ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)

UDC 004.42 DOI: https://doi.org/10.30837/ITSSI.2022.19.064

M. PERETIATKO, M. SHIROKOPETLEVA, N. LESNA

RESEARCH OF METHODS TO SUPPORT DATA MIGRATION BETWEEN RELATIONAL AND DOCUMENT DATA STORAGE MODELS

The **subject** matter of the article is heterogeneous model-inhomogeneous data migration between relational and document-oriented data storage models, existing strategies and methods to support such migrations, the use of relational algebra and set theory in the context of databases in building a new data migration algorithm. The **goal** of the work is to consider the features and procedure of data migration, explore methods to support data migration between relational and documentary data models, build a mathematical model and algorithm for data migration. The following **methods** were used: analysis and comparison of existing approaches to data migration, choice of strategy for further use in compiling the migration algorithm, mathematical modeling of the algorithm of heterogeneous model-inhomogeneous data migration, formalization of the data migration algorithm. The following **tasks** were solved in the article: consideration of the concept and types of data migration, justification for choosing a document-oriented data model as a target for data migration, analysis of existing literature sources on methods and strategies of heterogeneous model data migration from relational to document-oriented data model, highlighting advantages and disadvantages existing methods, choosing an approach to the formation of the data migration algorithm, compiling and describing a mathematical model of data migration using relational algebra and set theory, presentation of the data migration algorithm, which is based on the focus on data queries. The following **results** were obtained: the possibilities of relational algebra and set theory in the context of data models and queries are used, as well as in model redesign, the strategy of migration of data models is chosen, which provides relational and document-oriented data models, the algorithm of application of this method is described. **Conclusions:** because of the work, the main methods of migration support for different data storage models are analyzed, with the help of relational algebra, set theory a mathematical model is built, and an algorithm for transforming a relational data model into a document-oriented data model is taken into account. The obtained algorithm is suitable for use in real examples, and is the subject of further research and possible improvements, analysis of efficiency in comparison with other methods.

Keywords: database; heterogeneous migration; data model; set theory.

Introduction

Today, the role of information technology is increasing in the world; there is a growing number of software applications covering a variety of areas of people's lives. Most software applications involve the storage of data in one form or another. As the role of software systems grows, the scope of their use expands, the amount of data storage required increases, and the structure of data becomes more complex. Data storage methods are also evolving: new approaches to data storage are developed, new types of databases are created, existing database management systems are improved, hybrid databases that contain properties and functions of several other databases appear, etc.

Eventually the software system may face the problem of failure to function fully due to the limitation of the database used:

- with an excessive load on the database reaching its limits with a large number of system users;
- with increasing complexity of business logic and, as a consequence, difficulty of using the data model of the current database for the needs of this business logic;
- with the transition of a software application to a new technology stack and the technical or logical complexity of using the current database with the new technology stack;
- the impossibility of development and competitiveness of a software system in today's market while using an outdated database in that system (according to Moore's law [1], approximately every two years there is a significant increase in the speed and capabilities of technology, which means that to maintain competitiveness it is always important to be one step ahead of progress);
- when there are risks of full-fledged security and data integrity for outdated DBMSs, etc.

One way to solve the above problem is to migrate to another DBMS (newer, with advantages in features required for a particular software system), with the existing data being migrated to the new database without loss or damage and ready to fully function in the new database this process is called migration.

Database migration is a rather complicated and time-consuming process as the source database and the database to be migrated to may be of different types and have completely different data storage models, data types, ways of working with data, specifics of functioning (for example, migration from relational databases to document-based, event-based, graph-based, etc.).

In order for the migration process to be successful, it is necessary to have a clear migration plan, which includes:

- all preparatory actions for migration;
- conditions and activities in the framework of the migration itself;
- actions after data migration is completed (how the software application will migrate to the new database and how the old database will be liquidated).

Statement of the problem

Within the framework of this work it is necessary to investigate questions of support of migration from relational data model to document-oriented data model, this research should include consideration and analysis of existing methods for this kind of migration and development of own method, as a result of which

Сучасний стан наукових досліджень та технологій в промисловості. 2022. № 2 (20)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

introduction the newly created data scheme will as much as possible respond to requests to database. Thus, it is necessary to formalize the developed method, i.e. to execute its mathematical modeling and to present in the form of the full-fledged algorithm that in the further researches can be applied on real databases, to carry out the analysis of efficiency at various conditions of use in comparison with other methods.

Literature review

A large number of literary sources are devoted to the issue of database migration. When writing this paper, we used scientific literature, articles from periodicals, publications, as well as web resources that discuss information related to the concepts, principles and methods of data migration.

The concept and fundamental aspects of the theory of database migration are presented in the work of John Morris [2]. A description and comparison of two types of migration - homogeneous and heterogeneous - is presented in a technical web resource [3]. Preston Zhang in his book [4] and Andreas Meyer in his publication [5] describe the principles and process of database migration from a practical viewpoint. Lim Fung Gi et al. in their paper [6] address the issue of the need to redesign the data schema when migrating NOSQL databases with different data models.

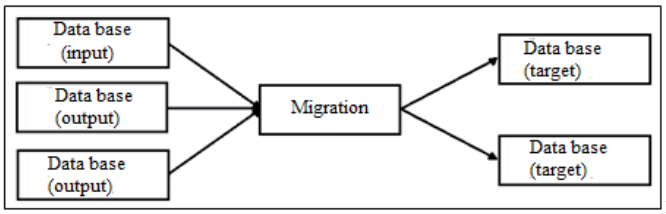
A large number of authors devote their research papers to data migration methods between different data models [7-15]; most of these sources assume not only the physical migration of data between different data stores, but also the data schema re-designing procedure. These methods will be discussed and analyzed in more detail in the following parts.

Analytical review

Database migration is the process of transferring data from one or more source databases to one or more target databases using a specific method [2]. After the migration is completed, the complete, possibly restructured set of source data is contained in the target databases. Customers who used the source databases are migrated to the target databases, and the source databases are not used and can be deleted within a specified period. Figure 1 schematically depicts the database migration process.

In the context of technologies, there are two types of database migration: homogeneous and heterogeneous.

