DOI: https://doi.org/10.15276/aait.02.2020.1
UDC 004.8

INFORMATION TECHNOLOGY OF SUPPORTING ARCHITECTURAL SOLUTIONS USING POLYGLOT PERSISTENCE CONCEPT IN LEARNING MANAGEMENT SYSTEMS

Olena O. Arsirii[1)]
ORCID: https://orcid.org/0000-0001-8130-9613, e.arsiriy@gmail.com
Maria G. Glava[1)]
ORCID: https://orcid.org/0000-0002-9596-9556, glavamg@gmail.com
Matthias Kolonko[2)]
ORCID: https://orcid.org/0000-0002-8296-1758, matthias.kolonko@hs-augsburg.de
Alina O. Glumenko[1)]
ORCID: https://orcid.org/0000-0002-6085-3422, alina.glumenko@gmail.com
[1)] Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
[2)]Augsburg University of Applied Sciences (AUAS), 1, An der Hochschule. Augsburg, 86161, Germany

ABSTRACT

This paper shows that performance of the learning management systems heavily depends on the choice, made during designing, of architectural solution for storage and processing of data. Based on analysis of evolution of the various architectural solutions during the information system design, beginning with monolith platform and ending with decentralized microservices, it has been determined that architecture based on microservices for a server side with code-level isolation and database-level decentralization for components is proved to be effective solution for high-performance system complexes for learning management system. However, for implementation of polyglot persistence concept based on multiple database management systems with various logical schemas, there is also a need for developing an information technology to support such architectural solutions. It has been shown that the development of databases for such learning management system, that operate with a large amount of various information, consists of the stages of conceptual, logical and physical modeling, and, precisely during the creation of logical models the requirements for the storage and processing of data, that are used by the selected entities for the implementation of business functions, are determined. The peculiar properties of using relational and non-relational database management systems such as: document, key-value, graph and column storages have been examined and analyzed in detail. A method for automated selection of logical data models based on initial information about a limited context has been developed, then used to develop a classifier. The efficiency of the classifier was tested on a dataset for two hundred thirty entities. As a result of the experiment, the reliability of the classification was ninety-three percent. The advantages of the developed information technology are shown on the example of designing JustStart learning management system. Analysis of the stress testing results of the developed system shows that due to the distribution of the load between the three databases, its average response time with simultaneous operation of one hundred fifty users was one point two seconds. At the same time, simulation of the same number of users with only one database management system, the response time increased and the average was approximately two point six seconds. Thus, the use of the developed information technology of supporting architectural solutions for organizing storage of large volumes of diverse data according to the polyglot persistence concept, that allowed to design and implement learning management system, the performance of which, if it is used simultaneously by a large audience, is on average twice as fast as the average educational resource on the market

Keywords: Learning Management Systems; Polyglot Persistence; Microservice Architecture; Decision Tree

***For citation*: Olena O. Arsirii, Maria G. Glava, Matthias Kolonko, Alina O. Glumenko. Information Technology of Supporting Architectural Solutions Using Polyglot Persistence Concept in Learning Management Systems. *Applied Aspects of Information Technology*. 2020; Vol.3 No.2: 13–31. DOI: https://doi.org/ 10.15276/aait.02.2020.1**

INTRODUCTION AND THE RESEARCH PROBLEM STATEMENT

According to the latest business software market research, the amount of platform providers for learning management systems (LMS) in 2019-2020 exceeded one thousand [1]. Despite the high market saturation, companies that decide to organize online training for the first time often choose to develop their own LMS, which would take into consideration the specifics of the company business process. From a technical point of view, LMS is a system which ensures the functioning of the following subsystems (services): support for educational content (content standards and formats for online learning), content creation using the course designer (authoring tool), user management, reporting system etc. All of the mentioned subsystems should be developed and function taking into accounts the specifics of the business and the created LMS in general. An essential part of the LMS is database (DB) and database management system (DBMS).

Thus, there is a need to develop a technology that will simplify and speed up the process of developing LMS, taking into account the distribution of modules of such a system and increasing performance. In current realities microservice architecture is used to develop information systems whose subsystems are built around business needs. This is an approach in which a single application is built as a set of small services, each of which works in its own process and communicates with the rest, using lightweight mechanisms based on the HTTP protocol [2]. Services are deployed independently using a

fully automated environment with a minimum of centralized management. Another advantage of services is the possibility of their implementation using various programming languages (Polyglot Programming) and different technologies of data storage (Polyglot Persistence) [3, 4].

Thus, the usage of a microservice architectural approach in the development of LMS is justified, on the one hand, consisting of sufficiently isolated subsystems, and on the other, providing access to a large amount of updateable multimedia content, as well as storing and processing a large amount of *various* operational data, such as information about teachers and students, tests, quizzes, course statistics, student grades, tracking student progress, course recommendations, etc.

In this case, in addition to solving the tasks of highlighting a limited context from the description of the subject area during the development of each microservice (function), the LMS architect is facing the need to use different models (schemas) of data storage and processing (various DBMS), taking into consideration the nature of the source data. Such approach is called the Polyglot Persistence concept. While implementing such concept in the process of making architectural decisions on the development of each of the sufficiently isolated LMS microservices, arises the problem of the *reasonable choice* of an appropriate technology for storing the data that can alleviate or solve at data processing level the performance issues that appear due to simultaneous usage of the system by many users at the same time.

To solve the problem, it is necessary to develop information technology (IT), which, based on the processing of contextual requirements of business functions for microservices, would be able to highlight specialized characteristics (attributes) of storage and processing of various types of source data and to offer solutions for using modern DBMS during the creation of LMS based on microservices to the developer.

ANALYSIS OF EXISTING SCIENTIFIC AND TECHNOLOGICAL ACHIEVEMENTS AND PUBLICATIONS

As analysis of information technologies used in the development of various LMS show, there is an evolution of server architectural solutions from building the monolithic applications to distributed microservices [2], [5]. At the same time, the only advantage of the monolithic architecture of the server side in relation to the LMS is the simplicity of development, which is then discarded by the complexity of maintenance and operating of such a solution (Fig. 1a). That includes low fault tolerance, horizontal scaling issues, overly complicated code refactoring, etc. Market research shows that the provision of online educational services is a rapidly growing business, which leads to the need of constant maintenance and updates of LMS platforms, so the second architectural solution was to combine two monoliths or a monolith with services (Fig. 1b) [5]. Although this approach partially solves the issues of fault tolerance, scalability, and single technology stack of the previous solution, using the third architectural solution, a service-oriented architecture allows to achieve full modularity in development with high independency levels of components (Fig. 1c).

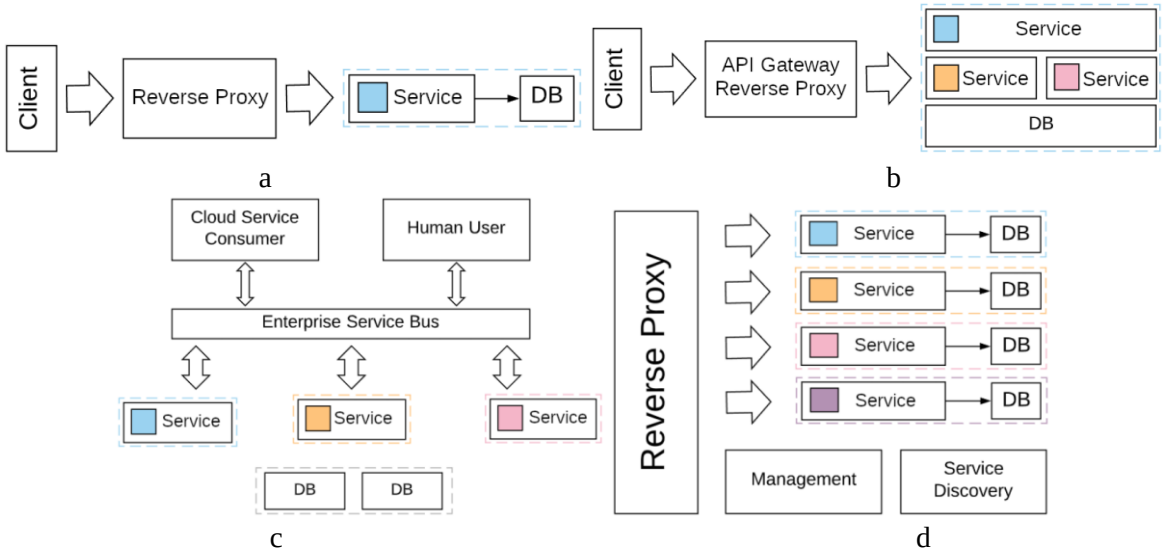


Fig. 1. **Architectural decisions in the design of the server side of the LMS: a – monolith; b – mix of monolith and microservices; c – service-oriented architecture; d – microservice architecture**

***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

The main disadvantage of this solution is the common Enterprise Service Bus, which acts as the only intermediary between the client and the server, as well as between its components (services and DBMS), which greatly affects the performance of the LMS.

Thus, the use of microservices to build the server side with *isolation of components both at the code level and at the database level* creating a distributed system is a good solution when designing high-performance complex LMS (Fig. 1d). Additionally, since the databases and DBMS in the microservice architecture are isolated, their practical use has the additional advantage of an increased level of data security due to the communication of services occurring only through the Remote Procedure Call (RPC) mechanism. However, problems appear when the used data is stored in databases within different microservices. In this case, ambiguous decisions are possible, for instance, storing the same value in two microservices, ensuring data migration by writing RPC or retrieving data from all databases for analytics, etc.

Therefore, the problem of organizing local databases for usage with isolated microservices and choosing the types of appropriate DBMS to control the storage and use of heterogeneous data that is present in the LMS arises. To solve it, the existing technological approaches for database design should be considered [1-2], [5-6].

