Expert Systems With Applications 115 (2019) 442–464

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A system for transformation of sentences from the enriched formalized Node of Knowledge record into relational database

Sanja Čandrlić[*], Martina Ašenbrener Katić, Mile Pavlić

Department of Informatics, University of Rijeka, Radmile Matejčić 2, Rijeka 51000, Croatia

A R T I C L E   I N F O

Article history:
Received 17 January 2018
Revised 8 June 2018
Accepted 7 July 2018
Available online 18 July 2018

Keywords:
Node of Knowledge (NOK)
Formalized Node of Knowledge (FNOK)
Relational database
Algorithm
Knowledge-based system

A B S T R A C T

Verbalized text contains knowledge necessary and sufficient for transfer of numerous human cognitions. The question is how this knowledge was saved into text. Authors believe that they found an important idea how to assemble knowledge into nodes of knowledge. Similar methods in the field of knowledge networks exist, but none of them does it in the same way as the Node of Knowledge (NOK) method. Basic terms and concepts in a language are represented in words whose meaning is final and cannot be divided in subterms. More complex meaning can be achieved by combining words in sentences. According to authors' opinion, each sentence includes connective medium for words in the sentence (related to semantic reasons), which is inwrought in the Node of Knowledge method. Using the Node of Knowledge method each sentence can be presented as a network of connected words. This network is enriched with links between words so a computer can interpret meaning and knowledge of the sentence in the same way an intelligent person does. A formalized and semantically enriched record of sentences (called Formalized Node of Knowledge – FNOK) is developed. Authors find that in this way, even without statistical text analysis, an algorithm can give correct answer to a question set based on written text. This paper presents the system for transformation of textual knowledge expressed in natural language sentences into a relational database. The system is a part of a larger knowledge-based system based on the Node of Knowledge (NOK) conceptual framework for knowledge-based system development. This paper starts with sentences written in the formalized and enriched form for which a logical transformation into the structure of a relational database is proposed. The system and algorithms for the transformation of formalized sentences into n-tuples of the relational database are distributed and represented in several steps. The research has shown that it is possible to store semantically enriched sentences in relational databases. The solution presented in this paper is important for further development of the system for receiving questions from users and providing answers (i.e. question-answering system), with the ability to use well-developed relational SQL languages. Relational database of texts enables numerous applications in the field of expert and intelligent systems.

1. Introduction

Data from business information systems is stored mainly in relational databases. In the relational databases, the entire sentences of natural language can also be stored, by creating a special attribute of the "memo" field type which can receive text as input. However, it is not possible to use the SQL query language over such non-structured record.

Search through sentences constitutes of indexing the text and searching for one or more words in a single text. The result of the search is the representation of the records in which the mentioned words appear. It is not possible to set a query to get the exact answer contained in the text.

There is a need for systems that would enable text expressed knowledge to be stored in the database. In addition, there is a need to ask questions and to search for answers in textual records.

The aim of this paper is to create a system model and algorithm for transforming textual knowledge into a relational database. Textual knowledge is represented as formalized records using *Formalized Node of Knowledge* (FNOK). In the database, the text would not be entered as a complete *Formalized Node of Knowledge* record, but instead a transformation of the formalized record into interconnected data sets would be conducted. The data thus obtained would be entered as values in the appropriate attributes, and the foreign keys would provide a link between n-tuples representing a single formalized sentence.

* Corresponding author.
E-mail addresses: sanjac@inf.uniri.hr (S. Čandrlić), masenbrener@inf.uniri.hr (M. Ašenbrener Katić), mile.pavlic@ris.hr (M. Pavlić).

https://doi.org/10.1016/j.eswa.2018.07.021
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

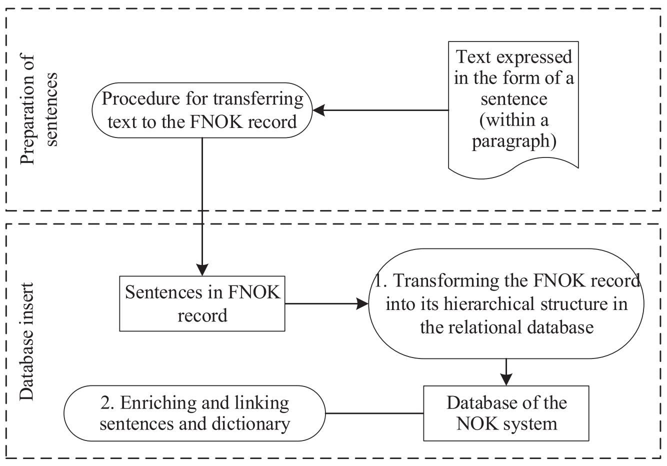
Preparation of sentences
Procedure for transferring text to the FNOK record
Text expressed in the form of a sentence (within a paragraph)
Database insert
Sentences in FNOK record
1. Transforming the FNOK record into its hierarchical structure in the relational database
2. Enriching and linking sentences and dictionary
Database of the NOK system

Fig. 1. Process model of the system for the transformation of formalized (FNOK) sentences into the relational database.

Natural language processing (NLP) deals with the processing of natural language texts and by extracting knowledge from the text. There are researches related to querying in natural language and translating the query text into an SQL query (Kanhe, Udawant, Bodke, & Chikhale, 2015;Kovács, 2011;Oracle, 2015). The problem of entering text into the database is not resolved in these works.

Development of information systems relies on development of technology and software tools in the field of information and communication technology. It is usual to divide each system to two related models: data model and algorithm model. Implementation of the data model is achieved by different database organizations, and especially developed and used is the paradigm for relational database organization. Implementation of algorithms is achieved by using programming languages. Some tools for expert systems development (such as Prolog) store algorithms and data in the same program. Authors think that development of applications in the field of intelligent systems should follow the same way as development of classic information systems, i.e. data and their structures in databases should be separated from algorithms written using program languages.

Following that objective, one can develop different applications using the same database, without the need to reinvest efforts to database organization. By using the *Node of Knowledge* database proposed in this paper, it is possible to develop different intelligent applications for verbalized textual knowledge.

The system presented in this paper is based on the *Formalized Node of Knowledge*. This formalism is defined in Jakupović, Pavlić, and Dovedan Han (2014) and Pavlić, Dovedan Han, and Jakupović (2015). The conceptual framework *Node of Knowledge* integrates: formalism for graphical representation (*Diagram Node of Knowledge*, DNOK), formalism for textual knowledge representation (*Formalized Node of Knowledge*, FNOK) and formalism for the questions formalization (*Formalized Node of Knowledge Question*, QFNOK). The *Node of Knowledge* method enables text expressed knowledge capturing based on identification of semantic relationships between words/phrases using wh-questions that clarify the role of the word/phrase in the relationship.

The results of this paper are essential for the continuation of the research of extending the existing relational databases with textual knowledge. In this paper, an algorithm for mapping enriched sentences of natural language into a relational database is given. Storing texts into databases is required just as much as storing data, but this has been difficult to achieve so far. The advantage of this approach has been achieved by applying the *Formalized Node of Knowledge* method, which enables storing and retrieving knowledge. The lack of this approach is that natural language sentences first need to be enriched with the described formalism.

Section 2 gives the overview of the related work. The research methodology is described in Section 3. The used *Node of Knowledge* method is presented, as well as the formalized and enriched sentence records (FNOK). A text database model is designed using the ER (Entity Relationship) method. For the same model, the relational database schema is presented. Linking of nodes from the formalized record to the dictionary is described. The idea of separating the sentences into a set of independent conditions that contains the initial sentence is described. A system for transformation of enriched formalized records into a relational database is proposed as a set of algorithms. The algorithms are represented in SQL programming language. Section 4 presents the results of the transformation algorithm application over a set of natural language sentences. Their transformation is performed into relational database tables. Section 5 gives the conclusion remarks.

2. Related work

Research of related work included exploration of the following areas: databases, information system development, artificial intelligence, and natural language processing. The approach described in this paper differs from other attempts of integration of artificial intelligence, databases and information systems. Other research dealt mostly with transforming natural language expressions (questions) to SQL query by using an intelligent interface (Hallett, 2006; Kanhe et al., 2015; Kovács, 2011; Li & Jagadish, 2014; Nihalani, 2010; Oracle, 2015). Our research aims to develop a framework that can translate any given text (any knowledge) into relational database.

In relational data mining field, research can be found which addresses the task of inducing models or patterns from multi-relational data (Peroperovvšek, Vavpetic, & Lavrac, 2012), as well as systems that represent text in natural language, such as product requirements, into formal specifications (Chen, Yao, Lin, Zeng, & Eberlein, 2007; Lami, Gnesi, Fabbrini, Fusani, & Trentanni, 2004). They sometimes use the formalism of conceptual graphs as a step that precedes the transformation into formal description (Fougères & Trigano, 1996). In contrast, the graphical representation of texts from natural language used with *Node of Knowledge* has a general objective to formalize any textual knowledge and translate it into the relational database model.

Similar problem of automatically integrating unstructured text into a multi-relational database is solved in Mansuri and Sarawagi (2006) by using statistical models for co-reference resolution and information extraction in a database setting. Sarawagi (2005) presents a system for using state-of-the-art statistical models for structure extraction and matching. The *Node of Knowledge* method does not deal with statistic approach, but proposes a completely different solution.

External knowledge, usually expressed as a domain ontology, needs to be linked to knowledge-based systems. The application of ontologies and text data repositories which store data in the form of an unstructured or semi-structured text and application of text mining with the aim of detecting relationships between data using natural language processing techniques and co-occurrence-based analysis is described in Abulaish and Dey (2007) and Yang, Bhowmick, and Madria (2005). These researches have their application in the narrow professional domain of medicine and biology, as opposed to the general approach adopted by *Node of Knowledge*.

