# Intellectual Analysis of Making Decisions Tree in Information Systems of Screening Observation for Immunological Patients

Lyubomyr Chyrun[0000-0002-9448-1751]1, Eugene Leshchynskyy[0000-0001-7627-5252]2, Vasyl Lytvyn[0000-0002-9676-0180]3, Antonii Rzheuskyi[0000-0001-8711-4163]4, Victoria Vysotska[0000-0001-6417-3689]5, Yuriy Borzov[0000-0002-0604-0498]6

[1]Ivan Franko National University of Lviv, Lviv, Ukraine
[2]LvivSoft, Lviv, Ukraine
[3-5]Lviv Polytechnic National University, Lviv, Ukraine
[6]Lviv State University of Life Safety, Lviv, Ukraine

chyrunlv@gmail.com[1], leshhynskyy@gmail.com[2], vasyl.v.lytvyn@lpnu.ua[3], antonii.v.rzheuskyi@lpnu.ua[4], Victoria.A.Vysotska@lpnu.ua[5], uob1968@gmail.com[6]

**Abstract.** The aspects of the design of an immunological patient screening observation information system related to the required system functionality are discussed. Practical solutions for individual operational challenges are suggested. Functional requirements for the designed information system are defined. Based on the outlined functional needs, the appearance of the kernel of the information system is determined, namely, in what form the entered data will be stored. Some functions of rules set formation for Machine learning and Neural Network building based on Making Decisions Tree, which have to be resolved by system, are defined. Practical solutions are proposed for specific tasks that are entrusted to the information system.

**Keywords.** Information system, Intellectual Analysis, Data mining, Immunological Patients, Making Decisions Tree, Machine learning

## 1 Introduction

Data mining is an integral part of the knowledge discovery in data bases process [1]. It allows us to uncover the essence of hidden dependencies in the data, to identify interactions between the properties of objects, information about which is stored in databases, to highlight the patterns inherent in a particular set of data [2]. The urgency of the problem of data exploration and processing is confirmed by the widespread practical and commercial use of intellectual analysis systems [3]. Most often they are used in the scientific field and business [4]. Hierarchically organized data and work with them is an important place in human life [5]. The structure of such data is characterized by the specific relations of its elements, namely, features of imitation and


2019 IDDM Workshops.

subordination between elements of the hierarchy [6]. Therefore, when creating an information system for the subject area, characterized by these features, an important issue to be paid attention to is the storage of not only data but also their structure [7].

## 2 Analysis of current research

The relatively new methodology of rough sets is naturally adapted to work with imperfect data [8]. The theory of rough sets was developed by Z. Pawlak [1] as a mathematical tool for overcoming data contradictions and revealing hidden patterns or patterns in them. A fundamental principle of the example learning algorithm using rough sets is to identify redundancy among the available features that describe the example [9]. This is how strong signs that influence the classification of an example are revealed. Next, remove the columns whose attributes do not affect the classification [10]. Only the attributes on which the classification of the table examples is depend on the analysis [11]. The remaining attributes are called redundant [12]. In other words, a reducer is a subset of all table attributes that provide the same classification result for all table instances as the entire set of examples [13]. Finding a reducer is an NP challenge [14]. However, there are enough effective methods to solve this task within an acceptable time [15]. Such methods include, in particular, Boolean reasoning [16-19], Johnson's algorithm [20-23].

## 3 Statement of the problem

The life cycle of information begins with its creation [24-29]. Therefore, as noted in [1], information about the patient can be obtained and pre-systematized to fill in the questionnaire of the established sample. A survey of the real questionnaire was also conducted and the shortcomings encountered in filling it out were identified [30-33]. It was also noted that the information submitted to the questionnaire has a clearly defined hierarchical structure [34-38]. To automate and simplify the process of filling out the questionnaire, the following provisions were defined [39]:

1. Determine the ratio of subordination of one item of the questionnaire to another (to actually determine the appearance of the hierarchy) [40];
2. To determine the domains of possible values to limit the “free” entry of data for the relevant items of the questionnaire [41].

A variant of practical solution to the problem of collecting and entering information by filling and saving the corresponding “electronic” analogue of the questionnaire was also proposed [42]. This avoided many inconveniences and uncertainties and, in general, automated the process of "converting" data to electronic form [43]. It should be noted that the following [1] had several "structural" disadvantages, namely:

1. The structure of the stored data had a linear appearance and did not in any way reflect the existing hierarchical dependencies between the corresponding items of the questionnaire [44];

2. Regardless of the number of questionnaire items filled in per patient, the values of all items were stored (even if these were blank). Obviously, this approach, despite the meager amount of "unnecessary" data, does not set a good tone when designing an information system [45].

In [46], a variant of hierarchical information using frames was proposed. In [1], as a practical extension of the idea outlined in [47], the mechanism of processing and storing data of a completed questionnaire based on the use of frames was considered.

## 4 Functional requirements definition

**Entering information.** As noted above, the life cycle of information begins with its creation. It should be noted that in item # 6 of the questionnaire (Fig. 1) it is envisaged to enter information about the stage of treatment (“primary”, “repeated”). So, let's distinguish the following tasks:

1. Entering the data of the patient being examined for the first time (“primary”);
2. Entering the data of the patient who has come for re-examination ("re-examination"). By the way, there can be several such calls.

