up to physical infrastructure delimited by the block size and storage perspective.

978-1-6654-8215-8/22/$31.00 ©2022 IEEE

In this paper, the main focus is done on the column integrity, and associated performance impacts related to the data processing and retrieval. Namely, NULL values can form a significant performance gateway. Firstly, undefined values cannot be mathematically compared. Any such operation results in getting NULL value as a result forming three-valued logics instead of binary (true and false). Secondly, NULL values cannot be directly sorted, so such value categorization inside the index is problematic. By applying B+tree indexing, undefined values are not part of that due to the comparison unavailability. As a consequence, locating undefined values requires sequential data scanning instead of index usage. And finally, although a particular value is not present, the value is often only partially undefined – it can be partially evaluated by the neighbor states. For the undefined validity, it is clear that such time has not been reached yet. Thus, although the particular time point value is not present, it is evident that it will occur in the future (if ever).

Although NULL values do not generally require additional storage capacity for the management, evaluation and data identification and retrieval can be demanding. Therefore, the DEFAULT option for a particular attribute often replaces NULL values. In this paper, there is a summary of existing approaches by focusing on the limitations. Then, the own solution using pointer referenced is introduced, supervised by the computational study.

Performance and storage perspective evaluations are done using the Oracle database system. The reason is related to the robustness of such a system providing the most complex features. Moreover, the implementation is covered by the CodeIn [17] and BeeApex [16] projects, which are devoted to the Oracle database system. However, the proposed solution is general and applicable to any database system and approach.

The paper is structured as follows. Section 2 deals with the current techniques for replacing an undefined value by the default option, focusing on the evolution and DEFAULT ON NULL clause. It supervises the process of data insertion and change management (Insert and Update operation). Section 3 focuses on the data retrieval by identifying and locating undefined value, which should be preferably done by the index. However, undefined values are not commonly part of the indexing. Transformation using the default value can bring the power and relevant solution. However, it brings additional storage demands compared to the NULL itself.

Therefore, the own solution is proposed by locating undefinition in the B+tree directly. Reflecting on the performance study, proposed B+tree enhancements can bring significant power by limiting sequential block scanning necessity. It is covered by section 4.

Finally, section 5 deals with the computational study focusing on the time processing of the data loading and retrieval process, supervised by the total storage demands.

II. Identifying and Replacing Undefined States – State of the Art

Undefined values are commonly modeled by the NULL notation. It does not require additional storage and is physically represented either by null, <null>, (null), or empty string can be used to visualize such component. The NULL value is considered as a marker of mission information but can also be used in case the value is not applicable. The sense of the NULL is not covered by any data type. The common representation is used across all domains.

L	R	L and R	L or R	not L
TRUE	TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE
NULL	TRUE	NULL	TRUE	NULL
NULL	FALSE	FALSE	NULL	NULL
NULL	NULL	NULL	NULL	NULL

Figure 1. Three-valued logic

Although NULL values do not require storage for the representation, they are hard for processing in the conditions. Arithmetical operations cannot be applied for NULL values resulting in getting NULL, which can significantly affect the conditional processing. Thus, the original bivalent representation (TRUE / FALSE) must be extended to cover the undefined value logic, forming a three-value mode. Fig. 1 shows the OR, AND, and NOT operations applied on 3-value logic. As evident, the NULL value is specific, and negation cannot be applied. Namely, the negation of the NULL is still NULL [7] [14].

Default

The core solution limiting non-present value is protected by the default option associated with the attribute. It is applied if the value is not stated. Thanks to that, undefined values are not present, which can even be forced by the column integrity constraint – NOT NULL. Thus, the attribute specification of the table is extended by the DEFAULT clause, like [2] [3] [5]:

```
create table sensor_tab_exp
(id integer,
 date_val date default sysdate,
 value integer default 1);
```

Default value specification can be either constant value or delimited by the function result. Since Oracle 12c version was introduced in 2012, it is possible to reference sequence pseudocolumns CURRVAL and NEXTVAL:

```
create table sensor_tab_exp
(id integer default seq_id.nextval,
 date_val date default sysdate,
 value integer default 1);
```

Based on [5], referencing sequence in the default option has some limiting factors to be highlighted. Firstly, the sequence must exist during the table definition. Hence, the owner must have the Select privilege on it. Secondly, any Insert operation fails if the sequence is dropped in the meantime. Thirdly, users performing the Insert operation must have sequence reference Select privilege, as well. Finally, by using sequences, gaps can be identified due to caching or rolling back the transaction. Any value request means the potential usage. Thus, the sequence cannot be shifted backward.