Existing researches rarely use a dictionary as an external knowledge resource, but more often use WordNet as a dictionary or ontology. WordNet is a large lexical database in which nouns, verbs, adjectives and adverbs are grouped into sets of cognitive

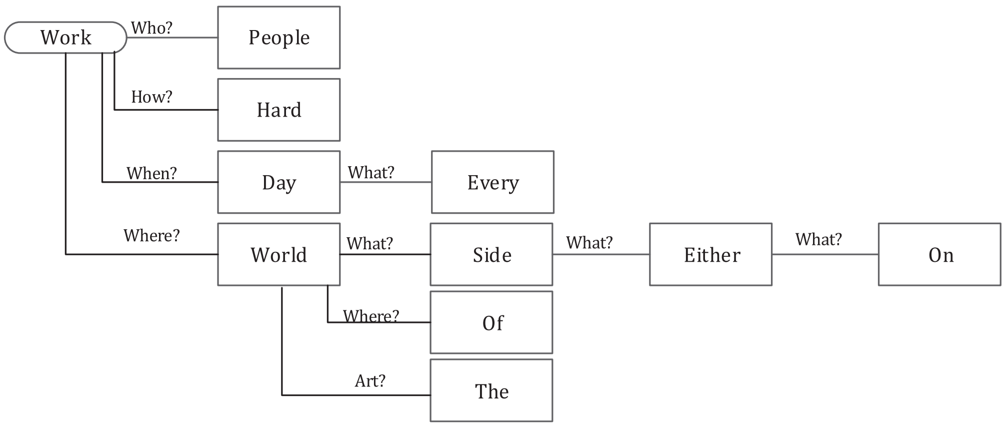

Fig. 2. Diagram Node of Knowledge for sentence „On either side of the world, people work hard every day.“.

Table 1
Basic concepts of the Node of Knowledge method.

Concept	Symbol
(Regular) node	People
Process node	Work
Link	Role 1

synonyms (synsets), each expressing a distinct concept (Fellbaum, 1998; Miller, 1995).

However, WordNet does not cover all languages, nor does it include all words of a language. In this paper we propose the application of a monolingual dictionary as external knowledge in enrichment of the text. The dictionary is implemented in the relational database used by *Node of Knowledge*. The implementation of the dictionary is not limited to WordNet, as in the case of Li and Shi (2010) which integrates WordNet as the base lexicon and ontology as the knowledge base of the semantic interpreter, but any monolingual dictionary of any language can be implemented into the relational database.

For the purpose of this research, comparison with several methods for knowledge representation was made (Jakupović, Pavlić, Meštrović, & Jovanovic, 2013): The Basic Conceptual Graphs (Chein & Mugnier, 2009), The Multi-Layered Extended Semantic Networks (Helbig, 2006), The Hierarchical Semantic Form (Stanojević & Vraneš, 2007), Semantic networks (Quillian, 1967), Frame theory (Minsky, 1974) and Tectogrammatical trees (Cinková, Hajič, Mikulová, Mladová, & Nedolužko, 2008; Sgall, Panevová, & Hajičová, 2004).

The Basic Conceptual Graphs (BCG) method recognizes two types of nodes (conceptual and relational) represented using different graphical symbols. The order of reading the model node by node is determined by numbers, which indicate the order of conceptual nodes in the relation. Reading of the *Node of Knowledge* model can start from any node, with the application of the corresponding link role (question). Relational node in BCG is process node in the *Node of Knowledge* method. However, *Node of Knowledge* enables connections between process nodes, which is important for representation of execution order of process nodes, their conditioning, etc.

The Multi-Layered Extended Semantic Networks (MULTINET) method is a complex method that does not graphically distinguish between node types as the *Node of Knowledge* method does, but it recognizes 29 classes of nodes. It defines 7 attributes for each node and 89 connection types. In *Node of Knowledge*, users can freely add classes based on their needs. Although with MULTINET is possible to express all statements that can be created in *Node of Knowledge*, MULTINET is a very complex method with many options and it is primarily used to model aspects of natural languages.

The Hierarchical Semantic Form (HSF) method consists of two concepts: a group and a link, represented with empty and filled circle respectively. They are connected with arrows, which indicate the proper reading direction. It uses groups to represent aggregation of knowledge and adds semantics to a certain sequence of terms by connecting them to a group concept that contains semantics descriptions. In the *Node of Knowledge* method, semantics is added to a sequence of terms by an adequate network on a higher level of abstraction.

In Quillian's semantic networks, each word is stored along with the configuration of the pointer to other words, and this configuration represents the meaning of the word. The *Node of Knowledge* method also has pointers to other words but these pointers have a semantic identifier, which indicates the reason for the existence of the pointer (or link). Semantic identifier is represented by a question.

In Minsky's frame theory, the frame is a data structure (which is stored in slots) that stores information such as: Facts or Data, Procedures, Default Values and Other Frames or Subframes. This type of representation results in the loss of the semantic relationship between words that exists in the sentence, which does not occur with the *Node of Knowledge*.

In Tectogrammatical trees each sentence is represented as a projective dependency tree with nodes and edges. Each node has a semantic label called functor, which renders the semantic relation of the given node to its parent node. There are 8 node types in Tectogrammatical trees. On the top of the hierarchy there is a technical root node which includes the identifier of the sentence in the treebank. In the Node of Knowledge there are three node types and on the top of the hierarchy there is a process node. All other nodes depend on it.

Tectogrammatical trees use Wordnet, while *Node of Knowledge* uses adapted MULTEXT-East (Erjavec, 2010). Semantic relation between nodes in the Tectogramatical trees is set by using 70 functors, while in *Node of Knowledge* it is establish by using wh-questions ("who?", "what?", "when?", "how?", "where?", "what?", "how much?", "how many?" etc.). Semantic identifiers (wh-questions) are determined automatically using MULTEXT-East (Jakupović et al., 2014). In some situations semantic identifiers cannot be determined automatically, and then manual assistance is needed.

The biggest similarity between Tectogramatical trees and *Node of Knowledge* is in hierarchical representation of a sentence. However, *Node of Knowledge* builds sentence hierarchy solely from the

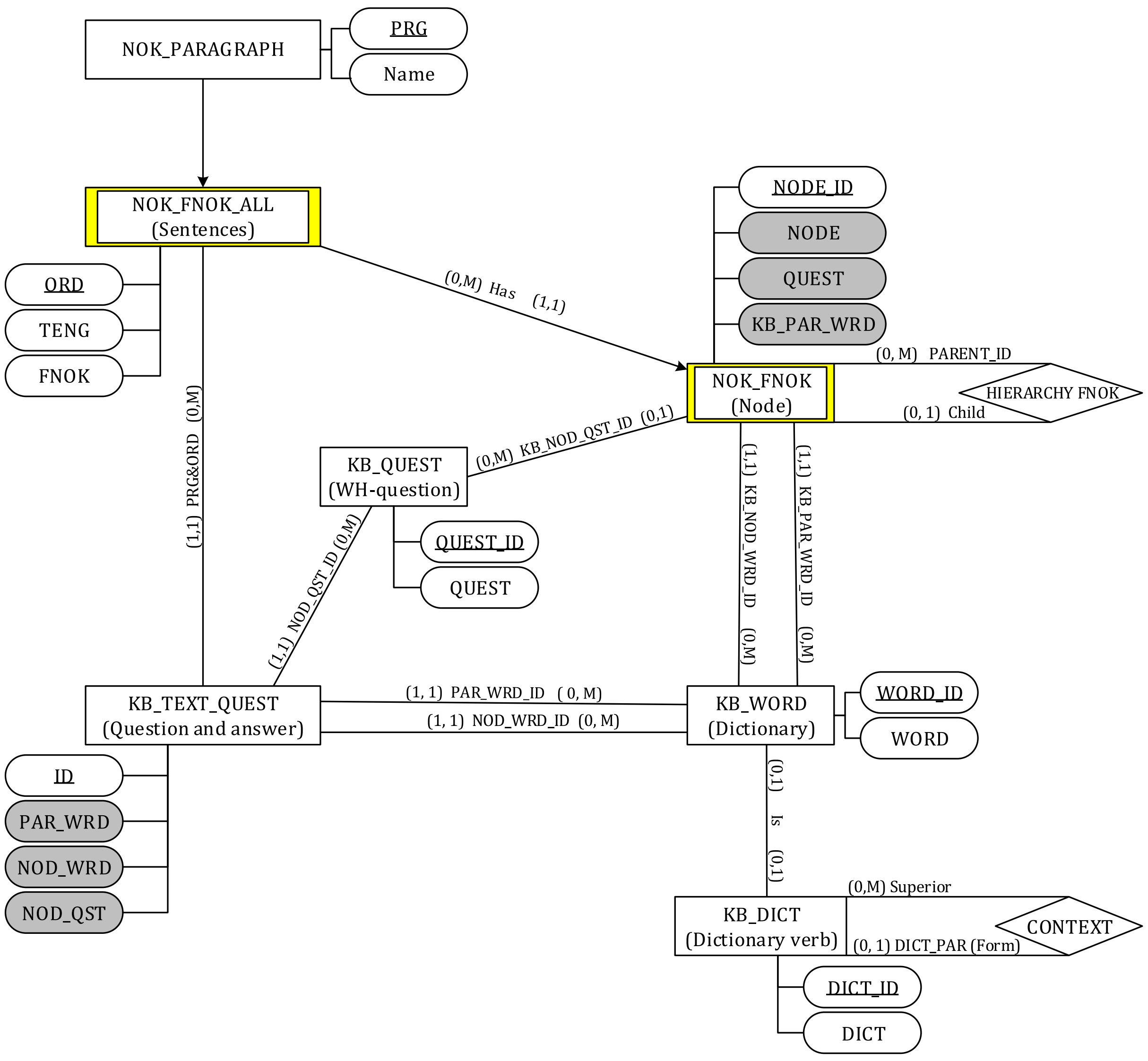

Fig. 3. Meta model for the structure and content of sentences in FNOK record.

words in a sentence, while Tectogrammatical trees build broader and higher hierarchy of nodes. In Tectogrammatical trees, it is possible to use complex sentences and different word types. The *Node of Knowledge* method is still developing its approach to complex sentences and some word types, such as conjunctions and pronouns.

Comparison of the *Node of Knowledge* method with other methods recognized several characteristics of the *Node of Knowledge* method (Jakupović et al., 2013): simplicity, expressiveness and simplicity of reading.

Simplicity – It has two basic elements: a node (regular, context and process) and a link (regular and context). They are used together with a role (the question) in reading the modelled knowledge.

Expressiveness – It enables to represent knowledge on various levels of abstraction. In lower levels, it uses regular and process nodes and in higher levels, it uses context and process nodes. The relationship between knowledge on lower and higher levels of abstraction is achieved using context links.

Simplicity of reading – It is possible to start reading knowledge from any node using link roles. The same knowledge from the same Node of Knowledge model is reachable in different ways.

3. Research methodology

This paper is limited to a "one-way" *Node of Knowledge* method in which two nodes are interconnected by one single role (question) which indicates the relationship between the two nodes. For the one-way *Node of Knowledge* method, a formalized (FNOK) record of sentences was developed (Jakupović et al., 2014). This paper uses this research results and applies the created method and its transformation of natural language sentences into an enriched and formalized record. This method is described in Section 3.1.