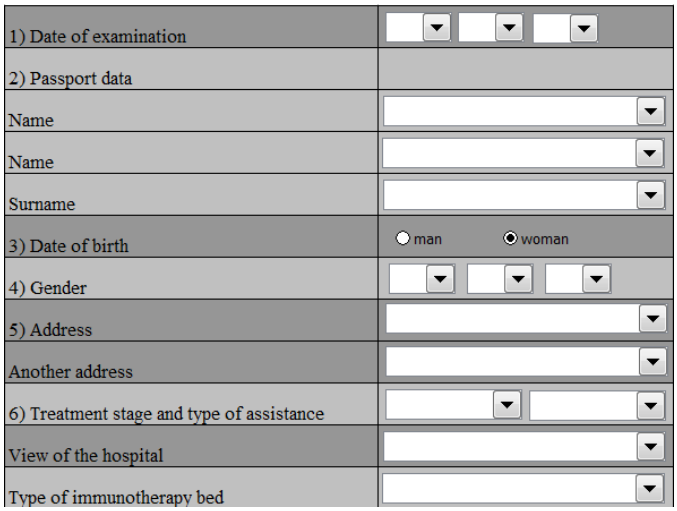


**Fig. 1.** Fragment of questionnaire form

In the first case it is necessary to fill in the questionnaire of the established sample. The second is the same, but it can be assumed that the "repeated" questionnaire does not undergo significant changes, so it would be convenient to re-enter the questionnaire only to correct the data entered in the primary card. Therefore, when making a "repeat" patient, it would be convenient to find the appropriate primary questionnaire, make a copy of it and edit the last one.

Therefore, the data entered should be stored in a convenient way:

1. Search for the required primary (or last completed) questionnaire;
2. Display of the entered data in the form of the second questionnaire.

**Development environment.** The prototype of the information system was implemented using the Web interface [1]. Recall that in this case, after filling in the questionnaire, pressing the "Save" button initiates the process of registration of the entered data in a special object REQUEST and transfer it for processing to the system server. The REQUEST object looks like a hash table [1].

Therefore, the look of the stored data should be easy to generate by simply processing the data in the corresponding hash table.

**Accounting information.** With the advent of data entry and storage tools, the system begins to perform database functions. There is a need for accounting that is natural for databases, for example:

1. Search of patients' cards according to a number of specified parameters (possibility to make a sample);
2. Removal and editing of completed questionnaires;
3. Backup and backup of the questionnaire;
4. Playback of stored data in a user-friendly way;
5. Obtaining a “hard” copy of the patient's card;
6. Archiving of data.

**Analysis and statistics.** Because the system is designed as a screening system, the look of the data should make it easy to find the information you need. This can be used both to sample according to specified criteria (for research) and to prepare reports of the required form.

**Transformation.** To date, the classification of diseases is defined in the document on the international classification of diseases MKH-10 [1]. Obviously, there is an opportunity to review and modify existing provisions in this classification. This will change the form in which the data is to be obtained (although the content of the information may not change). It is logical to assume that the stored data should also be brought in a new form as required. We substantiate the importance of this point in the example.

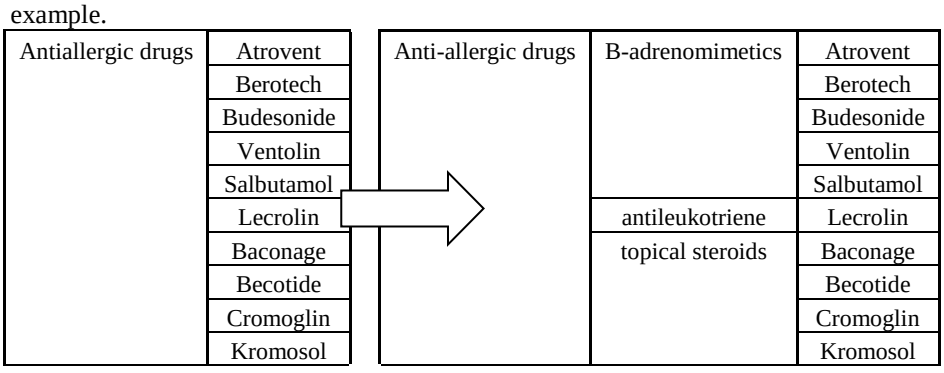


**Fig. 2.** An example of transformation

Fig. 2 illustrates the need for various transformations in the structure of the questionnaire. The second column on the left contains is a list of some drugs, and suppose you need to enter an additional classification for these drugs, which leads to the appearance of an additional column on the right. If certain drugs were prescribed to the patient and the relevant data were entered into the questionnaire prior to structural transformation, then obviously, after transformation of the questionnaire, the stored data should be modified and brought to the appropriate appearance.

We illustrate the previous case with the help of a tree (Fig. 3).

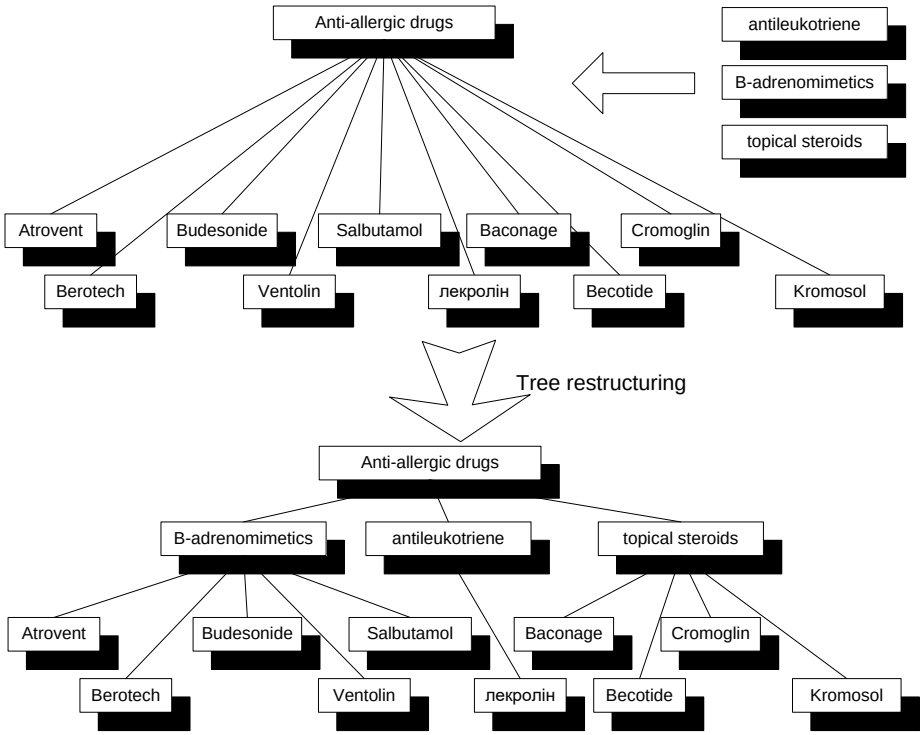


**Fig. 3.** An example of transformation in a tree structure

So, with the questionnaires already filled in, you need to review them and modify the hierarchical dependencies between the data. Obviously, the data form should allow it to be done quickly and conveniently.

