The efficiency of the Temporal Medical Data Retrieval

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

Abstract—**One of the significant aspects influencing patient treatment is associated with decision-making correctness based on reliable data inputs. Data should be managed, treated, and provided in a robust and performance-effective manner. Commonly, the patient needs to be monitored over time reflecting the evolution. This paper deals with the temporal database models and proposes an effective solution for the data evaluation during the retrieval, so the query request can end significantly sooner. It also points to the current problem of data management in the cloud technology as a central data repository, in which the data time perspectives can be shifted reflecting the time zone, so the results must be transformed to provide relevant data output in a client site.**

Keywords—medical data; temporal aspect; complex indexing; parallelism;

I. INTRODUCTION

Nowadays, great emphasis is done on complex patient treatment, to ensure correct health care across the world. A person is generally managed by the general practitioner, who gets the main overview and shits the patient to the specialist based on the assumptions and previous examination [10] [12]. Currently, data and results are commonly shared by the centralized hospital, regional, or even national system. As evident, data efficiency and proper management are crucial to getting the relevant data almost immediately. Performance and processing demand, mostly reflected by the time consumption cover the inevitable elements to be treated. Data are stored in a local on-premise server, but now, the strong demand to centralize management in a complex cloud system can be felt [6] [19]. The main reason is based on the data amount, which is still rising. The patient is treated more and more complexly, the medical care is still improved producing more reliable, more precise data outputs, which must be stored, analyzed, and retrieved on the demand. It is therefore clear, that the problem of the efficiency of the whole process is relevant and must be managed.

Moreover, nowadays, we can see the Covid-19 pandemics across the world influencing any sphere. Business and production are closed or strictly limited, transport is markedly monitored to limit the spread of the virus, which strongly influences the situation, fills the hospitals, and consequencing in getting the massive number of victims.

With the gradual relaxation of measures against the spread of coronavirus, there is again a strong need to monitor outbreaks and thus reduce the impact and overall problems of those infections.

Similarly, as already stated, all evaluations and systems produce various data structures but have two common aspects. Firstly, the produced data amount is significant and secondly, there is strict pressure to get the data in a massive, parallel, and mostly in a performance effective, robust, and reliable manner. Such data have a common definition of the time spectrum, as well as region assignment.

This paper deals with the temporal database architecture to provide support for complex medical data. In section 2, current technologies and temporal architectures are summarized, pointing to the specifications and limitations. The proposed extension is done by using an index relocation unit to shift the internal granularity to fit the request specified inside the query. Section 3 deals with the proposed architecture for medical data management and supervision. In section 4, the data retrieval process management handler, covered by the two-level parallel indexing strategy is defined. All data are time positioned. By shifting the environment to the cloud technology, the problem of the different time zones can be present, whereas most of the technology reference server time, not the transformation to the client [1] [4]. As a result, wrong values are provided. In section 5, we deal with the transformation covering the analysis of the additional demands to the processing. Internally, it is done by the query time replacement using the translation package. Section 6 deals with the performance evaluation.

II. TEMPORAL ARCHITECTURES

Significance of the temporal data management in database systems has been identified with the advent of the relational paradigm supervised by the transactions. Soon, the first temporal architecture delimited by the object-level temporal model was proposed. The identifier of the object itself was extended by the time attributes characterizing validity, transaction validity, or other temporal attributes. As a consequence, the object state cannot be uniquely identified by the object representation itself, the definition of the time-image should be handled, as well. The main disadvantage of such an approach is just the granularity itself. Each new state has a full definition, thus, each attribute value should be present. There is no need to get an image of several states, there is no need to compose a state from several fragments. Vice versa, such an approach can produce many duplicate values, if the update operation is not synchronized across the whole object state definition. Moreover, there is no solution for dealing with just the relevance defining the region or position of interest. Finally, such a solution can be significantly ineffective in terms of performance, but mostly in the aspect of the data storage demands. Original state values can be copied to new images if no change occurs. Synchronization across the whole state should be present to obtain the optimal solution. As evident, object-level temporal architecture is not commonly suited for the medical data with the dynamic evolution reflecting the region of the interest monitoring.