Process model of the system for the transformation of formalized sentences into the relational database is shown in Fig. 1.

The idea of this approach is that the sentences expressed in *Formalized Node of Knowledge* record are transformed into a relational database. The preparation of textual sentences in the formalized record is not a subject described in this paper, but is described in detail in Jakupović et al. (2014). System modeling was performed in accordance with the specialized methodology for the development of information systems - MIRIS (Pavlić, 2011; Pavlić, Jakupović, & Čandrlić, 2014). The relational database data model is created according to the content and struc-

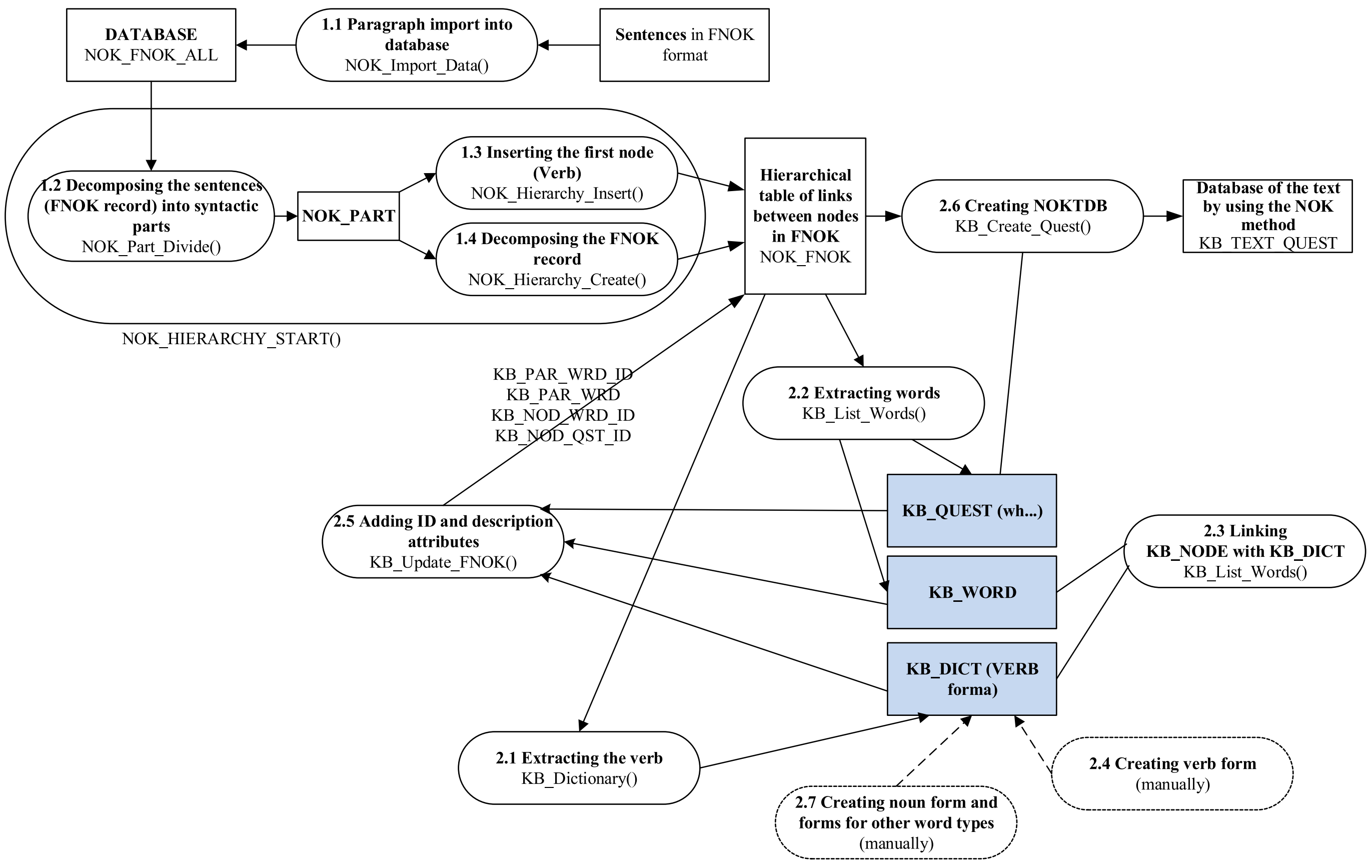

Fig. 4. Representation of transformation of formalized text into the relational database.

ture of *Formalized Node of Knowledge*. Let's describe the proposed system in more detail.

The transformation system uses a formalized sentence record (FNOK record) called "Sentences in FNOK Record" as input. These sentences are generated by applying a procedure for transferring text to the formalized record. Text paragraphs written in sentence form become paragraphs of the *Formalized Node of Knowledge* record. The procedure for transferring text in formalized record is described in detail in Jakupović et al. (2014), and is not described here.

The system for the transformation of formalized sentences into a relational database consists of two processes. The first process of the system transforms the input formalized records into the hierarchical structure of relationships between words in the sentence. The second process enriches the sentences and their parts (words) by linking them to the dictionary. This is how the *Formalized Node of Knowledge* database is created.

Although simple, the *Node of Knowledge* method is semantically rich. Its semantical richness, together with implemented algorithms for question answering, gave requested results during question answering, which is one of the biggest strengths of this research. In addition to that, the proposed method is adaptable to different languages, without the need to change algorithms (preliminary ongoing research is done using sentences in four European languages). Weaknesses of the proposed method include the need for text enrichment, i.e. text cannot be used in its original form, but needs to be transformed into formal record prescribed by the *Node of Knowledge* method. For this process, it uses MULTEXT-East. Mechanisms for text enrichment are not yet fully developed, and some manual interventions are needed, which is a methodological drawback. In some situations, it was necessary to rely on authors' judgement. Possibilities for application of the *Node of Knowledge* method are numerous. However, although the method is elaborated, its application in any field of industry depend on existence of the proper *Node of Knowledge Database* related to the requested field.

Limitation of this research is in rather small number of the analyzed sentences. The research is limited to simple sentences that use these types of words: nouns, verbs, pronouns, adjectives, adverbs, prepositions and numbers. The following types of words are missing: conjunctions, interjections, two types of pronouns (relative and interrogative pronouns). Complex sentences are not included in the research. Sentences that have homonyms and synonyms and the associated problems with dictionary form are not analyzed.

3.1. The Node of Knowledge method (NOK)

Node of Knowledge is the conceptual framework for knowledge-based system development (Pavlić et al., 2015) which consists of (Ašenbrener Katić, Čandrlić, & Pavlić, 2017):

- Node of Knowledge method (NOK)
- Formalism for graphical representation (called Diagram Node of Knowledge (DNOK))
- Formalism for textual knowledge representation (called Formalized Node of Knowledge (FNOK)), and
- Formalism for question representation (called Formalized Node of Knowledge Question (QFNOK)).

The *Node of Knowledge* method (as well as other graphical methods for knowledge representation (Chein & Mugnier, 2009;Helbig, 2006;Stanojević & Vraneš, 2007; Tolle, 2000) represents knowledge in the form of a graph consisting of nodes and links (Ašenbrener Katić, Pavlić, & Čandrlić, 2015). Nodes represent

Table 2
Algorithm for procedure NOK_Import_Data().

```
------------------------------------------------------------------------------------------
Procedure   NOK_Import_Data (
                                PrgId IN Number      -- Paragraph of Sentences
                             )
Is
Begin
--
-->>> PROCESS 1.1
-- Cancel old Data of the same Paragraph
   Delete
   From     NOK_FNOK_ALL
   Where    PRG = PrgId       -- Paragraph to delete
   ;
-- Insert new Data of the same Paragraph
   INSERT INTO NOK_FNOK_ALL
  ( PRG , ORD , TENG , FNOK
  )
SELECT PrgId -- Paragraph to insert
  ,
  ROWNUM +
  (SELECT COUNT(*) FROM NOK_FNOK_ALL WHERE PRG = PrgId
  ) -- Order Number between same Paragraph
  ,
  TENG ,
  FNOK
FROM NOK_FNOK_IMP ;
End;
---------------------------------------------------------------------------------------------
```

concepts, and links are used to represent relationships between concepts. It is based on the research and analysis of sentences, words and their meanings, and the order of linking words into complex terms. The aim of the method is to enable the structuring of the knowledge network contained in textual form in any natural language (Pavlić, Meštrović, & Jakupović, 2013).

In addition to the one-way variant of the *Node of Knowledge* method, which is the basis for this research, there is also a two-way variant of the same method. The one-way *Node of Knowledge* method uses a single-role link and represents a single mapping. With the two-way *Node of Knowledge* method, the link has two roles and represents two mappings (two links, from the first node to the other, and vice versa). It is semantically richer than the one-way method, which also means it is more complex.

3.1.1. One-way Node of Knowledge method

This paper uses the one-way *Node of Knowledge* method in which the link between the nodes has only one role and connects the two nodes so that it moves from one to the other. The link between the two nodes is accomplished by asking a question related to the first node, and the second node is the answer to the question. Conversely, the question we set to the second node in order to get the first node as the answer is ignored.

The basic concepts of the *Node of Knowledge* method (Pavlić, Jakupović, & Meštrović, 2013) used in this paper are shown in Table 1.

Using the concepts of the *Node of Knowledge* method, a diagram called *Diagram Node of Knowledge* (DNOK) is generated. The verb in a sentence of the natural language corresponds to the process node, and the other words correspond to nodes. Let us present the concepts of the *Node of Knowledge* method by using an example. *Diagram Node of Knowledge* for the sentence "On either side of the world, people work hard every day." is shown in Fig. 2.

Diagram Node of Knowledge for this sentence consists of 10 regular nodes: "People", "Hard", "Day", „Every", „World", „The", „Of", „Side", „Either" and "On" and the process node "Work".

Nodes "People", „Hard", „Day" and "World" are linked to the process node "Work" (and the depend on it hierarchically). Besides, node "Day" is linked to node "Every", while node "World" is linked to nodes "The", „Of" and „Side". Further on, node „Side" is linked to node „Either" and it is linked to node „On". On every link between the nodes there is a question (wh-question) which defines the role of the link between the nodes and it is relevant for storing and querying knowledge contained in the diagram.

For example, the node "People" and the process node "Work" are connected by a link with one role "Who?". If we ask "Who works?" the answer is "People". The question we would ask related to "People" and get "work" as the answer is not indicated and this side of interpreting diagram is neglected.