It is known that the development of databases for systems such as LMS, operating with a large amount of heterogeneous data, consists of the stages of conceptual logical and physical modeling [7]. At the conceptual stage, the architect presents online training in the form of basic, auxiliary and management business processes, defines their essence and functions and identifies the boundaries of its ability to take into account the features of any particular DBMS. During the development of logical models, the *requirements for data storage and processing* are determined, which are used by selected entities to implement business functions. Data is presented in a strict and structured format, taking into account the properties of a particular data storage and processing model, for example, relational, document, key-value, etc.

The next step is the physical modelling, i.e. detailing of logical models taking into account the data storage and processing system in specific DBMS, such as PostgreSQL, Oracle, MongoDB, ElasticSearch, Redis, etc. [8].

Thus, the problem of organizing local databases for working with isolated microservices is solved at the stage of logical modeling.

Additionally, at the stage of logical modeling, while choosing a DBMS type, it is necessary to take into account what data processing technology is needed to implement an isolated microservice on the server side of the developed LMS. There are two types of technology: OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing) technologies. OLTP is a way to organize a database, in which the system works with small *transactions* that are going in a large flow, and at the same time the client requires a minimum response time from the system. For example, in the LMS system, a user can register for a course only by paying for it. OLAP is a data processing technology consisting in the preparation of aggregated information based on large data arrays structured according to the multidimensional principle [9].

Since a transaction is a set of actions with data combined into a logical unit that either completes or rolls back, for OLTP a *relational data model* should be used, that will satisfy ACID requirements [10]:

– Atomicity guarantees that each transaction is treated as a single “unit”, which either succeeds completely, or fails completely;

– Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants;

– Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially;

– Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (e.g., power outage or crash).

However, not every DBMS fully implement ACID principles. In order to manage big data flows an open source solution has been proposed [11, 12]: NoSQL – non-relational and distributed databases with support for flexible database schemas and horizontal scalability, which do not use SQL as a query language. The inability to support ACID transaction properties to some extent compensated by the CAP theorem proposed by Brewer, who suggested that web services cannot provide all three of the following properties at the same time (denoted by the abbreviation CAP) [11].

Thus, according to CAP, all distributed DBMSs provide *only two* of the three properties: consistency of data, availability and partition tolerance (Fig. 2) [11,12], [13]:

– **C**onsistency – the existence of only a single state of the data corresponding to the latest update operation;

– **A**vailability – the guaranteed response for every request, without the guarantee that it contains the most recent write;

– **P**artition tolerance – the nodes ability to continue to operate independently of each other.

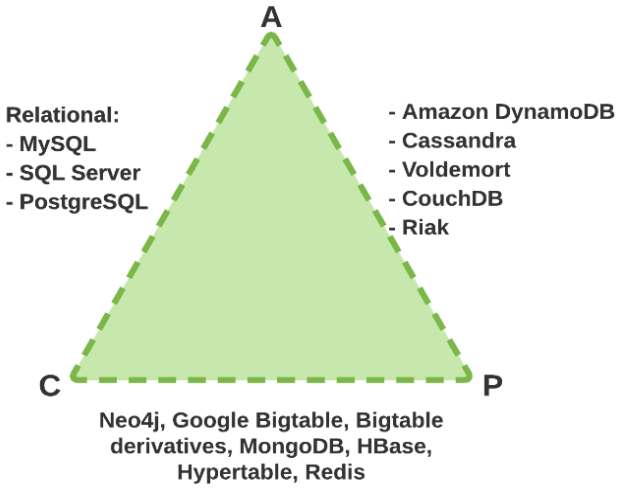


Fig. 2. **Illustration of the relationship of CAP and popular DBMS**

***Source*: compiled by the author**

Using the CAP theorem [13] to justify the choice of a DBMS in developing an LMS is needed to maximize combinations of consistency and availability. This approach assumes possibility to maintain the functioning of the system when dividing the network and restoring its connectivity. For example, at the beginning of the LMS operation, the data is consistent and remains so until the network is divided (Fig. 3).

After separation, in order to ensure data availability, the divided nodes continue to perform operations in the division mode, causing inconsistent data states S1 and S2. When network connectivity is restored, the process of recovering data after separation begins. During this process, the system combines the states S1 and S2 into a consistent state S ', and also compensates for all errors made in the division mode.

Therefore, distributed NoSQL DBMS is not a good solution for transaction management, however, studies show that ACID properties are almost impossible to maintain in LMS systems with a large international web audience. Thus, designers of NoSQL systems sacrifice data consistency in order to achieve two other properties from the CAP theorem [11-14].

At the same time, there is a possibility to create a universal solution in the form of multimodel databases [15]. The advantages of this approach include: the implementation of various logical models within a single product, data consistency, a single standard and query language, the presence of automatic optimization of combined queries [16-17]. However, when implementing microservice architecture with isolation of components at the database level, this approach is redundant [18].

In conclusion, based on [19-21], becomes apparent that the use of Polyglot Persistence for organizing data storage and processing in LMS based on multiple databases with different logical models on the server side has the following advantages: good system scalability, efficient heterogeneous data management, faster response times and, as a result, higher productivity.

However, based on the research of the authors [4], [17, 18] and experience [19, 20], [21] in the case of choosing Polyglot Persistence when creating LMS on microservices, in addition to such drawbacks as the need to integrate data into different storages, duplicate data, the complexity of the system, developers may encounter problems associated with the implementation, synchronization and support of various DBMS throughout the life cycle of the system.

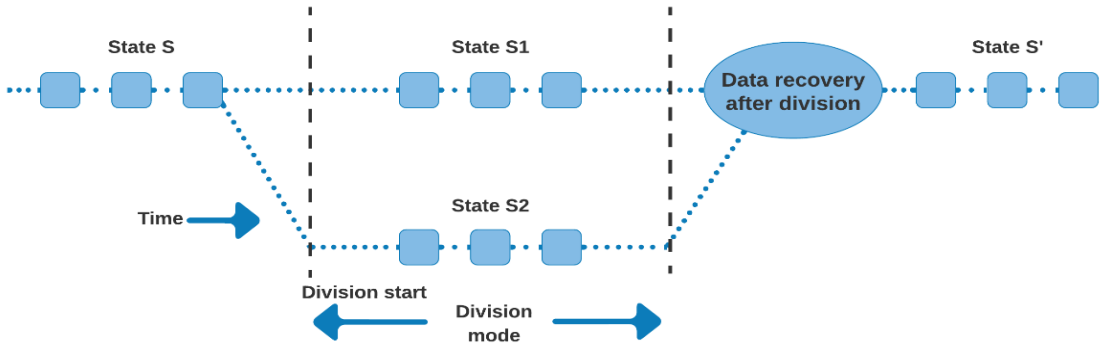


Fig. 3. Illustration of the work of the divided nodes in a network

***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

Therefore, the development of information technology to support the making of architectural decisions for the organization of heterogeneous LMS data using polyglot persistence concept and including the appropriate DBMS type choice is a relevant problem at the isolated microservices development stage, the solution of which will allow to reduce the negative impact of the above disadvantages.

THE PURPOSE AND OBJECTIVES OF THE RESEARCH

The purpose of the research is to develop the information technology (IT) of supporting reliable architectural solutions for organizing a various data by dint of polyglot persistence concept during the design of LMS using microservices to increase the performance of such systems.

To achieve the goal, the following objectives were accomplished:

I. Various features of architectural decisions for the organization of local databases in the design process of LMS have been analyzed.

II. Information technology able to make architectural decisions for organizing a various data according to polyglot persistence concept during the design process of LMS using microservices has been developed.

III. The advantages of the developed information technology in the design of LMS based on microservices have been shown.

STATEMENT OF THE MAIN RESEARCH MATERIAL

I. To develop information technology of supporting the architectural solutions for organizing various data according to polyglot persistence concept during the design of LMS using microservices *as part of the first objective*, the features of using relational and non-relational DBMS such as document, key-value, graph and column storage, have been analyzed.

It is known that *relational* DBMS better than others supports processing of small transactions, but occurring in a large stream, while providing minimal response time from the system to the client. At the same time, in a relational DBMS, the database is built as a strictly structured storage in which data is stored in interconnected tables (Fig. 4). So, for example, in the LMS a user can be enrolled in many courses and one course can have many users, taking into account the many lectures in each course and the many comments that the user can receive for each lecture, the essence of “courses” and “users”, “lectures” and “comments” should be connected by a *many-to-many* relationship (Fig. 4a). Then users can leave comments for each lecture of each course and see all their comments. The rigid structure of the lecture table is shown in Fig. 4b. Thus, despite the advantages of relational DBMSs such as security, data independence and duality, sophisticated multi-tenant solutions, data integration with cloud servers and data backup, the classical relational architecture performance takes a significant hit while processing large amounts of data [22].

Any attempts to adapt a relational DBMS to work with big data will lead to [23]:

– rejection of strict consistency;

– avoiding normalization and introducing redundancy;

– loss of SQL language expressiveness and the necessity to model part of its functions by additional code;

– significant complication of client software;

– the difficulty of maintaining the health and fault tolerance of the resulting solution.

Therefore, the cost of implementation and maintenance of relational solutions does not pay off. This leads to an increase in the practical use of various non-relational (NoSQL) models that process huge amounts of data, sharing it between servers, support a huge number of users, use a simplified, more flexible, unlimited database structure [24,25].

In *key-value* storages, data is presented in the form of an associative array or dictionary, which allows to quickly record and retrieve the data. Such structures allow data processing by a unique identifier – a key that can be a combination of identifiers [26].

The advantage of this model is the simplicity of the data format, which makes *writing and reading operations quick.* The disadvantage of the key-value storage is the *long search time* for values, the implementation of which requires scanning the entire collection or creating separate index values [27-28].

