20-22 June 2023

Zilina - Slovakia

Proceedings of

The International Conference on

Information and Digital Technologies 2023

IEEE

Visegrad Fund

ESRA

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194438

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194460

The International Conference on

Information and Digital Technologies 2023

20-22 JUNE 2023
ZILINA, SLOVAKIA

ISBN 979-8-3503-0586-9
ISSN 2575-677X
IEEE Catalog Number CFP23CDT-ART

The International Conference on Information and Digital Technologies 2023

IEEE Catalog Number: CFP23CDT-ART
ISBN: 979-8-3503-0586-9
ISSN: 2575-677X

Conference Secretariat Address for Contacts:
IDT'2023 Organizing Committee
Department of Informatics
University of Zilina
Univerzitna 1, 01026, Zilina, Slovakia
email: idt@fri.uniza.sk

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194461

Scientific Program Committee

Chair: Cepin Marko, Slovenia
Co-Chairs: Zaitseva Elena, Slovakia

Ablameyko Serge, Belarus
Baraldi Piero, Italy
Beer Michael, United Kingdom
Berenguer Christophe, France
Brinzei Nicolae, France
Bris Radim, Czech Republic
Coolen Frank, United Kingdom
Czapp Stanislaw, Poland
Deserno Thomas, Germany
Di Maio Francesco, Italy
El-Nouty Charles, France
Filatova Daria, Poland
Fiser Petr, Czech Republic
Izonin Ivan, Ukraine
Kameyama Michitaka, Japan
Kharchenko Vyacheslav, Ukraine
Kolowrocki Krzysztof, Poland
Kovalenko Andriy, Ukraine
Kovtun Viacheslav, Ukraine
Kor Ah-Lian, United Kingdom
Kostolny Jozef, Slovakia
Krsak Emil, Slovakia
Kvassay Miroslav, Slovakia
Kvet Michal, Slovakia
Levashenko Vitaly, Slovakia
Levitin Gregory, Israel
Lukac Martin, Japan/Kazakhstan
Luntovskyy Andriy, Germany
Majernik Jaroslav, Slovakia
Moraga Claudio, Germany
Muhamedyev Ravil, Kazakhstan
Podofillini Luca, Switzerland
Rauzy Antoine, Norway
Soda Paolo, Italy
Stankevich Sergey, Ukraine
Stankovic Radomir, Serbia
Steinbach Bernd, Germany
Subbotin Sergey, Ukraine
Sztrik Janos, Hungary
van Gulijk Coen, The Netherlands
Xie Min, Hong Kong
Yanushkevich Svetlana, Canada
Zio Enrico, Italy

Organizing Committee

Chair: Sedlacek Peter, Slovakia

Blahova Linda, Slovakia
Ilovska Anna, Slovakia
Macekova Denisa, Slovakia
Mrena Michal, Slovakia
Piatrikova Lucia, Slovakia
Rabcan Jan, Slovakia
Rusnak Patrik, Slovakia
Vaclavkova Monika, Slovakia

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194432

Reviewers

Ablameyko Serge
Baraldi Piero
Beer Michael
Belotserkovsky Alexei
Berenguer Christophe
Bolvashenkov Igor
Brinzei Nicolae
Bris Radim
Cepin Marko
Chatelet Eric
Coolen Frank
Czapp Stanislaw
Deserno Thomas
Di Maio Francesco
El-Nouty Charles
Filatova Daria
Fiser Petr
Frenkel Ilia
Hovorushchenko Tetiana
Izonin Ivan
Jämsä Timo
Kameyama Michitaka
Kharchenko Vyacheslav
Koháni Michal
Kolowrocki Krzysztof
Kor Ah-Lian
Kostolny Jozef
Kovalenko Andriy
Kovtun Viacheslav
Kozlova Anna
Krsak Emil
Kvassay Miroslav
Kvaššayová Nika
Levashenko Vitaly
Levitin Gregory
Lukac Martin
Lukyancuk Igor
Luntovskyy Andriy
Majernik Jaroslav
Moraga Claudio
Muhamedyev Ravil
Nedzved Alexander
Pastor Luis
Pattinson Colin
Perkowski Marek
Podofillini Luca
Rabcan Jan
Rusnak Patrik
Rauzy Antoine
Sedlacek Peter
Shmerko Vlad
Simic Zdenko
Soda Paolo
Stankevich Sergey
Steinbach Bernd
Subbotin Sergey
Sztrik Janos
Tatur Mikhail
Xie Min
Zio Enrico

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194434

Preface

We have the pleasure to present to you the Program of the International Conference on Information and Digital Technologies (IDT 2023). IDT 2023 provides a forum for the presentation and discussion of scientific contributions covering the theories and methods in the field of information and digital technologies, and their application to a wide range of industrial, civil, and social sectors and problem areas. IDT 2023 is also an opportunity for researchers, practitioners, academics, and engineers to meet, exchange ideas, and gain insights from each other. IDT 2023 offers a multidisciplinary platform to address information systems' technological, societal, and financial aspects.

The conference program includes several workshops that cover numerous trends, problems, and aspects of information and digital technologies:

- International Workshop on Reliability and Safety (RaS);
- International Workshop on Earth Observation for Early Warning of Land Degradation at European Frontier (EWALD);
- International Workshop of Advanced Centre for PhD Students and Young Researchers in Informatics (ACeSYRI);
- International Workshop on Biomedical Technologies (BT).