Similarly, two regular nodes („Day" and „Every") are related with a link with role („What?"). If we ask "What day?", the answer is "Every".

A detailed description of the transformation algorithm and the corresponding software product in Python for the selected sentences is given in Jakupović et al. (2014) and Pavlić et al. (2015).

The semantic relationship between the adjective, verb and other word types by using the *Node of Knowledge* method is shown in Pavlić, Dovedan Han, Jakupović, Ašenbrener Katić, and Čandrlić (2017) and Ašenbrener Katić, Čandrlić, and Pavlić (2018).

3.1.2. Formalized Node of Knowledge

In addition to the graphical representation of textual knowledge, the formalism for textual knowledge representation (*Formalized Node of Knowledge,* FNOK) was developed, whereby sentences with semantic relationships between words (using wh-questions) can be represented textually. Wh-questions are a question type that use wh-words: who, whom, whose, what, which, why, when, where, how, how often, how long, etc. (Biber, Johansson, Leech, Conrad, & Finegan, 1999; Brinton & Brinton, 2010; Quirk, Greenbaum, Leech, & Svartvik, 1985).

Generally: Each word in a natural language sentence corresponds to one node in the formalized sentence record. The verb in the natural language sentence corresponds to the process node in the formalized record.

Formalized sentence record and its corresponding diagram are hierarchically organized. The process node is at the highest level of hierarchy. All the other nodes are in the hierarchy below the process node. The open bracket "(" and closed bracket ")" show

Table 3
Algorithm for function NOK_Hierarchy_Start().

```
-----------------------------------------------------------------------------------
Function    NOK_Hierarchy_Start (
                                   PrgId IN Number      -- Paragraph of Sentences
                                )
Return Number Is
-----------------------------------------------------------------------------------
--Prepare Data and Start Transformation of a FNOK record into Hierarchical order.
-----------------------------------------------------------------------------------
   NOKID    Number;        -- NOK IDENTIFIER (used as SEQUENCE)
   Ret      Number;        -- Return Value
   NumS     Number;        -- Number of Sentences
   OrdNo    Number;        -- Sentence Order Number
   FNOK     Varchar2(400); -- FNOK string to divide into syntactical Parts
   NumP     Number;        -- Number of syntactical Parts
   Pos      Number;        -- Actual Position
   NodId    Number;        -- Node   Identifier
   ParId    Number;        -- Parent Identifier
   Part     Varchar2(99);  -- Single Syntactical Part
   Quest    Varchar2(99);  -- Question
--
Begin
-- Prepare tables and data
--
-- cancel all records from hierarchical table NOK_FNOK belonging to requested Paragraph
   Delete   From  NOK_FNOK    Where PRG = PrgId;

-- select Number of Sentences at requested Paragraph
   Select count(*)  Into  NumS  From  NOK_FNOK_ALL  Where PRG = PrgId;
--
-- Transforming all FNOK records from the Paragraph in their Hierarchical structure
--
   For OrdNo in 1 .. NumS Loop    -- MAIN LOOP - FOR EACH FNOK
--
-- get a single FNOK record
   Select   FNOK           Into  FNOK  From  NOK_FNOK_ALL   Where PRG = PrgId And ORD =
OrdNo;

--
-->>> PROCESS 1.2
-- Divide a FNOK record in his syntactical Parts
      NumP := NOK_Part_Divide(FNOK);
      If NumP = 0 Then
         Return 0;
      End If;
--
-->>> PROCESS 1.3
-- Prepare a First Node
      NOKID := 0;                                 -- NOK IDENTIFIER initialization
      ParId := NOKID;                             -- Parent Identifier for first Node
      NOKID := NOKID + 1;                         -- NOK IDENTIFIER increment
      NodId := NOKID;                             -- Node Identifier for first Node
      Quest := '_';                               -- first Question is empty

-- First syntactical Part must be a VERB (but we are not verifying a syntax)
      Pos   := 1;                                 -- first Position between syntactical
Parts of FNOK record
      Part  := NOK_Part_Read(Pos);                -- read a First syntactical Part
      Ret   := NOK_Hierarchy_Insert(PrgId, OrdNo, ParId, NodId, Part, Quest);   --INSERT
THIS NODE
--
-->>> PROCESS 1.4
-- Look for Second syntactical Part that must be '('
    If Pos < NumP Then
         Pos  := Pos + 1;
         Part := NOK_Part_Read(Pos);              -- read a Second syntactical Part
         If Part = '(' Then                       -- if next syntactical Part is equal
to '(' then can continue on the next level of hierarchy  -- Go to the next level of
hierarchy
         Pos := NOK_Hierarchy_Create(NOKID, PrgId, OrdNo, NodId, NumP, Pos);-- go to the
next level of hierarchy
         End If;
      End If;
   End Loop;                 -- END MAIN LOOP - FOR EACH FNOK
--
   Return NOKID;             -- return Number of Transformed Records
End;
-----------------------------------------------------------------------------------------
```

hierarchy levels. In front of each node in the hierarchy comes the question word (wh-question/question/role) with which the node at the lower level of the hierarchy is linked to a node at a higher level, or a node at a lower level hierarchy is the answer to the question related to the node at a higher level of hierarchy.

A question is marked by quotation marks and the question mark ("?"). For verbs, we can set different questions, for example "Who?", "What?", "When?", "How?", "Where?" and similar, which will open places for other word types hierarchically dependent on the verb. For example, adjectives correspond to the question "what kind?", "Whose?", "Which?". The numbers correspond to the question "how much?" And "which one?". The wh-question mark for the article is "art?".

Each node can have its own hierarchy. At the same hierarchical level, the order of listing nodes is arbitrary, and nodes and their related questions are separated by commas.

In the hierarchy, the verb (or process node) has the advantage. It is followed by nouns (if there are more at the same level of hierarchy, the order is arbitrary), then the adjectives, followed by numbers, and finally prepositions. Adverbs are at the same level of hierarchy as nouns.

Let us present the concepts of the *Formalized Node of Knowledge* method by using an example. For sentence „On either side of the world people work hard every day.“ formalized record looks like this:

- work
    - (
    - "who?" people,
    - "how?" hard,
    - "when?" day
        - (
        - "what?" every
        -),
    - "where?" world
        - (
        - "what?" side
            - (
            - "what?" either
                - (
                - "what?" on
                -)
            -),
        - "where?" of,
        - "art?" the
        -)
    -)

The record of the same sentence in an uninterrupted line is:

work ("who?" people, "how?" hard, "when?" day ("what?" every), "where?" world ("what?" side ("what?" either ("what?" on)),"where?" of, "art?" the))

3.1.3. Meta model of sentences in the Formalized Node of Knowledge record

To answer the research question "how to enter formalized records of sentences into a relational database?", it is necessary to analyze the data contents of formalized records and formalism itself, and find the appropriate meta model. Based on the data and meta model of the *Formalized Node of Knowledge*, it is necessary to define a relational database schema that will contain sentences represented in the formalized records and the algorithms for the transformation of the initial record of sentences into the related data in the database.

For the meta modeling of data, the entity relationship method (Pavlić, 2011) was used, the result of which is the ER diagram shown in Fig. 3. Based on the ER diagram, through the translation process (Pavlić, 2011) of concepts of a semantically rich model into a relational data model, the relational schema of the database is defined and shown in Section 3.3.1. Based on the database schema, a database is defined and presented in Section 4. Based on the ER diagram and a more detailed analysis of the transformation process, the required algorithms for transformation of formalized records into the relational database are defined, as shown in Section 3.3.2.

Formalized Node of Knowledge allows the representation of natural language sentences into formalized records enriched with symbols and question words. By analyzing the rules of recording sentences, we come to the model of the *Formalized Node of Knowledge* (meta model). The meta model is shown in Fig. 3.

Rectangles on the meta model represent entity types that will be translated and implemented as tables in the relational model. Attributes are shown using ovals. Lines with or without arrows represent types of links between entity types, which will represent related tables using foreign keys. Because of these links between the tables, attributes (foreign keys) are added to the database schema and there are more of them than on ER diagram. The foreign keys are underlined using a dashed line. Key attributes are underlined with a full line. The hierarchy among entities is represented by the rhombus symbol and links entities of the same type of entity (superior - subordinate) with themselves.

Let us describe the content of the proposed meta model in more detail. The texts of natural language sentences can be grouped in different shapes, and in this paper, we focus only on the group of sentences we call a paragraph. The paragraph list is contained in the entity type `Paragraph`. Each paragraph has a key attribute `PRG` and an attribute - short paragraph description (`NAME`).

Each paragraph has at least one sentence, and it can have many sentences. The sentence list is displayed by a weak entity type (shown by the double rectangle) `NOK_FNOK_ALL` (Sentences). One sentence always belongs to one and only one Paragraph. Entity Type `NOK_FNOK_ALL` (Sentences) has three attributes: ordinal number of the sentence (`ORD`), text of the sentence in natural language (`TENG`, abbreviated from Text in ENGlish) and formalized sentence record (`FNOK`).

Each formalized record `NOK_FNOK_ALL` (`Sentences`) consists of words (nodes), among which there is a hierarchical relationship. The nodes are displayed by entity type Node (`NOK_FNOK (Node)`). Each node belongs to one and only one sentence, or formalized record. Each node has its own identifier (`NODE_ID`) as the ordinal number of occurrence within the sentence, and attribute - node name (`NODE`), representing the node Node from the starting formalized record. A single entry may have multiple nodes, or none (if the process of transforming the record into nodes has not been processed yet). A node in `NOK_FNOK` (`Node`) always belongs to one and only one sentence from `NOK_FNOK_ALL` (`Sentences`).

The entity type `NOK_FNOK` (`Node`) has no more attributes. Other attributes of the entity type `NOK_FNOK` (`Node`) will be derived from types of links to other entity types in the ER diagram transformation process in the relational database schema. Thus, in the process of translating the data model into the relational scheme, the key of the starting sentence (`PRG` and `ORD`) will be lowered to each row of the table from which the node originated.