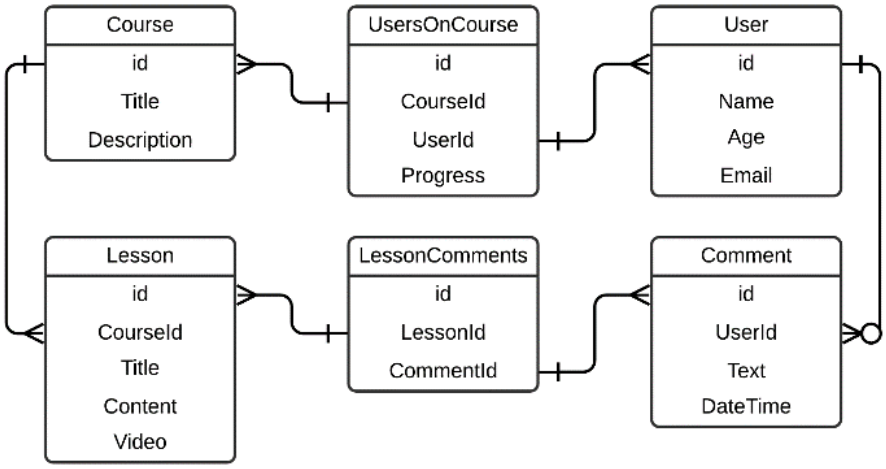
Fig. 5 shows an example of the key-value entries, containing a LMS student progress in the learning process. This is an example and frequently modified data. The combination of “user identifier” + “course identifier” can be used as a key. User’s progress percentage can be stored as “value”.

However, the “value” can be a complex structure in the “JSON” format, then such a “key-value” organization will resemble a document data model.

Document storages. This data model allows to combine many key-value pairs into an abstraction called a “document”. Documents can have a nested structure and combine into *collections* (Fig. 6a and Fig. 6b).

A data containing document can be formatted as an XML, JSON or YAML file. Processing of the documents is performed using a key that corresponds

to a specific document, however, there are solutions that allow to query by attribute values [24], [29].



a

Lesson				
Id	CourseId	Title	Content	Video
0	72	Python Variables	A **Python variable** is a reserved memory location to store values. In other words, a **variable** in a **python** program gives data to the computer for processing. Every value in **Python** has a datatype. Different data types in **Python** are Numbers, List, Tuple, Strings, Dictionary, etc.	<link_to_cloud_storage>
1	65	Java OOP	Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.	<link_to_cloud_storage>
2	162	Prototype JS	When a function is created in JavaScript, the JavaScript engine adds a prototype property to the function. This prototype property is an object (called a prototype object) that has a constructor property by default.	<link_to_cloud_storage>

b

Fig. 4. **Presentation of data storage about lessons of courses in RDBM during development of LMS: a – relations between tables in the RDBMS; b – lesson table in the RDBMS**

***Source*: compiled by the author**

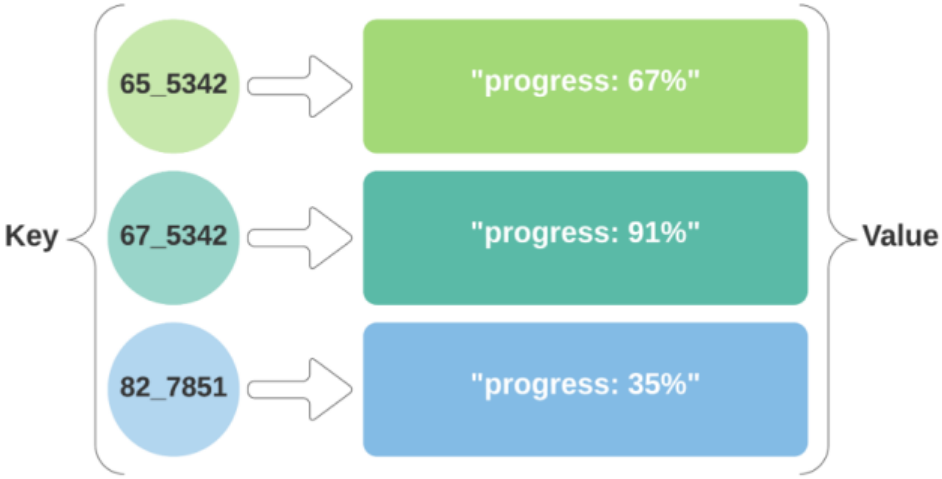
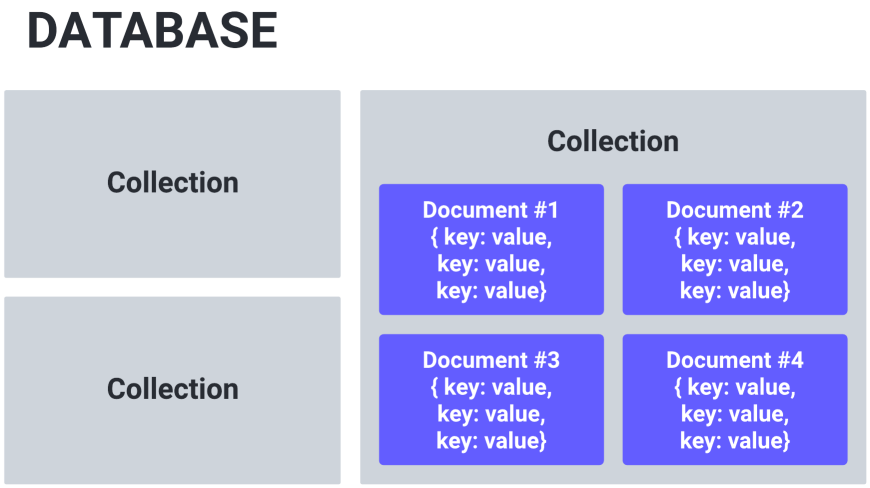


Fig. 5. **Presentation of data storage of the learning progress of users in NoSQL key-value storage**

***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)



a

```
{
    "id": "course_45665664",
    "title": "Python Starter",
    "mentor_id": 325345,
    "description": "It is good course.",
    "lessons": [
        {
            "id": "lesson_45443",
            "title": "Python Variables",
            "content": "APython variableis a reserved memory
            "video": "<link_to_cloud_storage>",
            "comments": [
                {
                    "id": "comment_344",
                    "user_id": "user_447678",
                    "datetime": "13:45-11/04/2020",
                    "text": "Must variable be declared befor
                }
            ]
        },
        {
            "id": "lesson_76754",
            "title": "Python Functions",
            "content": "A function is a block of code which
            "video": "<link_to_cloud_storage>",
            "comments": []
        }
    ]
}
```

b

Fig. 6. **Presentation of the storage of course data in document NoSQL storage: a – collections of the document storage; b – JSON presentation of course data**

***Source*: compiled by the author**

Due to the lack of a specific structure for storing such data, relational databases cannot be used, since such data cannot be represented in the form of tables.

In comparison of the solutions for organizing storage in the LMS shown in Fig. 4a and Fig. 4b, using the document model, it is possible to discard the additional table that is needed in a relational DBMS.

Fig. 6b shows the structure of the entity "course", which contains the identifier and the name of the course, the identifier of the teacher, a description, and also contains a list (array) of lessons that the course consists of, in turn, the lessons contain their identifier and name, a link to the video, description, as well as a list (array) of comments related to them, with the indicated authors (in the form of a user identifier) of the corresponding comments. Comments also contain the identifier of the comment itself, the date it was sent, and text (comment body).

Document storages organized in the form of such collections are able to immediately extract all the necessary data set for loading the course page with a list of the lessons contained in it, user comments, etc. It should be noted that the consistency and inconsistency of the structure of individual documents can be both the advantage and disadvantage of document DBMS, depending on the type of tasks to be solved for which they are used [30, 31]. The analysis shows that document models should be used to solve problems associated with the *need to frequently extract* large-volume documents, while *requests for storing such documents are much less common* [32].

Column storages are somewhat similar to traditional relational DBMS. Data in column storages forms a kind of non-interpreted byte arrays addressed by tuples.

The primary basic unit of that data model is a column. The number of columns for one table can be unlimited. Key columns are combined into families with a specific set of properties. Column storages are arranged in the form of long continuous data blocks and are good for *processing large data arrays* [23], [33,34], [35,36]. An example of storing information about the LMS user profile is shown in Fig. 7.

As a result, data organized in this way is good if there is a need to get statistics about users. For example, to find out what categories of users (schoolchildren, students, working people) are potential consumers of the resource. To obtain such information, it is enough to extract a single column with ages and then the result is obtained by simple calculations or as another example, a notification mailing list containing news or new educational content can be organized for users by extracting mail addresses.

Thus, column storages, provided the decentralization of data is present, maintain a high performance of data processing when obtaining analytical information by quickly extracting only the necessary data type from the database. In addition, compression is effectively applied in such storages due to the uniformity of data in the column [36].

Graph storages. Data is represented as vertices, edges, and properties. Data is searched by traversing the graph along the edges with the specified properties. They are used in case the relationship between the data is important.

It is known that, on large data sets, processing relationships in relational databases leads to a significant performance decrease, while the performance of the graph database remains constant, since the speed of traversing neighboring nodes of current vertex does not depends on the size of the graph. In addition, graph repositories use the principle of index-free adjacency, which means each vertex supports only the data about its neighbors [3], [31], [37, 38]. An example of representing entities "users" and "courses" in the LMS using a graph model is shown in Fig. 8.