Homogeneous (uniform) migration is a migration between databases in which the source and target databases belong to the same database technology [3], such as migrations from a MySQL database to a MySQL database, or from an Oracle database to an Oracle database. Homogeneous migrations also include migrations between databases systems hosted on its own server, such as PostgreSQL, to its managed version, such as Cloud SQL (a variant of PostgreSQL).



n

Fig. 1. Scheme of the database migration process

For homogeneous migration, the source and target database schemas are identical in most cases. If the schemas are different, the source database data must be converted during migration.

Heterogeneous (uneven) database migration is a migration in which the source and target databases belong to different database technologies [3], for example, migration from MS SQL database to MongoDB. Heterogeneous database migration can be between identical data models (e.g., relational to relational) or between different data models (e.g., relational to key-value). Migration between different database technologies does not necessarily involve different data models. In particular, Oracle, MySQL, PostgreSQL, and Spanner support a relational data model. However, multi-model databases such as Oracle, MySQL, or PostgreSQL support multiple data models. For example, if a multi-model database supports storing data as JSON documents, the data can be ported to MongoDB without the need for a practical conversion because the data model is the same in the source and target databases.

Though the difference between homogeneous and heterogeneous migration is based on database technologies, alternative categorization is based on the database models involved.

For example, migration from an Oracle database to Spanner is model-homogeneous because both databases use a relational data model, that is, only the technology used for the database changes. Figure 2 shows a diagram of heterogeneous model-homogeneous migration.

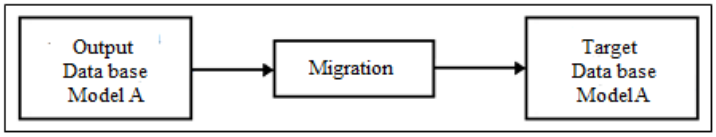


Fig. 2. Heterogeneous model-homogeneous data base migration

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)

Migration is model-heterogeneous when different data storage models are used in source and target databases, if, for example, data stored as JSON objects in Oracle are migrated to a relational model in Spanner. Fig. 3 shows a diagram of heterogeneous model-heterogeneous migration.

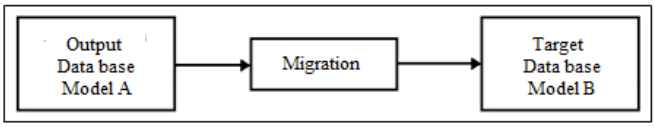


Fig. 3. Heterogeneous model-heterogeneous data base migration

The classification by data model categories more accurately reflects the level of complexity of data migration in comparison with the distribution of database systems.

The most difficult case of migration is the one in which the source and target databases are based on different technologies and at the same time have different data models, it is this case will be further considered and analyzed in the framework of this work.

Upon completion of data movement from the source base to the target base, switch client access to the target base and dispose of the source base.

The process of client switching from source to target databases has several parts [4]:

- to continue the migration clients must temporarily close their connections to the source databases and connect to the new databases;

- having closed client connections to the source databases, the process of transferring the data remaining in the source databases to the target databases. This process is called "draining" and is performed to ensure that all data is migrated to the new databases;

- after the data is migrated, the target databases and client connections should be checked for functionality.

The complete absence of downtime for clients during data migration is not possible, with downtime, there may be cases of inability to process requests, and this jeopardizes the operation of the application, so one of the goals in data migration is to minimize downtime. There are strategies by which downtime can be reduced [5]:

- running test clients in read mode with target databases ahead of time before the migration begins;

- analyzing and adjusting the amount of data to be migrated as migration approaches, partially migrating data in certain portions, the total migration time will increase, but the downtime will decrease;

- connecting new clients to the target databases while the old clients are working with the original databases, as a consequence, reducing the complete migration time to the new databases.

There are several options for the cardinality of database migrations:

- direct mapping (1:1): data from one source database is moved to one target database;

- consolidation (n:1): data from several source databases are moved to a smaller number of target databases, this approach can lead to a simplified database management procedure

- distribution (1:n): data from one source database are moved to a certain (>1) number of target databases. This cardinality can be used, for example, when moving an initial centralized database with regional data to several target regional databases;

- redistribution (n:m): data from a certain number of source databases is moved to a certain number of target databases. Such cardinality is useful in the situation of uneven number of data in the source databases (and, accordingly, uneven load), due to redistribution data are distributed evenly between the target databases.

During data migration, it is possible not only to carry out the actual migration of data, but also to redesign the database [6], more often architectural changes are introduced if the source and target databases have different models, because each type of model includes its own characteristics and principles in which the model will work more effectively, for example, when migrating data from the relational model to non-relational, instead of creating dependent tables, you can move them into nested tables (collections). This will reduce the number of queries to the database and reduce the processing time of related entities.

For this work, it was necessary to choose an example of what kind of NoSQL data model will be further consideration of the material. Considering the information about today's most popular and widely used NoSQL databases [19], it was found that the document-oriented database model MongoDB falls under such criteria. In the overall ranking of databases according to the resource [20], it is in the top 10, being in the fifth position, and ranks first in the NoSQL ranking, since all the previous ones are relational databases. In terms of database popularity, document databases are in second place after relational databases, the percentage of relational databases is 71.9% and document databases is 9.9% [20]. The structure and capabilities of this database have also been analyzed: in particular, there are effective database sharing technologies, formalized models of big data management [21].

A large amount of literature is devoted to comparing MongoDB with relational databases and detailing the advantages of the former [22]. Now MongoDB is used in many subject areas (3D visualization, predicting building thermal capacity usage, groundwater flow control and pollution transport, in health monitoring systems, IoT applications, etc.).

Hence, after the above analysis, it was decided to choose a document-oriented model (using MongoDB as an example) as a NoSQL database for use in the study.

Analysis of existing methods and algorithms

The issue of migration between databases represented by different models is a hot topic at the

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2022. № 2 (20) ISSN 2524-2296 (online)

moment; it has not been studied sufficiently for full formalization, so research related to this issue deserves special attention. There is no unambiguous recommendation for migration in one way or another and the developer himself must make decisions on the method of migration, depending on the characteristics of data and purposes for which migration is carried out.

Let's consider existing approaches and strategies for migrating from a relational database to MongoDB.