The entity type `NOK_FNOK` (`Node`) is linked to itself by a recursive relationship type, the `HIERARCHY FNOK`, shown by a rhombus. The *Formalized Node of Knowledge* record has a hierarchical structure. At the beginning of the record there is a verb (process node) that is superior to all other nodes. Within the formalized record, for each node having hierarchically subordinate nodes, a round parenthesis is used and all its subordinate nodes are listed within. It is possible to achieve an unlimited number of levels of hierarchy in this way. For each node it is valid that it can have one parent node (PARENT_ID) and more subordinate nodes (Child). Hence, the recursive relationship HIERARCHY FNOK has two roles: Parent and Child. One node Parent can have more subordinate nodes (for example, the first node) or minimum none subordinate node (if it is the last without division, for example, an adjective or an adverb). One node Child belongs to maximum one parent and minimum none parent (in the case of the first node there are no parents).

To all nodes in the formalized record, except for the first process node in the sentence, belongs one question word (role, question). For example, for the node "they", there is the question "who?" in the sentence “They aged”, i.e. in the formalized record *aged ("who?" they)*. Therefore, the attribute for “the name of the set question” - " QUEST " is added to the model as the attribute of the entity type NOK_FNOK (Node). Only the first node in the sentence has the value of this attribute set to "null value", as indicated in the QUEST column using the underscore symbol "_". The question word (QUEST) represents an embedded question added during the process of enrichment.

In Fig. 3. the attributes NODE, QUEST and KB_PAR_WRD, and PAR_WRD, NOD_WRD, and NOK_FNOK (NODE) are marked with gray colour. They are not entity types of NOK_FNOK (NODE) i.e. KB_TEXT_QUEST, but we have added them to the model for easier tracking of table content.

3.2. *Linking nodes from the* Formalized Node of Knowledge *record with the dictionary*

One of the important steps in machine comprehension of a text is to associate words in a sentence with words in the dictionary. Each word in the dictionary has a language-level identifier and a unique meaning. Assigning an identifier from the dictionary to the node in *Formalized Node of Knowledge* record is important for assigning the meaning of particular nodes. The dictionary implementation is possible in many ways. One of the ways is to use natural-language dictionaries adapted to computer data processing. One such dictionary for English language is the morphosynthactic lexicon of the English language MULTEXT-East (Erjavec, 2010), used to define the translation of English language sentences into the *Formalized Node of Knowledge* record (Jakupović et al., 2014). Classic dictionaries are rich in a set of attributes (properties). For the purposes of this research, we only need the word from the dictionary without any additional properties. In this research we have developed our own dictionary containing words that appear in text sources.

The system model is adaptable and allows us to automatically expand the dictionary with words that are not in the dictionary. It is possible that, no matter how large the initial set of words the selected dictionary contains, we come up with a sentence containing a new word, a new term, a new name of a person, or other. That is why it is necessary to enable expansion of the dictionary with new words.

In this paper we will neglect the problems of natural language dictionaries such as homonyms. If this problem occurs, it will be handled manually by the user. If a homonymous word (e.g. count, key, kind, ride) appears in the *Formalized Node of Knowledge* record that has multiple different occurrences in the dictionary, and for each meaning a special identifier, the problem of linking will be resolved manually. Part of the problem can be solved according to the type of words if homonyms are different types of words. If they are not, then a contextual analysis is required, and this does not fall within the scope of this paper.

Three types of entities (tables) are suggested to solve the problem of storing data from the dictionary. There are also more general solutions in the case of extending the system with new demands and new scientific issues such as: homonyms, synonyms, linking two or more sentences, the relationship between abstract and specific concepts, storing questions and answers to the questions.

Dictionary consists of three entity types, namely: KB_QUEST (Wh-question), KB_WORD (Dictionary) and KB_DICT (Dictionary verb).

The first entity type is a simple wh-questions list table (KB_QUEST) that enriches the formalized record of sentences. In English language there are three fundamental question types: polar questions, wh-questions, and alternative questions (Biber et al., 1999; Jespersen, 1964; Quirk et al., 1985). Wh-question is a question type that uses wh-words (who, whom, whose, what, which, why, when, where, how etc.). The KB_QUEST table has two attributes, the identifier (QUEST_ID), and the question itself (QUEST). The table has a small number of n-tuples (a few tenths of occurrences), and its large growth is not expected.

The second entity type is the Word Dictionary (KB_WORD (Dictionary)) and it represents an adaptable dictionary of all the words that appear in the text. Its attributes are: the word identifier (WORD_ID) and the word itself (WORD). Alternatively, you can manually add a description of the meaning of the word, word type, and other attributes, or retrieve that data together with the words from the existing dictionaries. For the purposes of this research, it was not necessary.

The third entity type belonging to the dictionary is the Dictionary of verbs and other open word classes (KB_DICT (Dictionary verb)). This entity type contains only verbs and nouns for now. KB_DICT (Dictionary verb) is linked via a recursive relationship CONTEXT to itself. Unlike the hierarchy of sentence parts in relationship type HIERARCHY FNOK, this recursive relationship models the hierarchy of the context relationship of a semantically richer word and individual subterms that this word encompasses.

If we observe the abstract verb *to be*, it has more concrete occurrences (*am, is, was, were, were, been, being*) depending on the person and tense, which are given the role Form. The model is general for all verbs and other open word classes. It is possible to use the same contextual linking of words in the dictionary and apply it to other types of open class words. For example, the noun *car* can have more occurrences, like *car* and *cars*, i.e. changes in the number (singular and plural). The proposed model provides multiple levels of hierarchy. One word as an abstract superordinate word can have at least one to maximum many concrete forms. One specific word can be the form of at least none word (the highest in the hierarchy) or maximum one abstract word.

The KB_DICT (Dictionary verb) and KB_WORD (Dictionary) are connected by the link type *Is*, so that in the upper limits, one word matches only one word by mapping in both directions. In the lower limits, zeros (0) are suggested, because all words from the dictionary are not verbs or nouns or have no contextual definition, while vice versa, it is possible to update the hierarchical relationships in the system, even before the dictionary update.

With such a defined dictionary in three types of entities, nodes must be linked in entity type NOK_FNOK (Node). A special algorithm will go through all *Formalized Node of Knowledge* records of a paragraph and link each node in NOK_FNOK (Node) with KB_QUEST using link KB_NOD_QST_ID. The leading node does not have to include any question, and all other nodes in the sentence contain at most one question from KB_QUEST. One question in KB_QUEST does not have to be in any node and can occur in many nodes. The node is also associated with the KB_WORD entity type. There are two "identification" links: KB_NOD_WRD_ID and KB_PAR_WRD_ID. Through this links, each node in the sentence gets a unique identification and its atomic interpretation described by the semantics of that term according to the dictionary. The identification link KB_NOD_WRD_ID assigns an identifier to a certain node in a sentence and word in the dictionary. The identification link KB_PAR_WRD_ID assigns an identifier for the parent word to a particular node in the sentence.

3.3. Decomposing sentence parts in Formalized record into a question–answer set

The transformation of *Formalized Node of Knowledge* record into the relational database ends with the algorithm that arranges individual nodes from NOK_FNOK (Node) and their recursive links HIERARCHY FNOK into a set of questions and answers in the entity type KB_TEXT_QUEST (Question and answer).

An entity from KB_TEXT_QUEST represents the aggregation of three types of entities, namely: words from KB_WORD (Dictionary) through link PAR_WRD_ID (node looking for an answer, node "parent"), questions from KB_QUEST (Wh-question) through link NOD_QST_ID that node "Parent" sets, and words from KB_WORD (Dictionary) through link NOD_WRD_ID (node which is the answer to the set question, node "child").

Accordingly, each formalized sentence is broken into a maximum set of atomic questions and answers to those questions contained in that sentence. These questions and answers represent a set of independent conditions (statements, n-tuples, rules) that all contain semantics of the initial sentence.

The entity type KB_TEXT_QUEST has its own key attribute (ID). Each entity in KB_TEXT_QUEST is associated with one and only one “parent” node in the NOK_FNOK (Node) which will ask one and only one question from KB_QUEST and will receive one and only one answer from NOK_FNOK (Node). This link is not implemented directly, but is implemented through KB_WORD (Dictionary).

Let us describe links of a particular KB_TEXT_QUEST (Question and answer) with the surrounding entity types in more detail.

One question from KB_QUEST can be used in many entities in KB_TEXT_QUEST. One entity from KB_TEXT_QUEST always takes one and only one question from KB_QUEST.

Entity type KB_TEXT_QUEST has as many occurrences as there are answers to the question in the formalized record. One entity from KB_TEXT_QUEST always has one and only one node from NOK_FNOK (Node) in the role of a parent and one in the role of the answer to the question. A node in the role of a parent requests an answer to the asked question. The leading nodes at the highest level of the hierarchy from NOK_FNOK (Node) are not mapped into their occurrences in KB_TEXT_QUEST because they are not an answer to any question contained in the formalized record. Other nodes are mapped into maximum one entity.

Since a unique key (QUEST_ID) from KB_QUEST for each question is familiar, we also want to have a unique identifier (WORD_ID) for each node from KB_WORD. For this, we use links PAR_WRD_ID and NOD_WRD_ID. This procedure replaces the ordinal number of node (NODE_ID) in NOK_FNOK (Node) with a general node identifier according to KB_WORD (Dictionary).

Link type PAR_WRD_ID is a mapping which enables that one word from KB_WORD does not have to be used in any sentence, and can be used in many entities in KB_TEXT_QUEST. One question and answer from KB_TEXT_QUEST is always mapped to one and only one word in the dictionary that represents the "parent" node asking the question. The second link NOD_WRD_ID is the same as the PAR_WRD_ID in number, and represents the connection between the question and the answer, i.e. the word in KB_WORD (Dictionary), which is the answer to that question.

Entity type KB_TEXT_QUEST represents a set of entities "question-answer", where the term question is not just a question word but a linked node that asks the question and the question word.

We have proposed a model that will transform hierarchically complex *Formalized Node of Knowledge* records into a set of independent statements that do not have direct dependencies. The semantics of the integral sentence is decomposed into a set of statements, n-tuples of the relational database, data over which further research and development of the question answering system can be carried out. The goal is accomplished, to transform the sentences into a relational database through the described formalism.

The transformation of natural language sentences into *Formalized Node of Knowledge* record is very important (Jakupović et al., 2014). Without the procedure of enrichment of sentences with questions, we could not bring a relational scheme to an arranged triplet: who is asking a question, questions and answers. There would be no links between words, and such a base could not be used to answer the questions.

3.3.1. Relational database schema

Using the rules for translation of the ER diagram from Fig. 3 into the relational database schema (Pavlić, 2011), a relational schema was obtained, consisting of seven tables (relations), shown below. Each type of entity (rectangles on ER diagram) becomes a table in the schema, where the name of the entity type corresponds to the table name. The table key is an attribute or set of attributes underlined by a full line. Foreign keys are underlined with dashed lines and italicized. The data content of these tables is shown in Section 4.

