DOI 10.52209/1609-1825_2024_4_394 UDC 004.043;424

Model of JSON To XML Format Conversion Based on XSD Data Structure Extraction

[1]***MUKHITOVA Aigul,** Senior Lecturer, mukhitova.aigul@gmail.com,*
[2]***YERIMBETOVA Aigerim,** PhD, Head of Laboratory, aigerian@mail.ru,*
[3]***BARAKHNIN Vladimir,** Dr. of Tech. Sci., Head of Department, bar@ict.nsc.ru,*
[1]NPJSC «Al-Farabi Kazakh National University», Al-Farabi Avenue, 71, Almaty, Kazakhstan,
[2]Institute of Information and Computational Technologies, Kurmangazy Street, 29, Almaty, Kazakhstan,
[3]Novosibirsk State University, Pirogov Street, 1, Novosibirsk, Russia,
**corresponding author.*

***Abstract.** Currently, the most common data formats for storing and processing data in web systems are JSON and XML. To ensure good compatibility and integration between different systems and applications, many industry standards and protocols prefer to use XML as the main data exchange format. JSON to XML transformations will allow convenient processing and transfer of structured data between different systems and devices. The goal of this work is to develop a model for JSON to XSD and XML output transformations that allows correct processing of data when transformed in both directions without changing the structure of the JSON document. The research method is based on the approach of finding and merging similar subgraphs within a graph of the same type, while the compared subgraphs have different types, the type is not extended by using the merge operation. The problem of differing structures and specifications of data formats is solved by developing a transformation model that takes into account the specificity of the particular data and the needs of the application.*

***Keywords:** web services, structured data, format conversion, heterogeneous, interaction, JSON, XML, XSD.*

Introduction

Most modern Web systems need a user interface – a website, a mobile application. It need to keep the benefits of XML (stringency, robustness) while simplifying development and improving performance [1]. REST APIs are dominant, while integrations with SOAP systems may be required. It may be required in the following cases: a response from the server is needed – it send a JSON request, the server converts the data into XML, processes it, and must send a JSON response. Requires integration with 3rd party systems using JSON on the server side [2-3]. This causes the following problems: loss of mandatory values; wrong data types; incorrect values; complexity and inconvenience of data after transformations; loss of metadata; wrong data order; data can be edited before being converted back.

Why it needs to transform JSON-XML-JSON:

- Javascript/Web 3.0 dominates fast message parsing, programmer convenience;

- Atom, XML and JSON message formats have become common and the XML API and REST API are being extended to support JSON;

- Enterprises want validation, declarative constraint, and data content rigor, but also want the benefits of web 3.0.

In such cases, converting data from JSON to XML can simplify data manipulation.

Analysis of existing solutions for generating types based on data in JSON format

During the analysis, no off-the-shelf solutions were found that could generate XML type schema based on data in JSON format, so a combination of two consecutive transformations was considered:

1. Generating JSON Schema based on JSON-formatted data.

2. Conversion of type schema from JSON schema to XML schema (XSD).

Moreover, there are no big data-compatible benchmarks for such database technologies. There are several libraries for converting JSON to XML, each with its advantages and disadvantages. Some of the more popular libraries for parsing JSON to XSD: Jsonix, X2JS, Xml-js, Quicktype.

In general, the choice of a library for converting JSON to XML depends on the specif-

ic task and data processing requirements. If full support for XML Schemas and the ability to generate JavaScript classes is required, then jsonix may be the most appropriate choice. If simple data processing is required with the ability to support non-standard JSON data, then xml-js may be a better option.

In our study, the choice was made in favor of the Quicktype. Having analyzed the software capabilities of each of the above solutions, the choice was made in favor of the quicktype library, which satisfies all the formed criteria and also creates the most accurate structure of types in JSON schema format. Another advantage of this library is that it is written in JavaScript programming language, which has native support for working with data in JSON format.

Research methods

The parametric algorithm that forms a type scheme in JSON schema format based on data represented in several JSON documents [4, 5] is taken as a basis. This algorithm generates a type graph specifying the type structure of input data, which is subsequently converted into JSON Schema. Алгоритм формирования графа типов разбивается на два этапа:

1. Formation of type graphs based on mapping each of the data elements in JSON to the corresponding type.

2. Merging of type graphs formed at the first stage.

The first stage of the algorithm is implemented using a depth-first search algorithm on a hierarchical data structure. Each element is matched with a corresponding type supported by the system based on its structure. At this stage, a set of strict types is formed, which is matched in accordance with the data specified along the hierarchical path. This approach allows formalizing the input data, as well as forming a type graph that matches the hierarchical structure of the information presented in the JSON document [6].