One of the more common ways to migrate data from a relational model to MongoDB is based on the fact that most relational databases support exporting tables to CSV format files. Even if there is no such built-in support, it is possible to export using auxiliary software and get the files in the right format. You can then import the files into MongoDB using the built-in command [7]. The disadvantage of this method is that, first, the existing relationships between tables are not taken into account, because essentially only data lists are obtained, and second, each table in the relational database will correspond to a collection in MongoDB, no logical. Architectural rebuilding of tables in document-oriented style, it will affect query execution time, because, as we know, MongoDB works harder with queries that access many document collections. Therefore, this method is reasonable to use only for databases of simple structure with few tables and links.

Another approach to migrating data from relational to MongoDB is migration, which consists of the following sequence of actions [8, 9, 10]:

- extracting data from the original database;
- working with the data, bringing them into the right form (working on data types, etc.)
- transfer of the processed data to the target database.

The disadvantage of this approach is that when working with data, insufficient attention is paid to database schemas and links between objects, because the emphasis is put on the data as such, rather than on their structuring, that is, as in the previous method, when using this strategy there is no restructuring of the data schema.

Another well-known way to migrate data from a relational model to MongoDB is to migrate data based on data structure and data queries [11]. Such migration is performed in three steps:

- describing the relational database structure, describing the data query requirements (according to the business logic);
- modeling the data in the query-oriented context of the NoSql database;
- modeling the database schema in the query-oriented context of the NoSql database.

The disadvantage of this method, as in previous cases, is that it does not take into account the dependencies between database objects, since the new database structure takes into account only metadata about objects and queries.

The next existing approach to this kind of migration is data migration by rules (six rules) describing three types of migration: Column-Based, Document-Based and Graph-Based [12]. The rules describe the cardinality of links between tables and special operations performed on one of the tables (aggregation operations, etc.). The disadvantage of this approach is that the structure of data queries is not taken into account during migration. To solve the problem of query duration that accesses multiple documents, this approach suggests combining all tables into a single NoSQL collection, but this action will inevitably lead to memory problems, because the size of such a collection will be too large. Applying such a method can be justified only if the database was voluminous.

Another method assumes that relationships in a document-oriented model can be represented in the form of embedded documents and relationships between these documents [13, 14]. For example, if there is a functional relationship between two attributes, both attributes will be transformed into a single data element in MongoDB. The same principle is applied for partial and transitive dependencies. The disadvantage is that embedded documents can only be used for a limited amount of data, and there is no clear identification of the form of this embedding.

There is another method, it involves using the theory of database schema normalization and using it in schema design for MongoDB [15], but this approach does not take into account the relationships of "many", primary and foreign keys.

Consequently, after considering the main methods of data migration between relational and json-like data models, we can say that the methods under consideration have their advantages and disadvantages and should be used depending on the specific situation, possibly in combination with each other.

Choice of methods

While studying and analyzing the methods of converting relational data model to MongoDB data model, three main general migration strategies were identified for further consideration and research:

- migration creates a corresponding collection in MongoDB for each relational database table [16];
- during migration, all the relational database tables are merged into a single MongoDB data collection;
- during migration, the database schema is redesigned so that it best meets the database queries (as a consequence, it would simplify the execution of those queries).

The first two strategies are clear and unambiguous and do not require the use of auxiliary methods for implementation.

The third strategy is more complicated and involves the use of auxiliary methods by which the collection schemas in MongoDB would be formed in response to queries.

One of the approaches in transforming data models is the use of relational algebra and set theory [17]. This adapts methods of relational algebra and set theory for convenient use in the context of database models and schemas. Using this approach, it is possible, using a

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)

formal language, to develop specific model transformation steps and compose a general algorithm to support heterogeneous model-inhomogeneous migration for selected data models.

The use of relational algebra and set theory is appropriate to implement the strategy of re-designing database schema according to queries, because the hierarchy of model structure and queries is quite convenient to represent by sets (compositions), analyze them, perform actions with these sets and present the results of these actions [18]. Therefore, this approach will be used as a tool for further development of migration maintenance methods.

Mathematical modeling of migration

Let us represent the incoming relational database schema using set theory. Let T be the set of all tables in the relational schema, the set T consists of elements T_i, where r is the number of table in the schema, in the form of a formula for representing the set of tables looks as follows:

$$T = \{T_i, i = 1,...n\}, \quad (1)$$

where T_i – i-th table in the table set T;

i – table number in a set of tables;

n – the total number of tables in the scheme. can be combined with one another.

Schematically, a set of tables is shown in fig. 4.

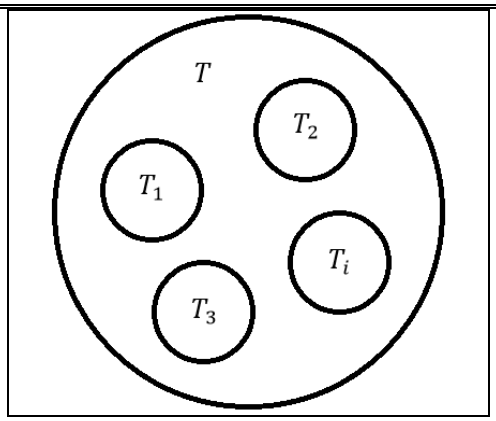


Fig. 4. Representing a relational database schema as a set.

Each table consists of the fields, that is, the table is a set of fields:

$$T_i = \{F_{ij}, i = 1,...n; j = 1,...i_k\}, \quad (2)$$

where F_{ij} – j-th field at the i-th table;

i – table number in a set of tables;

n – total number of tables in the scheme;

j – field number at the i-th table;

i_k – total number of fields in the i-th table. can be combined with one another.

The representation of the relational database schema in the form of sets is shown in fig. 5.