NOK_PARAGRAPH (PRG, NAME)
NOK_FNOK_ALL (PRG, ORD, TENG, FNOK)
NOK_FNOK (PRG, ORD, NODE_ID, NODE, QUEST, *PARENT_ID, KB_PAR_WRD_ID,* KB_PAR_WRD, *KB_NOD_WRD_ID, KB_NOD_QST_ID*)
KB_QUEST (QUEST_ID, QUEST)
KB_DICT (DICT_ID, DICT, *DICT_PAR*)
KB_WORD (WORD_ID, WORD, *DICT_ID*)
KB_TEXT_QUEST (ID, PAR_WRD, NOD_WRD, NOD_QST, *PRG, ORD, PAR_WRD_ID, NOD_WRD_ID, NOD_QST_ID*)

3.3.2. Algorithm for transformation of Formalized Node of Knowledge records into relational database

In this section, the description of the transformation algorithm describing the sentences initially represented in the Formalized Node of Knowledge record through a series of transformation steps, are entered into the relational database of the text according to the Node of Knowledge method. Fig. 4 gives a more elaborate and described processes, which were marked with number 1 and number 2 in the Fig. 1. Their sub-processes are marked with 1.1–1.4 and 2.1–2.7 in Fig. 4.

The initial text record (sentences) is given in *Formalized Node of Knowledge* format. Their import and entry of paragraphs into the Oracle database is done automatically, in the temporary table NOK_FNOK_ALL, and the sentences belonging to a paragraph are entered one by one. For example, sentence *My friend Sean drives a car.* is formally represented as: drives *("who?" Sean ("who?" friend ("whose?" my)), "what?" car ("art?" a)).* The procedure, which enters data from the table NOK_FNOK_IMP (natural language sentences and their corresponding formalized records) into the table NOK_FNOK_ALL, is shown in Table 2.

By transforming the sentence into a hierarchical record, using the function NOK_Hierarchy_Start(), a hierarchy is formed between the words in the sentence. The algorithm of this function is shown in Table 3.

Table 4
Algorithm for function NOK_Part_Divide().

```
-----------------------------------------------------------------------------------------
Function    NOK_Part_Divide (
                 FNOK  IN Varchar2-- FNOK string to divide into syntactical Parts
                            )
Return   Number Is
-----------------------------------------------------------------------------------------
-- Divide a FNOK string into his syntactical Parts
-----------------------------------------------------------------------------------------
--
   Chr1     Varchar (02);  -- Single Character
   Chr2     Varchar (02);  -- Single Character
   NumP     Number;        -- Number of syntactical Parts
   Part     Varchar2(99);  -- Single Syntactical Part
--
Begin
   Delete From NOK_PART ;                       -- cancel a temporary table NOK_PART
--
   NumP  := 0;
   Part  := NULL;
--
   For Pos In 1 .. length(FNOK) Loop            -- for each character in the FNOK string
--
      Chr1 := substr(FNOK, Pos, 1);             -- get a character
      If Chr1 In ('&','_') Then                 -- characters '&' and '_' can be a part of
text, not only a syntactical delimiters
         Chr2 := substr(FNOK, Pos+1, 1);        -- we must check the successive character
      End If;
--
      If  Chr1 In ('(',')',',','"')             -- syntactical divisors are '('  ')'  ','  '"'
      Or (Chr1 In ('&','_') And Chr2  =  ' ')-- but also '&' '_' only if the successive
character is space
      Then                                      -- if character is one of the syntactical
divisors (punctuation)
         NumP := NOK_Part_Write(NumP, Part); -- write a precedent Part
         NumP := NOK_Part_Write(NumP, Chr1); -- write the actual syntactical divisor
         Part := NULL;                          -- prepare a buffer for new syntactical Part
      Else                                      -- if character is one normal text character
         Part := Part || Chr1;                  -- add the character to the syntactical Part
      End If;
--
   End Loop;
--
  Return NumP;                                  -- Return Number of syntactical Parts
End;
-----------------------------------------------------------------------------------------
```

Function `NOK_Hierarchy_Start()`consists of three parts: decomposing the sentence into syntactic parts with the function `NOK_Part_Divide()` (Table 4.), entering the first node with the function `NOK_Hierarchy_Insert()` (Table 7.) which is a verb (`VERB`) and entering each remaining node using recursive procedure `NOK_Hierarchy_Create()` (Table 8).

Function `NOK_Part_Divide()` divides the *Formalized Node of Knowledge* record into its syntactic parts and enters them into the auxiliary table `NOK_PART`. For each character from the input formalized record, it is checked whether it is one of the delimiters (), " & _ (both brackets, comma, quotation mark, character "&"or underscore). In the case of "&" and "_", the next character must be checked and if it is " " (space) then it is a delimiter, otherwise it is a part of the text. If the delimiter is found, the previous syntactic part (word – text) `Part` is entered and after that, the delimiter itself is entered. Finally, `Part` is cleared for the beginning of a new entry. If the loaded character is not a delimiter, then it is attached to the previous syntactic part `Part`. The `NOK_Part_Divide()` function algorithm is given in Table 4.

To enter and read the syntactic parts of the formalized record into the auxiliary table `NOK_PART`, the functions `NOK_Part_Write(NumP, Part)` and `NOK_Part_Read(Pos)`, are used, described in Tables 5 and 6.

Within the process 1.3 First Node Entry, the function `NOK_Hierarchy_Insert()` is started (Table 7).

Process 1.3. First Node Entry (Verb) reads the first syntactic part `Part` of the formalized record from the auxiliary table `NOK_PART` with the help of the `NOK_Part_Read(Pos)` function (the syntax is not checked, but it is assumed that the first syntactic part is correct and that it is a verb– `VERB`). The first node `NodId:= 1, ParId:= 0 i Quest:= '_'` is prepared and inserted into the table `NOK_FNOK` by calling the function `NOK_Hierarchy_Insert (PrgId, OrdNo,ParId,NodId, Part, Quest)`. Values are assigned:

```
PrgId →Paragraph ID,
OrdNo →Sentence Order Number
```

Within the process 1.4 Decomposing the *Formalized Node of Knowledge* record (if there are more parts), the following syntactic part `Part` is read by calling the function `NOK_Part_Read(Pos)`. If there are more parts, the next syntactic part `Part` should be a bracket " (". If the next `Part` is not the bracket " (", the processing of that record will be interrupted (and proceed to the next record). If the next `Part` is the bracket '(', then there are more parts, so it moves to the next hierarchy level by calling the

Table 5
Algorithm for function NOK_Part_Write().

```
-----------------------------------------------------------------------------------------

Function    NOK_Part_Write (  NumP  IN Number    -- Number of precedent syntactical Parts
                           ,  Part  IN Varchar2  -- New syntactical Part
                           )
Return   Number Is
-----------------------------------------------------------------------------------------
-- Write a new syntactical Part into the temporary table NOK_PART.
-----------------------------------------------------------------------------------------
--
Begin
   If trim(Part) Is NULL Then    -- if Part is NOT OK exit
      Return NumP;
   End If;

-- increment Number of Parts and write the Part
   Insert Into    NOK_PART    ( ORD     ,  PART       )
          Values              ( NumP + 1,  trim(Part) ) ;
   Return NumP + 1;              -- Return Number of syntactical Parts
--
End;

-----------------------------------------------------------------------------------------
```

Table 6
Algorithm for function NOK_Part_Read().

```
-----------------------------------------------------------------------------------------

Function    NOK_Part_Read (
                             NumP  IN Number      -- Number of precedent syntactical Parts
                          )
Return Varchar2 Is
-----------------------------------------------------------------------------------------
-- Read requested syntactical Part of FNOK from temporary table NOK_PART.
-----------------------------------------------------------------------------------------
   Part     Varchar2(99);        -- Single Syntactical Part
Begin
--
   Select   trim(PART)
   Into     Part
   From     NOK_PART
   Where    ORD = NumP;
--
   Return Part;                  -- Return Number of syntactical Parts
End;

-----------------------------------------------------------------------------------------
```

Table 7
Algorithm for function NOK_Hierarchy_Insert().

```
-----------------------------------------------------------------------------------------
Function    NOK_Hierarchy_Insert (
                                    PrgId IN Number   -- Paragraph of Sentences
                                 ,  OrdNo IN Number   -- Order Number
                                 ,  ParID IN Number   -- Parent Identifier
                                 ,  NodID IN Number   -- Node Identifier
                                 ,  Node  IN Varchar2 -- Node content
                                 ,  Quest IN Varchar2 -- Question
                                 )
Return Number Is
-----------------------------------------------------------------------------------
-- Insert one hierarchical Node to hierarchical table NOK_FNOK
-----------------------------------------------------------------------------------
--
Begin
--
   Insert Into    NOK_FNOK    (  PRG   ,  ORD   ,  PARENT_ID,  NODE_ID,  NODE,  QUEST )
   Values                     (  PrgId,  OrdNo,  ParId    ,  NodId  ,  Node,  Quest ) ;
--
   Return 1;
End;
-----------------------------------------------------------------------------------
```