In summary, it can be said that the form of presentation in the designed system should allow the following operations to be performed efficiently and conveniently:

1. Formation (creation) of data by processing the hash table;
2. Search for analysis and statistics;
3. Implementation of transformations;
4. Interpretation of the data entered in order to:
5. Obtaining a “hard” copy of the data in a convenient form;

6. Displaying data for editing purposes;
7. Displaying data for the purpose of introducing a 'repeat' patient.

## 5 Method of storing hierarchical data using N-tree

Let's take a closer look at the data storage process that has been implemented so far. Let, in Fig. 4 provides a form in which to obtain information about the availability of a patient referral for examination.

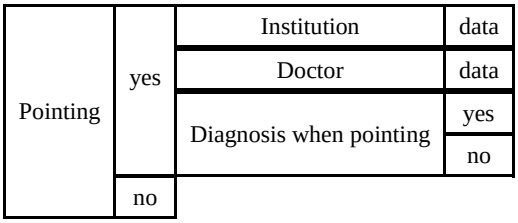


**Fig. 4.** The form in which you need to get information about the presence of directions

We construct an interface (questionnaire form) that corresponds to this form (Fig. 5). In Fig. 5 also lists the names of the corresponding visual form controls.

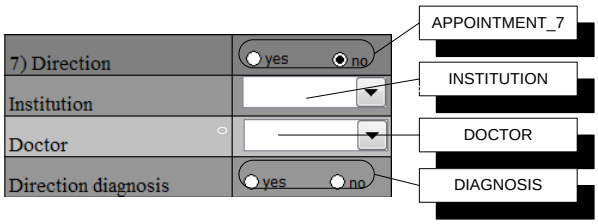


**Fig. 5.** The interface corresponding to the form in fig. 4

Following the principles of object-oriented programming, the PATIENT class was created, whose properties were all the data provided in the questionnaire (Fig. 6).

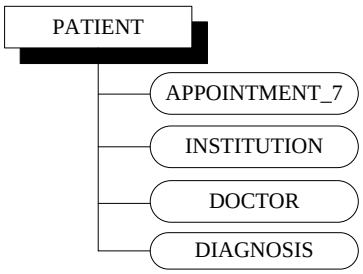


**Fig. 6.** Structure of the PATIENT class

As stated above, the structure of the class PATIENT is linear and in no way reflects the hierarchical dependence that exists between the individual items of the questionnaire.

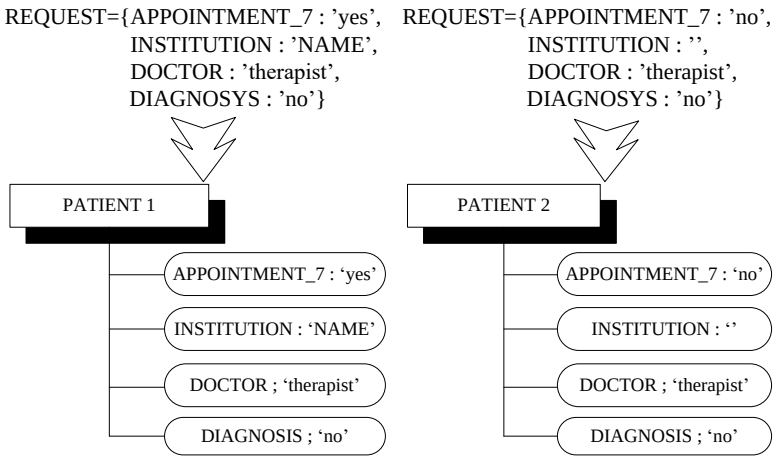


**Fig. 7.** The process of storing data

In Fig. 7 depicts a data retention process for two different patients. In both cases, when trying to save the completed questionnaire data, they are formed into a REQUEST object and passed to the server for processing, then PATIENT physical objects are created and, since the class structure is static, in PATIENT 1 and PATIENT 2 objects Stores the values of all properties available in REQUEST. For the first patient, the value of "Point" (APPOINTMENT_7) is marked as "yes", so it is advisable to keep hierarchically subordinate to him the INSTITUTION, DOCTOR and DIAGNOSYS items. For the second patient, the value of "Pointing" is marked as "no", so saving the remaining points is not appropriate, but due to the static structure of the class "PATIENT", their values are stored (even when they are empty). In [1], a mechanism for storing questionnaire data using a frame was proposed, but the disadvantage of this was the need to view all frame slots, regardless of whether or not the slot data was required. Consider the mechanism of storing the input data using the N-ary tree. For example, consider the fragment of the questionnaire shown in Fig. 8.

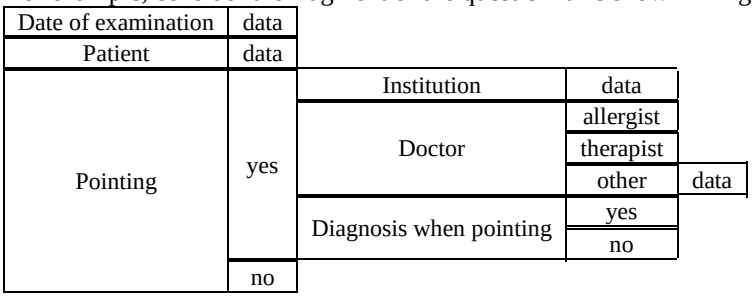


**Fig. 8.** Fragment of the questionnaire

Let us now depict this fragment as an N-ary tree (Fig. 9).