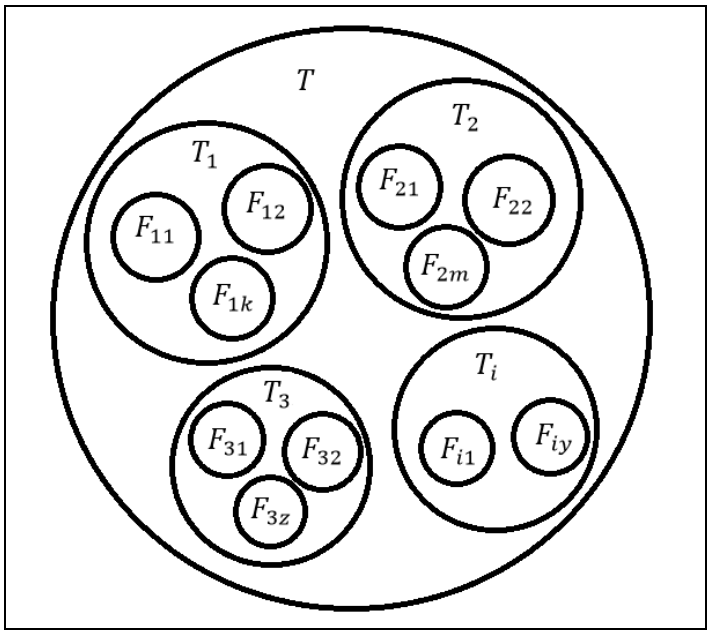


Fig. 5. Presentation of the relational data base scheme in the form of sets

Let Q be the set of all database queries provided by the business logic application:

$$Q = \{Q_l, l = 1,...m\}, \quad (3)$$

where Q_l is a l-th query in a set of queries Q;

l is a number of the query in a set of queries;

m is a total number of queries in a set of queries.

In its turn, each of the queries refers to a certain set of all database fields, i.e. each query can be represented as the following set of fields:

$$Q_l = \{F_{ij}, i \le n; j \le i_k; l = 1,...x\}, \quad (4)$$

where F_{ij} is a j-th field at the i-th table;

i is a table number in a set of tables;

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2022. № 2 (20) ISSN 2524-2296 (online)

n is a total number of tables in the scheme;
j is a field number at the *i*-th table;
i_k is a total field number at the *i*-th table;
x is a total number of queries.

The presentation of queries to the database in the form of sets is shown in fig. 6.

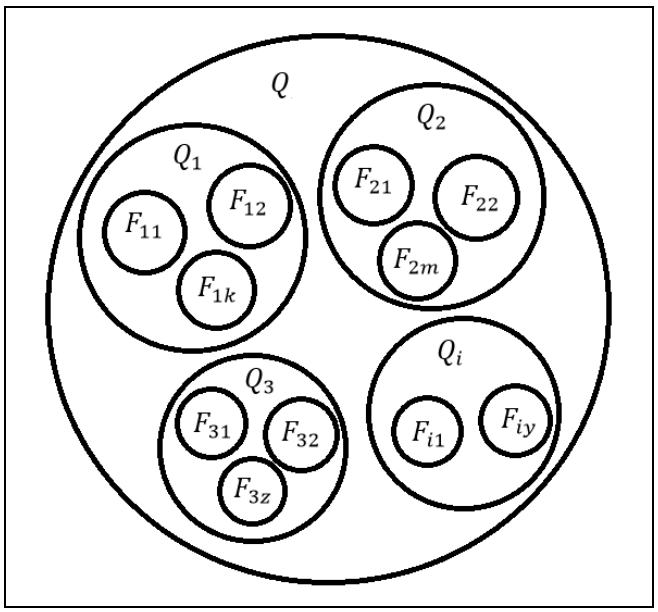


Fig. 6. Presentation of queries to the database

For each field, it is necessary to find a set of queries in which the field is used:

$$F_{ij}^{q} = \{Q_l, l = 1,...z; z \le x \mid F_{ij} \in Q_l\}, \quad (5)$$

where F_{ij}^{q} is the set of queries in which the field F_{ij} is used;

Q_l is a *l*-th query in which the field F_{ij} is used;

z is a number of queries in which the field F_{ij} is used;

x is a total number of queries.

At the beginning of model conversion, it is necessary to remove the fields that are not used in any query, and put these fields in a separate collection. These fields correspond to the following conditions:

$$\left|F_{ij}^{q}\right| = 0; \quad (6)$$

All such fields can be included into one (or several) collection C_1 $(C_1 \ldots C_k)$. In the set representation this collection looks like this:

$$C_1 = \{F_{ij}, i \le n; j \le i_k \mid \left|F_{ij}^{q}\right| = 0\}, \quad (7)$$

where F_{ij} is a *j*-th field at the *i*-th table;

i is a table number in a set of tables;
n – total number of tables in the scheme;
j – number of the field at the *i*-th table;
i_k – total number of the fields at the *i*-th table.

At this stage, several collections are created if the fields falling into the above category are not connected in any way and cannot be combined into one collection from the point of view of the logical representation of the data schema.

Next, it is necessary to select those fields that take part in only one query:

$$\left|F_{ij}^{q}\right| = 1; \quad (8)$$

Fields satisfying the above condition are part of the new collection Cz:

$$C_z = \{F_{ij}, i \le n; j \le i_k \mid F_{ij} \in Q_l \,\&\, \left|F_{ij}^{q}\right| = 1\}, z > 1, \quad (9)$$

where F_{ij} – *j*-th field at the *i*-th table;

n is a total number of tables in the scheme;
i_k – total number of fields at the *i*-th table;
Q_l – *l*-th query;
F_{ij}^{q} – the set of queries in which the field F_{ij} is used;
z – collection number.

The same as for the previous set, more than one collection can be created if the fields are not linked and logically cannot be combined into one collection.

At this stage, remove by consideration those fields that have already been used in previous collections.

Let's make a set of queries *H*, which we will work with next, by extracting from the general set of all queries those queries, all fields of which already belong to the found collections.

Let us make even sections of the sets $Q_l \in H$, these sections have the following form:

$$Q_k^{'} = Q_i \cap Q_j; j \ne i; i, j = 1,...\left|H\right|, \quad (10)$$

where $Q_k^{'}$ is the *k*-th pairwise section;

H is a set of queries that are considered;

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)

Q_i, Q_j – i-th and j-th query sets.

The resulting non-empty sets form a new set M. Provided that $H - M \neq Ø$, the new collection will consist of the fields included in this difference:

$$C_g = \{F_{ij}, i \leq n; j \leq i_k \mid F_{ij} \in (H - M)\}, \quad (11)$$

where C_g is a new collection;

H – set of queries under consideration;

M – set of non-empty intersections;

F_{ij} – field from difference $H - M$.

