ACADEMICIA: An International Multidisciplinary Research Journal
ISSN: 2249-7137 Vol. 11, Issue 12, December 2021 SJIF 2021 = 7.492
A peer reviewed journal

DEVELOPMENT OF AN ORDER FLOW MANAGEMENT INFRASTRUCTURE IN THE FORMATION OF A RELIGIOUS DATABASE

Alimjon Dadamuxamedov*

*Senior Teacher,
"Department of Modern ICT",
International Islamic Academy Of Uzbekistan,
Tashkent, UZBEKISTAN
Email id: a.dadamuhamedov@iiau.uz

DOI: 10.5958/2249-7137.2021.02624.0

ABSTRACT

The article suggests the use of replication methods to increase the reliability and optimal presentation of data stored in distributed information systems of a religious database. The software architecture of the replication mechanism determines the interaction of software components in the process of exchanging information with a separate node of the system.In the context of parallel operation of the server with many client applications, the operation of a system with such an architecture is considered. The problems that arise and ways to solve them are also considered and suggested.

KEYWORDS: *Religious Database. Replication, Distributed Information System, Query Intensity, Telecommunication Network, Datacenter,*

INTRODUCTION

E. Tanenbaum in his book "Distributed Systems. The fundamental monograph "Ideas and Principles" **[1]** describes the distributed system as follows:"A distributed system is an interconnected set of independent computers using communication channels, which is viewed from the user's point of view as a single system based on special software."It is known that the level of practical logic and the level of data access perform the functions required by the software components to work with the set of applications that make up the system.A set of client components on the server automatically creates a set of copies of software components during simultaneous operation.A similar confirmation can be added to the simultaneous running account of the application using the replication mechanism function.Activating copies of application layer components involves activating copies of data entry layer components.

The recent data warehouse explosion has led to the development of innovative systems for large-scale data processing.In processing BigData databases, Google Megastore and other systems have been introduced to process large amounts of data across data center boundaries.However, most transactional databases are less than 1-2 TB in size, which shows that data overgrowth is not a mandatory requirement for small and medium enterprises. Modern applications that are currently in use tend to experience the rapid growth and variability of users due to the advent of the internet and devices connected to the internet.Therefore, the increase in workload volume,

i.e., the ability to meet the increasing demands of the workload, as well as support the flexibility to process changes in these workloads, is critical for existing database samples**[2].**

LITERATURE REVIEW

The strategy we used to create the search strings was as follows **[3]:** • finding papers about distributed information system. • Listing keywords mentioned in primary studies, which we knew about. • Use synonyms word (usage) and sub subjects of network technology in data replications such as (distributed information system, database, replication, query intensity, telecommunication network, datacenter). • Use the Boolean OR to incorporate alternative spellings and synonyms. • Use the Boolean AND to link the major terms from population, intervention, and outcome. The complete search string initially used for the searching of the literature was as follows: network technology AND data replications. It has been highlighted in **[4]** that there are two main issues on conducting an SLR search which are the sensitivity and specificity of the search. In our preliminary search, when we used the complete search string defined above we retrieved a very high number of articles. For instance, Google scholar, Scopus, ProQuest education, IEEEXplore, Science Direct, Springer Link retrieved more than two hundred results. Therefore, we have deepened our search and used this search string: (Adoption OR Usage)AND (religious database. OR database) AND (query intensityOR telecommunication network). The revised search string has given us a reasonable number of studies and we finally selected relevant empirical studies

THE MAIN PART

Reliability of information storage and functional stability of servers within distributed databases are one of the main factors determining the quality of modern distributed systems.A common way to protect yourself from data loss is to back up these databases. When data is lost from memory for a variety of reasons, the availability of a backup allows the database to be restored quickly.The backup process does not require additional hardware to be added to the system, as backups can be made on the server itself or on external storage media **[5].**

In this case, each group of copies of components activated by one application is installed in the database as shown in Figure 1.1. Thus, there is no limit to the number of components that can be activated, leading to an uncontrolled increase in the number of active connections to the database.