In the past, default values were applied only if the particular attribute value was not stated. Thus, it did not secure undefined values properly. Namely, if the NULL value was defined explicitly, the default value option was not applied, resulting in raising an exception if the NOT NULL constraint was associated with such column:

```
create table sensor_tab_exp
(id integer default seq_id.nextval,
 date_val date default sysdate,
 value integer default 1);

insert into sensor_tab_exp(value)
  values(null);

select count(*)
  from sensor_tab_exp
    where value is null;              ==> 1
```

The previous section used default values only if the column is not explicitly referenced. Oracle 12c has introduced the default option extension by focusing on the NULL values stated explicitly – DEFAULT ON NULL enhancement. Therefore, even though the value is stated explicitly, the default option can be applied if the particular value is undefined, represented by the NULL value [5].

```
create table sensor_tab_exp
(id integer default seq_id.nextval,
 date_val date default sysdate,
 value integer default on null 1);

insert into sensor_tab_exp(value)  values(null);

select count(*)
  from sensor_tab_exp
    where value is null;              ==> 0
```

Above stated represents the final solution limiting undefined values in the relational systems, delimited by the default option management, which is robust and can cover any conditional transformation.

The different stream dealing with the undefined value that can be used across the database system versions is a trigger. It can be associated with any data manipulation operation. Oracle database can group multiple operations on one table to the common block. The function usually encapsulates it in other systems, which is called for each operation. One way or another, trigger definition is more powerful by referencing the state to be operated using the NEW record of the trigger. An analogous solution dealing with the default value is stated in the following code block [12] [15].

```
create or replace trigger trig_null
 before insert on sensor_tab_exp
  for each row
   begin
    if :new.date_val is null
        then :new.date_val:=sysdate;
    end if;
    if :new.value is null
        then :new.value:=1;
   end if;
   end;
```

In the performance evaluation section, individual approaches are compared reflecting the time cost demands and the storage demands. It is evident that the physical value transformation brings additional storage demands based on the used data type. Compared to the NULL, which can be used generally for any data type, and it does not require storage capacity at all.

By using physical transformation, particular values can be part of the indexing, and the data retrieval process is easy and straightforward, limiting the sequential block necessity.

III. Indexing

The database index forms a robust access model during the data retrieval process. Instead of sequential scanning of all associated blocks, traversing across the index can be used to locate relevant data portions. A typical structure used in database systems is B+tree [4] [8] [9], which is always balanced, and the leaf layer blocks are sorted based on the index key. The reference to the database layer is done by the address pointers specifying the data file, data block, and position of the row inside the block (ROWID). Thus, the data retrieval process is delimited by two stages. Firstly, the index access path extracts individual ROWIDs, which pass the defined query conditions. Secondly, the Table Access by Index Rowid method is used, which takes the ROWID set from the previous step and loads particular blocks into the instance memory for the evaluation. Such a process is typically significantly cheaper and more efficient than the sequential scanning – Table Access Full (TAF). Namely, the TAF method can degrade if the data block fragmentation is used or dynamic data update operations are present, resulting in storing a partially free block set.

In the B+tree indexes, the mathematical operations do the traverse path. However, they cannot cover undefined values, resulting in refusing NULL values part of the B+tree indexing. Thus, if the query references an attribute that can hold undefined value, B+tree index is not used to shift the processing to the TAF method, which can be really demanding. Therefore, undefined value transformation brings sufficient power, whereas NULLs are not present in the system. On the other hand, physical value transformation requires additional capacity for block storage.