Table 8
Algorithm for function NOK_ Hierarchy_Create().

```
---------------------------------------------------------------------------------------

Function    NOK_Hierarchy_Create (
  NOKID IN OUT   Number  -- NOK IDENTIFIER (used as
                                                             SEQUENCE)
                                 ,  PrgID IN       Number  -- Paragraph of Sentences
                                 ,  OrdNo IN       Number  -- Order Number
                                 ,  ParID IN       Number  -- Parent Identifier
                                 ,  NumP  IN       Number  -- Number of syntactical Parts
                                 ,  Pos   IN OUT   Number  -- precedent Position
                                 )
Return Number Is
-----------------------------------------------------------------------------------
-- Transform a FNOK record in Hierarchical order.
-----------------------------------------------------------------------------------
--
   Ret      Number;        -- Return Value
   NodId    Number;        -- Node  Identifier
   Part     Varchar2(99);  -- Single Syntactical Part
   Quest    Varchar2(99);  -- Question
--
Begin
--
   While Pos < NumP Loop                  -- for all - until the last syntactical Part
     Pos   := Pos + 1;
     Part  := NOK_Part_Read(Pos);         -- read next syntactical Part
     If    Part  = '(' Then               -- if divisor is equal to '('
                                          -- then can continue on the next level of
                                               hierarchy (recursive call)
          Pos := NOK_Hierarchy_Create(NOKID, PrgId, OrdNo, ParID, NumP, Pos);
          Continue;
     ElsIf Part  = ')' Then  Exit;        -- if divisor is equal to ')' then it is the
                                                 end of this level of hierarchy
     ElsIf Part  = ',' Then  Continue;    -- if Part is equal to ',' we skip to the
                                                 next Part
     End If;
--
      Pos   := Pos + 1;
      Part  := NOK_Part_Read(Pos);       -- read PART as NODE - ANSWER
      NOKID := NOKID + 1;                -- NOK IDENTIFIER increment
      NodId := NOKID;                    -- Node Identifier for first Node
      Ret  := NOK_Hierarchy_Insert(PrgId, OrdNo, ParId, NodId, Part, Quest);
                                          -- INSERT THIS NODE
      Pos   := Pos + 1;
      Part  := NOK_Part_Read(Pos);       -- read next syntactical divisor (punctuation)
      If    Part  = '(' Then             -- if divisor is equal to '(' then can
                                    continue on the next level of hierarchy (recursive call)
         Pos := NOK_Hierarchy_Create(NOKID, PrgId, OrdNo, NodId, NumP, Pos);
      ElsIf Part  = ')' Then             --if divisor is equal to ')' then it is the
                                              end of this level of hierarchy
         Exit;
      Else                               -- repeat the last read
         Pos := Pos - 1;
      End If;
--
   End Loop;
   Return Pos;                           -- return Actual Position of the last
                                              syntactical Part
End;
-----------------------------------------------------------------------------------
```

function `NOK_Hierarchy_Create(NOKID, PrgId, OrdNo, NodId, NumP, Pos)`, that is given in Table 8.

The recursive function `NOK_Hierarchy_Create()` continues to read the following syntactic parts of the record, create nodes and enter them into the hierarchical table `NOK_FNOK`, using the read and write functions already mentioned.

The specificity of this recursive function is that at each open bracket " (" in the record syntax, it creates one lower level of hierarchy and at each closed bracket ") ", it returns to the previous level until all the syntactic parts are processed. The node hierarchy `NODE` is created in the `NOK_FNOK` table by adding and filling in the `PARENT_ID` column for each `NODE_ID`. By carrying out the procedures described, the data is inserted in the table `NOK_FNOK`. Over the data in the table, the process of enriching and linking sentences and dictionary (process 2) is run. The dictionary consists of three tables: `KB_DICT, KB_WORD i KB_QUEST`.

The node that is `VERB` by syntax is always at the first level of the hierarchy (`First PARENT`). Verbs are, through process 2.1 i.e. procedure `KB_Dictionary()`, read from the table `NOK_FNOK` (all different occurrences of the first node are read, `First PARENT` = 0) and inserted into the table `KB_DICT`. This procedure is given in Table 9.

The verb nodes are especially singled out because the verbs in a sentence often have different forms (e.g., 3rd person singular in Present Simple or other regular or irregular verbal forms). At this

Table 9
Algorithm for procedure KB_Dictionary().

```
-------------------------------------------------------------------------------------
Procedure   KB_Dictionary   (
                              PrgId IN Number   -- Paragraph of Sentences
                           )
Is
Begin
-------------------------------------------------------------------------------------
-- Insert distinct VERBS to KB_DICT from NOK_FNOK
-------------------------------------------------------------------------------------
-->>> Process 2.1
--
   Insert   Into   KB_DICT ( DICT_ID          , DICT  ,  DICT_PAR )
   --
   Select                    M.MaxId + ROWNUM, K.NODE,  null
   --
   From   (  Select nvl(max(DICT_ID), 0) MaxId
             From   KB_DICT
          )  M
        , (  Select distinct NODE
             From
                  ( Select   NODE
                    From     NOK_FNOK
                    Where    PRG        = PrgId             -- requested Paragraph
                      And    PARENT_ID = 0      )           -- First Node = VERB
         Where  NODE Not In (Select DICT From KB_DICT)
          )  K
   Order by NODE ;
--
End;
-------------------------------------------------------------------------------------
```

time, the verb forms and the nouns are processed in this way, but the same procedure can be applied to other types of words.

The NOK_FNOK table contains all the words that appeared in the input formalized records. This table is the first usable *Formalized Node of Knowledge* record in the relational database. Each node from the table NOK_FNOK is inserted into the dictionary using the procedure KB_List_Words() (Table 10), into one of the following tables: KB_QUEST or KB_WORD, as indicated in process 2.2 in Fig. 4. Each node (each word from the sentence) is retrieved and inserted in the table KB_WORD, or the table KB_QUEST if it is a question (wh-question). Given the characteristics of the formal record, the system can automatically recognize which of the following tables should be used for entry: words from column NOK_FNOK.NODE are inserted into table KB_WORD, and data from column NOK_FNOK.QUEST is inserted in the KB_QUEST table.

Within the same procedure, support was given to process 2.3 for linking words from KB_WORD with KB_DICT. If the procedure KB_List_Words() finds the same word in tables KB_DICT and KB_WORD (KB_DICT.DICT = KB_WORD.WORD), it fills the node identifier with the word identifier in the dictionary (KB_WORD.DICT_ID is filled in with KB_DICT.DICT_ID).

In order to identify words that are not in their basic form (tense, plural / singular...), each occurrence of verb / noun in a form that is different from the basic form, must be related to the basic form. This is done in the table KB_DICT, and in Fig. 4 it is shown using the process 2.4. It is manually verified whether VERB NODE is, for example, an irregular verb form, or a variant that differs from the verb in infinitive. If this is the case, this word is related to its "original". For example, the word is linked with *be*, and the word *sang* with *sing*.

However, since it is realistic to expect verbs to appear in other forms before the original form (and are not yet entered into the KB_DICT dictionary), the input of the original form is often done manually. The process within which verb forms are created in the dictionary is labeled with 2.4.

In order to display the dictionary hierarchy visually, the query shown in Table 11 may be used.

Since nouns, as well as other types of words can appear in a sentence in a form that does not correspond to the original dictionary form (plural with suffix -s and the like), they will also need to be linked to their original forms. For example, if the word "girls" appears in the sentence, it should be linked to singular form *girl* in the dictionary.

The entry of those original forms that have not yet appeared in sentences or even in the dictionary, is shown in process 2.7. A dashed line indicates that this part is performed manually.

The initial filling of the NOK_FNOK table transformed the *Formalized Node of Knowledge* record of sentences into a relational database and the table NOK_FNOK was filled with nodes. Table 19 shows the structure of the NOK_FNOK table after the initial transformation. The rows in the table remain to be linked with the words form the dictionary (tables KB_WORD and KB_QUEST). For this purpose, the structure of the table NOK_FNOK has been supplemented by attributes KB_NOD_WRD_ID and KB_NOD_QST_ID, that is, the link with the given tables from the dictionary was generated through the ID. However, for easier tracking of results (human-friendly), columns KB_PAR_WRD_ID and KB_PAR_WRD were also added to describe the parent node. Filling these four attributes is implemented within process 2.5 using the procedure KB_Update_FNOK() (Table 12).

The presented system is a system in which two important elements occur:

- Each sentence enriches the dictionary with new words (when the word appears for the first time in a sentence of text, it is entered in the dictionary because it does not exist there yet.) Identification (ID) is assigned to it.
- The dictionary enriches each word in the sentence (in each sentence (when all the words of the sentence exist in the dictionary), during the entry process into the database, it concatenates the actual occurrence of the word in the sentence with the corresponding word in the dictionary. This is how the dictionary enriches the source word record in a sentence).

Table 10
Algorithm for procedure KB_List_Words().

```
-----------------------------------------------------------------------------------
Procedure   KB_List_WORDS   (
                              PrgId IN Number  -- Paragraph of Sentences
                           )
Is
Begin
--
-->>> Process 2.2
--
-- -- Inserting distinct QUEST to KB_QUEST from NOK_FNOK
   Insert   Into   KB_QUEST ( QUEST_ID        ,  QUEST   )
   --
   Select                    M.MaxId + ROWNUM,  K.QUEST
   --
   From  (  Select nvl(max(QUEST_ID), 0) MaxId
            From   KB_QUEST
         )  M
       , (  Select distinct QUEST
            From   NOK_FNOK
            Where  PRG    = PrgId                -- requested Paragraph
And  QUEST != '_'
              And  QUEST Not In ( Select QUEST From KB_QUEST )
         )  K ;
--
-- -- Inserting distinct NODE  to KB_WORD  from NOK_FNOK
   Insert   Into   KB_WORD  ( WORD_ID         ,  WORD  ,  DICT_ID )
   --
   Select                    M.MaxId + ROWNUM,  K.NODE,  null
   --
   From  (  Select nvl(max(NODE_ID), 0) MaxId
            From   KB_WORD
         )  M
       , (  Select distinct NODE
            From   NOK_FNOK
            Where  PRG = PrgId                   -- requested Paragraph
              And  NODE  Not In   ( '_','&','X')
              And  NODE  Not In   ( Select WORD From KB_WORD )
         )  K ;
--
-->>> Process 2.3
-- -- Updating DICT_ID from KB_DICT (only for Verbs)
   Update   KB_WORD
   Set      DICT_ID = Select DICT_ID From KB_DICT Where DICT = KB_WORD.WORD ;
--
End;
-----------------------------------------------------------------------------------
```

Table 11
Query for visual representation of dictionary hierarchy.

```
-----------------------------------------------------------------------------------
   Select   DICT_ID                       ID
        ,   lpad(' ', LEVEL * 5) || DICT   VERB
        ,   DICT_PAR                      PARENT
   From     KB_DICT
   Start With                 DICT_ID = DICT_PAR
   Connect By NOCYCLE PRIOR   DICT_ID = DICT_PAR ;
-----------------------------------------------------------------------------------
```

After creating question coding table KB_QUEST and node coding table KB_WORD and their linking to table NOK_FNOK using question identifiers (question words) and node identifiers, a final table can be created and filled with questions KB_TEXT_QUEST obtained from the initial input formalized sentence records (NOK_FNOK_ALL (Sentences)). Table KB_TEXT_QUEST contains attributes ID, the paragraph mark, and the sentence mark based on which the question was created. In addition to the above attributes, the table also contains:

```
PAR_WRD_ID Parent Word Identifier
NOD_WRD_ID Node Word Identifier
NOD_QST_ID Node Question Identifier.
```

In order to facilitate the monitoring of the test results and taking into account the fact that identification marks are numerical, their textual values are also added: PAR_WRD, NOD_WRD, NOD_QST. These textual records are only for visual verification and are not used in further algorithms.