The International Workshop on Reliability and Safety is organized under the support of Visegrad Grant Programme of the International Visegrad Fund by the project "*Exchange Reliability and Safety Experience in the V4 region*" (ERaSEV4) with reg.no.: 21830194. Collaboration between academic staff, representatives of enterprises and young researchers in the areas of Reliability and Safety will be established at this event. The modern trends of teaching and research aspects of the RaS will be discussed at the workshop. The conception of the RaS network in V4 will be discussed and clarified. Together with the International Visegrad Fund, this workshop is supported by the Slovak Research and Development Agency by the project "*New methods development for reliability analysis of complex systems*" (reg.no. APVV-18-0027).

Land degradation is the world's greatest environmental challenge affecting the environment, agriculture, and human well-being. Intensified by natural disasters and desertification, land degradation may present potential risks and socioeconomic tension. IT is one of the perspective tools for observing and eliminating such effects. The workshop on Earth Observation for Early Warning of Land Degradation at European Frontier proposes a platform for the discussion of relevant trends in the Early Warning System, responses to land degradation, and innovative geo-informational frameworks and solutions.

The workshop EWALD is organized with the support of the project of the same name EWALD which has received funding from the European Union's Framework Programme for Research and Innovation Horizon Europe – the Framework Programme for Research and Innovation (2021-2027), Grant Agreement No. ID 101086250. And this workshop develops of ideas for the project "*Risk assessment of environmental disturbance using Earth observation data*" (reg.no. SK-UA-21-0037) of the Slovak Research and Development Agency.

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194431

This workshop ACeSYRI is organized as an event of the ERASMUS+ project "*Advanced Centre for PhD Students and Young Researchers in Informatics*" (ACeSYRI, reg.no. 610166-EPP-1-2019-1-SK-EPPKA2-CBHE-JP). Also, this Workshop is post-project event of the project "*Centers of Excellence for young RESearchers*" (CERES, 2014-2017) - 544137-TEMPUS-1-2013-1-SK-TEMPUS-JPHS.

Co-funded by the Erasmus+ Programme of the European Union

This ACeSYRI workshop is intended for PhD students, young researchers, and educators. The main goal of the workshop is a presentation of ACeSYRI project results to representatives from the EU and other countries. The young researchers can present at the workshop with their lectures about own scientific research. The next aim is to establish and expand international contacts and co-operation of young researchers.

Initially, more than a hundred contributed papers were submitted for the review. Approximately half of these submissions have been recommended by reviewers for presentation at the Conference and publication in the proceedings. The review process was mainly organized by the Workshop chairs and the process was made by a large number of reviewers, which are gratefully acknowledged for their contributions to the improvement of quality of the accepted papers. Each paper was reviewed by at least two anonymous reviewers in order to ensure fair and high-quality reviews. Our organization team is grateful all reviewers who help us to select papers for the presentation at the Conference and publication in the proceedings.

In addition to regular sessions, IDT 2023 offers distinguished keynote lectures. At the conference will be presented lectures:

- *Self-sufficient electric energy supply at home* by prof. Marko Čepin (University of Ljubljana, Slovenia);
- *Benefits of Petri nets for systems modeling and probabilistic assessment in reliability engineering* by prof. Nicolae Brinzei (University of Lorraine, France);
- *Maintenance optimization of complex multi-component systems* by prof. Radim Briš (VSB—Technical University of Ostrava, Czech Republic);
- *Multi-Diagnosis Cough Classification Evaluation* by prof. Martin Lukač (Hiroshima City University / Nazarbayev University, Japan / Kazakhstan);
- *Data-driven decision making in practice: Experiences in academia and government* by Dr. Martin Komenda (Masaryk University, Czech Republic).

We thank Keynote Speakers for offering their unique perspectives on information technologies at the Conference.

We gratefully acknowledge the Faculty of Management Science and Informatics of the University of Žilina, European Reliability and Safety Association (ESRA), the Czechoslovakia section of IEEE, and the Reliability Society Chapter of IEEE Czechoslovakia Section for the sponsoring, organizational and technical support.

We also thank all the contributed paper authors for their submissions and presentations.

Organization team of IDT 2023

UNIVERSITY OF ŽILINA
Faculty of Management Science and Informatics

Contents

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194401

Using Boolean Data Type in Oracle Database - Performance Study

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

2023 International Conference on Information and Digital Technologies (IDT) | 979-8-3503-0586-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDT59031.2023.10194397

***Abstract*—Boolean data type was unavailable in Oracle Database for a long time, resulting in many workarounds negatively influencing the data retrieval and overall processing performance. In April 2023, Oracle Database version 23c was released by offering Boolean data types for SQL use. This paper deals with the performance study, modeling, and representation impacts. It summarizes existing solutions focusing on various data types, followed by integrity management by forming three-valued logic if undefined values can be present. Besides, this paper highlights implicit conversions and indexing strategies for the Boolean data type or its variants. Thus, it aims at defining a methodology to ensure performance in terms of processing time, indexing, and storage demands.**

Keywords—Oracle Database 23c, Performance, Boolean, Indexing

I. INTRODUCTION

Data integrity, security, and overall reliability are critical parts of information technology. Data management is an inseparable part of processing. In the past, data were embedded in applications or information systems. However, currently, it is inevitable to form a separate layer for data storage and representation. There are many database systems that differ in structure, processing approaches, and storage layer.