From the set of fields taken for consideration we remove those that are included in the above set. After receiving the collection we will re-recognize the set H to continue the algorithm:

$$H = M, \quad (12)$$

where H – set of queries to be considered;

M – set of non-empty intersections.

Now we find the intersection of the fields of queries, but already 3 elements:

$$Q_k^{"} = Q_i \cap Q_j \cap Q_m; j \neq i; i, j, m = 1, ... |H|, \quad (13)$$

where $Q_k^{"}$ – k-th intersection of three elements;

H – set of queries under consideration;

Q_i, Q_j, Q_m – i-th, j-th and m-th sets of queries.

Similarly, as with two elements, the resulting nonvoid sets form a new set M and if $H - M \neq Ø$, a new collection (11) is formed. Thus, the algorithm is repeated until only one intersection is left at a certain step, that is, the set H will not consist of one element, when this condition is reached, it is necessary to form a new set consisting of fields that entered the last intersection and are not included in all previous collections.

Thus, as a result of the method, a set of collections of documents with their field sets will be formed.

In further research, it is planned to compare such an algorithm for migration between data models and other algorithms on real database examples and to investigate its effectiveness

Algorithm development

The migration support algorithm is represented by means of the UML activity diagram in fig. 7.

The preparatory step for a relational model database migration algorithm to a query-based document-oriented model is to define a set of queries to the database. It is suggested to use one of the query definition options for the algorithm:

- for commercial projects, in most cases, in addition to the schema database itself, the queries to this database that are operated by the application are also known, in which case known queries are taken for the algorithm;

- if the set of queries is unknown, it is reasonable to make a deeper analysis of the subject area and the database schema and to independently compose the queries that are most likely to cover the business logic of the subject area and the application where the database is or can be used;

- if the previous two options cannot be used in a particular situation, it is proposed to consider the links between the tables as queries.

After defining a set of queries, it is necessary, using set theory, to work with the representation of tables and queries for the further work of the algorithm, namely:

- represent each table as a set of fields;

- represent each query as a set of fields used in it (in any part: sampling, grouping, etc.);

- represent each field as a set of queries in which it takes part.

If at this stage the presence of such fields that do not participate in any of the queries are detected, it is necessary to combine them into one collection (or several collections, if the business logic does not allow combining these fields into one collection) of links between the initial tables, possibly with different degrees of nesting.

The next step in the algorithm is to make links between sets of fields in queries, starting with pairwise links. For each query, i.e. set of its fields, a search for pairwise relations with all other sets of query fields is performed. If the next section is not an empty set, you must add queries from the section to the set of non-empty sections. Thus, all the paired sections are traversed and the queries from the next section are either attached or not attached to the set of non-empty sections (if a certain query is already contained in the resulting set of non-empty sections, it does not need to be added to the second set).

At the end of the formation of the resulting set of non-empty intersections, it is necessary to find the difference of the set of all queries considered at this stage (in pairs intersections - the whole set of queries) and the set of queries that entered the set of non-empty intersections. If the found difference is not an empty set, then the resulting difference is a new collection (it must be further checked that each field is included in this difference once, if there are more than one occurrence, then remove repeating fields). Also, as with the formation of the first collection, it is necessary to pay attention to the connections between the initial tables, and, if necessary, to form different degrees of the nesting of the newly created collection.

For the next step, it is necessary to override the set of queries under consideration by the set of non-empty intersections of queries obtained in the previous step.

Similarly, to the actions with pairwise sections, it is necessary to determine the set of non-empty sections queries (without repeating elements in the resulting set). Having obtained the resulting set of non-empty sections, you should find the difference between the sets of queries considered at this step and the current set of non-empty sections. If the found difference is not an empty set, then the resulting difference makes a new collection (taking into account the links between the initial tables, business logic, formalizing the necessary levels of nesting in the collection, etc.).

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2022. № 2 (20) ISSN 2524-2296 (online)