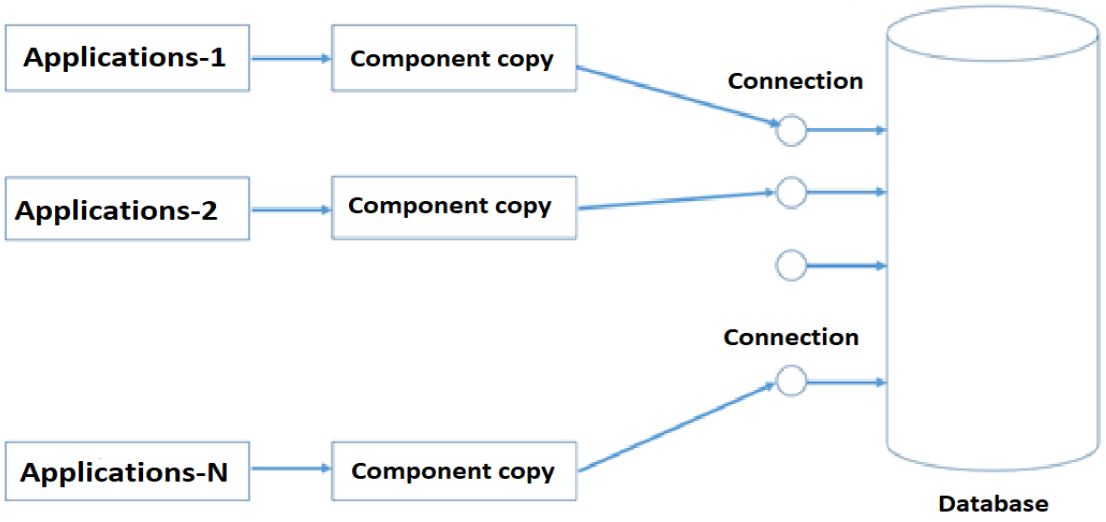

Figure 1.1 (install each group of copies of activated components in the database)

Any combination of a client application with a database server will, firstly allocate additional memory space and secondly the load on the server's computing resources will increase as the number of active connections running in parallel increases.As a result when the number of active connections increases to a certain critical level, there is a significant decrease in the efficiency of the database server.The development of large-scale distribution computing systems is based on the inefficiency of the client-server architecture.One of the conditions for the rapid operation of the system is to limit the number of active connections to the database **[6-9].**

SOLUTIONS METHOD

In addition, an unlimited increase in the number of running components reflects an increase in the number of active database transactions, which can negatively affect the stability of the system.The nature of this effect can be assessed by considering the following model.Suppose there is a database of m tables.There are n active transactions for writing and K transactions for reading in this database.We assume that each transaction works in a single table from its database.

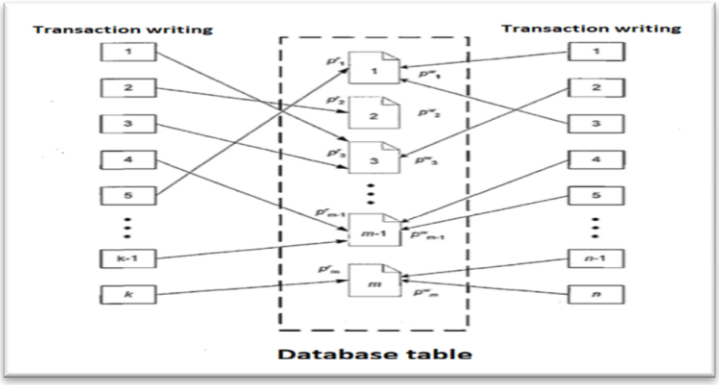

Figure 1.2 cannot be considered inconsistent.

As you know read and write transactions are generated in different encryption databases as shown in Table 1.1.

Transaction type	Reading	Writing
Concealment rate	A range of keys in individual records and indexes	Tables, indices
Compatibility with readable transactions	Yes	possible
Compatibility with writing transaction	possible	No

TABLE 1.1.

We enter probabilities for each table in the database and

$$\sum_{i=1}^{m} p_i^r = 1, \quad (1.1.1)$$

$$\sum_{i=0}^{m} p_i^r = 1, \quad (1.1.2)$$

In that p^r_t i-the probability that a newly started transaction to read the table will apply, p^r_t - in that 1-corresponds to the probability that a newly started transaction will apply to the record in the table.The distribution of tables between transactions is considered non-contradictory if no transaction between n transactions in writing refers to a table that uses at least one transaction in writing or reading at that time.Example of non-contradictory distribution. The distribution shown in Figure 1.2 cannot be considered inconsistent **[10,11].**