Attribute-oriented architecture was complexly defined in [3] [4] [15]. Each state is delimited by the composition of individual attributes along with the time definition. Each change is divided into the attributes themselves, which are maintained separately. Namely, each attribute change forces the definition of the new state physically stored by the temporal layer. Such architecture is suitable in terms of the physical layer perspective. It is, however, clear, that the data retrieval process can be demanding forcing the system to calculate the state on demand. In ad-hoc networks forcing the systém to obtain data dynamically with a strong impact on the data transfer, such architecture is not suitable, whereas the pre-processing and state composition can last too much time. In medical data management, the situation is similar with the particular emphasis on the processing efficiency of the state retrieval as the data are commonly centralized. It creates significant pressure on the optimization of data processing, retrieval, and transmission. The architecture of the attribute-oriented temporal model is in fig. 1. It consists of four-level architecture. The first level deals with the current state of the object. Physically, data are stored in the attribute granularity, however, the retrieval representation is done using the object perspective composition. The second level is formed by the core of the solution – the temporal evidence management layer. Any change is registered there, with the pointing reflection to the particular layer in an attribute perspective. Such a module is also responsible for the object granularity state composition during the data retrieval. Level 3 deals with the historical data, level 4 manages plans (level 4 is an optional level, if no plans or states valid in the future need to be monitored, such representation does not need to be used). The temporal level is interconnected to the index relocation unit, where the requested format is created, mostly modeled by the object granularity. The index relocation unit is then interconnected to the public interface providing result sets to the client site. In comparison with the original attribute-oriented granularity presented in [13], our proposed solution uses an external source to provide any data granularity perspective using an index relocation unit.

Extension to fuzzy management is discussed in [2] [5].

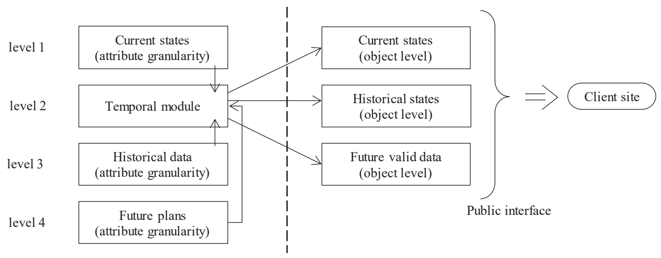

level 1
level 2
level 3
level 4
Current states (attribute granularity)
Temporal module
Historical data (attribute granularity)
Future plans (attribute granularity)
Current states (object level)
Historical states (object level)
Future valid data (object level)
Client site
Public interface

Figure 1.Attribute oriented granularity

The last solution is a group granularity composition. Instead of storing each attribute change separately, the dynamic temporal group can be identified (either manually or in an automatic manner using a data analytics perspective). Thanks to that, if several attributes are update synchronized, only one new row is added to the temporal module limiting the amount of the data to be stored, with no loss of the information value. Physically, it is covered by the analytical module extending the temporal module. In principle, each attribute represents the one element group, so the reference is always done on the group level, which can be temporarily grouped into the various segments. Fig. 2 shows the group state hierarchy. The input stream is routed to the data change barrier, where the attribute or synchronization groups are identified by the Grouper background process. The analytical module is responsible for the synchronization detection, so the storage demands are always up to date and optimized based on the data flow. Note, that in fig. 2, individual levels are visualized using the storage type, however, internally, current, future, and historical data are treated and stored separately.

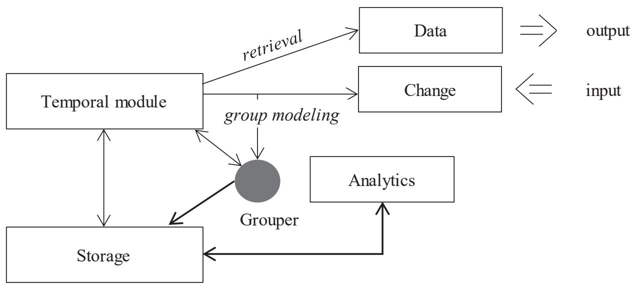


Figure 2.Group data management

Original group level architecture is defined in [13]. The stated solution introduces the Grouper background process, which brings additional benefits, whereas the group definition can be automated and treated dynamically in parallel using the Analytics module. The original solution required to specify groups either by the user or the optimizer could do that. And that is just the point. The optimizer was congested causing delays.

III. TEMPORAL GROUP REGION ARCHITECTURE

In the previous sections, we have summarized current temporal approaches by pointing to the limitations. Besides, we have proposed several extensions covered by this paper, mostly reflected by the synchronization group extension and index relocation unit. Section 2 deals just with temporality, in this part, we propose the Spatio-temporal solution covering the positional data in the region assignment principle. Thanks to that, medical data can be delimited by the patient himself, temporal elements, but the positions of the interest can be managed, as well.

The proposed architecture overview is shown in fig. 3, reflected by the data model located in the temporal layer. A core part of the model is formed by the temporal table referencing any data update by assigning its change_id chaining individual changes for the particular object. Statement_type delimits the data modeling operation (Insert, Update, Delete). Pointer to the data object is done by the table identifier (id_tab), attribute or temporal group (data_id) definition, and row (row_id). Non-current values are grouped into individual data type categories referenced by the data_type_cat. Note, that the data type categories are maintained automatically by the internal transformation opportunity (implicit conversions). As stated, the model is spatio-temporal meaning, that the temporal table consists of the validity time frame, reliability time dimension expressed by the transactions and spatial assignment managed either locally by spatial_positions or by dynamic region covering – region_assignment.