Relational databases were introduced in 60ties of the 20th century. They have been in practice for more than 60 years and are still very often used. The primary aspect relates to the processing efficiency, defined by the relational algebra [5], and integrity, operated by the Structured Query Language, which compared to the data embedding, is a non-procedural language, so the physical layer and indexing are not impacted by the physical data structures, layouts or even database architectures. Moreover, the data are checked and encapsulated by the transactions ensuring that all integrity constraints are passed at the latest at the end of the transaction. Thus, transaction shifts the database from one consistent state to another, which is protected by the integrity rules, as well [9] [10]. Integrity constraints can be categorized to the column (uniqueness of the attribute and definition, whether it can hold undefined value or not), domain (data type or check constraints limiting the available values for the attributes), entity (primary key as a unique identifier of the row), and referential (foreign key referring primary key candidate (primary key or unique constraint)) integrity. Among specified, user integrity is also present, setting application rules for validity and data consistency [10] [11]. Relational databases are part of any information technology and applicability [4] [8] [12].

Data integrity rules can evolve over time [12] [13], as well as the data model characteristics can change over time [14] [15]. In any case, relational databases and mechanisms ensure the correctness and appropriateness of the data stored in them.

Relational databases have become a default storage strategy soon after the first database releases. Each data model is characterized by entity and relationship sets. An entity consists of at least one attribute (column), and each of them is bounded by the data types. Individual database systems provide a wide range of data types for the character strings, numerical values, files, and Large objects (LOBs). A specific category is formed by the date and time processing and Boolean. Although both of them are part of the SQL and ANSI standards, they strongly differ across the database systems and individual versions.

This paper deals with the Oracle Database, which is a leader of database and information strategy. There are various reasons for selecting it, namely, based on multiple performance strategies [2] [6] [14] [15], Oracle provides the best solution for complex and robust information systems. Moreover, it offers autonomous databases in Cloud [1] [7], which are self-managed, patched, and optimized using advanced machine learning and artificial intelligence techniques. Last but not least, this paper is supported by the Erasmus+ project EverGreen [24] in cooperation with Oracle Corporation and Oracle Academy. This paper emphasizes the Boolean data type definition and its internal management. For decades, Oracle Database did not provide sufficient power and description for holding Boolean values directly in the SQL. Instead, a similar representation had to be modeled using a character string or numerical variant. We, therefore, summarize existing principles by evaluating them in processing time and storage demands. In April 2023, a significant milestone was introduced, defined by the release of the Oracle Database version 23c, which now offers a Boolean data type. We, therefore, evaluate the performance impacts, mapping, and processing strategies to make the methodology of the treating Boolean values [22].

This paper is structured as follows. Chapter 2 deals with the ordinary representation using character string or numerical expression meaning, protected by the performance evaluation and aspects. Chapter 3 deals with the implicit conversions expressing TRUE or FALSE values. Chapter 4 defines the Boolean data type introduced in Oracle Database 23c. Finally, in chapter 5, three-valued logic is defined by extending the Boolean characteristics with the undefined values (NULL). Besides, it takes emphasis on the data indexing in B+trees and bitmaps.

The parameters of the computer used for the evaluation study were:

- Operating system: Windows 11 Pro, 22H2

- Processor: AMD Ryzen 5 PRO 5650U with Radeon Graphics, 2.30 GHz
- Memory: 2x 32 GB DDR-4, 3200MHz, CL20
- Disc storage: 2 TB, NVMe, read/write 3500 MB/s

As stated, Oracle Database was used in the computational performance study. Namely, the existing solutions were evaluated in Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 – Production Version 21.3.0.0.0. Boolean data type was assessed in the test version of the 23c release bundle – Oracle Database 23c Free, Release 23.0.0.0.0 - Developer-Release Version 23.2.0.0.0.

The used data set was spatio-temporal [17] [18] [19] oriented, locating airplanes and assigning them to the individual Flight Information Regions (FIR) in aviation [14]. Each object was characterized by the positional GPS data, flight parameters, and particular FIR entry and exit time [14]. The source data table consisted of the 5 million rows (fig. 1), which were then sliced to create data images for each minute during five years. Instead of using the entry and exit time of the FIR assignment, a Boolean data type was used, reflecting whether particular FIR is assigned or not. As the FIR boundaries change over time, NULL values can also be present for the FIR assignment reference.

```
"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. Example of the source data

The processed data set made by data slicing in a minute granularity during five years had 2 635 200 rows with the positive assignment (TRUE value for the Boolean) and 63 244 800 rows for the negative assignment (FALSE value for the Boolean). So, in total, 25 FIRs have been selected. Fig. 2 shows the structure of the sliced data.