Let us consider the relationship property of the probability of a non-contradictory distribution obtained from the parameters m, n and k.We denote this probability by P (k, n, m).The case is maintained if k transactions only work with tables remaining n = m. The general expression for the probability of non-contradictory combinations in an equally probabilistic selection of tables is as follows.

$$P(0,m,m) = \prod_{i=1}^{m} p_i^w. \quad (1.1.3)$$

Since all tables are engaged in write operations, the occurrence of at least one read transaction leads to a conflicting situation in this case.If N = 1, if all the read transactions distribute the m-1 tables in any way, the situation is conflict-free, except for a single table involved in the status recording operation.The probability of this distribution is as follows

$$P(k,1,m) = \sum_{i=1}^{m} p^w{}_i \left[1-(p^w{}_i)^k\right]. \quad (1.1.4)$$

1 <n <m has non-conflict distribution options $\binom{m}{n}$ tables between write operations.If the read transactions split the remaining m - n tables, a conflict-free situation is maintained.

If $p^r{}_t = p^w{}_i = \frac{1}{m}, 1 \le i \le m,$ then the problem of finding the probability m of the distribution of tables between operations to write combinations is not reduced to the problem of selection by regression. As shown in **[12]**, $\left(\frac{m}{n}\right)$

There is a repetitive selection of N objects with a total number of possible combinations of M equal to M.Thus, for each record, the probability of distributing disputes for transactions is equal.

Tables $p^i(0, n.m) = \frac{\frac{m}{n}}{m^n} = \frac{1}{m^n} \cdot \frac{m!}{n!(m-n)!} = \frac{1}{m^n} \cdot \frac{(n+1)...(m-1)m}{(m-n)!}.$ (1.1.5)

If the read transactions only work with the remaining m-n tables, a conflict-free situation is maintained.Therefore, the general expression for the probability of a non-collision combination with the selection of tables with the same probability looks as follows:

$$P^i(k, n.m) = \frac{(m-n)^k}{m^k} \cdot \frac{1}{m^n} \cdot \frac{(n+1)...(m-1)m}{(m-n)!} = \frac{(m-n)^k}{m^{n+k}} \cdot \frac{(n+1)...(m-1)m}{(m-n)!}. \quad (1.1.6)$$

For each table, as mentioned above $\{p^r{}_i\}_i^m$ and $\{p^w{}_i\}_i^n$ if the specific values of the probabilities are determined, then the probability of a conflict-free combination is calculated as follows:

$P(k, n.m) = P(A)P(B) =$

$$= \sum_{i=1}^{m-n+1} p^w{}_i \sum_{j=i+1}^{m-n+2} p^w{}_j \sum_{l=j+1}^{m-n+3} p^w{}_l ... \sum_{u=i+1}^{m} p^w{}_u \cdot \left(\sum_{\substack{v \neq i, \\ v \neq j, \\ v \neq l, \\ .... \\ v \neq u}} p^r{}_v \right), \quad (1.1.7)$$

In this case, the events representing the transactions in records A and B are distributed without intersections, and the transactions in the reading work with the remaining m-n tables.

The ratio (1.1.5) - (1.1.7) allows us to conclude that the probability of distribution without contradictions quickly disappears as we approach n m. (1.1.5), (1.1.6) grow much faster than n in the relationship. (1.1.7).The probability of a non-contradictory distribution decreases with increasing n.In this model, even under the influence of n parameters, the stability of the system is much more importan**.**

Limiting the number of self-active transactions is an important condition for ensuring the stability of the system.Thus, in order to ensure high efficiency of the server part of the system in the mass operation of client applications, two problems need to be solved:

1. Enable a set of client applications to work with a limited number of active connections to the database

2. Limit and control the number of active transactions in the database.

The problem of limiting the number of active combinations with the database [13,14] can be solved in two ways.

- By setting the maximum number of connections that can be used in the BD server settings
- By creating a set of associations - with a specially organized set of components used by the objects to access the data when accessing the BD.

The disadvantage of these methods is that any new task that attempts to access the database while all connections are running will be rejected.This option is much preferred in the mass work environment of client applications.A message sequence should be used to organize such a scheme of work**.**

Using the message queue allows the client module and server components to interact asynchronously.By queuing the message, the client module takes control immediately.It can wait for the results of the message to be processed as needed, or continue to work on a state-of-the-art basis.If the required system resources are available at the time of queuing, the order will be processed immediately.The asynchronous type of interaction is required for the client module primarily if the system notifies the server part of certain events and the outcome of server operations is not critical to system performance.For example, in the case of a replication distribution mechanism, asynchronous interaction is used at the stage of recording evidence of uploading updated information about an object in a local database.