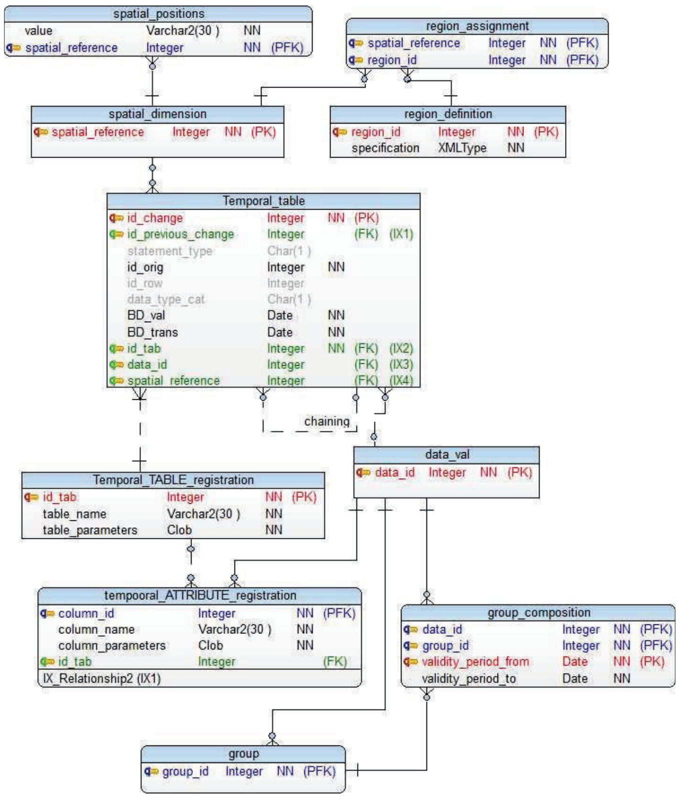

spatial_positions
value Varchar2(30) NN
spatial_reference Integer NN (PFK)
region_assignment
spatial_reference Integer NN (PFK)
region_id Integer NN (PFK)
spatial_dimension
spatial_reference Integer NN (PK)
region_definition
region_id Integer NN (PK)
specification XMLType NN
Temporal_table
id_change Integer NN (PK)
id_previous_change Integer (FK) (IX1)
statement_type Char(1)
id_orig Integer NN
id_row Integer
data_type_cat Char(1)
BD_val Date NN
BD_trans Date NN
id_tab Integer NN (FK) (IX2)
data_id Integer (FK) (IX3)
spatial_reference Integer (FK) (IX4)
chaining
data_val
data_id Integer NN (PK)
Temporal_TABLE_registration
id_tab Integer NN (PK)
table_name Varchar2(30) NN
table_parameters Clob NN
tempooral_ATTRIBUTE_registration
column_id Integer NN (PFK)
column_name Varchar2(30) NN
column_parameters Clob NN
id_tab Integer (FK)
IX_Relationship2 (IX1)
group_composition
data_id Integer NN (PFK)
group_id Integer NN (PFK)
validity_period_from Date NN (PK)
validity_period_to Date NN
group
group_id Integer NN (PFK)

Figure 3.Group spatio-temporal model

IV. DATA RETRIEVAL

Data retrieval perspective is a core element influencing the whole performance. Optimization and evaluation is a staged process to get reliable results. Namely, the optimizer has to check the prerequisites for the processing – semantic and syntactic check reflected by the access rules, object identification by the owner, etc., followed by the execution plan composition. Generally, several plans are identified, evaluated by the estimated costs originating from the data table and database statistics. The aim is to select the best suitable plan for the consecutive evaluation. The execution process itself can be done either by using sequential scanning of the data blocks associated with the table or by using the index layer [20]. Sequential scanning is the most demanding operation, whereas the data amount to be produced cannot be evaluated directly. Moreover, individual blocks can be fragmented, located in multiple physical discs, as well, bringing additional processing demands. Moreover, some blocks can be even empty, as a result of extent initialization, instead of the blocks themselves. Each extent is composed of the defined amount of data blocks, which allows you to associate blocks dynamically by limiting the necessity of allocation any time. The second current solution is based on using an index strategy to locate relevant data blocks. Namely, instead of block scanning, a defined index is used to locate data. Pointers to the direct data are located in the leaf layer, specifying ROWID value – address of the row inside the block and data file forming the database. The index is far smaller in comparison with the whole table, can be optimized [6] [7] [9], but mostly, it is directly accessible in the database instance memory. The index itself is defined by the named set of the individual table attributes or function calls, where the element order is significant forming the structure [8]. In database systems, the B+tree index structure is used most often as a default strategy. B+tree database index is formed by the root node, internal nodes, and leaf layer pointing to the data. The limitation of the indexing is just the reliability issue – undefined table data cannot be indexed, whereas NULL values cannot be mathematically sorted and evaluated. As a consequence, commonly, if the query can produce an undefined or untrusted value, the processing is shifted to sequential data scanning. In [16] [17], it is solved by introducing NULL modules, which hold undefined data directly inside the index by using mapping function [11] [14] [18]. In comparison with function-based index transforming undefined values, mapping is done internally with no additional storage demands. NULL values can be stored either internally or in an external module interconnected to the root element.