```
"ECTRL ID","Sequence Number","FIR ID","Entry Time","Exit Time",
"FIR_SK","FIR_GE", "FIR_AU", "FIR_CZ", "FIR_PL"
"186858226","0","TAXI_OUT","01-06-2015 00:38:00","01-06-2015 00:48:00",
"1","0","0","0","0"
"186858226","1","KKKKFIR","01-06-2015 00:48:00","01-06-2015 02:29:50",
"1","0","0","0","0"
"186858226","2","CCCCFIR","01-06-2015 02:29:50","01-06-2015 04:01:44",
"1","0","0","0","0"
"186858226","3","CZZZFIR","01-06-2015 04:01:44","01-06-2015 05:30:25",
"0","0","0","1","0"
"186858226","4","EGGXFIR","01-06-2015 05:30:25","01-06-2015 07:02:49",
"0","0","0","1","0"
"186858226","5","EGTTUIR","01-06-2015 07:02:49","01-06-2015 07:27:38",
"0","1","0","0","0"
```

Fig. 2. Sliced data

The next chapter deals with the Boolean data type emulation before the Oracle 23c version release.

II. Existing Approaches Before Oracle 23c

Boolean data type was initially considered a necessary element for value modeling and representation. Even just after the first releases of the relational paradigm and database system release, Boolean data type was part of the standardization. However, during the critical evaluation, such a premise of using explicit Boolean was refused. At the same time, it can be successfully emulated by a character string or numerical value stating "on "or "off ", represented by the values one or zero. As a consequence, the explicit Boolean data type was considered to be unnecessary since the values can be expressed with other, more commonly used data types. Thus, the first ANSI standards did not apply Boolean data type [23]. Later, Boolean was introduced either in ANSI standardization but also in common and widespread database systems, like MySQL, PostgreSQL, or MS SQL. However, in Oracle Database, SQL type did not provide such a type. Only procedural extension (PL/SQL) offered it, which caused various troubles when mapping and calling PL/SQL blocks in a SQL environment [3] [22] [23].

As stated, before Oracle 23c release, Boolean values were primarily modeled by the character strings or numerical values. In this chapter, we will evaluate performance impacts and storage demands.

A. Representation by the character string

In the initial phases, Boolean values were treated as character strings in a full format, delimited by the values "TRUE" and "FALSE" in textual forms, either characterized by the fixed size attribute (CHAR(5)) or variable size attribute (VARCHAR(5)). It brought additional demands for the storage, while exactly five characters were used for the CHAR(5) data type – an extra character was added for the value "TRUE" to ensure the compactness of the data type. Tab. 1 shows the results. Based on the evaluation study, it is evident variable character size and fixed size reflecting the difference using only one character does not have any storage demands impact, and the requirements are the same. Besides, fixed size string requires additional demands, compared to the variable character string reflecting 7.67%. It is caused by the data value transfer between the session state and server using the network, packeting, and evaluation. Therefore, from the performance point of view, variable character strings are preferred, even though the storage demands are the same.

TABLE I. Modeling Boolean using a character string

	Storage demands [MB]	Processing time [seconds]
CHAR(5)	6144	387.4
VARCHAR(5)	6144	359.8

Later on, it was clear that the full textual notation does not have any sense and it has the same information value if only one character is present. Thus, by reducing the source data to CHAR(1) or VARCHAR(1), the following results were obtained – Tab. 2.

TABLE II. Modeling Boolean using one character

	Storage demands [MB]	Processing time [seconds]
CHAR(1)	5120	238.7
VARCHAR(1)	5120	236.3

The results shown in Tab. 2 clearly declare that there is no significant change in the performance, comparing VARCHAR and CHAR, delimited by only one character, while there is no NULL value present, one character is always stored, no additional characters to form the output value need to be used. The difference between the processing time is tiny, reflecting less than a 1.02% difference.

 979-8-3503-0586-9/23/$31.00 ©2023 IEEE

B. Numerical representation

Another solution representing Boolean relates to the numerical representation, modeled by the value "1" for TRUE and "0" for FALSE. Physically, various data types can be used based on the value precision, modeling, and storage options. Whereas the integer is commonly used, although the same value can be served by the smallint or even 1 numeral, it is essential to evaluate whether the data type for the numerical value representation requires additional sources and, if, how many.

Tab. 3 shows the results. Based on the evaluation, all the solutions dealing with the numerical representations provide the same results. Furthermore, even the higher precision is used, it does not impact the disc storage because the data rows are not allocated directly, but they are always bundled in the fixed data blocks [9] [15]. The default value of the block size used in the Oracle Database is 8KB. The impact of the various block size structures to the performance of the data processing and retrieval can be found in [14].

TABLE III. MODELING BOOLEAN USING NUMERICAL REPRESENTATION

	Storage demands [MB]	Processing time [seconds]
INTEGER	4608	182.6
SMALLINT	4608	183.5
NUMBER{1)	4608	184.7

C. Domain integrity

In the above subsections, the performance of the Boolean workaround modeling in the Oracle Datatabase version 21c was discussed, either represented by the character string or numerical representation. However, there was no checking on the validity of the value. Namely, what if the value "X" is loaded into the character definition? Or value 2 for the numerical representation? How to represent them? Based on the integrity rules, it is inevitable to limit the available values for the domain, specified either by the check constraint, which is applied directly on the data model layer, or by firing a trigger anytime the data are manipulated in a changing manner. Tab. 4 shows the performance impacts on the loading. If no integrity enhancement and checking are used, 182.6 seconds are required for the numerical (integer) representation, while character mapping takes 238.7 seconds. Evidently, checking the correctness of the value at the level of the data model and the domain boundary using the check constraint is much more efficient compared to the trigger. Namely, the check condition requires 225.9, which reflects an additional 23.7% for the numerical representation and only 8.4% for the character. The significant difference between those data types is related to the original domain range, which is to be limited (one character string vs. integer range).
However, for the trigger, it is necessary to load the pre-compiled trigger definition into the instance memory for the execution, followed by the firing process, which is done individually for each processed row. This factor consequences in a rapid increase in the processing time demands. For the numeric, it reflects 312.9 seconds, expressed by a 38.5% rise compared to the check constraint and 70.4% rise compared to the no-domain boundary definition. Similar results were obtained by the trigger-enhancing character string. The firing, checking, and loading process takes 342.4 seconds in total, represented by 32.4% (compared to the check constraint) and 43.4% for the pure solution. Even in this case, a significant difference can be seen from the point of view of data types. It is really so because of the percentage reduction of the original value and range of the data type in terms of capacity and disk requirements. The graphical representation of the results is in fig. 3.

TABLE IV. DOMAIN VALUE CHECKING

	NUMERIC		CHAR	
	CHECK	TRIGGER	CHECK	TRIGGER
Processing time [seconds]	225.9	312.9	258.7	342.4

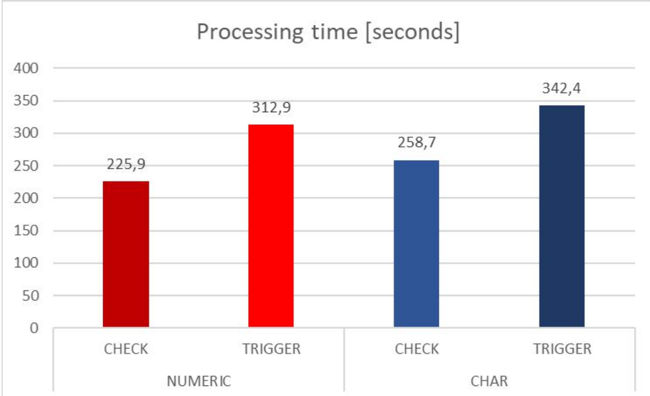

Fig. 3. Example of the source data

III. DATA CONVERSIONS

The data to be processed and queried, however, do not need to have the same data type as the attribute of the table. It results in the necessity to convert the value to the appropriate data type, which can be done either explicitly or implicit conversion selection can be applied based on the optimizer decision [15] [16. This section will evaluate the performance impacts of implicit and explicit conversions.

There are four models to be evaluated, separately processed for the character and numerical representation as a destination source.

The first model does not rely on the conversion explicitly and forces the system to apply implicit conversion on the fly. As evident, it provides the best solution, while these conversion methods are optimized for directly to be used in SQL. Explicit conversion (the second model) is not so powerful, although the conversion direction from the source to the destination is clearly defined. The reason is based on the indexing and data pre-processing, which cannot be done feasibly. Thus, for the character string, explicit conversion mechanisms require an additional 81.6% for the processing time demands and 58.3% for the numerical representation. The difference across the data types is caused by network data transfer. Character string takes only 1 byte, while integer transfers 4 bytes, irrespective of the 32 or 64-bit architecture.

The other evaluation perspective is associated with the own function definition, which does not process the query inside to make it SQL statement independent. Instead, pure IF constructs are used. The third model optimizes the function definition for the procedural language (PL/SQL), while the fourth model is optimized for the SQL function calls, reducing the content switch [3] [4] [9], done by the PRAGMA UDF keyword clause:

```
create function convertdUDF (id char)
return integer
is
  pragma udf;
begin
  if id='1'
    then return 1;
    else return 0;
  end if;
end;
/
```

The obtained results for the character string converted from the numerical representation are present in Tab. 5, graphically represented in the form of the chart in fig. 4. If the own function optimized for the PL/SQL usage is used, content switches are present, resulting in strong additional processing demands – 21.6%, compared to the solution limiting SQL and PL/SQL content switches, named as UDF. But when dealing with implicit conversion management, additional demands are almost three times. Even optimization for SQL usage does not provide sufficient power.

TABLE V. PROCESSING TIME – CONVERSION TO CHAR

CHAR	Processing time [seconds]
Implicit	198.8
Explicit (TO_CHAR)	361.0
Own function (PL/SQL)	599.8
Own function (UDF)	493.1

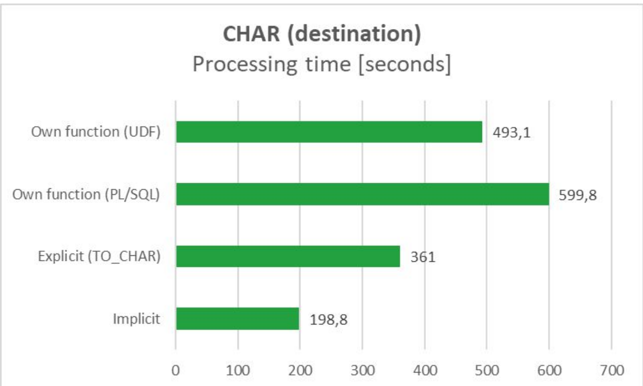

Fig. 4. Results – demands for the processing numerical value to character string

Reflecting the results for the numerical type as the table attribute data type, analogous solutions were obtained. There is no specific reason for getting an explicit conversion, either by the function packaged directly in the Oracle Database (to_number) or by coding own function, optimized for the SQL usage using the PRAGMA UDF clause.

TABLE VI. PROCESSING TIME – CONVERSION TO NUMBER

NUMERIC	Processing time [seconds]
Implicit	233.1
Explicit (TO_CHAR)	369.0
Own function (PL/SQL)	611.1
Own function (UDF)	548.8

Graphical representation fo the obtained results are in fig. 5.

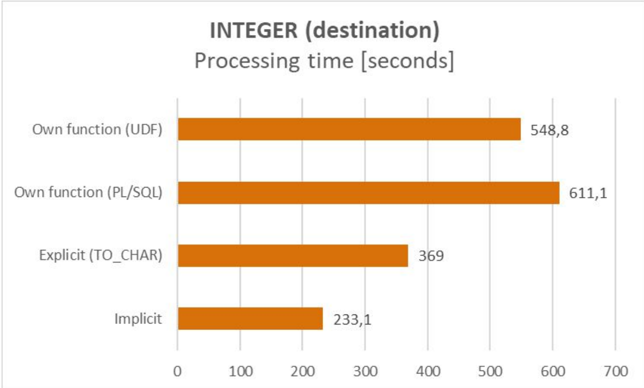

Fig. 5. Results – demands for the processing character converted to the numerical rerpresentation

IV. INTRODUCING BOOLEAN DATA TYPE

The Boolean data type has been valid for the PL/SQL for long time. However, Oracle Database did not recognize it until Oracle Database 23c was released in April 2023, in which the data type Boolean was introduced [22] [23]. Thank's to that, there is no longer a necessity to use numeric or character strings to emulate it, requiring limiting the set of available values, either by the check constraint on the data model layer or trigger. In this section, we will summarize the values, which can be mapped to the Boolean representation. Please note that implicit conversion from the character string, as well as numerical representation, is done automatically. However, it is impossible to call those conversions explicitly. Own function mapping does not make sense because it would produce a considerable offset in terms of the processing time, but also overall costs, content switch, loading pre-compiled version, parsing, etc.