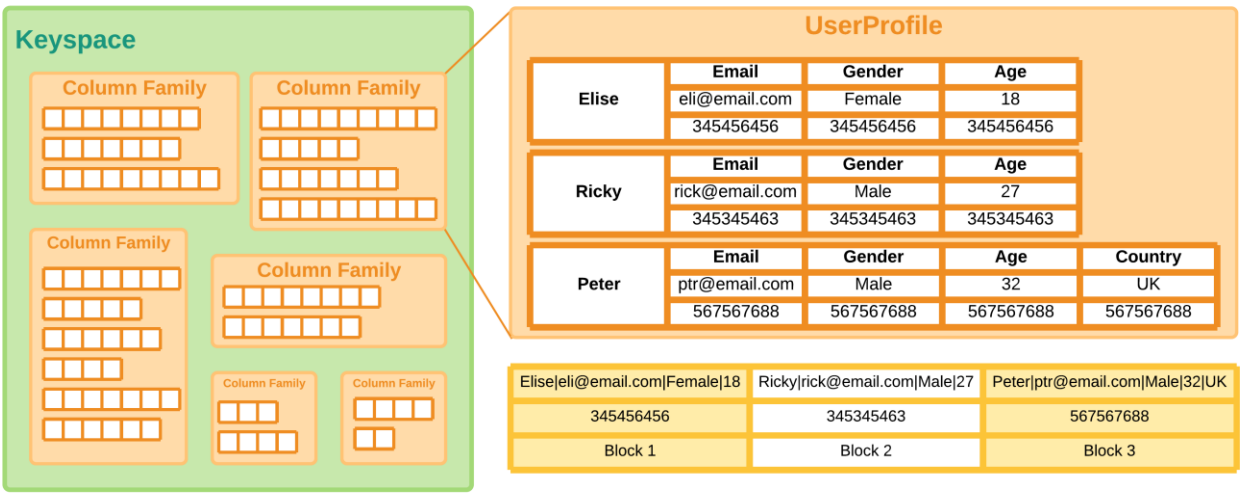


Fig. 7. **Representation of storing user data in column NoSQL storage**
***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

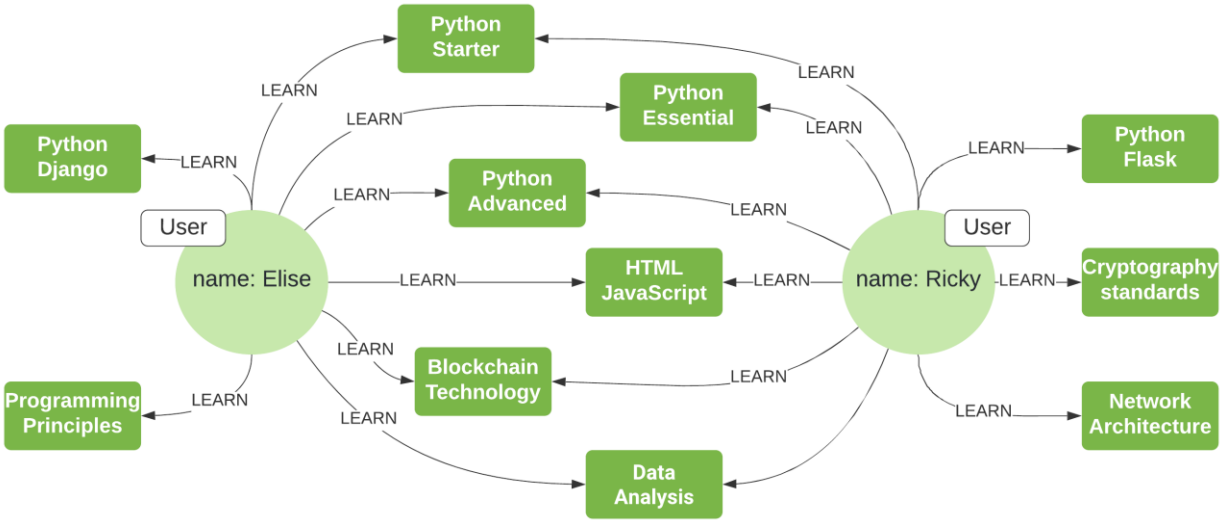


Fig. 8. **Presentation of user preference data storage in NoSQL graph storage**
***Source*: compiled by the author**

As a result, with the usage of a graph data model combined with well-known graph traversal algorithms, matches of the "tastes" or interests of users can be found, and then used to recommend similar courses that might interest other users with similar interests. Thus, graph models are used to implement a recommendation system of courses for users when they search for similar content of their preferences and interests, or recommendations of materials related to a certain level of user knowledge.

In conclusion, it is determined that different logical models provide good performance and efficiency indicators for solving a certain kind of tasks (Table).

Table. **The correlation of database types and tasks**

Database type	Task type
SQL	Transaction processing
NoSQL	
Key-value	Fast data read and write
Document-based	Searching system
Column-based	Analytical tasks
Graph-based	Recommendation system

***Source*: compiled by the author**

II. As part of the solution for the second objective, information technology of supporting architectural solutions for organizing a variety of data according to polyglot persistence concept in the design of LMS using microservices was developed.

IT requires the following steps:

1. Description of the subject area using a conceptual diagram.

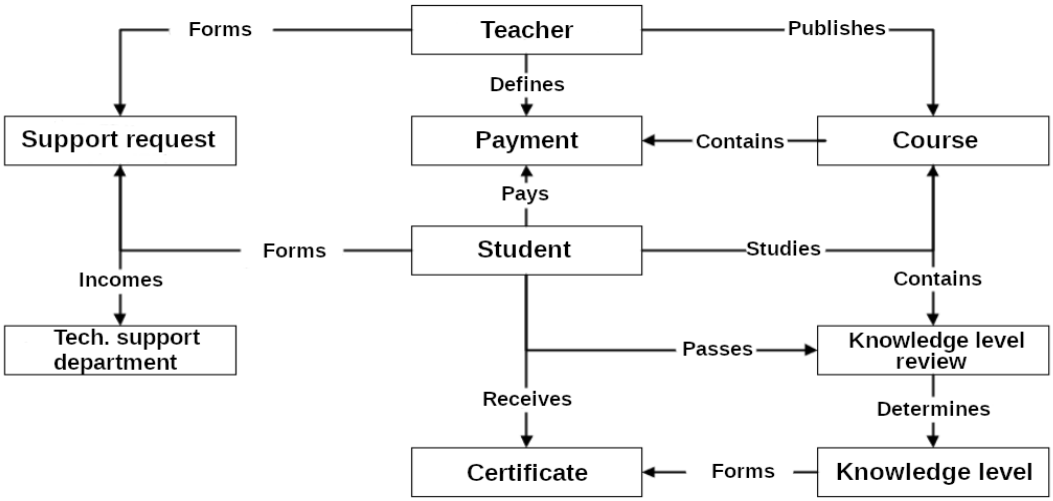
2. Definition of a limited context for each microservice (task) in the process of designing the microservice architecture of LMS.

3. Determining the type of data storage of dedicated microservices.

At the first stage, the developer analyzes the subject area – the company's business processes, which should be automated with the help of LMS. The primary entities and the relationships between them are determined. Tools for constructing conceptual diagrams are suggested to use for their representation, an example of which is shown in Figure 9a. The developer has identified and represented the following business processes using entities and relationships: the teacher publishes the course, students (users) subscribe to it, pay for it, study the course materials, and then take the knowledge test to determine if they are eligible to receive the certificate or not, in case there are problems during the learning process or resource publications, users (teachers and students) will form a request for appropriate technical support. Arrows indicate the direction of communication between entities.

At the second stage, to highlight a limited context, usage of a CRUD matrix (Create, Read, Update, Delete) is suggested, the construction of which allows to correlate the actions of the system with data elements (individual or their combination), which gives an idea of where and how each data item is created (C), read (R), updated (U) and deleted (D).

The first row of the matrix (Fig. 9b) contains the entities shown in the conceptual diagram, while the "student" and "teacher" are combined into a "user" entity. The first column of the matrix contains a set of the basic functionality of the system (options for using the entity). Matrix-filled areas of the matrix correspond to selected limited contexts, for each of which a separate microservice is designed.



a

Use case / entity	User	Knowledge level	Course	Tuition fee	Knowledge control	Certificate	Tech. support request	Tech. support department
User registration	C	C						
Deleting user	D	D	D	D	D	D		
Editing user data	U							
Course search			R					
Adding a Course	R		C	C	C	C		
Deleting a Course	R		D	D	D	D		
Editing a Course			U	U	U	U		
Course study			R	R	R			
Determining the knowledge level	R	U			R			
Obtaining a Certificate	R	R				R		
Support request creation	R						C	
Support request processing	R						RD	

b

Fig. 9. **Description of the subject area:**
a – a conceptual diagram; b – CRUD matrix
***Source*: compiled by the author**

Given the analysis and information in Table, within each such area, the most priority operations are concentrated, namely, the creation and updating (C, U). Operations such as deleting and reading (D, R) are considered less significant, so they are excluded from some areas marked with color.

It is worth noting that the “Course Search” function is marked by a reading operation and highlighted in a separate context, because the system can store a huge number of courses at which users can carry out quick searches.

In addition, it is worth paying attention to the fact that the limited context, marked in blue on the matrix, is quite large. In the future, when it is implemented in the form of a microservice, it will be advisable to use data caching mechanisms in order to reduce the load and the number of requests to the same database.

Therefore, five limited contexts were highlighted.

1. Users processing: registration, account deletion, which will entail the removal of courses published by corresponding user, editing user data.
2. Search by courses, full-text search, keyword search.
3. Courses processing: add, delete, update. Each course contains tasks for reviewing user’s knowledge, information about the cost of the course and a certificate that is issued at the end of the course.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

4. Technical support requests processing: creating and reading with its subsequent removal.

5. Data caching.

At the third stage, in order to build the architectural solution of the server side of the LMS with components that are independent both at the code and database levels (Fig. 1d), it is necessary to determine the type of data storage for the selected limited contexts. With that purpose, a technique for automated selection of logical data models based on the initial information about a limited context was developed (Fig. 2 and Fig. 3). The flowchart for the automated selection of a logical data model is shown in Fig. 10. The input receives a description of the limited context in the form of a document; the output determines the type of logical data model.

Data in the source space is presented as a document in text format. For example, a description of a Python beginner course in an online educational resource would be as follows: "The Python Starter" video course is suitable for those who are just starting to understand programming. Of the seven logically interrelated video lessons, you learn the basics of legitimization and programming in the Python language."

In the process of *preprocessing the input data*, features are highlighted (Fig. 10).