Flower Index Approach (FIA) is formed by a specific robust index method usable in case of unavailability of the (sub)optimal access path. In that case, sequential scanning would be necessary to perform. As stated in the research paper [14] [15], if the data fragmentation caused by a huge update stream or volatility aspect is present, total performance would be poor, several data blocks would be necessary to be memory loaded, with no relevant data there. FIA approach is used as a data block locator, where at least one existing data row is present. In the leaf layer of the index, BLOCKIDs are present pointing to the whole block, instead of the data row position. By loading the data, the whole block is evaluated to focus on the data.

The performance of the index is limited by the structure, which can, in principle, degrade over time, if several updates and delete operations are present. The property of the B+tree is based on the balancing, so the traverse path from the root to any leaf node is always the same in terms of depth. If the node inside the index does not hold any data after the Update or Delete operation, a particular node remains in the index structure still, because it is assumed that the node will be used again in the early future. Such definition can have, however, significant performance impact, if the type and values are evolving responding to the accuracy and precision. In [14], database index balancing strategies are defined. Balancing operation is extracted from the main transaction and is operated separately consequencing in the ability to approve the original change operation and transaction sooner. Changed values are stored either in a separate structure by applying them by the introduced Balancer background process [14] or are placed in the leaf index layer with no reflection to the balancing itself [15].

Thanks to that, it is ensured, that the index is still up-to-date, even in terms of the structure and tree depth. Based on the computational study, it has minimal additional data retrieval demands – less than 1 percent.

Proposed spatio-temporal solution

The proposed solution covers spatial and temporal perspectives together. Architecture can be split into two internal regions supervised by the background processes and available through the public interface. The internal layer consists of two index types. B+tree index set is formed by the primary index set, where objects irrespective of the spatial and temporal dimension are registered and indexed. The secondary index set is managed by the user specification. It commonly stores B+tree definition, as well, however, the bitmap, compressed, hash, or reverse indexes can be covered by such module, as well. Spatial and temporal dimensions are secured by the separate index set. These data portions can be processed and evaluated in parallel followed by the merging operation, which is done by the bitmap index pointing to the ROWIDs. Data blocks are then loaded, the result set is created and provided to the client via a public interface. The internal B+tree index set does not point to the data blocks (like in a common conventional system, but the leaf layer references the bitmapper located in the second interval layer). Fig. 4 shows the architecture.

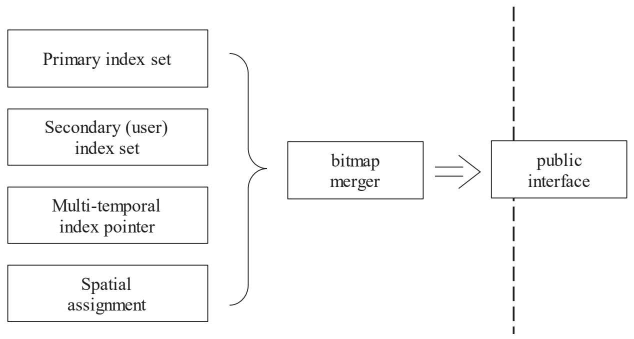


Figure 4.Group spatio-temporal model

V. Time management

Common characteristics of current information systems (not only regarding the medical data perspective) are based on time spectrum limitation, mostly reflected in the server. Most of the systems represent just server time definition, instead of the local client. It was based on the time zones - it was assumed that the server settings are always correct and provide the correct values regardless of the time zone parameter setting on the client-side. Thanks to that, it was ensured that the provided data were relevant - the server was stored directly in the medical center, in the local server room. With the spread of cloud technologies, security standards, and complexity of the requests, there has been a legitimate demand to move data to cloud storage, to manage them autonomously [21]. As we mentioned in the description of the architecture, the amount of data to be processed is constantly growing, and therefore the dedicated cloud storage provides a suitable space for growth and ensuring robustness, performance and security. The problem is just the time perspective. Based on the study provided by Oracle, most of the current systems use server time reflection (sysdate), instead of the client, which can produce incorrect data (curret_date) for getting the current date and time. The difference is just the time zone, however, if the cloud storage is in another region or is even placed in a different continent, it would be necessary to rewrite the whole code to reflect the client perspective and time zone. It would to very demanding and prone to errors. Moreover, the time evaluation perspective in the medical systém is crucial. Therefore, it is necessary to find another solution to represent local client time. In this paper, we propose and evaluate the costs and benefits of the translation profiles of the SQL.