Another solution is formed by the function-based index [4], by which the negatives can be limited. Undefined values can be directly physically placed in the database, while during the retrieval, they are transformed to the required form using the function result. Thus, the index does not cover the pure attributes, but they are enclosed by the function calls, which results are part of the index. The particular function must be deterministic. Concluding the function-based index definition, it provides a robust solution by limiting additional database storage demands, the transformation is done dynamically, and the index stores such results. However, also this technique brings some negative aspects. Firstly, function transformation extends the processing time of the index management covered by the Insert operation, whereas the function result must be calculated before processing and referencing the tuple by the index. Secondly, the defined function must be robust and distribute the data as uniformly as possible, removing the data cluster potential. And thirdly, that function must be explicitly and precisely used and referenced in the query. Otherwise, the index will not be used. Thus, it is not enough to refer to the source attribute itself.

Focusing on the performance and existing functionality, it is clear that there are two streams, either by limiting NULL values supervised by the default option or trigger (in that case, additional storage demands are present), or transformation is done via the index. However, it requires functional coverage by the query. Therefore, although the translation profile can do it, it can be non-reliable, influencing the original statement.

Therefore, specific B+tree enhancements are proposed in the following section to cover undefined values in the indexes as a particular module. It combines both approaches by reaching optimized solutions, emphasizing the storage demands and capacity, but mostly performance is ensured.

IV. PROPOSED SOLUTION

The proposed solution extends the B+tree definition by focusing on undefined value management. Therefore, from the storage perspective, original NULL values are treated, limiting the additional storage demands. However, for the purposes of the index, dynamic transformation is done to reflect the index structure. The first proposed solution (**SOL1**) extends the leaf layer of the index by the external module referencing undefined representations of the index key values. Such a module is placed either in the leftmost or rightmost part. The limitation of that solution is reflected by the path traversing necessity to locate undefined states. The different solution (**SOL2**) locates undefined state module reference just to the root node so that they can be directly accessed.

Using the above solutions, undefined values are not categorized by placing them into a common structure using the NULL transformation. Fig. 2 shows the architecture of the proposed SOL2. Internally, whereas the NULL values cannot be mathematically compared, individual references are then stored in a linear linked list with no specific order. As a result, to obtain and locate undefined states, a particular external module must be fully scanned. As evident, if the number of (partially) undefined states is high, the proposed solution can be too demanding, even degrading the performance up to original sequential block scanning. Notice that the external module must be memory-loaded in the first phase, followed by the extraction process. Then, the ROWID values are obtained, forcing the system to load the blocks into the memory Buffer cache for the consecutive processing and evaluation. If the size of the external module is large, almost all table-associated blocks need to be processed. However, the processing itself is preceded by the index management itself.

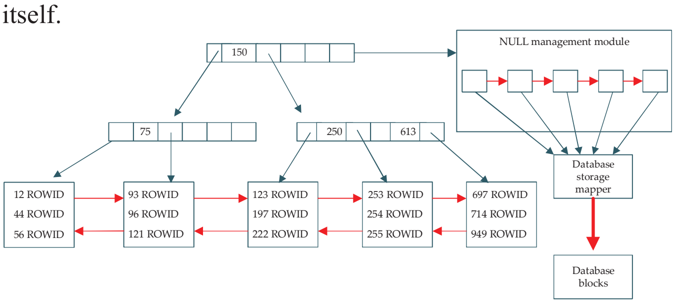

Figure 2. SOL2 architecture – NULL management module

Therefore, the third solution (**SOL3**) uses another approach. Instead of using a linked list with no sorting (ordering) criterion, the Referencer module using B+tree is used. The primary key secures the order of individual elements placed there. A relational database requires each table to have the primary key for the tuple identification. It is delimited by the unique constraint, as well as the NOT NULL value for each element. The set for the primary key definition can consist of multiple attributes (composite primary key) but should be minimal to ensure performance. Thus, a unique constraint would be lost if any element was removed from the primary key attribute set.

For the proposed third solution (SOL3), the index key differs from the Referencer module state identification. Therefore, it can be said that the global solution uses the index inside the index.

The difference between the External and Referencer modules is based on the internal structure. The external module uses a linked list, whereas the Referencer module uses the index, mostly composed of the B+tree.

A. *Referencer module sharing*

External or Referencer modules can benefit from the fact that they are only logically interconnected with the index itself. The association is done by the pointer to the particular storage elements, mostly mirroring the root node or the first node, generally. Therefore, multiple indexes can share a common structure without specifying their own. Thanks to that, particular External or Referencer modules are typically already loaded in the instance memory, so the loading operation does not need to be present. It is a precise consequence of such structure sharing. Frequent accesses to these structures keep them in the memory. Even if the memory needs to be freed, reloading will only be a matter of time. In addition, it is still there, at least partially.