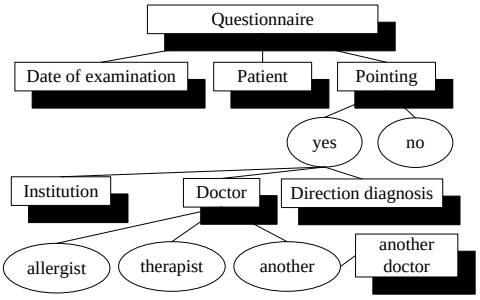


**Fig. 9.** Presentation of the questionnaire from fig. 8 in the form of an N-ary tree

From Fig. 9 it can be seen that certain values of the individual items of the questionnaire make it possible to descend to the next level of the hierarchy. For "Pointing" this value is "Yes" and for "Doctor" the value is "Other". Now suppose some data were collected in the questionnaire and the corresponding REQUEST objects were sent to the server for processing.

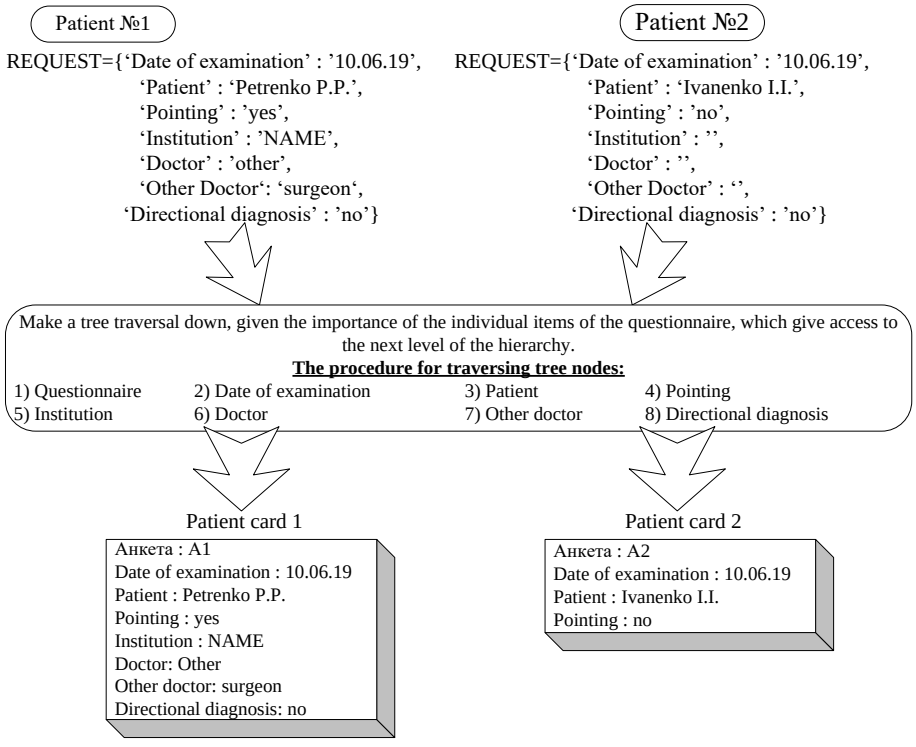


**Fig. 10.** The process of processing a REQUEST object using a tree

As can be seen from Fig. 10, a top-down tree is traversed corresponding to the hierarchical structure of the questionnaire. However, for individual nodes, the next level of the hierarchy can be accessed only with a certain parent node value defined. Thus, for the first patient, the value of "Pointing" is defined as "yes", therefore, the transition to a lower level takes place and the units "Institution", "Doctor" and "Diagnosis at referral" are taken for consideration. Since the value "Other" is set for the Doctor node, the value of "Other Doctor" is considered at the subordinate level. For the second patient, the value "no" for "Direction" does not allow access to the next tree level, so subordinate nodes are not considered at all. Thus, for the second patient, only the data with the contents will be stored on the card. Note that this approach not only saves space, but also eliminates the need to consider "unnecessary" data. So, the general idea is that when processing each typed questionnaire, you need to crawl from top to bottom of a given tree, taking into account the "bypass" values for the corresponding nodes of the tree. The question is: how to build such a tree? We use a frame for this purpose. For the example above, the frame will look like this (Figure 11):

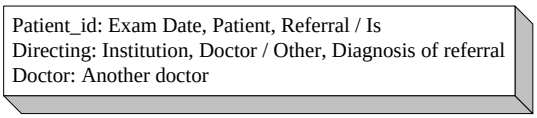


**Fig. 11.** The frame view for the questionnaire from fig. 8

As can be seen from the picture, the frame is actually a two-dimensional list, which specifies the tree structure and the "bypass" values for the corresponding items of the questionnaire. In each frame of the frame, the first element denotes the parent node, and all subsequent elements in the row represent child nodes. In Fig. 12 illustrates the process of constructing a tree from a given frame.