Following features are suggested to describe the type of data storage:

– *Letters / numbers / specific characters' occurrence frequency in the line*, tracking of these signs will help distinguish the description of the course or webinar, user information from the fields with the dates of the webinar. In such fields numbers and less often specific signs (slash, period, etc.) are often found, progress represented as percent (numerical value), cached data contain numbers and letters approximately equally. It is also important to highlight data that does not carry any value within the framework of this system, and filter it out (high frequency of occurrence of specific characters).

– *Data volum*e, introduction of this feature allows to detect and deal with data that could potentially clog the system. For example, a database dump is usually large and does not make sense to store it in a DBMS specifically within the framework of this system.

– *The searchability / viewability requirement*, tracking of this feature will help to identify the data which will be the subjects to search, namely courses, lessons and webinars in the online educational resource.

– *The binary sign of the presence / absence of meaning*, will help with distinguishing the description of the course, lesson, webinar from user data (name, surname, place of residence), and also discard a meaningless incoherent set of words.

– *The format and shelf life* – the introduction of these features in addition to the size of the data will help to understand whether it is worth saving such data in the system. For example, a document in the *".html"* format will be received, which may contain a description of the course, the previous parameter will determine that it makes sense, but because it's just a page layout and not even a cache, it makes no sense to save it to the database.

According to the proposed methodology, for the *presentation of data in the attribute space*, it is necessary to conduct a structural analysis by constructing data flow diagrams (DFD) (Fig. 11).

Data flow diagrams simulate information transfer (information and material flows) between participants in the information exchange process (functions, data warehouses, external entities). The diagram (Fig. 11) shows how the functions (numbered blocks) exchange data streams. In the same way, they interact with external entities (indicated by a closed block) and data warehouses (indicated by an open block).

Next comes the analysis of the information flows (entities) (for example, "Progress" and "Complaint") and their representation in the attribute system: the storage period and purpose of the data, the searchability requirement and data volume and storage format. Unfortunately, for now the values of the characteristics are filled in manually by the developer.

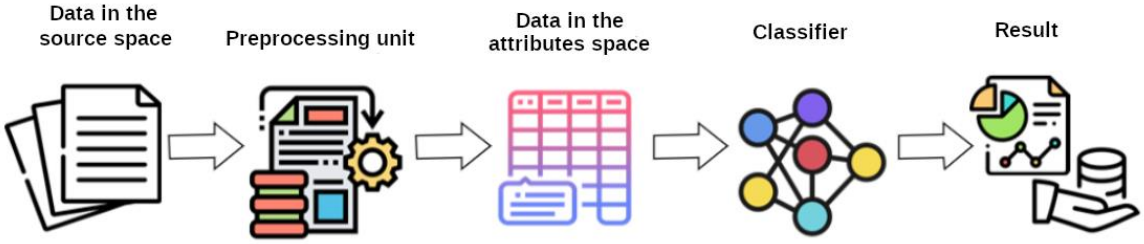


Fig. 10. **Steps of the methodology for automated selection of a logical data model**

***Source*: compiled by the author**

For example, the document storage format can be defined using specialized libraries; for Python, these are "fleep-py" and "python-magic". The frequency of occurrence of letters / numbers / specific characters in a string is calculated as the total number of characters divided by the number of letters, numbers or specific characters, depending on what outcome is required or expected. The existence of a meaning in the document is determined using the NLP (Natural Language Processing) tools for analyzing text using contextual hints for machine learning. Moreover, the algorithm can identify, differentiate and, therefore, classify words and phrases [41].

A fragment of LMS entities represented in the attribute space is shown in Fig. 12.

Building a *classifier* is suggested to determine the type of logical model appropriate for the data represented in the attribute space. (Fig. 13) [42].

The CART decision tree algorithm using the Gini coefficient [43] was selected to solve the classification problem. The efficiency of the classifier was tested on a dataset containing 230 entities, which was randomly divided into training and test samples with a ratio of 200/30. In the target field in the training sample, the answer was the number of one of the three possible classes, i.e. types of logical models: SQL (1), NoSQL (2), is not subject to storage (3).

The classification reliability (true positive rate, TPR) was calculated by the formula:

$$TPR = \frac{TP}{TP + FN} \cdot 100\ \%, \qquad (1)$$

where: TP – truly positive decisions; FN – positive decisions, classified as negative (type II error).

The result of the experiment has shown that the reliability of the classification was 93 %. An analysis of the results showed that a relational DBMS was suggested to use for user entities of small and medium size, text format and long-term storage, and NoSQL storage model was suggested for informational entities with a searchability requirement or in XML and JSON formats. System entities are not subject to storage at all. Further studies showed the necessity to expand the classifier for the five types of logical models shown in Table.

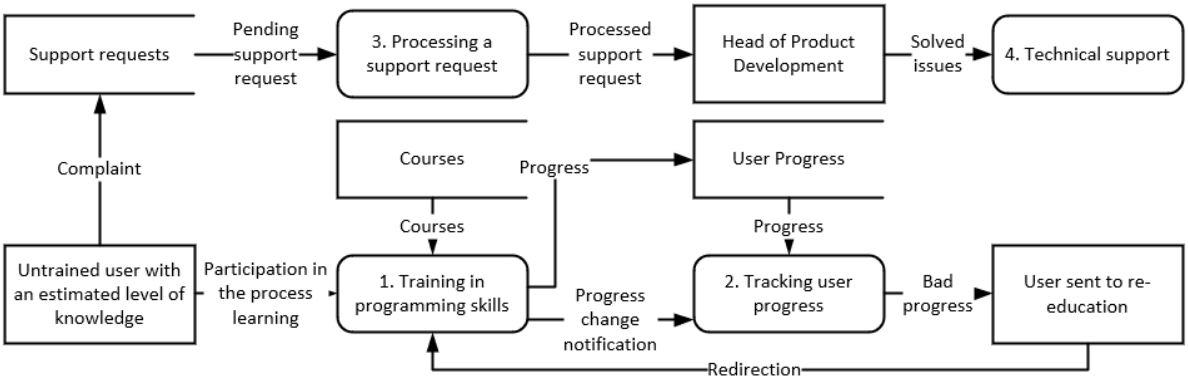


Fig. 11. **Example DFD to review user's knowledge level in the LMS**
***Source*: compiled by the author**

#	Shelf life	Data purpose	Search ability	Data volume	Format
1	long-term	system	no	average	HTML
2	short-term	informational	no	large	JSON
3	short-term	system	no	average	HTML
4	short-term	informational	no	small	JSON
5	short-term	system	no	large	JSON

Fig. 12. **Presentation of data in attribute space**
***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

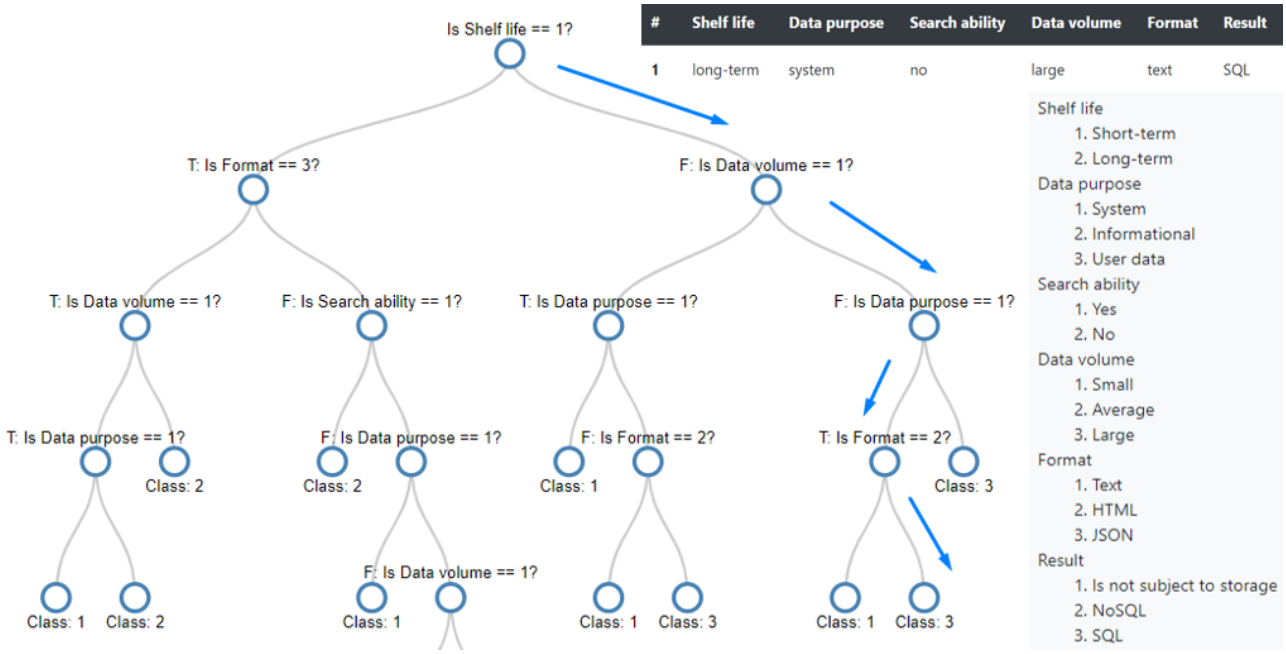
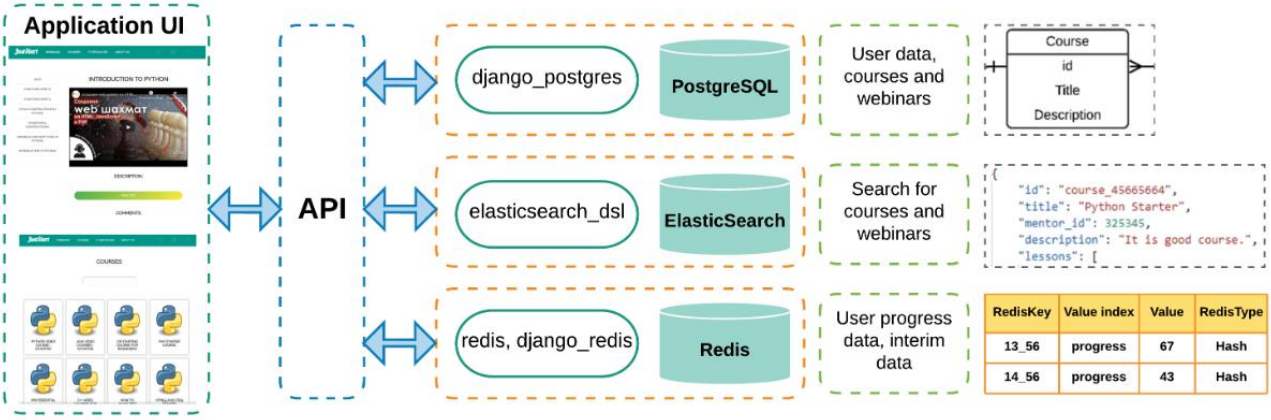