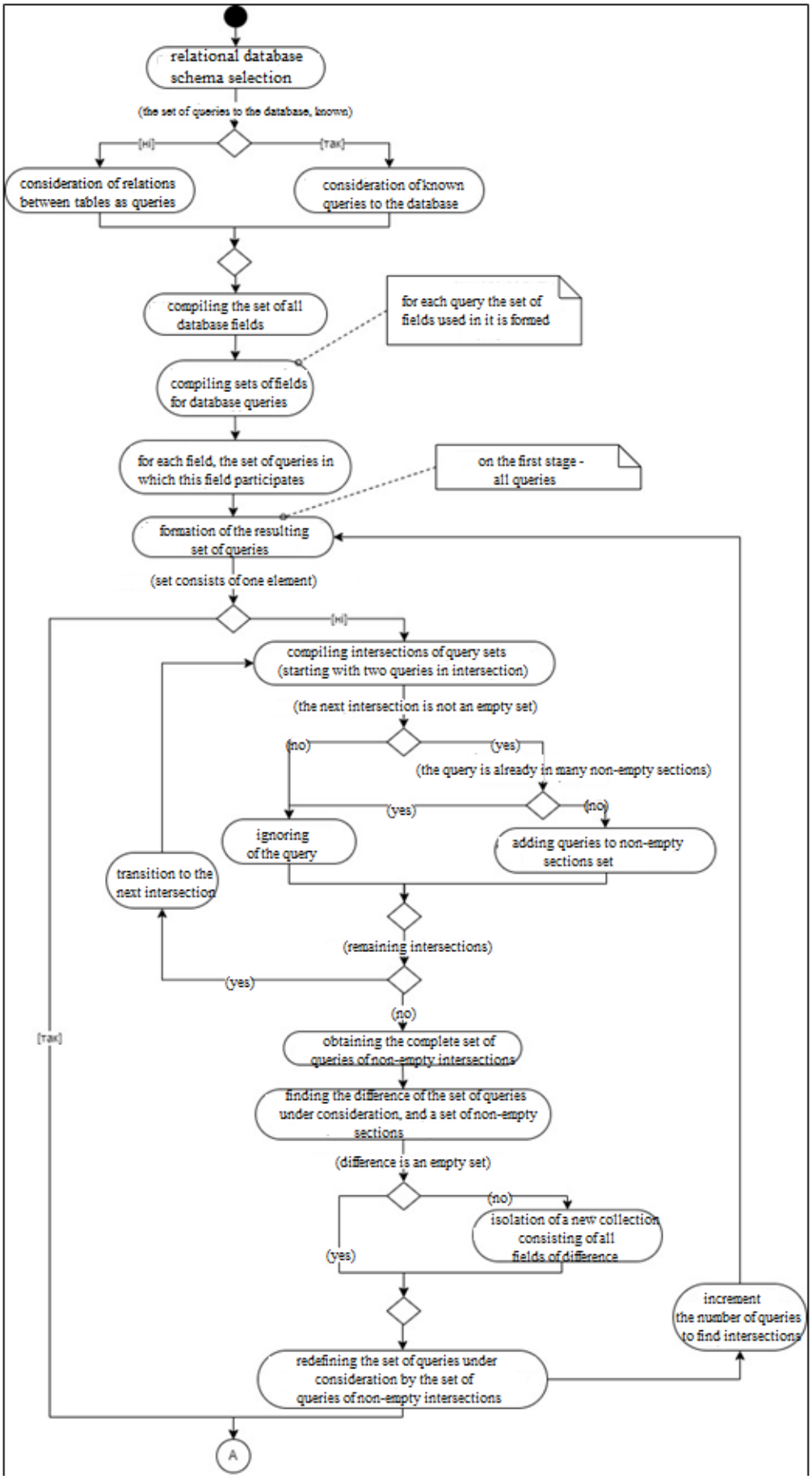
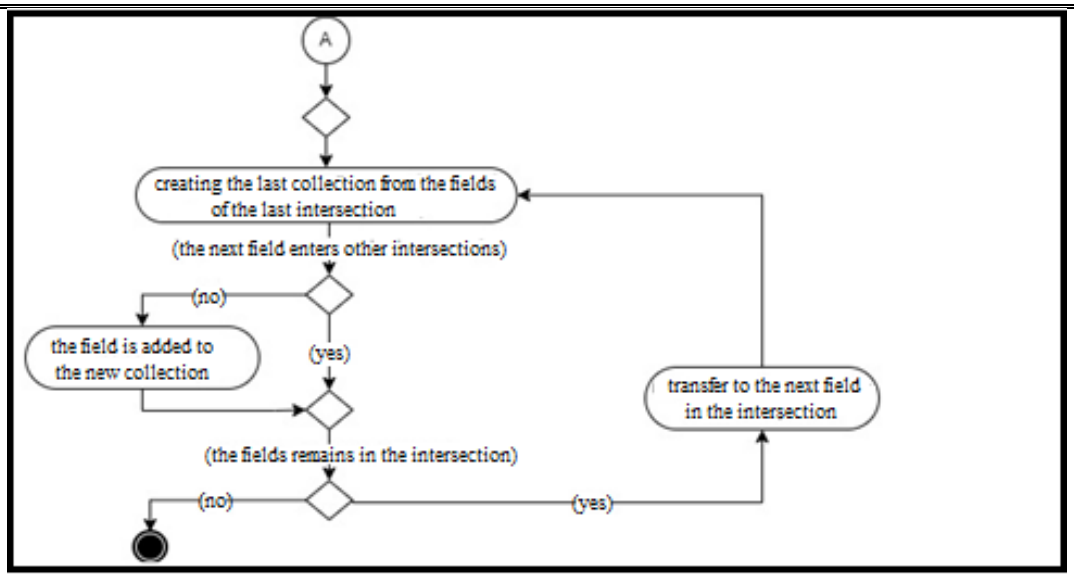


Fig. 7. UML-activity diagram for the algorithm

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)



The end **Fig. 7.**

Similar actions to form new collections with an incremental number of queries to intersect at each step, are repeated until there remains one element in the set of queries considered at a particular step - that is, one query, and since with this query is no longer possible to build a cross section, the transition to the next stage of the algorithm. In this step, each field is checked from the set of fields of this query and, if this field does not yet belong to any of the collections created during the previous work of the algorithm, it is added to the last collection.

Thus, after the algorithm finishes, all fields are distributed among the collections in the newly created document-oriented data model.

Discussion of the results

To automate the migrations, the software was developed in the programming language C#, on the platform .NET Core. This software inputs a MS SQL relational database and a set of queries to that database, converts the database schema according to the algorithm described in this study, shows the result and creates a MongoDB database after confirmation by the user. After the successful creation of collections, the process of data transfer on the corresponding fields takes place. Such a process was run on a test database, after the migration was completed, it was verified that the data was migrated in full and without corruption. The results of query execution in the target database were also tested and matched the results obtained on the source database. Such results prove that the algorithm obtained in the study is workable and can be used for heterogeneous model-heterogeneous data migration between relational and document-oriented storage models.

Conclusions

As a result of the study it was:

- reviewed the theoretical aspects of data migration, related terms and processes;
- analyzed the popularity and relevance of using non-relational databases, on the basis of this analysis non-relational document-oriented storage model (using MongoDB as an example) was selected for consideration as a target database when migrating from the relational model;
- an analytical review of existing methods and strategies for migration between relational models and document-oriented data storage models was performed, features of such methods, limitations of such methods, cases of expediency of use and disadvantages of such methods were given;
- the possibility of using relational algebra and set theory in the context of data and query models, as well as in redesigning models was considered;
- selected a data model migration strategy that involves redesigning the database schema in accordance with database queries;
- a mathematical model for an algorithm for heterogeneous model-inhomogeneous data migration between relational and document-oriented data storage models using set theory has been compiled
- formalized and described an algorithm for data migration according to the principles of the aforementioned strategy (data query-oriented strategy).

The obtained algorithm is suitable for use on real examples, and is also an object for further research and possible improvements, analysis of efficiency in comparison with other methods.

References

1. "International Roadmap for Devices and Systems. More Moore White Paper", available at: https://irds.ieee.org/images/files/pdf/2016_MM.pdf (last accessed: 25.03.2022).
2. Morris, J. (2012), *Practical Data Migration*, BCS, The Chartered Institute for IT, London, 266 p.
3. "Homogeneous vs Heterogeneous migration", available at: https://rtfm.co.ua/aws-database-migration-service-obzor-i-primer-migracii-self-hosted-mariadb-v-aws-aurora-rds/#Homogeneous_vs_Heterogeneous_migration (last accessed: 30.03.2022).