The next step of the algorithm is to combine the type graphs for the data presented in JSON documents based on the reduction operation over the set of graphs formed based on the input data [7, 8]. The function that is called at each step of iteration over the set of graphs takes two graphs as input:

- a graph formed as a result of the reduction operation at previous iteration steps;

- a graph that is represented in the set at the current iteration step.

Based on the above, *an algorithm for generating XML schema based on data in JSON format* has been created, in which the following steps are sequentially implemented:

1. Formation of type graph on the basis of input data.

2. Combining class types in the type graph.

3. Formation of XSD based on the type graph.

1. Formation of type graph on the basis of input data

The essence of the algorithm is to form a type graph based on input data. The algorithm is divided into two consecutive stages.

The first stage traverses the hierarchical data structure in depth and maps each element from the data in JSON to the corresponding type (figure 1).

This stage also forms groups of types for further comparison if the types are represented within the elements of one array or an object.

The first stage of the algorithm execution is necessary to form a strict set of types that corresponds to the data specified by the hierarchical JSON document path

The second stage combines the types grouped earlier on the basis of the equivalence function:

If the result of calling the equivalence function is negative, a type-union is formed, specifying the mutually exclusive structure of one of the compared types

If the result of the comparison is positive, the process of comparing nested structures based on the equivalence function is recursively called for object or array types. Similar parts refer to a single structure, and different parts form types-units. The positive result of comparison of primitive types is trivial – it refers to a single type (figure 2).

2. Combining class types in the type graph

The process of performing this step consists in searching and merging similar structures represented as type-classes inside the type graph. This procedure does not require unification of array types, because the nested type of an array element unambiguously formalizes its representation and can be used as a union in case of comparability with one of the type-classes represented in the graph.

For the type-union step, a restriction on creating type-unions only on the basis of primitive types was additionally added, otherwise the compared types are not united.

The equivalence function for the compared types was defined. If object types are compared, the similarity measure of property keys is calculated based on the Jaccard coefficient. If the minimum similarity threshold is exceeded, a comparison of matching property structures is performed. If other types of types are compared, a comparison by full structural correspondence is performed.

A set of type-classes represented in a graph is taken as input. Initially, a traversal of