Fig. 13. **Visualization of the decision tree**
***Source*: compiled by the author**

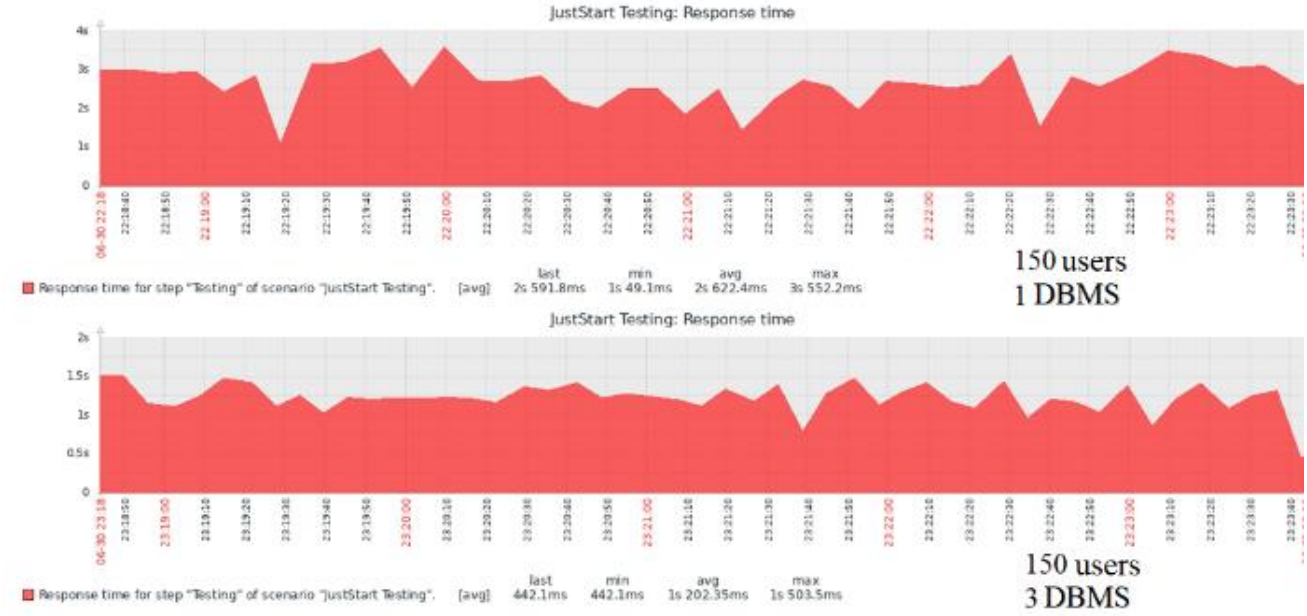
III. The advantages of the developed information technology of supporting architectural solutions for organizing various data according to polyglot persistence concept are shown on the example of designing the JustStart LMS, the logical presentation diagram of which is shown in Fig. 14a.

The list of tasks that the resource performs included: the ability of taking courses and viewing webinars which contain video content, as well as knowledge level review after each lesson and upon completion of the course. Each user is required to register to obtains access to the resource and to data on their own progress. In addition, the resource provides a search function for the basic textual information of courses and webinars, as well as the caching of the pages to ensure their quick loading.

The standard technology stack of HTML / CSS, JavaScript and Python, including the corresponding libraries for interacting with various DBMS was used to develop both client and server side. The usage of three different DBMS was implemented: PostgreSQL, Redis, ElasticSearch. The client interaction with the resource is performed through the Application Programming Interface (API). User data and educational content is stored in PostgreSQL. The Redis DBMS is used for cached pages and often updated or retrieved data; the search among educational content is performed using ElasticSearch (Fig. 14a).



a



b

Fig. 14. **An example of the implementation of LMS using developed IT: a – a logical representation; b – stress test results**

***Source*: compiled by the author**

The stress testing was carried out to evaluate the speed of the developed JustStart LMS, the results of which are shown in Fig. 14b. An analysis of the results shows that due to the load distribution between the three databases, the average response time of the system with 150 users working simultaneously was 1.2 s. At the same time, when simulating the work of the same number of users with only one DBMS, the response time increased and averaged about 2.6 s. (Fig. 14b). Therefore, the usage of the developed information technology of supporting architectural solutions for organizing large volumes of various data according to polyglot persistence concept allowed to design and implement LMS that is under a load of simultaneous usage by a large audience performs on average twice as good as the average educational resource on the market.

CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

As a result of the research, it was proved that the performance of learning management systems significantly depends on the architectural solution for data storage and processing chosen during its design. The use of microservices architecture for the server side with components independent at the code level and distributed at the database level was found to be the good solution in the design of high-performance learning management systems. Therefore, in order to implement polyglot persistence concept in the learning management system based on several databases with different logical models on the server side, it is necessary to develop an information technology to support such architectural solutions.

The analysis of the modern authors researches has led to the conclusion that the development of databases for systems such as LMS that operate with a large amount of heterogeneous data, although requires conceptual, logical and physical modeling, but the requirements for storage and processing of the data used by selected entities to implement business functions, are determined during the creation of logical models process [4], [7], [17,18], [19, 20], [21], [44, 45].

Therefore, during the development of information technology of supporting architectural solutions for organizing various data according to polyglot persistence concept, it is suggested to automate problem solving at the following three stages: description of the subject area using a conceptual diagram; allocation of a limited context for each microservice (task) in the process of designing the microservice architecture of the LMS; determination of data storage type for dedicated microservices.

To implement the third stage, a method for automated selection of logical data models based on initial information about a limited context has been developed. This method consists of the following steps: obtaining data in its original form, preprocessing, highlighting the attributes, representation of data in the attribute space, classification and analysis of classification results.

The CART decision tree algorithm using the Gini coefficient [43] was selected to solve the classification problem. The efficiency of the classifier was

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

tested on a dataset containing 230 entities. The result of the experiment has shown that the reliability of the classification was 93 %, as a relational DBMS was suggested to use for user entities of small and medium size, text format and long-term storage, and NoSQL storage model was suggested for informational entities with a searchability requirement or in XML and JSON formats.

The advantages of the developed information technology of supporting architectural solutions for organizing a variety of data according to polyglot persistence concept is shown in the JustStart LMS design example. An analysis of the results shows that due to the load distribution between the three databases, the average response time of the system with 150 users working simultaneously was 1.2 s, while during the simulation of the same number of users with only one DBMS, the response time increased and averaged about 2.6 s.

Therefore, the usage of the developed information technology of supporting architectural solutions for organizing large volumes of various data according to polyglot persistence concept allowed to design and implement LMS that is under a load of simultaneous usage by a large audience performs on average twice as fast as the average educational resource on the market.

In conclusion, it is necessary to note that the authors consider promising the continuation of research in the direction of expanding the dimension of the attribute space to represent highlighted entities, and classification algorithms for five types of logical data storage models in the development of learning management systems.

REFERENCES

1. “List of Top LMS Software Companies of 2020”. *Finances Online*. – Available from: https://financesonline.com/top-20-lms-software-companies/. – Active link – 9 May 2020.

2. “Martin Fowler, Microservices”. – Available from: https://martinfowler.com/articles/microservices.html. – Active link – 9 May 2020.

3. “Martin Fowler, Polyglot Persistence”. – Available from: https://martinfowler.com/bliki/PolyglotPersistence.html. – Active link – 10 May 2020.

4. Srivastava, K. & Shekokar, N. “A Polyglot Persistence Approach for E-Commerce Business Model”. *International Conference on Information Science (ICIS).* 2016. p. 7–10. DOI: https://doi.org/10.1109/infosci.2016.7845291.

5. “Po stopam luchshih: mikroservisnaja arhitektura v razreze” [Following the steps of the best: microservice architecture breakdown] (in Russian). – Available from: https://proglib.io/p/po-stopam-luchshih-mikroservisnaya-arhitektura-v-razreze-2019-11-07. – Active link – 9 May 2020.

6. Di Francesco, P., Lago, P. & Malavolta, I. “Migrating Towards Microservice Architectures: An Industrial Survey”. *IEEE International Conference on Software Architecture (ICSA)*. 2018. DOI: https://doi.org/10.1109/icsa.2018.00012.

7. Martinez-Mosquera, D., Navarrete, R. & Lujan-Mora, S. “Modeling and Management Big Data in Databases”, *A Systematic Literature Review. Sustainability*. 2020; 12(2): 634 p. DOI: https://doi.org/10.3390/su12020634.

8. Storey, V. C. & Song, I.-Y. “Big Data Technologies and Management: What Conceptual Modeling Can Do”, *In: Data & Knowledge Engineering* 108. 2017. p. 50–67. DOI: https://doi.org/10.1016/j.datak.2017.01.001.

9. Mohammad Sadoghi, Souvik Bhattacherjee, Bishwaranjan Bhattacharjee & Mustafa Canim. “L-Store: A Real-time OLTP and OLAP System”, *In: Proceedings of the 21st International Conference on Extending Database Technology (EDBT).* March 26-29 2018. p. 540–551. DOI: https://doi.org/10.5441/002/edbt.2018.65.