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2022. № 2 (20) *ISSN 2524-2296 (online)*

4. Preston, Z. (2021), *Practical Guide to Large Database Migration*, CRC Press, USA, 198 p.
5. Andreas, M. (2015), "Providing Database Migration Tools. A Practitioner's Approach", *21st International Conference on Very Large Data Bases (VLDB)*, P. 635 – 641.
6. Ji, L. F., Azmi, N. F. M. (2020), "The development of a new data migration model for NOSQL databases with different schemas in environment management system", *Journal of Environmental Treatment Techniques*, No. 8 (2), P. 787–793.
7. Ceresnak, R., Dudas, A., Matiasko, K. (2021), "Mapping rules for schema transformation : SQL to NoSQL and back", *International Conference on Information and Digital Technologies*, P. 52–58. DOI: https://doi.org/10.1109/IDT52577.2021.9497629
8. Hanine, M., Bendarag, A., Boutkhoum, O. (2015), "Data Migration Methodology from Relational to NoSQL Databases", *International Journal of Computer, Electrical, Automation, Control and Information, Engineering*, No. 9 (12), P. 2566–2570.
9. Alalfi, M. H. (2018), "Automated Algorithm for Data Migration from Relational to NoSQL Databases", *Al-Nahrain Journal for Engineering Sciences (NJES)*, No. 21 (1), P. 60–65. DOI: https://doi.org/10.29194/NJES2101
10. Fouad, T., Mohamed, B. (2019), "Model transformation from object relational database to NoSQL document database", *NISS19*, No. 49, P. 1–5. DOI: https://doi.org/10.1145/3320326.3320381
11. Li, X., Ma, Z., Chen, H. (2014), "QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases", *IEEE Workshop on Advanced Research and Technology in Industry Applications*, P. 338–345.
12. Alotaibi, O., Pardede, E. (2019), "Transformation of Schema from Relational Database (RDB) to NoSQL Databases", *Data*, No. 4 (4), P. 148. DOI: https://doi.org/10.3390/data4040148
13. Ain El Hayat, S., Bahaj, M. (2020), "Modeling and transformation from temporal object relational database into mongodb: Rules", *Advances in Science, Technology and Engineering Systems*, No. 5 (4), P. 618–625. DOI: https://doi.org/10.25046/aj050473
14. Mason, R. T. (2015), "NoSQL databases and data modeling techniques for a document-oriented NoSQL database", *Informing Science & IT Education Conference (InSITE)*, P. 259–268. DOI: https://doi.org/10.28945/2245
15. Alekseev, A. A., Osipova, V. V., Ivanov, M. A. (2016), "Efficient data management tools for the heterogeneous big data warehouse", *Physics of Particles and Nuclei Letters*, No. 13 (5), P. 689–692. DOI: https://doi.org/10.1134/S1547477116050022
16. Gu, Y., Wang, X., Shen, S., Wang, J., Kim, J.-U. (2015), "Analysis of data storage mechanism in NoSQL database MongoDB", *2015 IEEE International Conference on Consumer Electronics*, P. 158–159.
17. Dabowsa, N. I., Maatuk, A. M., Elakeili, S. M. (2021), "Converting Relational Database to Document-Oriented NoSQL Cloud Database", *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, P. 381–386. DOI: https://doi.org/10.1109/MI-STA52233.2021.9464488
18. Date, C. J. (2012), *SQL and Relational Theory: How to Write Accurate SQL Code*, O'Reilly Media, London, 448 p.
19. Kuzochkina, A., Shirokopetleva, M., Dudar, Z. (2018), "Analyzing and Comparison of NoSQL DBMS", *International Scientific-Practical Conference on Problems of Infocommunications Science and Technology*, P. 560–564. DOI: https://doi.org/10.1109/INFOCOMMST.2018.8632133 "DB-Engines Ranking", available at: https://db-engines.com/en/ranking (last accessed: 10.04.2022).
20. Chickerur, S., Goudar, A., Kinnerkar, A. (2015), "Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications", *8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, P. 41–47. DOI: https://doi.org/10.1109/ASEA.2015.19
21. Stepovik, A. N, Efanov, N. V. (2019), "Analysis of relational and non-relational databases", *Digitization of the economy: directions, methods, tools*, P. 414–416.

Received 13.04.2022

Відомості про авторів / Сведения об авторах / About the Authors

Перетятко Марія Вікторівна – Харківський національний університет радіоелектроніки, магістрант кафедри програмної інженерії, м. Харків, Україна; e-mail: mariia.peretiatko@nure.ua; ORCID ID: https://orcid.org/0000-0003-3711-3765.

Перетятко Мария Викторовна – Харьковский национальный университет радиоэлектроники, магистрант кафедры программной инженерии, г. Харьков, Украина.

Peretiatko Mariia – Kharkiv National University of Radio Electronics, undergraduate at the Department of Software Engineering, Kharkiv, Ukraine.

Широкопетлєва Марія Сергіївна – Харківський національний університет радіоелектроніки, старший викладач кафедри програмної інженерії, заступник директора центра післядипломної освіти, м. Харків, Україна; e-mail: marija.shirokopetleva@nure.ua; ORCID ID: https://orcid.org/0000-0002-7472-6045.

Широкопетлева Мария Сергеевна – Харьковский национальный университет радиоэлектроники, старший преподаватель кафедры программной инженерии, заместитель директора центра последипломного образования, г. Харьков, Украина.

Shirokopetleva Mariia – Kharkiv National University of Radio Electronics, senior lecturer at the Department of Software Engineering, Deputy Director of the Center for Postgraduate Education, Kharkiv, Ukraine.

Лєсна Наталя Совєтівна – кандидат технічних наук, професор, Харківський національний університет радіоелектроніки, професор кафедри програмної інженерії, м. Харків, Україна; e-mail: natalya.lesna@nure.ua; ORCID ID: https://orcid.org/0000-0001-9816-3251.