The client application does not have to wait for the server to confirm that the data storage evidence has been successfully recorded in the central database.An interruption in this operation will result in the object data being re-uploaded to the workstation during the next update session.At this time, asynchronous interaction involves ways to notify the server of an interruption in the execution of an order if the interruption is critical to the operation of the system **[13].**

While the client application expects to receive certain information from the server as a result of message processing, it is advisable to develop a compromise option that combines both synchronous and asynchronous interaction features.Its essence is that after the server sends a message to the queue, the client application for a predetermined period of time, firstly a confirmation that the message was successfully queued, and secondly, a confirmation that the server has started processing the message, the third waits for the results of the operation.When any of the three listed breaks is completed, the operation is considered invalid and the application user receives an appropriate error message.

The main stages of the client module operation in the interaction with the server are described in the status diagram shown in Figure 3.5. The duration of the waiting period for each stage of the work is set individually depending on the operating conditions of the system.

The second task is to limit the number of objects to be activated in the system, taking into account that excessive reduction of the maximum allowable number of objects to be activated leads to an unwanted reduction in the bandwidth of the system.This leads to a rapid accumulation of orders in the queue, which leads to an increase in the time of the order in the system and the

response time.To optimally solve this problem, it is recommended to use a two-factor decomposition of a set of practical components in the system**.**

The first factor is the functional orientation of the components.In a multifunctional database system, as a rule, information is collected that belongs to different subsystems.Tables containing this information form a set of groups that are independent of each other.Based on the analysis of the practical logic of the components, they can be divided into appropriate functional groups.This division allows you to minimize the number of possible conflicts between components during data access.

The second factor in decomposition is the way the database components are accessed: read or write.As mentioned above, to increase the stability of the system, it is necessary to limit the number of transactions running in parallel in writing, which leads to a limit on the number of activated components running in writing.

Based on the results of the two-factor decomposition, component packages are formed, on the basis of which server applications are created.Server applications represent multi-threaded processes that process incoming messages.The number of streams in a process determines the number of copies of practical components received from a single component packet, which in turn determines the limit number of connections used by the database**.**

By summarizing the above, it is possible to form a generalized approach to the decomposition of a set of components of the practical level of the system. It includes the following main steps:

1. Separating logically separated pieces of information from a system database
2. Divide the set of practical components of the system into groups corresponding to the allocated information subsystems
3. Separate the components of each group according to the form of operations performed on the data
4. Formation of component packages based on the performed decomposition
5. Defining the scope of server applications that manage component packages.

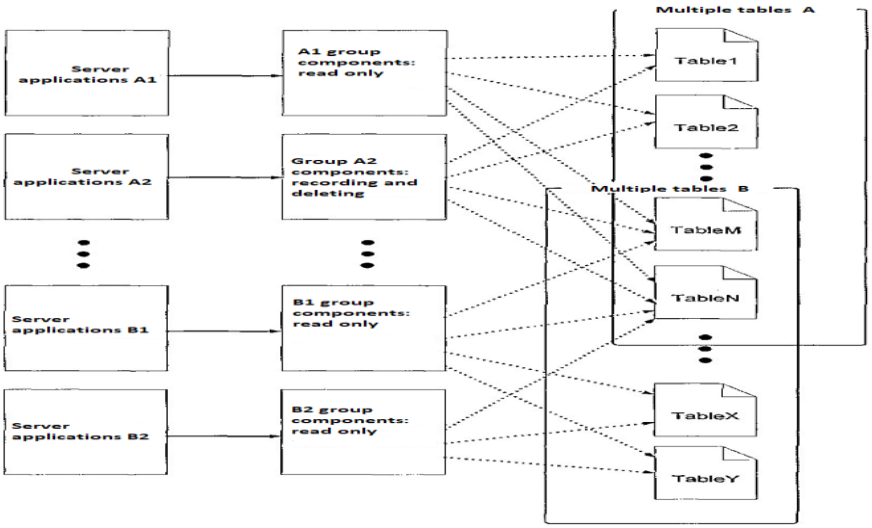

Figure 1.6(cross-section of logical subset sets of database tables)