Global sharing

In common conditions and environment, the index covers all the data tuples. Thus, the added modules can be completely shared. So, even if several indexes are present in the system for the specific table, the External or Referencer modules are added just once. Statement evaluation and parsing process are listed in fig. 3. In the Parse operation, data access method is selected based on the optimization and statistics. Bind applies dynamic conditions. Execute and Fetch provide the process of retrieval and result set composition.

Whereas the transaction can be approved only after applying changes to the whole index set to ensure reliability and consistency, it is strongly recommended to use the post-indexing layer, by which several change operations can be grouped and applied at the same time [7] [8].

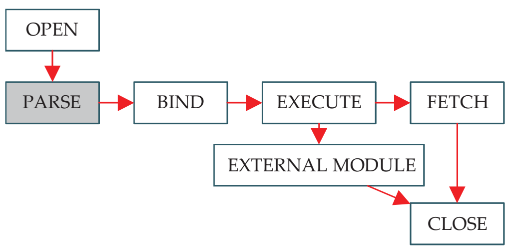

Figure 3. Statement execution process

Partial sharing

By moving the processing to the temporal sphere, multiple structure enhancements can be present to monitor and ensure performance. Namely, current valid states have the highest priority, which is consecutively lowered over the time flow. Thus, historical values are gradually losing their importance, which is reflected by the index sets, as well. From the architecture point of view, techniques of data distribution [5], partitioning [4] [11], stand-by types [5], or warehouses [9] are present. As a result, individual indexes are local and point only to the data subset. E.g., one partition reference only the current valid state, while the historical images are separated by the expiration date. Each partition can have own index to ensure parallel access, identification, and processing. Therefore, global sharing of the External or Referencer module is not applicable.

There are two options available, either total unsharing or just partial sharing. Partial sharing is more convenient and preferable. At least some data demands on the index layer can be reduced. In that case, the module itself is extended by the shared and own local data, which can be identified and processed separately, but still in parallel. Thus, three separate structures can be identified:

- index with defined values (mostly modeled by the B+tree index structure),
- local pointers to the undefined data states,
- shared modules (External / Reference).

B. Undefined state categorization

In the above section, there was a discussion related to the undefined value management and transformation. The aim is to ensure indexing possibilities. However, undefined states or individual attribute values themselves can originate from various sources – non-applicable value, value not known, non-reliable value occurrence, value out of range, delayed value, etc. Thus, the next step is to provide categorization opportunities. Input values are evaluated by extracting NULL values, which are then categorized and indexed. Categorization can be applied in both structures – The external module, as well as Referencer module. For the purposes of the performance evaluation, we will focus only on the indexing, which allows traverse opportunity and reflect performance aspects.

Multi-tree

The first proposed solution is done by the multitree index, formed by two layers. The first layer deals with the categorization, which can be formed either by the linked list (**SOL4a**) or B+tree can be used (**SOL4b**). In that case, the traverse is done by priority, so the most relevant sources can be obtained sooner. Regarding the [9], it can lower demands for the huge data sources. Vice versa, if the data source size is not extremely large, performance impacts are minimal and do not provide significant benefits.

The second layer consists of the B+tree for each undefinition source type. A particular solution is optimal if the user wants to query data based on the categorization. On the other hand, if specific objects are to be referenced, they can be spread across multiple indexes, which must be scanned either sequentially or in parallel.

Fig. 4 shows the architecture of the multi-tree index.

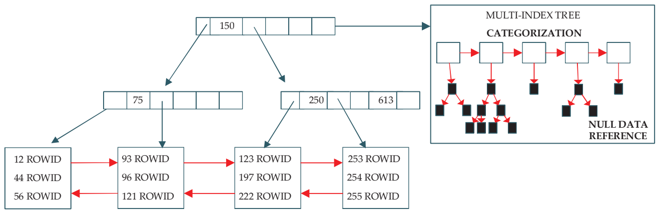

Figure 4. Multi-index tree

Bitmap

The limitation of the B+tree indexing inside the Referencer module is just the mapping association, which must always be 1:1. Namely, the origin of the undefinition must be part of one category only.