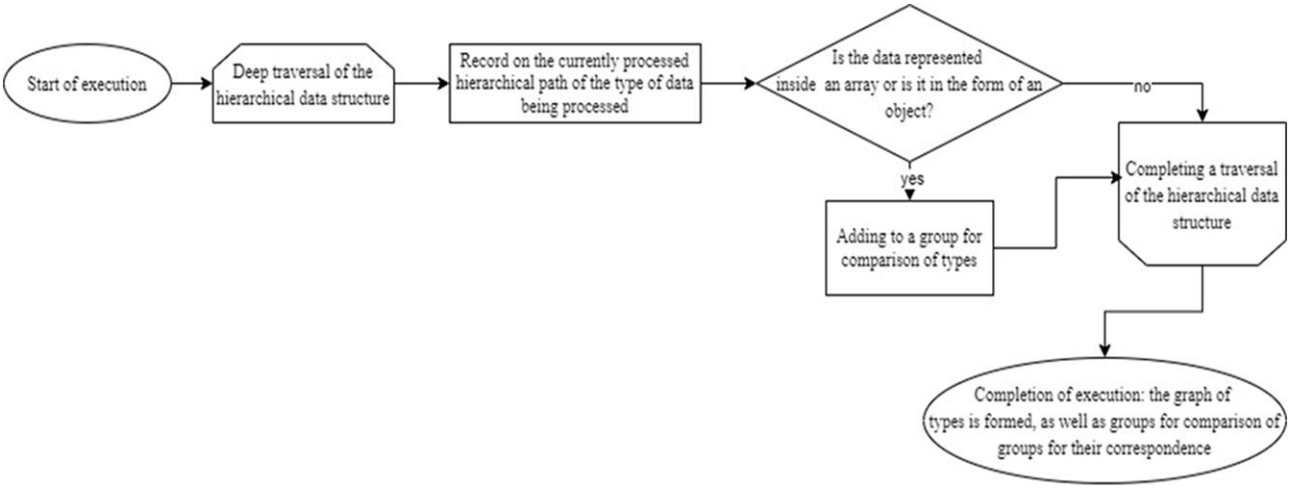

Figure 1 – Mapping model to each data element in JSON of the appropriate type

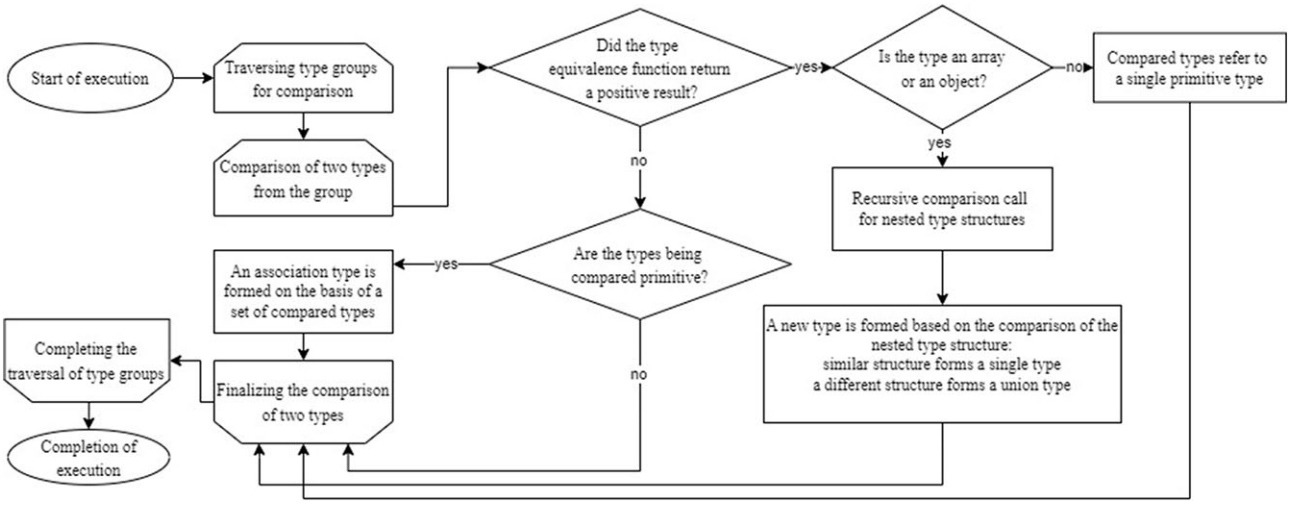

Figure 2 – Model of combines the types grouped earlier on the basis of the equivalence function

all class types is performed, identifying similar structures, which are combined into groups for the subsequent formation of a single structure. If the current type is structurally similar to one of the types already included in the existing group, the function of adding the type to the merge group is called (figure 3). Otherwise, if no similarity with the types from the formed groups is found, a new group is created for the current type, in which similar types are recorded, as well as a set of matching property structures represented as key-value pairs, where the key is the property name, the value is the property type.

When the comparison of all class types is complete, a type-class merging process is performed based on the groups to merge, forming a different type-class for each group.

Generating input data for creating the type graph involves creating the following processing functions:

- Function to handle the key parsing event.
- Function of processing the event of the beginning of an object or array.
- Function for handling the complex object completion event.
- JS primitive value parsing event processing function.
- Function for writing data to the current context.

3. Formation of XSD on the basis of the type graph

At this stage, the type graph is converted into a representation of the type structure in