An important condition for system efficiency when using this approach is the low cross-section of logical subset sets of database tables.In the example of the scheme shown in Figure 1.6, this condition can be expressed as follows:

$$\begin{aligned} &\square = A \cap B, \\ &\dim(C) \square \dim(A), \\ &\dim(C) \square \dim(B). \end{aligned} \qquad (1.1.8)$$

In the example of the architecture of the data replication mechanism discussed earlier, we consider the use of two-factor decomposition of components.According to this scheme, the components of SourseTable1 ..... SourseTableN refer to the database only for reading data.The RegUpdate component performs reverse write operations.The Fetcher application level component is read-only, and the Registrar is a group of components that work with data from subset sets of tables.Components that work with other sub-collections are grouped separately while maintaining read-write constraints **[14].**

RESULTS

To organize the work of a set of server applications, a multi-channel process is implemented in the system structure, which performs multi-channel processing of incoming messages, which are then redirected to the server applications.During start-up, the dispatcher must define the following configuration parameters:

- The amount of identifying information in the queue

- Parallel execution flows of processing for each queue

- Criteria for choosing the direction of forwarding incoming messages

Adjusting the number of incoming messages allows you to ensure high flexibility of the dispatcher to the system environment.Normally, a single queue that receives messages from all applications will suffice to organize the interaction of a client module with a level of practical logic.In the case of integration of several functional subsystems into a single core of practical logic, there may be a need to create an additional queue.To avoid the need to include a separate dispatcher for each queue, it is recommended to consider the possibility of parallel processing of several queues within one dispatcher.

It is recommended that server applications use a range of message codes as a criterion for choosing a forwarding route.A schematic of the order flow management infrastructure structured taking into account the above is shown in Figure 1.7.