10. Khine, P. P. & Wang, Z. “A Review of Polyglot Persistence in the Big Data World”, *Information 2019.* 10(4). 141 p. DOI: https://doi.org/10.3390/info10040141.

11. Pritchett, D. “BASE: An ACID Alternative”. *Queue.* 2008; 6(3): 48–55. DOI: https://doi.org/10.1145/1394127.1394128.

12. Yashraj Sharma & Yashasvi Sharma. “Case Study of Traditional RDBMS and NoSQL Database System”. *In: International Journal of Research – Granthaalayah.* 2019; 7(7): 351–359. DOI: https://doi.org/10.5281/zenodo.3364448.

13. Kleppmann, M. “A Critique of the CAP Theorem”. 2015. DOI: https://doi.org/10.17863/CAM.13083.

14. Klemenkov, P. & Kuznetsov, S. “Bol'shie dannye: sovremennye podhody k hraneniju i obrabotke” [Big data: modern approaches to storage and analysis] (in Russian. *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*. 2012; 23. DOI: https://doi.org/10.15514/ISPRAS-2012-23-9).

15. Pete Aven & Diane Burley. “Building on Multi-Model Databases”. *Published by O’Reilly Media*. Inc. 1005 Gravenstein Highway North, CA 95472. – Available from: http://info.marklogic.com/rs/371-XVQ-609/images/building-on-multi-model-databases.pdf. – Active link – 10 May 2020.

16. Jiaheng Lu, Irena Holubová & Bogdan Cautis. “Multi-model Databases and Tightly Integrated Polystores: Current Practices, Comparisons, and Open Challenges”. *In: CIKM ’18*. October 22-26 2018. Torino: Italy. DOI: https://doi.org/10.1145/3269206.3274269.

17. Lu, J. & Holubová, I “Multi-model Databases”, *In: ACM Comput. Surv*. 52, 3, Article 55. June 2019. p. 1–38. DOI: https://doi.org/10.1145/3323214.

18. Irena Holubova, Meike Klettke & Uta Storl. “Evolution Management of Multi-model Data”. *Springer Nature Switzerland AG 2019*. V. Gadepally et al. (Eds.): DMAH 2019/Poly 2019. LNCS 11721. 2019. p. 139–153. DOI: https://doi.org/10.1007/978-3-030-33752-0_10.

19. Olena O. Arsiri, Alina O. Glumenko & Diana E. Stelmakh. “Characteristics of the Development of an IT-project for Creation of Open Educational Resource”. *The Third International Research Conference “Project, Program, Portfolio Management, 2018”*.p. 21–23. Odessa: Ukraine.

20. Olena O. Arsiri, Alina O. Glumenko & Diana E. Stelmakh. “Rozrobka Onlajnovogo Osvitn"ogo Resursu z Vikoristannjam Bagatovariantnoï Persistentnosti”. [The Development of an Educational Resource Based on Polyglot Persistence] (in Ukrainian). *Materials of the Ninth International Scientific Conference of Students and Young Scientists on the 55th Anniversary of the Institute of Computer Systems. “Modern Information Technology – 2019”*. MES of Ukraine; Odessa. Nat. polytechnic. univ.; Inst. of computers. Systems. *Ecology*. May 23-24, 2019. p. 97–98. Odessa: Ukraine.

21. Arsiri, Olena O. & Glumenko, Alina O. “Rozrobka Onlajnovogo Navchal"nogo resursu z Vikoristannjam Bagatovariantnoï Persistentnosti”. [The Development of an Educational Resource Based on Polyglot Persistence] (in Ukrainian). *VII International scientific & technical conference “Informatics. Culture. Technologies*”. 23.09.-25.09.2019. p. 35–36. Odessa: Ukraine.

22. Bellatreche, L., Valduriez, P. & Morzy, T. “Advances in Databases and Information Systems*”. In: Inf. Syst Front 20.* 1-6 (2018). DOI: https://doi.org/10.1007/s10796-017-9819-2.

23. Jannatul Maowa, A. H. M. Sajedul Hoque, Rashed Mustafa & Mohammad Osiur Rahman. “A Comparative Study on Big Data Handling Using Relational and Non-Relational Data Model”. *In: International Journal of Data Mining & Knowledge Management Process (IJDKP)*. May, 2017; Vol.7 No.3. DOI: https://doi.org/10.5121/ijdkp.2017.7302.

24. Klemenkov, P. A. “Postroenie Novostnogo Rekomendatel’nogo Servisa Real’nogo Vremeni s Ispol"zovaniem NoSQL SUBD” [Building Real-time News Recommendation Service Using NoSQL DBMS] (in Russian). *In: Informatics and Applications.* September, 2013; Vol.7 Issue 3: 14–21. DOI: https://doi.org/10.14357/19922264130302.

25. Moniruzzaman, A. B. & Hossain, S.A. “NoSQL Database: New Era of Databases for Big Data Analytics – Classification, Characteristics and Comparison”. *In: International Journal of Database Theory and Application.* ArXiv, abs/1307.0191. 2013; Vol.6 No. 4.

26. Li, Z. “NoSQL Databases”. *In: The Geographic Information Science & Technology Body of Knowledge. (2nd Quarter 2018 Edition).* John P. Wilson (Ed). DOI: https://doi.org/10.22224/gistbok/2018.2.10.

27. Hieu Nguyen (Jack). “The Pros and Cons of Different Data Formats: Key-values vs Tuples” [Digital Resource]. – Available from: https://www.freecodecamp.org/news/the-pros-and-cons-of-different-data-formats-key-values-vs-tuples-f526ad3fa964/. – Active link – 11 May 2020.

28. Alves Florencio, D., Ricardo Freitas de Oliveira, D., Laisa Soares Xavier Freitas, E. & da Fonseca de Souza, F. “Which Fits Better? A Comparative Analysis about NoSQL Key-Value Databases”. *In: IEEE Latin America Transactions* (in Portuguese). Nov. 2017; Vol.15 No.11: 2251–2256. DOI: https://doi.org/10.1109/TLA.2017.8070434.

29. Chevalier, M., El Malki, M., Kopliku, A., Teste, O. & Tournier, R. “Document-oriented Models for Data Warehouses – NoSQL Document-oriented for Data Warehouses”, *In: Proceedings of the 18th International Conference on Enterprise Information Systems.* ICEIS, ISBN 978-989-758-187-8. 2016; Vol.1: 142–149. DOI: https://doi.org/10.5220/0005830801420149.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

30. Davoudian, A., Chen, L. & Liu, M. "A Survey on NoSQL Stores*", In: ACM Computing Surveys,* Vol. 51, No. 2, Article 40. *Publication date: 2018.* p. 1–43. DOI: https://doi.org/10.1145/3158661.

Ameya Nayak, Anil Poriya & Dikshay Poojary. "Type of NOSQL Databases and its Comparison with Relational Databases". *In: International Journal of Applied Information Systems (IJAIS)* – ISSN: 2249-0868. *Foundation of Computer Science FCS.* March, 2013, Vol. 5 No. 4. New York: USA.

31.Kumar, K. B. S. & Srividya Mohanavalli, S. "A Performance Comparison of Document-oriented NoSQL Databases". *In: 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*. DOI: https://doi.org/10.1109/icccsp.2017.7944071.

32. Weidong Wen, Yang Li, Wenhai Li, Lingfeng Deng & Yanxiang He. "CORES: Towards Scan-Optimized Columnar Storage for Nested Records", *In: ACM Transactions on Storage*. June 2019; Vol.15 No. 3. 46 p. Article 16. DOI: https://doi.org/10.1145/3321704.

33."What are the Pros and Cons of Using a Columnar Database for Data Mining? Are there Any Characteristics of the Databases that Can Be Used for Deciding?". – Available from: https://www.quora.com/What-are-the-pros-and-cons-of-using-a-columnar-database-for-data-mining-Are-there-any-characteristics-of-the-databases-that-can-be-used-for-deciding. – Active link – 10 May 2020.

34. "What are the Pros and Cons of Document and Column-oriented Databases?" – Available from: https://www.quora.com/What-are-the-pros-and-cons-of-document-and-column-oriented-databases. – Active link – 10 May 2020.

35. Bhagat, V. & Gopal, A. "Comparative Study of Row and Column Oriented Database". *In: 2012 Fifth International Conference on Emerging Trends in Engineering and Technology*. DOI: https://doi.org/10.1109/icetet.2012.56.

36. Martins de Sousa, V. & Del Val Cura, L. M "Logical Design of Graph Databases from an Entity-Relationship Conceptual Model", *In: Proceedings of the 20th International Conference on Information Integration and Web-Based Applications and Services*. Yogyakarta: Indonesia. November 2018. p. 183–189. DOI: https://doi.org/10.1145/3282373.3282375.

37."What are the pros and cons of using a graph database?" – Available from: https://www.quora.com/What-are-the-pros-and-cons-of-using-a-graph-database. – Active link – 10 May 2020.

38. Roopa Tangirala & Thomas Betts. "Polyglot Persistence Powering Microservices". *In: QCon San Francisco 2018*. – Available from: https://www.infoq.com/presentations/microservices-polyglot-persistence. – Active link – 10 May 2020.

39. Khine, P. P. & Wang, Z. "A Review of Polyglot Persistence in the Big Data World", *In:* Information 2019, 10(4). DOI: https://doi.org/10.3390/info10040141.

40. Nazaruka, E. & Osis, J. "Determination of Natural Language Processing Tasks and Tools for Topological Functioning Modelling". *In: Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering.* MDI4SE. 2018; Vol.1: 501–512. DOI: https://doi.org/ 10.5220/0006817205010512.

41. Çığşar, B. & Ünal, D. "Comparison of Data Mining Classification Algorithms Determining the Default Risk". *In: Scientific Programming.* 2019. p.1–8. DOI: https://doi.org/10.1155/2019/8706505.