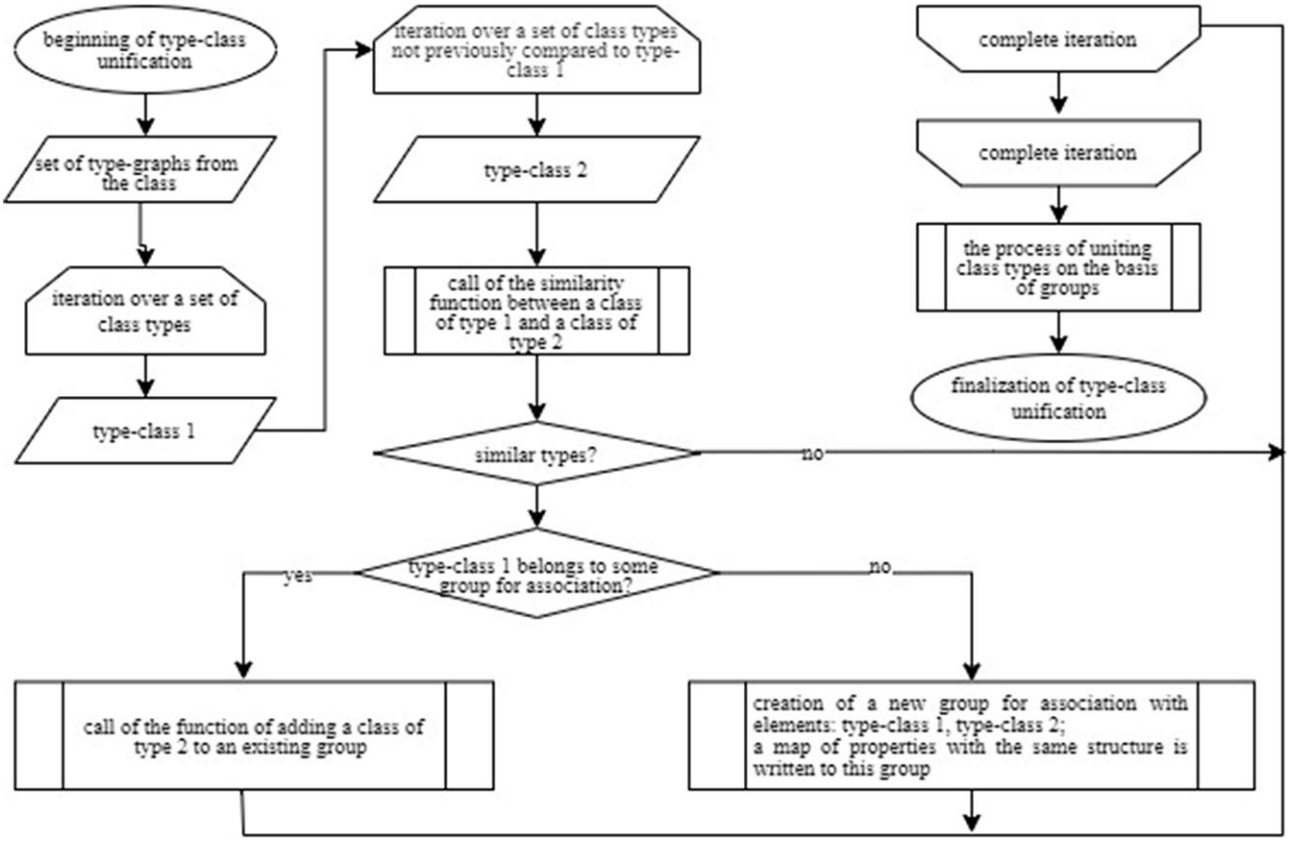

Figure 3 – Model of combining class types in the type graph

the XSD language. The input is a graph represented as a hierarchical model corresponding to a data type in JSON format. The process of performing this step consists in traversing the type structure using a depth-first search algorithm, matching each type from the graph with the corresponding representation in the XSD language. Earlier in the course of the description of the work, the set of types supported by the system was defined, as well as their formalized descriptions in the XSD language. Application of these rules of type mapping allows to form the final XSD scheme, which provides data consistency and the possibility of their validation.

Research results

An algorithm for forming a hierarchical structure of types based on data in JSON format was developed, which allows forming the structure of complex objects regardless of their local location in a JSON-document. On the basis of the developed algorithm, the software that forms XML schema on the basis of data in JSON-document was implemented.

Another problem in translating JSON to XML is namespace management. JSON does not have the concept of namespaces, which is essential for XML. Therefore, when translating JSON to XML, it need to consider how to manage namespaces in order to maintain data compatibility.

As part of testing the correctness of the developed software, different documents in JSON format containing data in the form of a hierarchical structure were fed to the program as input. For each test case, hypotheses were made regarding the expected structure of the XSD schema. After performing the process of XSD schema generation, its validity was checked and the schema was analyzed for the expected result.