Лесная Наталья Советовна – кандидат технических наук, профессор, Харьковский национальный университет радиоэлектроники, профессор кафедры программной инженерии, г. Харьков, Украина; e-mail: natalya.lesna@nure.ua

Lesna Natalya – Candidate of Technical Sciences (PhD), Professor, Kharkiv National University of Radio Electronics, Professor at the Department of Software Engineering, Kharkiv, Ukraine.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2022. No. 2 (20)

ДОСЛІДЖЕННЯ МЕТОДІВ ПІДТРИМКИ МІГРАЦІЙ ДАНИХ МІЖ РЕЛЯЦІЙНИМИ І ДОКУМЕНТНИМИ МОДЕЛЯМИ ЗБЕРІГАННЯ ДАНИХ

Предметом дослідження в статті є гетерогенна модельно-неоднорідна міграція даних між реляційними та документо-орієнтовними моделями зберігання даних, існуючі стратегії та методи підтримки такого роду міграцій, використання реляційної алгебри та теорії множин у контексті баз даних при побудові нового алгоритму міграції даних. **Мета** роботи – розглянути особливості та порядок міграції даних, дослідити методи підтримки міграції даних між реляційними і документними моделями даних, побудувати математичну модель та алгоритм для міграції даних. Використовуються такі **методи:** аналіз та порівняння існуючих підходів до міграції даних, вибір стратегії для подальшого використання при складанні алгоритму міграції, математичне моделювання алгоритму гетерогенної модельно-неоднорідної міграції даних, формалізація алгоритму міграції даних. В статті вирішуються наступні **завдання:** розгляд поняття та різновидів міграції даних, обґрунтування вибору документо-орієнтовної моделі даних в якості цільової для міграції даних, аналіз існуючих літературних джерел, що стосуються методів та стратегій гетерогенної неоднорідно-модельної міграції даних з реляційної до документо-орієнтовної моделі даних, виділення переваг та недоліків існуючих методів, вибір підходу до формування алгоритму міграції даних, складання та опис математичної моделі міграції даних за допомогою реляційної алгебри та теорії множин, представлення алгоритму міграції даних, в основі якого лежить орієнтація на запити до даних. Отримано наступні **результати:** використано можливості реляційної алгебри і теорії множин у контексті моделей даних та запитів, а також при перепроектуванні моделей, обрано стратегію міграції моделей даних, яка передбачає перепроектування схеми бази даних у відповідності до запитів до бази даних, створено математичну модель методу гетерогенної неоднорідно-модельної міграції між реляційною та документо-орієнтовною моделями даних, описано алгоритм застосування цього методу. **Висновки:** в результаті проведеної роботи проаналізовано основні методи підтримки міграції для різних моделей зберігання даних, за допомогою реляційної алгебри та теорії множин побудовано математичну модель та складено алгоритм перетворення реляційної моделі даних до документо-орієнтовної моделі даних з урахуванням запитів до даних. Отриманий алгоритм є придатним для використання на реальних прикладах, а також є об'єктом для подальших досліджень і можливих удосконалень, аналізу ефективності у порівнянні з іншими методами.

Ключові слова: база даних; гетерогенна міграція; модель даних; теорія множин.

ИССЛЕДОВАНИЕ МЕТОДОВ ПОДДЕРЖКИ МИГРАЦИЙ ДАННЫХ МЕЖДУ РЕЛЯЦИОННЫМИ И ДОКУМЕНТНЫМИ МОДЕЛЯМИ ХРАНЕНИЯ ДАННЫХ

Предметом исследования в статье является гетерогенная модельно-неоднородная миграция данных между реляционными и документо-ориентированными моделями хранения данных, существующие стратегии и методы поддержки такого рода миграций, использование реляционной алгебры и теории множеств в контексте баз данных при построении нового алгоритма миграции данных. **Цель** работы – рассмотреть особенности и порядок миграции данных, исследовать методы поддержки миграции между реляционными и документными моделями данных, построить математическую модель и алгоритм для миграции данных. Используются следующие **методы:** анализ и сравнение существующих подходов к миграции данных, выбор стратегии для дальнейшего использования при составлении алгоритма миграции, математическое моделирование алгоритма гетерогенной модельно-неоднородной миграции данных, формализация алгоритма миграции данных. В статье решаются следующие **задачи:** рассмотрение понятия и разновидностей миграции данных, обоснование выбора документо-ориентированной модели данных в качестве целевой для миграции данных, анализ существующих литературных источников, касающихся методов и стратегий гетерогенной неоднородно-модельной миграции данных из реляционной к документо-ориентированной модели данных существующих методов, выбор подхода к формированию алгоритма миграции данных, составление и описание математической модели миграции данных с помощью реляционной алгебры и теории множеств, представление алгоритма миграции данных, в основе которого лежит ориентация на запросы к данным. Получены следующие **результаты:** использованы возможности реляционной алгебры и теории множеств в контексте моделей данных и запросов, а также при перепроектировании моделей, выбрана стратегия миграции моделей данных, предусматривающая перепроектирование схемы базы данных в соответствии с запросами к базе данных, создана математическая модель метода гетерогенной неоднородно-модельной миграции между реляционной и документо-ориентированной моделями данных, описан алгоритм применения этого метода. **Выводы:** в результате проведенной работы проанализированы основные методы поддержки миграции для различных моделей хранения данных, с помощью реляционной алгебры и теории множеств построен математическая модель и составлен алгоритм преобразования реляционной модели данных в документо-ориентированную модель данных с учетом запросов к данным. Полученный алгоритм пригоден для использования на реальных примерах, а также является объектом для дальнейших исследований и возможных усовершенствований, анализа эффективности по сравнению с другими методами.

Ключевые слова: база данных; гетерогенная миграция; модель данных; теория множеств.

Бібліографічні описи / Bibliographic descriptions

Перетятко М. В., Широкопетлєва М. С., Лєсна Н. С. Дослідження методів підтримки міграцій даних між реляційними і документними моделями зберігання даних. *Сучасний стан наукових досліджень та технологій в промисловості*. 2022. № 2 (20). С. 64–74. DOI: https://doi.org/10.30837/ITSSI.2022.20.064

Peretiatko M., Shirokopetleva M., Lesna N. (2022), "Research of methods to support data migration between relational and document data storage models", *Innovative Technologies and Scientific Solutions for Industries*, No. 2 (20), P. 64–74. DOI: https://doi.org/10.30837/ITSSI.2022.20.064