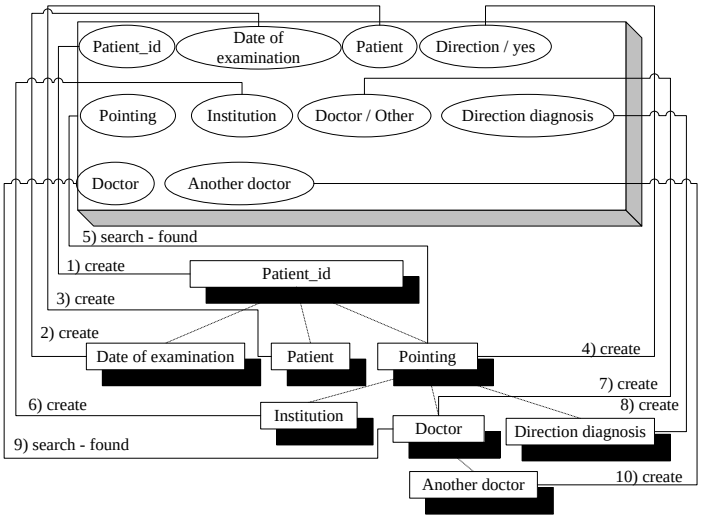


**Fig. 12.** The mechanism of construction of N-pillar tree from a frame

As can be seen from the figure, the tree of the desired structure is constructed by sequentially viewing the corresponding frame. The first element (Direction, Doctor) of each frame of the frame (except for the first line where all elements must be created) is searched for a node already built that corresponds to this element, after which the found node descendants are built. Note that in the process of construction, it is necessary to store the "bypass" values for the elements in which it is defined. Thus, the scheme of storing only the necessary data becomes quite convenient and transparent. The sequence of this process is shown in Fig.13.

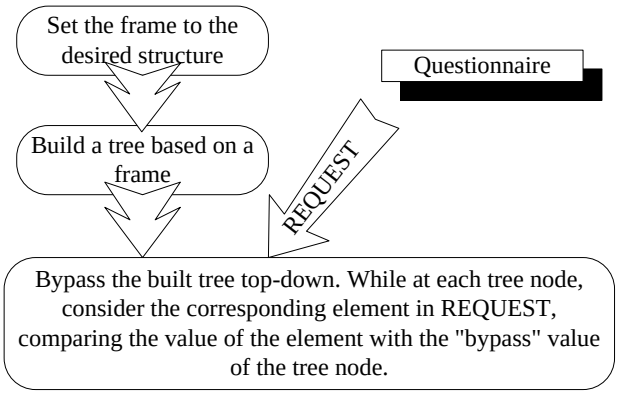


**Fig. 13.** Data storage scheme

As to the appearance in which the data will be stored for the moment, let's say: bypassing the tree, we will write each considered pair "element - value of element" in a separate line (like the one shown in Fig. 9). Until now, we have considered the case where the questionnaire item had only one value, which allowed to move to the next level of the hierarchy (the value "yes" for "Pointing" and "other" for "Doctor"). However, there may be situations where multiple values (or even each) lead to a lower level. For example, consider the fragment of the questionnaire depicted in Fig. 14.

<table>
<tr><td rowspan="3">By etiology</td><td rowspan="2">known</td><td>congenital (antenatal)</td><td rowspan="2">infectious</td><td>HIV / AIDS</td></tr>
<tr><td>acquired (postnatal)</td><td>herpes viruses I, II (HerSVI,II)</td></tr>
<tr><td>unknown</td><td colspan="3"></td></tr>
</table>

**Fig. 14.** Fragment of the questionnaire illustrating the “gravitas” of some points of the questionnaire

Note that the values "congenital" and "acquired" are mutually exclusive, and the choice of each leads to the next level of hierarchy ("infectious"). In this situation, the infectious and other subordinate levels are the same for the two values, but they may be different. Consider how the previously described storage mechanism works in this case. But we are modifying the frame somewhat, as some of the questionnaires now

have more than one "bypass" value. The questionnaire interface, corresponding frame and tree are shown in Fig. 15.

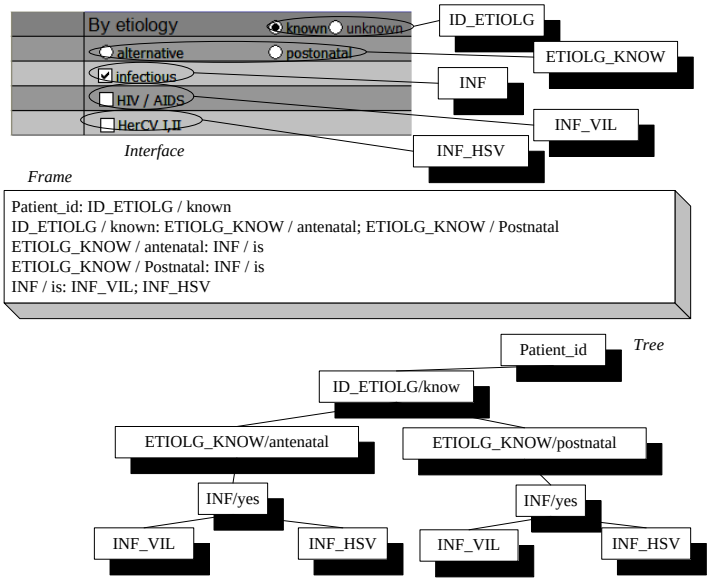


**Fig. 15.** Interface, frame, and tree for multiple bypass values

As you can see in the picture, the idea of defining a tree structure with a frame remained, but now the tree node is further identified by a "bypass" value. Now let's consider the process of data storage directly (see Fig. 16).