In the real environment, the assignment can be done to multiple categories, like delayed and inconsistent data simultaneously. However, to ensure robustness, each relational database tuple should be present in the system only once. But by capturing multiple undefinition categories, a particular tuple would be part of numerous indexes violating the integrity.

Therefore, the proposed **SOL5** uses a bitmap index to create the Referencer module. Individual rows inside the index are sorted based on the primary key as the index rule. Bivalent variables are expressing the undefinition category assignment. Thanks to that, each tuple can be part of multiple categories. Identification is made by applying superfast conditions, so the processing and evaluation are straightforward and performant. Adding a new undefinition type is straightforward. It requires adding one column to the bitmapper. By default, a particular type is not used for the existing tuples:

Alter **Reference_module**
  manage table <*table_name*>
    add **undefinition_source** <*description*>
        [default {0 | 1}];

Type reference

The last solution (**SOL6**) uses a reverted approach, Referencer module is formed just by one index, and individual categories are located on the leaf layer for each date tuple. Categorization association is modeled either by the linked-list (**SOL6a**) or B+tree (**SOL6b**). Each leaf layer element points to the data storage (ROWID) but also to the undefinition category – NULL_category module. Mapping is one-to-many so that each node tuple can be associated with multiple categories complicating the architecture.

Regarding the optimization techniques, it can be easier to create the list of all possible values and reference the code representation itself (**SOL6c**).

V. Performance Study

Performance evaluation has been executed using *Oracle Database 19c Enterprise Edition* (Release 19.0.0.0.0 – Production) with the following server parameters: *processing unit:* Intel Xeon E5620; 2,4GHz (8 cores), *operation memory:* 16GB, *SSD drive:* 1 000GB.

The database consisted of the spatio-temporal data locating and identifying airplane objects by the occurrence time, GPS position, as well as other parameters – speed, destination,

```
"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"
"186858227","1","EGGXOCA","01-06-2015 04:08:00","01-06-2015 05:01:00"
"186858227","2","EISNCTA","01-06-2015 05:01:00","01-06-2015 05:34:00"
"186858227","3","EGPXCTA","01-06-2015 05:34:00","01-06-2015 06:18:10"
```

Figure 5. Data tuples

current airspace association (entry and exit time), planned route vs. current route, as well as the weather conditions influencing the flight itself. There were 20 attributes. Each state has begun and end point of the validity (entry and exit time). There are *1 808 390* data tuples in total. Examples of the data are in fig. 5.

Undefined states were originated from the sensor failure, non-reliable communication network, and processing delays.

A performance evaluation study can be divided into two experiments. The first part deals with state-of-the-art managing physical transformations using the trigger and default option. The second part emphasizes the indexing, focusing on the proposed methods and architectures.

A. Performance study - physical transformation

For evaluating costs and processing time demands of the physical transformation, the following models were used. The reference system (M0) requires no transformation, whereas undefined values are not present at all. The aim is to point to the additional demands covered by the identification and processing. On the other hand, the current attribute value requires storage capacity and operations of the loading and block allocation:

M0 – no-undefined values present (Referencer model).

M1 – undefined data caused by the time delays of the input data.

M2 – data set is provided at the required time, but the data quality is not suitable due to a communication channel or broken sensor.

M3 – this data model is associated with processing delays, as well as non-reliable data processing. Thus, the NULL value can be categorized into two segments.

Tab. 1 shows the processing time demands (expressed in seconds). Undefined states are not explicitly identified nor transformed. For the processing, 100 000 row tuples were associated with the undefined state (NULL). NULL values have no size demand reaching the reduction in the block allocation and I/O operations. Based on the reached results, data types of the undefinition do not play a significant role. Results of the M1 and M2 are analogous. In the first case, time validity is not present, whereas M2 is delimited by missing GPS positions. Compared to M3, both undefined values are present, getting 51.36% reduction of the total processing time costs (M0), 43.22% for M1, and 44.84% for M2.