```
Create table flight_mapper
 (ECTRL_ID integer,
  sequence_number integer,
  . . .,
  fir_eu_assign Boolean
);
```

The attribute can be set as 1 as a numeral, '1' as a character string, TRUE (Boolean) or 'TRUE' as string format, and abbreviated value 'T' or 'on'. All the above values will be properly and successfully mapped into the TRUE value.

From the opposite side, 0 as numeral, '0' as a character string, FALSE (Boolean) or 'FALSE' as string format,

979-8-3503-0586-9/23/$31.00 ©2023 IEEE

abbreviated value 'F' or 'off'. All those values are then mapped into the FALSE value [22].

For the performance study evaluation, we have used seven models dealing with the Boolean values:

- M1 takes Boolean *value* as an input, in uppercase format,
- M2 takes '0' or '1' in a textual format.
- M3 takes 0 and 1 in a numerical format.
- M4 takes a *Boolean* value as an input in lowercase format,
- M5 takes character value as an input in lowercase format,
- M6 takes character value as an input in uppercase format,
- M7 takes 'on' or 'off' in a textual format.

The results are in Tab. 7, expressing the processing time in seconds.

TABLE VII. PERFORMANCE - BOOLEAN DATA TYPE IN ORACLE 23C

Model	Processing time [seconds]
M1	187.7
M2	186.5
M3	192.8
M4	193.2
M5	186.1
M6	186.9
M7	187.4

Dealing with the Boolean data type, there are no significant differences, and implicit conversion necessity does not play a key role during the processing and data transformation. Specifically, if the M1 is considered as a reference for the Boolean definition, whereas no conversion is done, there, implicit conversion necessity ranges the additional demands from almost 0 up to 2.9%, which is tiny even for the huge data sets in the analytical databases, like data warehouses, lakes or marts. Thus, even the implicit conversion needs to be done, the overall results are better compared to the numeric value processing, enhanced by the domain value limitation using the check constraint. This brings a substantial change from the point of view of the approach to the bivalent values. Namely, treating Boolean values brings significant benefits. However, existing statements (insert, update, delete, and select) do not need to be rewritten, while the implicit conversion techniques can be directly applied. The only change can be done on the table definition level by dropping the existing check constraint and changing the data type to Boolean (from the original character or numeral). Moreover, pointing to the reached results, the Boolean data type requires approximately 3 percent as additional processing costs compared to a solution that does not check the correctness of the values in any way (i.e. for a number, any value in the range of the original domain, can be entered: -2^{31} to 2^{31}. Fig. 6 shows the results in the form of a chart.

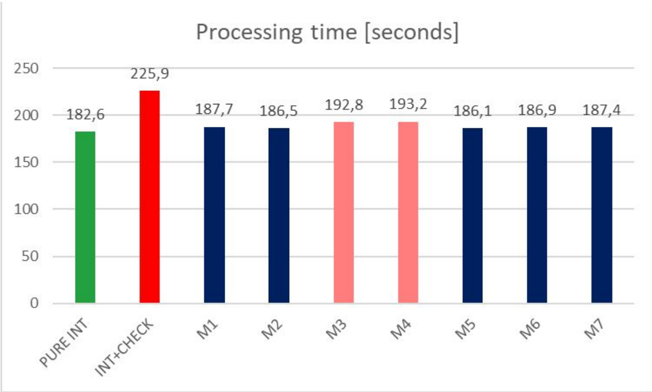

Fig. 6. Results – processing time for Boolean value conversion

The total size demands are stated in Tab. 8.

TABLE VIII. PERFORMANCE – STORAGE DEMANDS IN ORACLE 23C

Boolean representation	Storage demands [MB]
CHAR(5)	6144
CHAR(1)	5120
NUMERIC	4608
BOOLEAN	4608

V. 3-VALUED LOGIC AND INDEXING

Commonly, it is assumed that the Boolean data type holds only TRUE and FALSE values. However, if there is no column restriction specified for the attribute, NULL values can also be present, forming the 3-valued logic. Tab. 9 shows the logical sum, logical count, and negation, reflecting the aspect of undefined values for the Boolean representation. As evident, if the input value is NULL, also the whole condition can be treated as NULL by placing the evaluation to the ELSE clause. This definition can impact the performance related to the indexing because NULL values cannot be mathematically compared and consecutively located through the index.

TABLE IX. 3-VALUED LOGIC

OR	T	F	N		AND	T	F	N		NOT	
T	T	T	T		T	T	F	N		T	F
F	T	F	N		F	F	F	F		F	T
N	T	N	N		N	N	F	N		N	N

Oracle Database prefers B+tree indexes, considering them as a default option, whereas they maintain the performance with the data changes and rise of the tuple number. It is formed by one root and multiple internal nodes navigating the processing to the leaf layer based on the index key. The leaf layer is extended by the pointers to the physical data storage using ROWID pointers, which provide the physical address to the repository – data file, data block, and the position of the row in the block [14] [20] [21]. Each ROWID value consists of 10 bytes enhanced by the object identifier. The performance of the B+tree is strongly related to the aspect of uniqueness – high selectivity of the index key [9]. However, that's not the case for the Boolean or its emulation variants. At the same time, the available values are strongly limited to two or three values, depending on the column integrity, which is applied (whether NULL values can be present or not). Based on this assumption, the bitmap index should provide more robust solutions for the data querying because each Boolean type is represented by one bit. Thus, the following results were obtained to identify the FIR assginment for the particular flight edfined by the 100 000 positional data. The system preferred bitmap index, even if the flight reference has to be compared based on the physical data loaded from the database. It is, however, not done for each row separately, but the bulk loading process is used (TABLE ACCESS BY INDEX ROWID BATCHED). Thus, the bitmap index is used for the value and assignment consideration (BITMAP INDEX SINGLE VALUE), followed by the data conversion to the ROWID value (BITMAP CONVERSION TO ROWIDS). The total processing steps are shown in the following code snippet:

```
SQL> create bitmap index ind1 on idtb1(val);
        Index created.
        Elapsed: 00:00:00.09
SQL> create index ind2 on idtb1(val, id);
        Index created.
        Elapsed: 00:00:00.29
```

So two index types were used. The query aims at getting the FIR assignment for the flight (fig. 7).

```
Execution Plan
----------------------------------------------------------
Plan hash value: 812055720

-------------------------------------------------------------------------------------------
| Id  | Operation                           | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |       | 99251 |  1453K| 75   (0)   | 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| IDTB1 | 99251 |  1453K| 75   (0)   | 00:00:01 |
|   2 |   BITMAP CONVERSION TO ROWIDS       |       |       |       |            |          |
|*  3 |    BITMAP INDEX SINGLE VALUE        | IND1  |       |       |            |          |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("VAL"=TRUE)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       7005  consistent gets
          0  physical reads
          0  redo size
    2542787  bytes sent via SQL*Net to client
     166758  bytes received via SQL*Net from client
       6668  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
     100000  rows processed
```

Fig. 7. Execution plan using a bitmap index

Besides, the bitmap index benefits from one significant aspect – it can treat and cover NULL values, so if there is a requirement to identify states with no FIR assignment, the bitmap index can serve the request directly. Compared to the B+tree index, which is evaluated in fig. 8. It uses the TABLE ACCESS FULL method by scanning the data table block by block sequentially. The total costs have risen from 75 to 92, reflecting 22.7% additional demands.

```
Execution Plan
----------------------------------------------------------
Plan hash value: 47197185