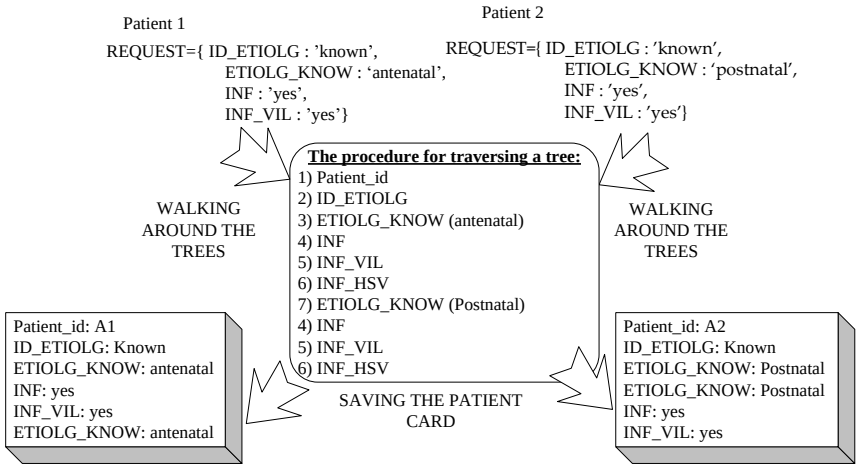


**Fig. 16.** Scheme of the process of data storage for the case of several "bypass" values

The process of data storage takes place according to the scheme already considered (Fig. 13), but it can be noticed that the value of the element ETIOLG_KNOW is stored in the card twice (exactly two mutually exclusive values are provided for this item of the questionnaire). This happens for the following reason: when bypassing the tree for the first patient while in the node ETIOLG_KNOW / antenatal, REQUEST looks for the key ETIOLG_KNOW, takes its value (antenatal), comparing the latter with the bypass value of the node (antenatal) gives a positive result and moves to the next level of the hierarchy; however, in the further crawl, being at the node ETIOLG_KNOW / postnatal, again in REQUEST the key ETIOLG_KNOW is searched, its value (again "antenatal") is taken, comparing the latter with the bypass value of the node (postnatal) gives a negative result, there is no transition to the next level, but since the call to the ETIOLG_KNOW / postnatal node has been completed, this is recorded in the patient card as another line ETIOLG_KNOW: antenatal. A similar situation occurs with the second patient, however, the order in which the lines in the card are followed differs slightly from the first patient (due to tree crawling). The identified deficiency can be corrected by removing unnecessary (unnecessary) branches with a root in the corresponding "multivalued" node before processing the questionnaire data. This process is illustrated in Fig. 17.

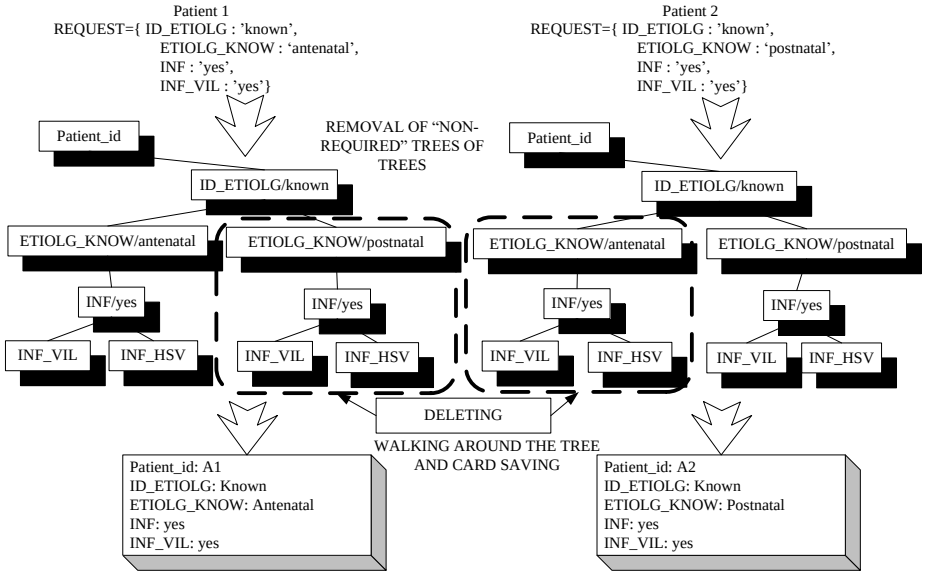


**Fig. 17.** Removal of unnecessary tree branch with root in "multivalued node"

## 6 Interpretation of stored data

It has been determined that the input of the input data is required for several purposes. Consider the process of interpreting data to obtain a "hard" copy of a patient's card. It is clear that such a copy can be obtained by printing the corresponding text file on

paper. At present, patient data is stored as a sequence of rows of element-value element pairs, and it is obvious that this form of data does not reflect the hierarchical dependencies defined between the items of the questionnaire. An acceptable print layout might be, for example, the shape shown in Fig. 18. It is easy to see that the written sequence of rows corresponds to the sequence of traversing the corresponding tree from top to bottom. A general diagram of the process of obtaining a card for printing is shown in Fig. 19.

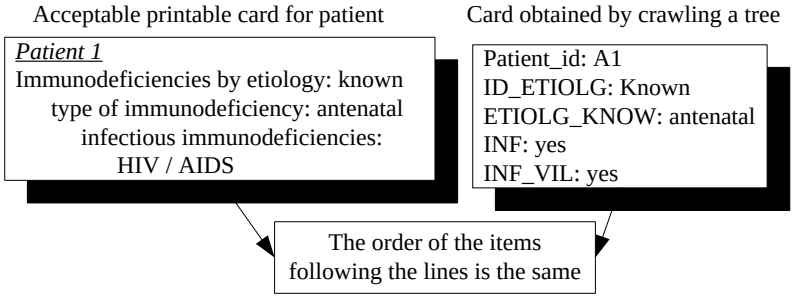


**Fig. 18.** Matching the tree-to-bottom crawl sequence of the data display se*quence*

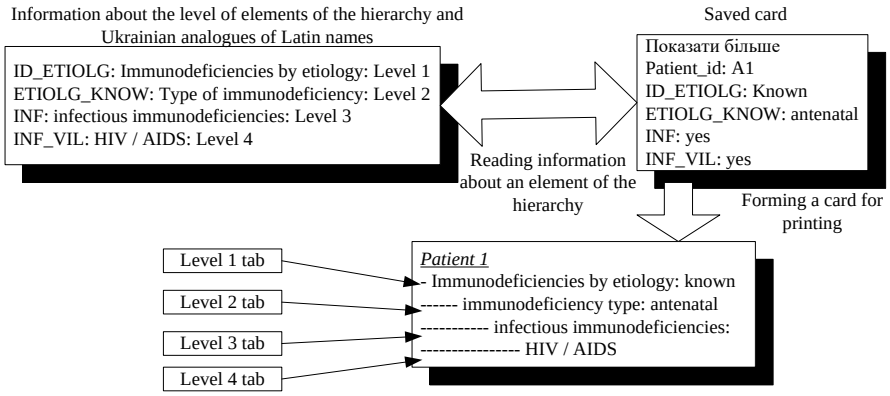


**Fig. 19.** Receiving a card for printing by adding a tab to the saved card

Therefore, you can retrieve the print card from the saved card by adding indentations on the left. The amount of indentation is determined by the specified hierarchical element level.