TABLE I. RESULTS – NO TRANSFORMATION

Model	M0	M1	M2	M3
Processing time demands (s)	17.409	14.915	15.352	8.468

TABLE II. RESULTS – PURE DEFAULT

Model	M0	M1	M2	M3
Processing time demands (s)	18.101	15.024	15.431	8.512

The second evaluation stream in this category is associated with the Pure default. The undefined state can be defined explicitly by the NULL, which is, however, not transformed using pure default. In total, 50 000 rows were defined explicitly, while the rest 50 000 rows do not state the value. Tab. 2 shows the results. As evident, there is no significant difference between the previous solutions. Thus, extending the solution by default option does not bring significant additional evaluation costs:

- M0 3.97%
- M1 0.73%
- M2 0.51%
- M3 0,52%

In the case of using the Default on NULL option, all undefined values are identified and transformed using predefined values. The performance varies on the used model, ranging from the 4,08% improvement up to 2.60% slowdown (processing time costs). Tab. 3 shows the processing time demands using the Default on NULL option.

Introduced Default on NULL option in oracle 12c is powerful and can transform undefined values defined explicitly, as well as non-present data themselves. A similar solution obtained requires trigger firing for each tagged data row. As evident, it requires high additional costs. Namely, for one undefined sphere management, compared to the Default on NULL option, additional demands are 47,04% (M1) and 22,44% (M2). If two undefinition categories are maintained, the reached difference is even sharper – 81,14%. The reason is the trigger firing necessity. Firstly, the trigger execution code in a parsed form must be instance loaded and associated with each changed operation, preceded by the NULL value identification. Finally, the data row record is enhanced by the transformation and processed. Tab. 4 shows the results for trigger management.

Grouped results in a graphical form are in fig. 6.

Default on NULL provides sufficient power if available in the database system version. If not, a trigger must be used, but additional processing time demands can range up to 80%.

TABLE III. RESULTS – DEFAULT ON NULL

Model	M0	M1	M2	M3
Processing time demands (s)	17.362	15.019	15.212	8.733

TABLE IV. RESULTS – TRIGGER

Model	M0	M1	M2	M3
Processing time demands (s)	20.488	22.084	18.626	16.081

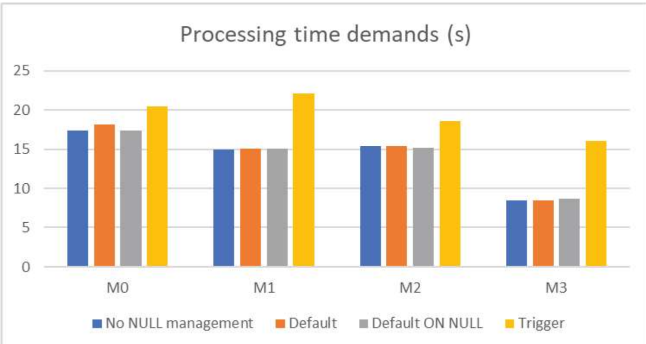

Figure 6. Processing time demands

Although limiting undefined values can be powerful in terms of processing and demands, storage capacity has to be extended based on the data type associated with the undefined value attribute. For date and positional values, original storage demands are 25600 KB. Transformation requires 49152 KB in total. 8 KB block was used.

B. Performance study – dynamic index management

Input stream transformation offers a robust solution, but brings additional demands. Instead of a NULL definition, a specific value is stored. It can even be problematic to identify such value if the domain is unlimited. E.g. if the value can be any integer, an undefined value cannot be transformed, whereas it would not be possible to distinguish between the original and transformed value. The second evaluation stream deals with indexing and focuses on the structures managing undefined values. **SOL0** does not cover undefined values, therefore, sequential block scanning must be done. It is used for reference. **SOL1** locates the External module on the leaf layer of the B+tree index, whereas **SOL2** points to it directly from the root. Undefined values can be sorted for easier identification and location, therefore, **SOL3** uses the principle of sorting data based on the primary key, which should always be present.

SOL4a and **SOL4b** use multi-tree index approach, shaped by the linked-list (**SOL4a**) or B+tree (**SOL4b**) is used internally. Bitmap index categorization is done in **SOL5**.

Finally, reference list categorization can be supervised either by linked-list (**SOL6a**) or B+tree (**SOL6b**). As evident, these approaches do not benefit, whereas they are not dynamic. By adding a new undefinition element, the whole index must be reconstructed, marking the original type as invalid.