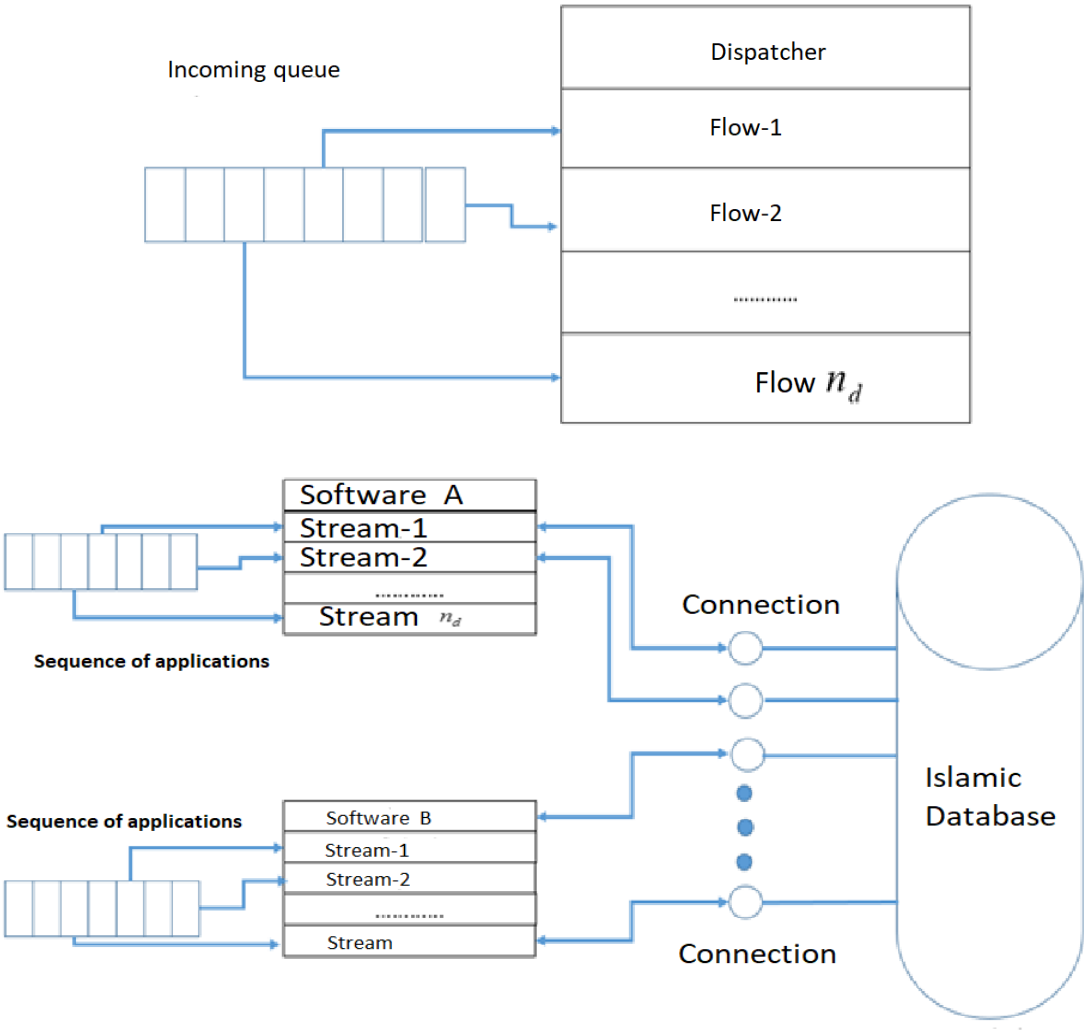

Figure 1.7. (A schematic of the order flow management infrastructure)

The proposed approach to the organization of public service infrastructure within the level of practical logic is more universal.For example, server applications can manage both local nodes and component packages installed on a database server.The placement of the incoming message dispatcher is optional. It can be extended in the server node or in the server application.

CONCLUSION

As a replication-based mediation system, it can provide workload scalability for existing databases without compromising consistency.The parallel replication algorithm allows faster access to extension databases, the routing algorithm prevents delay by redirecting the read data to sequential replications. We believe that automated flexibility and cloud-based innovation can be the basis for exciting research.

The described principle of building a public service system has the following advantages:

1. Universality and flexibility to system environment conditions

2. Ability to control the number of activated copies of components

3. Minimize the risks associated with parallel flow collisions when accessing resources

4. Provide queue processing at high intensities.

REFERENCES

1. Tanenbaum E, Van-Steen M. Distributed system. Principles and pa-radigmy. Spb.: Peter, 2003. P.877.

2. Georgiou M, Panayiotou M, Odysseos L, Paphitis A, Sirivianos M, Herodotou H. Attaining Workload Scalability and Strong Consistency for Replicated Databases with Hihooi. Proceedings of the International Conference on Management of Data. 2021

3. Belousov VE. Algorithms of replication of data in distributed systems of information processing (Doctoral dissertation, Penza: PGU, 2005. 184 p..

4. Nishonboev T. Software Configured Networks. Study Guide. (Griffith). TUIT printing house named after Muhammad al-Khwarizmi. 2017 (p. 186)

5. Nishanbayev TN,, Abdullayev MM,, Maxmudov SO. The model of forming the structure of the 'cloud' data center. International Conference on Information Science and Communications Technologies: Applications, Trends and Opportunities, ICISCT 2019.

6. Belousov V E. Algorithms of replication of data in distributed systems-max processing of information. Int mat: http://diss.rsl.ru/diss/05/0591/050591031pdf.

7. Irgashevich DA. Development of national network and corporate networks (in the case of Tas-IX network). International Journal of Human Computing Studies. 2019;1(1):1-5.

8. Irgashevich DA. Development of national network (tas-ix). ACADEMICIA: An International Multidisciplinary Research Journal, 2020;10(5):144-151.

9. Dadamuhamedov IA. Cloud technologies in islamic education institutions. The Light of Islam, 2020;2 (23).

10. Dadamuxamedov A, Mavlyuda X, Turdali J. Cloud technologies in islamic education institutions. ACADEMICIA: An International Multidisciplinary Research Journal. 2020;10(8):542-557.

11. Dadamuxamedov A. The impact of online communication on youth education. International Engineering Journal For Research & Development, 2020;5(6):10.

12. Jumayev TS, Mirzayev NS, Makhkamov AS. Algorithms for segmentation of color images based on the allocation of strongly coupled elements. Studies of technical sciences. 2015; 4: 22-27.

13. Abdujabborovich MA. Human personal identification algorithms from the image of the ear. International Engineering Journal For Research & Development, 2020;5(6):5-5.

14. Тухтаназаров ДС. Computer models for process management of developing oil and gas fields. Проблемы вычислительной и прикладной математики, 2017;(2):41-46.