Solution – SQL translation profile

Our proposed solution covering local client time is based on the SQL translation profile, which dynamically replaces the original definition of the server time by the client time zone. SQL translation profile is commonly set by the database administrator (DBA) by invoking the CREATE_PROFILE procedure of the DBMS_SQL_TRANSLATOR package. Each profile is delimited by its unique name, defined attributes with specified values, and a pointer to the package method, which is invoked any time the SQL query is to be executed.

The invoked function is associated with the ATTR_TRANSLATOR parameter. The translator function must be packaged with the following two procedures:

- Procedure TRANSLATE_SQL has two parameters by passing the original SQL query resulting in providing the second parameter as an output in form of character LOB value.
- Procedure TRANSLATE_ERROR is called if the translation ends with a raising exception. It has three parameters – binary integer value ERROR_CODE, TRANSLATED_CODE, and TRANSLATED_SQLSTATE.

To cover the reliability of the medical data processing, the translation function is associated with the regular expression calling REGEXP_REPLACE method. The syntax of the REGEXP_REPLACE procedure is in fig.5 [22]:

- SOURCE_CHAR is a string used as a search value.

- PATTERN is a regular expression with the same data type as SOURCE_CHAR, respectively transformable implicitly.
- REPLACE_STRING
- POSITION – positive integer defining the starting position for the evaluation.
- OCCURRENCE – non-negative value (n) delimiting the replacement of the n-th occurrence. Value 0 expresses all occurrences to be replaced.
- MATCH_PARAMETERS characterizes the sting format to be handled. Value "i" in our case defines case insensitivity.

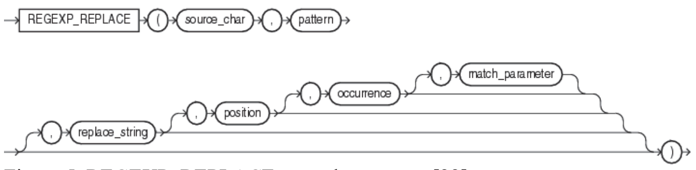


Figure 5. REGEXP_REPLACE procedure syntax [22]

Implementation

Medical data transformation is done by using the translator defined in the following code snippet. In case of ensuring server-client time transformation, SQL translation profile is defined by the following steps. First of all, the translation profile has to be created and mapped to the package definition, which consists of two methods. Then, the session is mapped to the profile.

Package definition

```
create or replace package date_translator
 is
  procedure translate_sql(sql_text in clob,
                  translated_text out clob);
end;
/

create or replace package body date_translator
 is
  procedure translate_sql(sql_text in clob,
                  translated_text out clob)
    is
     begin
       translated_text:=regexp_replace(sql_text, 'SYSDATE',
                                   'CURRENT_DATE,1,0,'i');
     end;
end;
/
```

Note, that the procedure TRANSLATE_ERROR is optional. If omitted, if any error occurs, translated code is ignored and the original definition is used instead.

Profile definition and association

```
begin
 dbms_sql_translator.create_profile
  (
   profile_name => 'MEDICAL_PROFILE'
  );
 dbms_sql_translator.set_attribute
  (
   profile_name => 'MEDICAL_PROFILE',
   attribute_name =>
     dbms_sql_translator.ATTR_FOREIGN_SQL_SYNTAX,
   attribute_value =>
     dbms_sql_translator.ATTR_VALUE_FALSE
  );
 dbms_sql_translator.set_attribute
  (
   profile_name => ''MEDICAL PROFILE',
   attribute_name => dbms_sql_translator.attr_translator,
   attribute_value => 'DATE_TRANSLATOR'
  );
end;
/
```

Profile definition and association are covered by three-step blocks inside the anonymous block execution. The first calls the procedure create profile delimited by the name. The second block is based on the changing default parameter *ATTR_FOREIGN_SQL_SYNTAX* from the value *ATTR_VALUE_TRUE* to *ATTR_VALUE_FALSE. Such a parameter specifies the type of SQL syntax.* The third call sets the ATTR_TRANSLATOR pointer to the defined package (DATE_TRANSLATOR).

Mapping

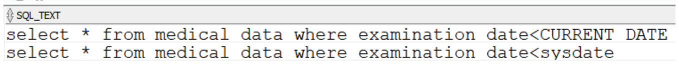
```
alter session
  set sql_translation_profile=MEDICAL_PROFILE;
```

Mapping is done by associating profile to the session.