42. Harsh H. Patel & Purvi Prajapati. "Study and Analysis of Decision Tree Based Classification Algorithms". *International Journal of Computer Sciences and Engineering*. 2018; 6(10): 74–74. DOI: https://doi.org/10.26438/ijcse/v6i10.7478.

43. Glava, M. G., Malakhov, E. V., Arsiri, O. O. & Trofymov, B. F. "Information Technology for Combining the Relational Heterogeneous Databases Using an Integration Models of Different Subject Domains". *Applied Aspects of Information Technology. Pub. Science i Technical.* Odessa: Ukraine. 2019; Vol.2 No.1: 29–44. DOI: https://doi.org/10.15276/aait.02.2019.3.

44. Müllenbach Sabine, Kern-Bausch Lore & Kolonko Matthias. "Conceptual Modeling Language Agila Mod". *Herald of Advanced Information Technology. Publ. Science i Technical.* Odessa: Ukraine. 2019; Vol.2 No.4: 246–258. DOI: https://doi.org/10.15276/hait.04.2019.1.

Conflicts of Interest: the authors declare no conflict of interest

Received 05.04. 2020
Received after revision 10.06. 2020
Accepted 15.06. 2020

DOI:https://doi.org/10.15276/aait.02.2020.1
УДК 004.8

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПІДТРИМКИ АРХІТЕКТУРНИХ РІШЕНЬ БАГАТОВАРІАНТНОГО ЗБЕРІГАННЯ ДАНИХ В СИСТЕМАХ ДИСТАНЦІЙНОГО НАВЧАННЯ

Олена Олександрівна Арсірій[1)]
ORCID: https://orcid.org/0000-0001-8130-9613, e.arsiriy@gmail.com
Марія Геннадіївна Глава[1)]
ORCID: https://orcid.org/0000-0002-9596-9556, glavamg@gmail.com
Маттіас Колонко[2)]
ORCID: https://orcid.org/0000-0002-8296-1758, matthias.kolonko@hs-augsburg.de
Аліна Олегівна Глуменко[1)]
ORCID: https://orcid.org/0000-0002-6085-3422, alina.glumenko@gmail.com
[1)] Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна
[2)] Університет прикладних наук, 1, An der Hochschule. Аугсбург, 86161, Німеччина

АНОТАЦІЯ

В роботі показано, що продуктивність систем дистанційного навчання істотно залежить від, обраного при її проектуванні, архітектурного рішення по зберіганню і обробці даних. На основі аналізу еволюції серверних архітектурних рішень, при проектуванні інформаційних систем, починаючи з побудови монолітної платформи до розподілених мікросервісів встановлено, що використання архітектури мікросервісів для серверної частини з ізоляцією компонентів на рівні коду і розподілу на рівні баз даних, є хорошим рішенням при проектуванні високопродуктивних комплексних систем дистанційного навчання. Однак, для реалізації багатоваріантного зберігання даних в системі дистанційного навчання на основі декількох баз даних з різними логічними моделями на стороні сервера необхідно розробити інформаційну технологію підтримки таких архітектурних рішень. Показано, що розробка баз даних для таких систем як системи дистанційного навчання, що оперують великим обсягом різноманітної інформації, складається з етапів концептуального логічного і фізичного моделювання і саме при створенні логічних моделей визначаються вимоги до зберігання та обробки даних, якими оперують виділені сутності для реалізації бізнес функцій. Детально проаналізовано особливості використання реляційних і нереляційних систем управління базами даних таких як: документні, ключ-значення, графові і колонкові сховища. Розроблено методику автоматизованого підбору логічних моделей даних на основі вихідної інформації про обмежений контекст, на основі якої в подальшому було розроблено класифікатор. Працездатність класифікатора перевірялася на наборі даних для двохсот тридцяти сутностей. В результаті проведення експерименту достовірність класифікації склала дев'яносто три відсотки. Переваги розробленої інформаційної технології підтримки архітектурних рішень по організації багатоваріантного зберігання різноманітних даних показані на прикладі проектування системи дистанційного навчання JustStart. Аналіз результатів навантажувального тестування розробленої системи показує, що завдяки розподілу навантаження між трьома базами даних, її середній час відгуку при одночасній роботі ста п'ятдесяти користувачів становить близько однієї і двох десятих секунди. У той час як, при моделюванні роботи цієї кількості користувачів тільки з однією системою управління базами даних час відгуку збільшився і становив в середньому близько двох і шести десятих секунди. Таким чином, використання розробленої інформаційної технології підтримки архітектурних рішень по організації багатоваріантного зберігання великих обсягів різноманітних даних, дозволило спроектувати і реалізувати систему дистанційного навчання, швидкодія якої в разі її одночасного використання численною аудиторією в середньому в два рази більше , ніж у середньостатистичного освітнього ресурсу, представленого на ринку.

Ключові слова: система дистанційного навчання; багатоваріантна персистентність; мікросервісна архітектура; дерева прийняття рішень

DOI: https://doi.org/10.15276/aait.02.2020.1
УДК 004.8

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ПОДДЕРЖКИ АРХИТЕКТУРНЫХ РЕШЕНИЙ МНОГОВАРИАНТНОГО ХРАНЕНИЯ ДАННЫХ В СИСТЕМАХ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

Елена Александровна Арсирий[1)]
ORCID: https://orcid.org/0000-0001-8130-9613, e.arsiriy@gmail.com
Мария Геннадьевна Глава[1)]
ORCID: https://orcid.org/0000-0002-9596-9556, glavamg@gmail.com
Маттиас Колонко[2)]
ORCID: https://orcid.org/0000-0002-8296-1758, matthias.kolonko@hs-augsburg.de
Алина Олеговна Глуменко[1)]
ORCID: https://orcid.org/0000-0002-6085-3422, alina.glumenko@gmail.com
[1)] Одесский национальный политехнический университет, пр. Шевченко, 1. Одесса, 65044, Украина

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

[2)] Университет прикладных наук, 1, An der Hochschule. Аугсбург, 86161, Германия

АННОТАЦИЯ

В работе показано, что производительность систем дистанционного обучения существенным образом зависит от, выбранного при ее проектировании, архитектурного решения по хранению и обработке данных. На основе анализа эволюции серверных архитектурных решений, при проектировании информационных систем, начиная с построения монолитной платформы до распределенных микросервисов установлено, что использование архитектуры микросервисов для серверной части с изоляцией компонентов на уровне кода и распределенности на уровне баз данных, является хорошим решением при проектировании высокопроизводительных комплексных систем дистанционного обучения. Однако, для реализации многовариантного хранения данных в системе дистанционного обучения на основе нескольких баз данных с разными логическими моделями на стороне сервера необходимо разработать информационную технологию поддержки таких архитектурных решений. Показано, что разработка баз данных для таких систем как системы дистанционного обучения, оперирующих большим объемом разнородной информации, состоит из этапов концептуального логического и физического моделирования и именно при создании логических моделей определяются требования к хранению и обработке данных, которыми оперируют выделенные сущности для реализации бизнес функций. Детально проанализированы особенности использования реляционных и нереляционных систем управления базами данных таких как: документные, ключ-значение, графовые и колоночные хранилища. Разработана методика автоматизированного подбора логических моделей данных на основе исходной информации об ограниченном контексте, на основе которой в дальнейшем был разработан классификатор. Работоспособность классификатора проверялась на наборе данных для двухсот тридцати сущностей. В результате проведения эксперимента достоверность классификации составила девяносто три процента. Преимущества разработанной информационной технологии поддержки архитектурных решений по организации многовариантного хранения разнообразных данных показаны на примере проектирования систем дистанционного обучения JustStart. Анализ результатов нагрузочного тестирования разработанной системы показывает, что благодаря разделению нагрузки между тремя базами данных, ее среднее время отклика при одновременной работе ста пятидесяти пользователей составило около одной и двух десятых секунды. В то время как, при моделировании работы этого количества пользователей только с одной системой управления базами данных время отклика возросло и составило в среднем около двух и шести десятых секунды. Таким образом, использование разработанной информационной технологии поддержки архитектурных решений по организации многовариантного хранения больших объемов разнообразных данных, позволило спроектировать и реализовать систему дистанционного обучения, быстродействие которой в случае ее одновременного использования многочисленной аудиторией в среднем в два раза больше, чем у среднестатистического образовательного ресурса, представленного на рынке.

Ключевые слова: система дистанционного обучения; многовариантная персистентность; микросервисная архитектура; деревья принятия решений

ABOUT THE AUTHORS

Olena O. Arsirii, Dr. Sci. (Eng), Professor, Head of the Information Systems Department, Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
e.arsiriy@gmail.com. ORCID: https://orcid.org/0000-0001-8130-9613
***Research field*:** Information Technology, Decision Support Systems, Machine Learning, Neural Networks

Олена Олександрівна Арсірій, д-р техн. наук, проф., зав. каф. Інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

Maria G. Glava, PhD, Assistant Professor of the Information Systems Department, Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
glavamg@gmail.com, ORCID: https://orcid.org/0000-0002-9596-9556
***Research field*:** Domain Modeling, Relational Databases

Марія Геннадіївна Глава, PhD (Eng), доцент каф. Інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

Matthias Kolonko, Assistant of Computer Science Faculty, Augsburg University of Applied Sciences (AUAS), 1, An der Hochschule. Augsburg, 86161, Germany
matthias.kolonko@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-8296-1758
***Research field*:** Database Development for Applied Information Systems, Conceptual Modeling of Subject Areas, Semantic Data Modeling

Маттіас Колонко, асистент факультету комп'ютерних наук, Університет прикладних наук. An der Hochschule 1, Аугсбург, 86161, Німеччина

Alina O. Glumenko, Master Student of the Information Systems Department, Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
alina.glumenko@gmail.com., ORCID: https://orcid.org/0000-0002-6085-3422
***Research field*:** Database Systems, Machine Learning, Problem Solving, Programming Languages

Аліна Олегівна Глуменко, магістрант каф. інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)