The following cases were considered in the testing:

1. Formation of the XSD schema based on a set of primitive types matched according to the input data (figure 4).

2. Schema formation for an array with different array element types (figure 5).

The developed software allows to formalize data in JSON format, providing them with the properties of integrity and consistency. Based on the generated schemas, it is possible to perform model generation in software systems that have their own specific requirements for data representation. In addition, the generated schema sets rules for the formation of new data in XML-documents, which contributes to

Figure 4 – XSD schema based on data in primitive types

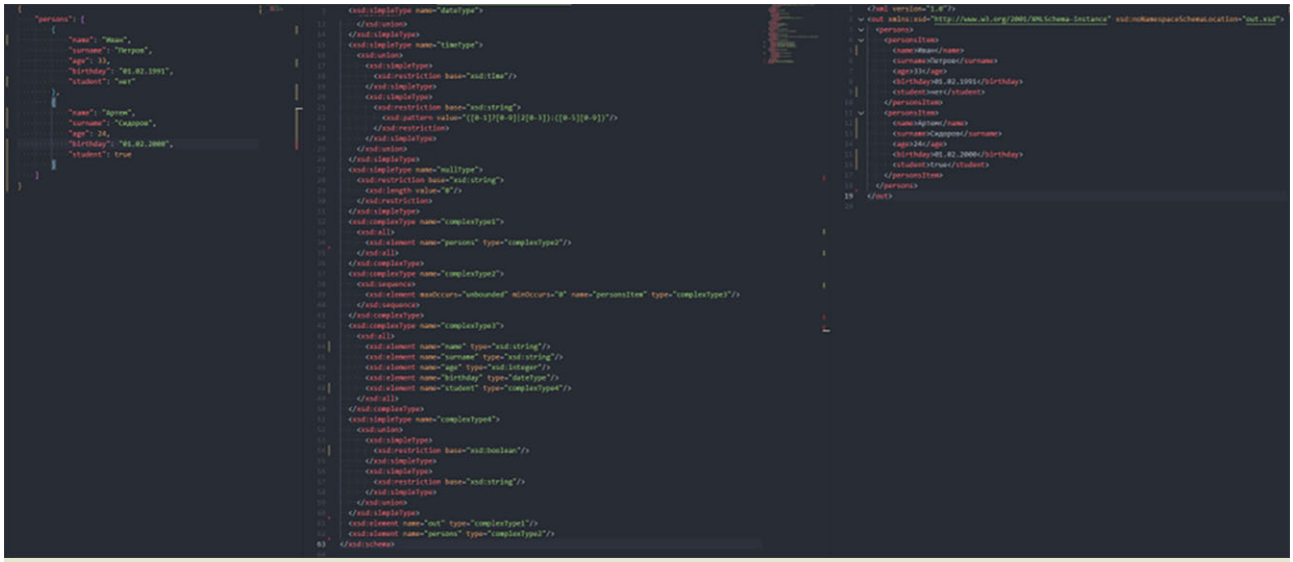

Figure 5 – Checking the merging of type-classes during XSD schema generation

the acceleration of this process.

The advantage of the algorithm is to generate type structures that are not bound to a local location within a hierarchical structure. It means that the type structure can be used to formalize data elements that have different locations within the JSON structure.

The disadvantage is that it violates the rule for XML schema formation of not being able to combine multiple complex types for a single element.

Requirements for a program to form XML schema based on data in JSON format were formulated:

- to be able to analyze the structure of a JSON document, forming compound and primitive types;
- generate type graph on the basis of input data structure;
- take into account the specifics and rules used when forming the structure of types in XSD language.

Testing has demonstrated the correctness of the algorithm realization on generation of XSD scheme on the basis of JSON with the help of the developed software.

Conclusion

The developed model for extracting the XSD data structure from the JSON format is a model for forming a hierarchical structure of complex object types regardless of their local location in the JSON document. Based on the developed model, software was implemented that generates an XML schema based on the data in the JSON document.

During testing of the resulting model, it was found that the selected quicktype library for translating JSON into XSD and XML meets our requirements and provides reliable and efficient data conversion. Quicktype was modified to implement a new approach to merging

types, and support for converting a type graph into an XML schema was added.

The developed model allows for a qualitative increase in the performance of the JSON document processing system via an XML editor [10], due to the correct formation of the XSD structure. A correctly formed schema allows for converting a document into JSON-XML-JSON formats without changing the structure and losing data.