## 7 Conclusions

For the projected information system, a number of necessary executable functions were defined, based on which the functional requirements to the kernel of the system

were formed and the question was asked: in what form the data should be stored in order to allow the system to efficiently solve the tasks assigned to it. The main unresolved issues up to this point were: how to store only the required data and how to save information about the hierarchical structure of this data? To solve the first problem, an approach based on the use of n-tree arrays and frames was proposed, as well as some specific situations that were encountered. With regard to the second problem, it was solved in part by the example of obtaining a "hard copy" of the patient's card. At the practical level, the problems of hierarchical data editing and transformation implementation remain unresolved. Thus, we can conclude that the approaches described in the article are general and can be applied to any subject areas with hierarchical organization of information.

## References


1. Pawlak, Z.: Rough sets: Theoretical aspects of reasoning about data. In: Springer Science & Business Media, 9. (2012)
2. Iturrioz, J., Azpeitia, I., Díaz, O.: Generalizing the like button: empowering websites with monitoring capabilities. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, 743-750. (2014).
3. Maggi, F., Robertson, W., Kruegel, C., Vigna, G.: Protecting a moving target: Addressing web application concept drift. In: International Workshop on Recent Advances in Intrusion Detection, Springer, Berlin, Heidelberg, 21-40. (2009).
4. Christensen, J. H.: Using RESTful web-services and cloud computing to create next generation mobile applications. In: the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, 627-634. (2009).
5. Neugschwandtner, M., Neugschwandtner, G., Kastner, W.: Web services in building automation: Mapping knx to obix. In: Int. Conf. on Industrial Informatics, 87-92. (2007).
6. Tkachenko, R., Izonin, I., Kryvinska, N., Chopyak, V., Lotoshynska, N., Danylyuk, D.: Piecewise-linear Approach for Medical Insurance Costs Prediction using SGTM Neural-Like Structure. In: CEUR Workshop Proceedings, 170-179. (2018)
7. Tepla, T., Izonin, I., Duriagina, Z., Tkachenko, R., Trostianchyn, A., Lemishka, I., Kulyk V., Kovbasyuk, T.: Alloys selection based on the supervised learning technique for design of biocompatible medical materials. In: Archives of Materials Science and Engineering, 93/1, 32-40. (2018)
8. Vysotska, V., Lytvyn, V., Burov, Y., Gozhyj, A., Makara, S.: The consolidated information web-resource about pharmacy networks in city. In: CEUR Workshop Proceedings, 239-255 (2018)
9. Kravets, P.: The control agent with fuzzy logic. In: Perspective Technologies and Methods in MEMS Design, MEMSTECH'2010, 40-41 (2010)
10. Bisikalo, O., Ivanov, Y., Sholota, V.: Modeling the Phenomenological Concepts for Figurative Processing of Natural-Language Constructions. In: CEUR Workshop Proceedings, Vol- 2362, 1-11. (2019)
11. Babichev, S., Taif, M.A., Lytvynenko, V., Osypenko, V.: Criterial analysis of gene expression sequences to create the objective clustering inductive technology. In: 2017 IEEE 37th International Conference on Electronics and Nanotechnology, 244-248. (2017)
12. Babichev, S., Gozhyj, A., Kornelyuk A., Litvinenko, V.: Objective clustering inductive technology of gene expression profiles based on SOTA clustering algorithm. In: Biopolymers and Cell, 33(5), 379–392. (2017)

13. Babichev, S.A.: An evaluation of the information technology of gene expression profiles processing stability for different levels of noise components. In: Data, 3(4), art. 48. (2018)
14. Shakhovska, N. B., Noha, R. Y.: Methods and tools for text analysis of publications to study the functioning of scientific schools. In: Journal of Automation and Information Sciences, 47(12). (2015)
15. Bobalo, Y., Stakhiv, P., Mandziy, B., Shakhovska, N., Holoschuk, R.: The concept of electronic textbook "Fundamentals of theory of electronic circuits. In: Przegląd Elektrotechniczny, 88 NR 3a/2012, 16-18. (2012)
16. Shakhovska, N., Shvorob, I.: The method for detecting plagiarism in a collection of documents," Computer Sciences and Information Technologies (CSIT), 142-145. (2015)
17. Arzubov, M., Shakhovska, N., Lipinski, P.: Analyzing ways of building user profile based on web surf history. In: Computer Sciences and Information Technologies (CSIT), 1, 377-380. (2017)
18. Mukalov, P., Zelinskyi, O., Levkovych, R., Tarnavskyi, P., Pylyp, A., Shakhovska, N.: Development of System for Auto-Tagging Articles, Based on Neural Network. In: CEUR Workshop Proceedings, Vol-2362, 106-115. (2019)
19. Shakhovska, N., Basystiuk, O., Shakhovska, K.: Development of the Speech-to-Text Chatbot Interface Based on Google API. In: CEUR Workshop Proceedings, Vol-2386, 212-221. (2019)
20. Nazarkevych, M., Klyujnyk, I., Nazarkevych, H.: Investigation the Ateb-Gabor Filter in Biometric Security Systems. In: Data Stream Mining & Processing, 580-583. (2018).
21. Lytvynenko, V., Wojcik, W., Fefelov, A., Lurie, I., Savina, N., Voronenko, M. et al.: Hybrid Methods of GMDH-Neural Networks Synthesis and Training for Solving Problems of Time Series Forecasting. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 513-531. (2020)
22. Rusyn, B., Lytvyn, V., Vysotska, V., Emmerich, M., Pohreliuk, L.: The Virtual Library System Design and Development, Advances in Intelligent Systems and Computing, 871, 328-349 (2019)
23. Rusyn, B., Lutsyk, O., Lysak, O., Lukeniuk, A., Pohreliuk, L.: Lossless Image Compression in the Remote Sensing Applications. In: Int. Conf. on Data Stream Mining & Processing (DSMP), 195-198 (2016)
24. Emmerich, M., Lytvyn, V., Yevseyeva, I., Fernandes, V. B., Dosyn, D., Vysotska, V.: Preface: Modern Machine Learning Technologies and Data Science (MoMLeT&DS-2019). In: CEUR Workshop Proceedings, Vol-2386. (2019)
25. Rusyn, B., Vysotska, V., Pohreliuk, L.: Model and architecture for virtual library information system. In: Computer Sciences and Information Technologies, CSIT, 37-41 (2018)
26. Lytvyn, V., Sharonova, N., Hamon, T., Cherednichenko, O., Grabar, N., Kowalska-Styczen, A., Vysotska, V.: Preface: Computational Linguistics and Intelligent Systems (COLINS-2019). In: CEUR Workshop Proceedings, Vol-2362. (2019)
27. Burov, Y., Vysotska, V., Kravets, P.: Ontological Approach to Plot Analysis and Modeling. In: CEUR Workshop Proceedings, Vol- 2362, 22-31. (2019)
28. Vysotska, V., Lytvyn, V., Burov, Y., Berezin, P., Emmerich, M., Basto Fernandes V.: Development of Information System for Textual Content Categorizing Based on Ontology. In: CEUR Workshop Proceedings, Vol- 2362, 53-70. (2019)
29. Veres, O., Rishnyak, I., Rishniak, H.: Application of Methods of Machine Learning for the Recognition of Mathematical Expressions. In: CEUR Workshop Proceedings, Vol- 2362, 378-389. (2019)