Tab. 5 shows the results in terms of processing time of the data retrieval and total costs of the operation.

SOL0 does not manage NULL values inside the index reaching sequential block scanning necessity (Table Access Full). The problem arises if the blocks are fragmented, or even empty blocks are present. In that case, even such blocks need to be memory-loaded, so the efficiency of the whole solution is getting worse and worse. SOL1, SOL2, and SOL3 form a specific category dealing with the External modules with no particular format. The emphasis is done on the location of such module inside the index. Traversing across the index requires 0,16s, which represents 3.88%. The second category delimits multi-tree index structures (SOL4a and SOL4b). Specific primary key sorting allows faster location. Instead of the structure searching, traverse activity can be done, reducing the demands up to 0.27s (7,48%).

TABLE V. DYNAMIC INDEX MANAGEMENT - PROCESSING TIME AND COSTS

Model	Processing time (s)	Costs
SOL0	38.21	103
SOL1	4.28	19
SOL2	4.12	18
SOL3	3.93	16
SOL4a	3.88	15
SOL4b	3.61	15
SOL5	3.37	14
SOL6a	4.44	19
SOL6b	4.37	19

Moreover, multi-index can cover several undefinition categories. The one-to-many mapping can be done. Compared to the SOL3, even better performance can be reached, whereas the evaluation can be done in parallel – for each category, one extra slave scanning process can be assigned. Reference lists are not relevant. They cannot extend the structure without rebuilding the particular reference list is placed for each node making the management hard to maintain.

The solutions have been tested using Oracle 19c database system. However, whereas they are based on the B+tree index structure, which is part on any database system, proposed techniques are universal and generally applicable.

VI. CONCLUSIONS

Dealing with data management requires accessing and maintaining individual states with regard to reliability and integrity. Relational database transaction ensures the shift from one consistent state to another consistent. Data input can be influenced by various factors resulting in delays or getting data out of the range. When dealing with the sensor-based network, the problem can be even sharper due to wireless infidelity or outages. It results in getting undefined values, mostly modeled by the NULL notation. Although such an approach is suitable in terms of database storage demands, performance can degrade due to the inability to use the index. The existing solutions dealing with the undefined values or whole states are based on transformation in the physical layer. It is done by the trigger or default value used in the case of non-present data. As a result, NULL values are not present at all. Thus, the index can benefit by improving the overall data retrieval performance. On the other hand, physical transformation increases the storage capacity demands. If the data types are complex, additional requirements can range significantly. Another existing perspective is associated with the function-based index, which limits undefined values in a logical layer. However, the disadvantage is that it is necessary to use this function also in the query. Otherwise, index cannot be used.

Our proposed solution interconnects both principles by introducing a specific module associated with the index, by which the undefined values can be covered. Sortability is ensured by the primary key or the linear linked list is used with no specific order. Thanks to that, original NULL values can be directly stored and indexed. Discussed solutions can also manage undefined value categorization by multiple architectures.

In the future, we would like to emphasize the data distribution and various database instance architectures mirroring and fragmenting data. Physical infrastructure in terms of data block size will also be the object of interest. It is assumed that various block size can significantly influence the performance - data loading, as well as the capacity and evaluation process. Moreover, by grouping individual states into categories physically located in the same segment can bring additional power correlating the neighborhood.

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] 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. Apress, 2021, ISBN – 9781484274989

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

[7] M. Kvet, "Managing, locating and evaluating undefined values in relational databases". Conference ICITS 2021. February 2020.

[8] M. Kvet and M. Kvet, "Relational pre-indexing layer supervised by the DB-index-consolidator background process". Conference of Open Innovation Association, FRUCT. 2021-January, (2021).

[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. 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

[14] 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.

[15] R. Wang, B. Salzberg, D. Lomet, "Transaction support for log-based middleware server recovery," Proceedings - International Conference on Data Engineering. 353–356 (2009). https://doi.org/10.1109/ICDE.2009.45

[16] R. Leskovar, et al. "Erasmus+ project Better Employability for everyone with APEX." https://beeapex.eu/ (accessed Aug. 1, 2022)

[17] F. Urem, et al. "Erasmus+ project Cloud cOmputing for Digital Education Innovation. "https://code-in.org/ (accessed Aug. 1, 2022)