---------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       | 99251 |  1453K|    92   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| IDTB1 | 99251 |  1453K|    92   (2)| 00:00:01 |
---------------------------------------------------------------------------
```

Fig. 8. Execution plan using NULL value assignment

VI. Conclusions

Relational databases form a significant part of information technology and is still being improved to stay powerful, reflecting the continuous increase in the amount of data. One of the key elements, which was uncovered over the decades, relates to the Boolean value definition, processing, and evaluation, either in the whole SQL language but primarily in Oracle Database because other systems often provide own solutions for applying Boolean definitions and constraints.

This paper discusses available performance aspects related to the implicit and explicit conversions and techniques used before releasing Oracle 23c. It provided the methodology guide on how to use the Boolean data type by transferring existing solutions to the updated architecture. We have declared that the existing code does not need to be changed at all. Only data layer represented by the table definitions can be enhanced to serve the Boolean data type directly. The implicit conversion methods do impact the performance and do not require additional processing time demands. Moreover, it limits the necessity to focus on valid values for the data type by defining check constraints or triggers. Thanks to that, even if the implicit conversion has to be done, overall performance would be better, whereas the value checking in directly embedded.

Among that, we have declared performance using the indexes by focusing on the bitmap index, whereas the value range is limited, consisting of no more than three values for the Boolean if the NULL values can be present.

During future research and performance studies, we will emphasize explicit conversions from and to Boolean data type to reduce the content switches. Besides, we will apply index enhancements to cover NULL values in the index by building 3-valued logic directly in the B+tree index structure. It is assumed that by applying all the values in the B+tree index, overall performance would be more efficient than a bitmap, precisely in the case of dynamic changes over time with various data stream strengths. Bitmap changes would require its reconstruction, which is not present in B+tree structures. There is only balancing, which must be done before releases.

Acknowledgment

It was supported by the Erasmus+ project: Project number: 022-1-SK01-KA220-HED-000089149, Project title: Including EVERyone in GREEN Data Analysis (EVERGREEN).

Co-funded by the Erasmus+ Programme of the European Union

EverGreen DATA Analytics

References

[1] Abhinivesh, A., Mahajan, N.: The Cloud DBA-Oracle, Apress, 2017

[2] Anders, L.: Cloud computing basics, Apress, 2021

 979-8-3503-0586-9/23/$31.00 ©2023 IEEE

[3] Cunningham, T.: Sharing and Generating Privacy-Preserving Spatio-Temporal Data Using Real-World Knowledge, 23rd IEEE International Conference on Mobile Data Management, Cyprus, 2022.

[4] Dudáš, A., Škrinárová J., and Vesel, E.: 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.

[5] Greenwald, R., Stackowiak R., and Stern, J.: *Oracle Essentials: Oracle Database 12c*, O'Reilly Media, 2013.

[6] Idreos, S., Manegold S., and Graefe, G.: Adaptive indexing in modern database. In: ACM International Conference Proceeding Series, 2012

[7] Jakóbczyk, M.: Practical Oracle Cloud Infrastructure: Infrastructure as a Service, Autonomous Database, Managed Kubernetes, and Serverless, Apress, 2020

[8] Janáček, J. and Kvet, M.: Shrinking fence search strategy for p-location problems, 2020 IEEE 20th International Symposium on Computational Intelligence and Informatics (CINTI), Hungary, 2020

[9] Kuhn, D. and Kyte, T.: *Expert Oracle Database Architecture: Techniques and Solutions for High Performance and Productivity.* Apress, 2021.

[10] Kuhn, D. and Kyte, T.: *Oracle Database Transactions and Locking Revealed: Building High Performance Through Concurrency*, Apress, 2020.

[11] Kumar, Y., Basha, N., et al.: Oracle High Availability, Disaster Recovery, and Cloud Services: Explore RAC, Data Guard, and Cloud Technology, Apress, 2019

[12] Kvet, M.: Developing Robust Date and Time Oriented Applications in Oracle Cloud: A comprehensive guide to efficient date and time management in Oracle Cloud, Packt Publishing, 2023, ISBN: 978-1804611869

[13] Kvet, M. and Matiaško, K.: Analysis of current trends in relational database indexing, 2020 International Conference on Smart Systems and Technologies (SST), Croatia, 2020.

[14] Kvet, M., Papán, J.: The Complexity of the Data Retrieval Process Using the Proposed Index Extension, IEEE Access, vol. 10, 2022.

[15] Lewis, J.: *Cost-Based Oracle Fundamentals*, Apress, 2005.

[16] Liu, Z., Zheng Z., Hou, Y. and Ji, B.: Towards Optimal Tradeoff Between Data Freshness and Update Cost in Information-update Systems, 2022 International Conference on Computer Communications and Networks (ICCCN), USA, 2022.

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

[18] Su S.Y.W., Hyun S.J. and Chen, H.M.: Temporal association algebra: a mathematical foundation for processing object-oriented temporal databases, IEEE Transactions on Knowledge and Data Engineering, vol. 4, issue 3, 1998.

[19] Yao, X., Li, J., Tao, Y. and Ji, S.: Relational Database Query Optimization Strategy Based on Industrial Internet Situation Awareness System, 7th International Conference on Computer and Communication Systems (ICCCS), China, 2022.

[20] Wang W., Jin, Y. and Cao, B.: An Efficient and Privacy-Preserving Range Query over Encrypted Cloud Data, 2022 19th Annual International Conference on Privacy, Security & Trust (PST), Canada, 2022

[21] Waseem, Q., Maarof, M.A., Idris, M.Y., and Nazir A.: A Taxonomy and Survey of Data Partitioning Algorithms for Big Data Distributed Systems. In: Lecture Notes in Networks and Systems, 2020

[22] https://oracle-base.com/articles/23c/Boolean-data-type-23c

[23] https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/Data-Types.html#GUID-A3C0D836-BADB-44E5-A5D4-265BA5968483

[24] Erasmus+ project EverGreen dealing with the complex data analytics: https://evergreen.uniza.sk/