The target audience for this program is web application developers, who can use this solution to automatically generate validation rules and ensure consistency of input data.

This research was funded by a grant for Financing This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23484329).

REFERENCES

1. Song E., Haw S.-C. XML-REG: Transforming XML Into Relational Using Hybrid-Based Mapping Approach, IEEE Access, 8, 2020, pp. 177623-177639.
2. Haroune-Belkacem N., Semchedine F., Al-Shammari A., Aissani D. SMCA: An efficient SOAP messages compression and aggregation technique for improving Web services performance, J. Parallel Distrib. Comput., vol. 133, Nov. 2019, pp. 149-158.
3. Leal R., Queiros M. Generation of Document Type Exercises for Automated Assessment, Licensed under Creative Commons License CC-BY 4.0 11th Symposium on Languages, Applications and Technologies (SLATE 2022). Article No. 4, 2022, pp. 41-46.
4. Baazizi M.-A., Colazzo D., Ghelli G., Sartiani C. Parametric schema inference for massive JSON datasets. The VLDB Journal, 2019. doi: 10.1007/s00778-018-0532-7
5. Guo S., Xia H., Xiang G. Research on the translation from XSD to JSON schema. 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017. doi: 10.1109/iccsn.2017.823033
6. Lee J., Anjos E., Satti S.R. SJSON: A succinct representation for JSON documents, Information Systems 97, 101686, 2021, pp. 27-38.
7. Vodnansky D. Three Metric-Based Method for Data Compatibility Calculation, Acta Informatica Pragensia, vol. 10 (1), 2021, pp. 38-60.
8. Breje A.R., Gyorodi R., Gyorodi C., Zmaranda D., Pecherle G. Comparative study of data sending methods for XML and JSON models, Int. J. Adv. Comput. Sci. Appl, vol. 9, no. 12, 2018, pp. 198-204.
9. Pezoa F., Reutter J.L., Suarez F. Foundations of JSON Schema. International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 263-273.
10. Mukhitova A., Yerimbetova A., Cherikbayeva L. Development of an adaptive graphic web interface model for editing XML data, Eastern-European Journal of Enterprise Technologies, 2 (122), 2023, pp. 26-35.

XSD деректер құрылымын шығару негізіндегі JSON-нен XML-ге түрлендіру моделі

[1]*МУХИТОВА Айгуль Ариповна, *аға оқытушы, mukhitova.aigul@gmail.com,*
[2]ЕРІМБЕТОВА Әйгерім Сембекқызы, *PhD, зертхана меңгерушісі, aigerian@mail.ru,*
[3]БАРАХНИН Владимир Борисович, *т.ғ.д., кафедра меңгерушісі, bar@ict.nsc.ru,*
[1]«Әл-Фараби атындағы Қазақ ұлттық университеті» КеАҚ, әл-Фараби даңғылы, 71, Алматы, Қазақстан,
[2]Ақпараттық және есептеуіш технологиялар институты, Құрманғазы көшесі, 29, Алматы, Қазақстан,
[3]Новосибирск мемлекеттік университеті, Пирогов көшесі, 1, Новосибирск, Ресей,
**автор-корреспондент.*

Аңдатпа. *Қазіргі уақытта веб-жүйелерде сақтауға және өңдеуге арналған ең көп тараған деректер форматтары JSON және XML болып табылады. Әртүрлі жүйелер мен қолданбалар арасында жақсы үйлесімділік пен интеграцияны қамтамасыз ету үшін көптеген салалық стандарттар мен хаттамалар XML-ді бастапқы деректер алмасу форматы ретінде пайдалануды жөн көреді. JSON-ды XML түрлендірулері құрылымдық деректерді әртүрлі жүйелер мен құрылғылар арасында ыңғайлы өңдеуге және тасымалдауға мүмкіндік береді. Бұл жұмыстың мақсаты – JSON құжатының құрылымын өзгертпей екі бағытта түрлендіру кезінде деректерді дұрыс өңдеуге мүмкіндік беретін JSON-ды шығыс XSD және XML-ге түрлендіру моделін жасау. Зерттеу әдісі әртүрлі типтегі салыстырылған тармақшалар біріктіру операциясын қолдану арқылы типті кеңейтпесіне қарамастан, бір типті графиктің ішінде ұқсас тармақшаларды іздеу және біріктіру тәсіліне негізделген. Мәліметтер форматтарының әртүрлі құрылымдары мен спецификациялары мәселесі нақты деректердің ерекшеліктерін және қолданбаның қажеттіліктерін ескеретін түрлендіру моделін жасау арқылы шешіледі.*