Result

Data transformation can be obtained by using V$SQL dynamic performance view.

```
select sql_text
 from v$sql
  where sql_text
         like 'select%medical  data%examination  date%';
```



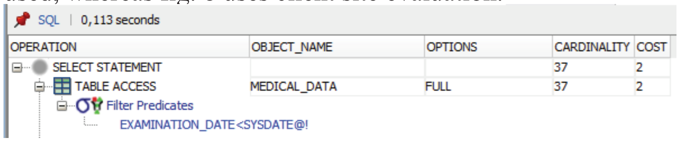
```
SQL_TEXT
select * from medical data where examination date<CURRENT DATE
select * from medical data where examination date<sysdate
```

Figure 6. Transformation results

Analysis of the additional processing demands

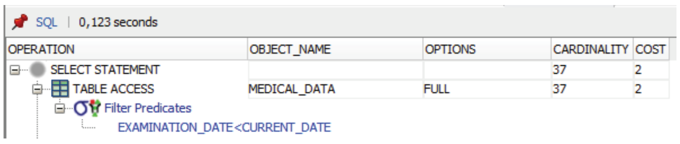
Cloud environment can provide scalable dynamic solutions ensuring the global performance of the system. By transferring the implemented systems to the cloud environment, reference of the time zones is highlighted, as well. In this part, there is an analysis of the additional demands regarding the SQL translation profiles. Generally, there are two available domains to be implemented. The first one is simpler in terms of management, whereas the translation is done automatically. The second approach is delimited by the parse code reference delimited by the base lines. It will work, however, robustly, only if the reference parse is always accessible and loaded, even after the system restart, so the administrator needs to ensure it, otherwise the processing will be aborted.

Experiments were done in magnetic resonance imaging and patient monitoring, where time precision is crucial. As evident from the following figures, the solution is scalable with minimal additional demands caused by the translation – 10ms. Note, however, that it is done only once before the hard parsing, afterwards, just the reference to the already stored library cache parse is used directly with no transformation, at all. Fig. 7 shows the original demands, fig. 8 shows the results by applying translation. Only time demands are slightly changed, access methods, as well as total processing costs expressed by the size, processing time, and resource consumption, remain the same [3]. In fig. 7, server date is used, whereas fig. 8 uses client site evaluation.


SQL | 0,113 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			37	2
TABLE ACCESS	MEDICAL_DATA	FULL	37	2
Filter Predicates				
EXAMINATION_DATE<SYSDATE@!				

Figure 7. Original statement


SQL | 0,123 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			37	2
TABLE ACCESS	MEDICAL_DATA	FULL	37	2
Filter Predicates				
EXAMINATION_DATE<CURRENT_DATE				

Figure 8. Translated statement

VI. COMPUTATIONAL STUDY

Performance characteristics have been obtained by using the Oracle 19c database system (Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 – Production). Parameters of used computer are processor: Intel Xeon E5620; 2,4GHz (8 cores), operation memory: 16GB, HDD: 500GB.

The results of the patient monitoring using magnetic resonance imaging are used. 1000 patient data were used, each of them is delimited by the 10 results provided over time. The brain tumour is detected, located, and monitored. In the first phase, whole data about the brain are stored, later, just regions of interest with significant changes between individual results are stored to optimize consecutive evaluation and limit storage demands. The average size of the data set is 96 MB consisting of 20 markers.

In [13] [14], individual granularity types and architectures are compared technically, mostly reflecting the efficiency of the data transfer. In this paper, the evaluation study will focus on the proposed spatio-temporal architecture supervised by the parallel processing through indexes followed by the bitmap merger. In comparison with already existing approaches, the whole data should be treated, there is no robust region of interest temporal solution. Based on the provided data, just 20% of the data are monitored over time based on the significance evaluation.

For the study, we used two streams. In the first experiment, monitoring of just one patient is used. Data, which are not stored, are replaced by the baseline data results, whereas there is not a necessity to store them, they do not point to any change over time, so there is no reason to store duplicate values. The second experiment deals with the monitoring of multiple patients with the same diagnosis to compare and highlight the treatment methods and reached results.

EXPERIMENT 1

As stated, each patient is represented by 10 examination results, mostly covered by the biannual time-frequency. The first result set is always stored completely by locating potential anomalies, which are marketed as regions of interest. Besides, also the positional neighborhood is maintained to evaluate the detection progress. Besides, for each evaluation, Epsilon perspectives are handled to ensure any significant change is covered properly. Based on the evaluation, the average data amount to be stored and processed was limited to the value of 20% reflecting the temporality and spatial dimension. The significant aspect is just the reliability, such data amount reduction does not influence the information values, at all. The data retrieval process is evaluated, reflected by the storage demands and data retrieval process. Non-stored values are obtained by the base line and marked by the obtain time point.

Fig. 9 shows the results comparing temporal group dimension with the proposed spatio-temporal architecture. Fig. 9 represents the storage demands expressed in MB. In comparison with individual attribute management, storage demands are lowered to 45%, thanks to dynamic data processing and group detection. Just 27% of the storage capacity is necessary for the spatio-temporal dimension, in comparison with the reference model – attribute granularity.