30. Furgala, Y., Rusyn, B.: Peculiarities of melin transform application to symbol recognition. In: International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 251-254. (2018)
31. Kapustiy, B., Rusyn, B., Tayanov, V.: Peculiarities of application of statistical detection criteria for problems of pattern recognition. In: Journal of Automatioin and Inrormation Science, 37(2), 30-36. (2005)
32. Rusyn, B., Kosarevych, R., Lutsyk,O.,Korniy,V.: Segmentation of atmospheric clouds images obtained by remote sensing. In: International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 213-216. (2018)
33. Kapustiy, B., Rusyn, B.,Tayanov, V.: A new approach to determination of correct recognition probability of set objects. In: Upravlyaushchie Sistemy i Mashiny, 2, 8-12. (2005)
34. Rusyn, B., Prudyus, I., Ostap,V.: Fingerprint image enhancement algorithm. In: The Experience of Designing and Application of CADSM, 193-194. (2001)
35. Rusyn, B., Torska, R., Kobasyar, M.: Application of the cellular automata for obtaining pitting images during simulation process of thei growth. In: Advances in Intelligent Systems and Computing, 242, 299-306. (2014)
36. Varetskyy, Y., Rusyn, B., Molga, A., Ignatovych, A.: A new method of fingerprint key protection of grid credential. In: Advances in Intelligent and Soft Computing, 84, 99-103. (2010)
37. Pukach, P., Il'kiv, V., Nytrebych, Z., Vovk, M., Pukach, P.: On the Asymptotic Methods of the Mathematical Models of Strongly Nonlinear Physical Systems. In: Advances in Intelligent Systems and Computing, 689, 421- 433. (2018)
38. Pukach, P.: Investigation of bending vibrations in Voigt-Kelvin bars with regard for nonlinear resistance forces. In: Journal of Mathematical Sciences, 215(1), 71-78. (2016)
39. Lavrenyuk, S., Pukach, P.: Mixed problem for a nonlinear hyperbolic equation in a domain unbounded with respect to space variables. In: Ukrainian Mathematical Journal, 59(11), 1708-1718. (2007)
40. Pukach P.: Qualitative Methods for the Investigation of a Mathematical Model of Nonlinear Vibrations of a Conveyer Belt. In: J. of Mathematical Sciences, 198, 31-38. (2014)
41. Nytrebych, Z., Malanchuk, O., Il'kiv, V., Pukach, P.: Homogeneous problem with two-point conditions in time for some equations of mathematical physics. In: Azerbaijan Journal of Mathematics, 7(2), 180-196. (2017)
42. Nytrebych, Z., Il'kiv, V., Pukach, P., Malanchuk, O.: On nontrivial solutions of homogeneous Dirichlet problem for partial differential equations in a layer. In: Kragujevac Journal of Mathematics, 42(2), 193-207. (2018)
43. Lytvyn, V., Vysotska, V., Rusyn, B., Pohreliuk, L., Berezin, P., Naum O.: Textual Content Categorizing Technology Development Based on Ontology. In: CEUR Workshop Proceedings, Vol-2386, 234-254. (2019)
44. Basyuk, T.: The main reasons of attendance falling of internet resource. In: Proc. of the X-th Int. Conf. Computer Science and Information Technologies, CSIT'2015, 91-93. (2015).
45. Lytvyn, V., Vysotska, V., Rzheuskyi, A.: Technology for the Psychological Portraits Formation of Social Networks Users for the IT Specialists Recruitment Based on Big Five, NLP and Big Data Analysis. In: CEUR Workshop Proceedings, 2392, 147-171. (2019)
46. Vysotska, V., Burov, Y., Lytvyn, V., Oleshek, O.: Automated Monitoring of Changes in Web Resources. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 348–363. (2020)
47. Demchuk, A., Lytvyn, V., Vysotska, V., Dilai, M.: Methods and Means of Web Content Personalization for Commercial Information Products Distribution. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 332–347. (2020)