Кілт сөздер: *веб-қызметтері, құрылымдық деректер, форматты түрлендіру, гетерогенділік, өзара әрекеттесу, JSON, XML, XSD.*

Модель преобразования формата JSON в XML на основе извлечения структуры данных XSD

[1]*МУХИТОВА Айгуль Ариповна, *старший преподаватель, mukhitova.aigul@gmail.com,*
[2]ЕРИМБЕТОВА Айгерим Сембековна, *PhD, зав. лабораторией, aigerian@mail.ru,*
[3]БАРАХНИН Владимир Борисович, *д.т.н., зав. кафедрой, bar@ict.nsc.ru,*
[1]НАО «Казахский национальный университет имени аль-Фараби», пр. аль-Фараби, 71, Алматы, Казахстан,
[2]Институт информационных и вычислительных технологий, ул. Курмангазы, 29, Алматы, Казахстан,
[3]Новосибирский государственный университет, ул. Пирогова, 1, Новосибирск, Россия,
**автор-корреспондент.*

Аннотация. *В настоящее время наиболее распространенными форматами данных для хранения и обработки в веб-системах являются JSON и XML. Для обеспечения хорошей совместимости и интеграции между различными системами и приложениями многие отраслевые стандарты и протоколы предпочитают использовать XML в качестве основного формата обмена данными. Преобразования JSON в XML позволят удобно обрабатывать и передавать структурированные данные между различными системами и устройствами. Целью данной работы является разработка модели преобразования JSON в выходные XSD и XML, позволяющей корректно обрабатывать данные при их преобразовании в обоих направлениях без изменения структуры JSON-документа. Метод исследования основан на подходе поиска и слияния схожих подграфов внутри графа одного типа, при том, что у сравниваемых подграфов разного типа тип не расширяется за счет использования операции слияния. Проблема различия структур и спецификаций форматов данных решается путем разработки модели преобразования, учитывающей специфику конкретных данных и потребности приложения.*

Ключевые слова: *веб-сервисы, структурированные данные, преобразование форматов, гетерогенность, взаимодействие, JSON, XML, XSD.*

REFERENCES

1. Song E., Haw S.-C. XML-REG: Transforming XML Into Relational Using Hybrid-Based Mapping Approach, IEEE Access, 8, 2020, pp. 177623-177639.
2. Haroune-Belkacem N., Semchedine F., Al-Shammari A., Aissani D. SMCA: An efficient SOAP messages compression and aggregation technique for improving Web services performance, J. Parallel Distrib. Comput., vol. 133, Nov. 2019, pp. 149-158.
3. Leal R., Queiros M. Generation of Document Type Exercises for Automated Assessment, Licensed under Creative Commons License CC-BY 4.0 11th Symposium on Languages, Applications and Technologies (SLATE 2022). Article No. 4, 2022, pp. 41-46.
4. Baazizi M.-A., Colazzo D., Ghelli G., Sartiani C. Parametric schema inference for massive JSON datasets. The VLDB Journal, 2019. doi: 10.1007/s00778-018-0532-7
5. Guo S., Xia H., Xiang G. Research on the translation from XSD to JSON schema. 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017. doi: 10.1109/iccsn.2017.823033
6. Lee J., Anjos E., Satti S.R. SJSON: A succinct representation for JSON documents, Information Systems 97, 101686, 2021, pp. 27-38.
7. Vodnansky D. Three Metric-Based Method for Data Compatibility Calculation, Acta Informatica Pragensia, vol. 10 (1), 2021, pp. 38-60.
8. Breje A.R., Gyorodi R., Gyorodi C., Zmaranda D., Pecherle G. Comparative study of data sending methods for XML and JSON models, Int. J. Adv. Comput. Sci. Appl, vol. 9, no. 12, 2018, pp. 198-204.
9. Pezoa F., Reutter J.L., Suarez F. Foundations of JSON Schema. International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 263-273.
10. Mukhitova A., Yerimbetova A., Cherikbayeva L. Development of an adaptive graphic web interface model for editing XML data, Eastern-European Journal of Enterprise Technologies, 2 (122), 2023, pp. 26-35.