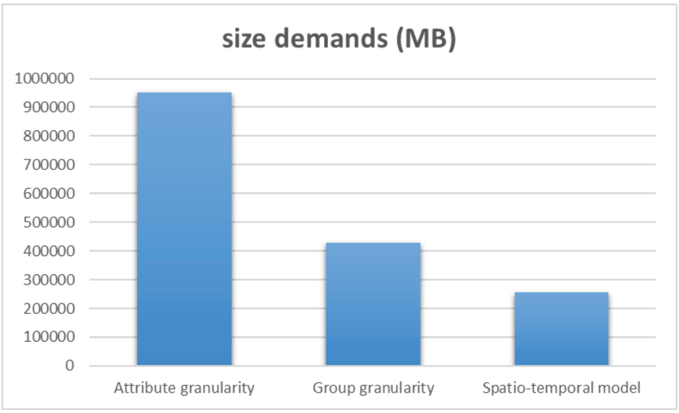


Figure 9. Size demands

Fig. 10 shows the processing time demands, which are lowered, similarly, as well. Namely, attribute granularity requires 45,74 seconds. By applying group detection, only 37,84 seconds are required, whereas the synchronization groups are managed as one element stored in the temporal layer. Proposed group granularity of the temporality and spatial dimension requires only 25,03 seconds, 3,87 seconds are used for the non-stored values obtaining from the base line, which reflects 15,46% of the total processing for the proposed solution.

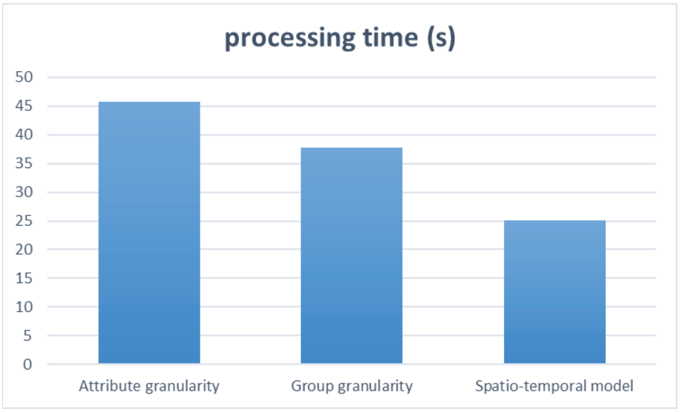


Figure 10. Processing time

EXPERIMENT 2

The second evaluated study is based on monitoring several patients over time by using the proposed spatio-temporal technique. We have identified the group with the same treatment category containing 37 patients. Fig. 11 shows the results reflecting the size demands, which are reduced from the value 35,5 GB for the attribute level architecture to the value 16,2 GB for temporal group dimension. By applying spatio-temporality, 12,72 GB. The main size demands drop possibility is based on the categorization, it is clear, that all the patients have the same disease, so the location can point to the specific region directly.

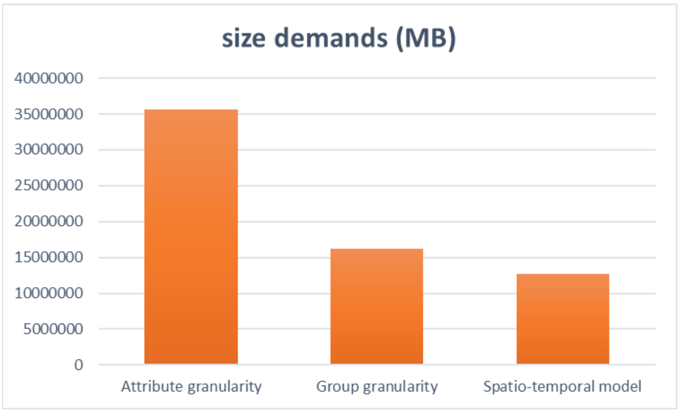


Figure 11. Size demands

Processing of such data amount is demanding and should be reduced as much as possible. Spatio-temporal model can directly locate data of interest, whereas it is just the same for each patient (with regards to the data neighborhood). Thanks to that, processing demands are lowered from the value 1712 seconds for the referential attribute granularity to the 1378 seconds for the group detection model (19,5% improvement by reducing processing time). The proposed spatial model focuses on just the spatial positions, not only temporality, thus the time demands are 946 seconds (44,7% referencing attribute model and 31,3%).

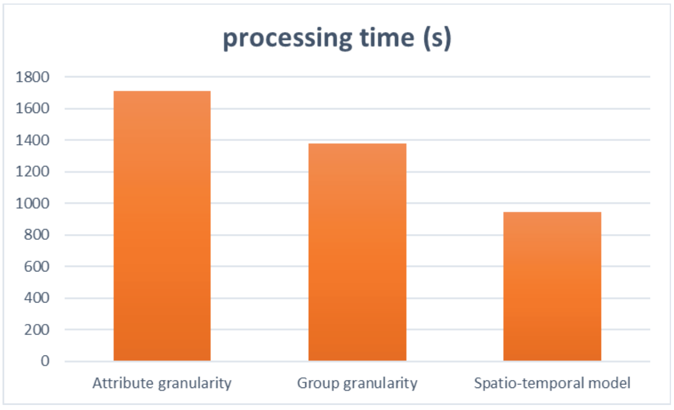


Figure 12. Processing time

VII. CONCLUSIONS

Data efficiency is the main point expressing the complexity, robustness, and reliability of the whole information system. If the data are not provided in a defined time spectrum, consecutive analysis, management, evaluation, and decision making cannot be done. In medical systems, it is necessary to manage data of the patient over time, to be provided immediately, and cover real-time activities. Therefore, the strict impact should be taken into the whole optimization of the database processing as an internal layer providing and managing data. In this paper, we summarize existing temporal systems, which can cover the whole data tuples and their changes in the whole time spectrum. The analysis emphasizes the available granularity levels by forming spatio-temporal solutions covered by four-level architecture splitting data based on the time spectrum.

The main contribution of this paper is defined in section 3 proposing spatio-temporal group solutions covering all existing approaches to reach a robust effective model dealing with Flower Index Approach, undefined states, etc. The aim is to ensure the effectiveness of the processing, mostly pointing to the data retrieval process. The proposed solution extends the existing index set management by using parallel processing merged by the bitmap system.

The limitation of the current systems is just the temporality management, mostly represented by the server time, which brings problems if the data are migrated to the cloud environment to propose a robust, reliable, and secure solution. In this paper, we also analyze the impact of the SQL translation profiles definition on performance. As evident from the reached results, the transformation is done only once, followed by the storing execution plan in the memory cache.

In the future, our emphasis will be taken to the cloud environment block storage and baseline definitions, by mapping the SQL statement to the pre-stored parsing process. Thanks to that, the translation can be hosted internally, which can lower the additional processing time demands completely.

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

References

[1] Abdalla, H. I.: A synchronized design technique for efficient data distribution, Computers in Human Behavior, Volume 30, 2014, pp. 427-435

[2] Behounek, L., Novák, V.: Towards Fuzzy Patrial Logic. In 2015 IEEE Internal Symposium on Multiple-Valued Logic, 2015.

[3] Bryla, B.: Oracle Database 12c The Complete Reference, Oracle Press, 2013, ISBN – 978-0071801751

[4] Burleson, D. K.: Oracle High-Performance SQL Tuning, Oracle Press, 2001, ISBN - 9780072190588

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

[6] Dostál, J., Wang, X., Steingartner, W., Nuangchalerm, P. , - Digital Intelligence - New concept in context of future school education, In: 10th Annual International Conference of Education, Research and Innovation (ICERI2017), Seville, SPAIN, NOV 16-18, 2017, pp. 3706-3712, 2017

[7] Eisa, I., Salem, R., Abdelkader, H.: A fragmentation algorithm for storage management in cloud database environment, Proceedings of ICCES 2017 12th International Conference on Computer Engineering and Systems, Egypt, 2018

[8] Feng, J., Li, G., Wang, J.: Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units, IEEE Transactions on Knowledge and Data Engineering (Volume: 23, Issue: 12, Dec. 2011) , 2011.

[9] Honishi, T., Satoh, T., Inoue, U.: An index structure for parallel database processing, Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, 1992.

[10] Janáček, J., Kvet, M. (2016). Sequential approximate approach to the p-median problem. In Computers & Industrial Engineering 94 (2016), Elsevier, ISSN 0360-8352, pp. 83-92.

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

[12] Kvet, M. (2019). Complexity and Scenario Robust Service System Design. In Information and Digital Technologies 2019: conference proceedings, Žilina, 2019, ISBN 978-1-7281-1400-2, pp. 271-274.

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

[14] Kvet, M.: Database Index Balancing Strategy, in print (2021)

[15] Kvet, M., Kršák, E., Matiaško, K.: Study on effective temporal data retrieval leveraging complex indexed architecture, Applied Sciences 10 (2020)

[16] Lien, Y.: Multivalued Dependencies With Null Values In Relational Data Bases. In Fifth International Conference on Very Large Data Base, 1979.

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

[18] Moreira, J., Duarte, J., Dias, P.: Modeling and representing real-world spatio-temporal data in databases, Leibniz International Proceedings in Informatics, LIPIcs, Volume 142, 2019

[19] Schreiner, W., Steingartner, W. and 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

[20] Vinayakumar, R. Soman, K., Menon, P.: DB-Learn: Studying Relational Algebra Concepts by Snapping Blocks, International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018, India, 2018

[21] https://www.oracle.com/database/what-is-autonomous-database.html

[22] https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions130.htm