Within process 2.6, procedure KB_Create_QUEST(PrgId) fills this table by retrieving the finished data prepared in the "working" table NOK_FNOK (Table 13).

4. Research results and discussion (Case study)

In this section, we will demonstrate the results of the transformation algorithm application on a set of sentences. The system

Table 12
Algorithm for procedure KB_Update_FNOK().

```
-------------------------------------------------------------------------------------
Procedure   KB_Update_FNOK   (
                                PrgId IN Number   -- Paragraph of Sentences
                             )
Is
Begin
--
-->>> Proces 2.5
-- -- Update NOK_FNOK From List KB_QUEST And List KB_WORD
--
-- -- Parent Node Word KB_PAR_WRD
   Update   NOK_FNOK      K
   Set      KB_PAR_WRD    = Select   NODE
                            From     NOK_FNOK
                            Where    PRG       = K.PRG
                              And    ORD       = K.ORD
                              And    NODE_ID   = K.PARENT_ID
   Where    PRG           = PrgId                      -- requested Paragraph
     And    PARENT_ID     >  0      ;
--
-- -- Parent Node Word Identifier KB_PAR_WRD_ID
   Update   NOK_FNOK      K
   Set      KB_PAR_WRD_ID = Select   WORD_ID
                            From     KB_WORD
                            Where    WORD      = K.KB_PAR_WRD
   Where    PRG           = PrgId ;                    -- requested Paragraph
--
-- Node Word Identifier KB_NOD_WRD_ID
   Update   NOK_FNOK      K
   Set      KB_NOD_WRD_ID = Select   WORD_ID
                            From     KB_WORD
                            Where    WORD      = K.NODE
   Where    PRG           = PrgId ;                    -- requested Paragraph
--
-- Node Question Identifier KB_NOD_QST_ID
   Update   NOK_FNOK      K
   Set      KB_NOD_QST_ID = Select   QUEST_ID
                            From     KB_QUEST
                            Where    QUEST     = K.QUEST
   Where    PRG           = PrgId ;                    -- requested Paragraph
--
-- Node Question Identifier KB_NOD_QST_ID for QUEST = '&'
   Update   NOK_FNOK    K
   Set      KB_NOD_QST_ID  =  -1    -- non important Question
   Where    QUEST          = '&'
     And    PRG            = PrgId                     -- requested Paragraph
--   And    KB_NOD_QST_ID  is null
   ;
-------------------------------------------------------------------------------------
```

was tested on 100 simple sentences of English language, shown in Appendix A. Let us present a portion of the relational database content obtained after the transformation of sentences from Appendix A.

The first table in the database is simple and contains a list of paragraphs: NOK_PARAGRAPH (PRG, NAME), i.e. text segments (group of sentences). In our case, its content has only one row and is shown in Table 14. The first row in the table represents the attribute list, while the other rows represent data in the database. The key attribute is underlined with a full line.

The second table is: NOK_FNOK_ALL (PRG, ORD, TENG, FNOK) and it is displayed in Table 15. It represents the list of natural language sentences (text in English, TENG) and its *Formalized Node of Knowledge* (FNOK) record, by paragraphs. Only the first six sentences are shown.

Three tables representing the Dictionary are shown in Tables 16–18.

The foreign key is underlined using a dashed line. The foreign key DICT_PAR was generated by translating the recursive link CONTEXT from Fig. 3.

The NOK_FNOK table is generated by the transformation of record from the table NOK_FNOK_ALL and is shown in Table 19. For each node in the record, a row is created in the table and this row is linked to the initial sentence, i.e. the record and the paragraph in which it is located. This table is a list of nodes in the record and the mark of a parent node (PARENT_ID). Each row of the table is linked to the dictionary in which the word is located with its unique semantics. The table row is also linked to the parent word in the dictionary using a key. Finally, the table row is also linked to the key of the question set in the initial sentence. In this way, the words in the initial sentences are denoted with a unique meaning from the dictionary. Process nodes in the record do not have a parent node and their value is PARENT_ID = ∅. Three attributes (in strikethrough) NODE, QUEST and KB_PAR_WRD are not attributes of this table, but are added for easier tracking of table content.

The final table KB_TEXT_QUEST is a transformation of the initial *Formalized Node of Knowledge* records into the relational database, shown in Table 20. Each node link from the formalized record becomes one row in this table and has its own unique key ID in the database. Through foreign keys, it is related to the original sentence to which it belongs to and to two words in the dictionary and one question.

Table 13
Algorithm for procedure KB_Create_QUEST().

```
-----------------------------------------------------------------------------------
Procedure   KB_Create_QUEST   (
                                PrgId IN Number   -- Paragraph of Sentences
                             )
Is
Begin
--
-->>> Proces 2.6
-- -- Insert KB_TEXT_QUEST From NOK_FNOK
   Delete
   From     KB_TEXT_QUEST
   Where    PRG = PrgId  ;                         -- requested Paragraph
--
   Insert   Into KB_TEXT_QUEST (  ID              ,  PRG             ,  ORD
                             ,  PAR_WRD_ID      ,  NOD_WRD_ID      ,  NOD_QST_ID
                             ,  PAR_WRD         ,  NOD_WRD         ,  NOD_QST    )
   Select                       M.MaxId + ROWNUM,  K.PRG           ,  K.ORD
                             ,  K.KB_PAR_WRD_ID ,  K.KB_NOD_WRD_ID,  K.KB_NOD_QST_ID
                             ,  K.KB_PAR_WRD    ,  K.NODE          ,  K.QUEST
   From   (  Select   nvl(max(ID), 0)   MaxId
             From     KB_TEXT_QUEST
          )  M
        , (  Select   *
             From     NOK_FNOK
             Where    PRG = PrgId                  -- requested Paragraph
               And    PARENT_ID > 0
             Order by ORD
                  ,   NODE_ID
          )  K ;
End;
-----------------------------------------------------------------------------------
```

Table 14
List of paragraphs.

PRG	NAME
333	Appendix A

Table 16
Wh-questions in the table KB_QUEST.

QUEST_ID	QUEST
1	art
2	how
3	how far
4	how many
5	how much
6	how often
7	what
8	what time
9	when
10	where
11	which
12	who
13	whom
14	whose
…	…

The last three attributes (in strikethrough text) PAR_WRD, NOD_WRD and NOD_QST are not attributes of this table, but we have added them for easier tracking of the table content. One sentence is broken into as many rows as it has questions. The question is set by the Parent node and the answer to that question is the node Node itself.

The result of applying all algorithms is a relational database containing the knowledge from the starting textual sentences. The database structure and part of its content are shown in Tables 14–20. This database is the basis for further development of a question-answering system.

In other words, sentences of natural language grouped in paragraphs are put into relational database in a specific way: for each sentence expressed in the natural language, its *Formalized Node of Knowledge* record is defined and inserted into table NOK_FNOK_ALL. Algorithms are used to transform sentences one by one into hierarchical structure – hierarchy between words is formed. In addition, words are inserted into dictionary, which consists of three tables (KB_QUEST, KB_DICT, KB_WORD) necessary to create the list of all nodes and their parent nodes in the higher hierarchy level (table NOK_FNOK). Every row of the table is linked to the dictionary, which stores the word and its semantics, parent word from the dictionary and question set in the starting sentence. In this way, words from the starting sentences are assigned with a unique meaning from the dictionary.

Table 15
Sentences in the table NOK_FNOK_ALL.

PRG	ORD	TENG	FNOK
333	1	Daniel drives.	drives ("who?" Daniel)
333	2	Jacob drives a truck.	drives ("who?" Jacob, "what?" truck ("art?" a))
333	3	Thomas drives a car.	drives ("who?" Thomas, "what?" car ("art?" a))
333	4	John drives an automobile.	drives ("who?" John, "what?" automobile ("art?" an))
333	5	My friend Sean drives a car.	drives ("who?" Sean ("who?" friend ("whose?" my)), "what?" car ("art?" a))
333	6	Catherine drives a red car on the highway.	drives ("who?" Catherine, "what?" car ("art?" a, "what?" red), "where?" highway ("where?" on, "art?" the))
	…	…	…

Table 17
Dictionary that relates every word with its original form - KB_DICT.

DICT_ID	DICT	DICT_PAR
1	be	1
2	am	1
3	are	1
4	is	1
5	was	1
6	were	1
7	been	1
8	being	1
9	brush	9
10	buy	10
11	buys	10
12	came	61
13	can	13
14	could	14
15	cross	15
…	…	…

Table 18
Dictionary of all words that appear in the text – KB_WORD.

WORD_ID	WORD	DICT_ID
61	be	1
56	are	3
146	is	4
253	was	5
75	brush	9
76	buy	10
77	buys	11
78	came	12
79	can	13
89	could	14
275	cross	15
92	crossed	16

Transformation of the *Formalized Node of Knowledge* record to the relational database ends with execution of the algorithm, which generates data in the table `KB_TEXT_QUEST` (Question and answer), based on particular nodes from the `NOK_FNOK` (Node) table and their recursive relationships. This final table stores data in a form of question-answer, where question is not only question word, but in addition to it, linked node asking the question.

These transformations and enrichment of sentences in the *Formalized Node of Knowledge* record evolved relational schema to the set of three elements: who is asking the question, question, answer. Relational database can now serve for further applications, for example question answering system to support learning.

5. Conclusion

In this paper, we have presented the results of research of entering enriched English language sentences into a relational database in accordance with the *Node of Knowledge* conceptual framework for knowledge-based system development. The chosen *Formalized Node of Knowledge* record for sentences is enriched with explicit reference to links between the words in a sentence, with marking these links by natural questions used in everyday speech and the hierarchy among the word types.

In our earlier research of data processing through algorithms, unstructured textual files were used. The question arises as to whether the well-developed relational databases and their SQL languages can be used in the field of knowledge-based system based on *Node of Knowledge* conceptual framework. This work presents a system for transforming *Formalized Node of Knowledge* sentences into relational databases.

The logical model of the data shows the structure of the system based on *Formalized Node of Knowledge* sentence record. The transformation system itself is presented as a series of processes that load sentences and transform them into two key tables in a relational database. A dictionary is used and dynamically updated as needed, structured in three tables.

The relational database tables and algorithms of the transformation of the initial sentences into tables, represent new findings presented in this paper. It has been shown that enriched records can be entered into a relational database with the preservation of hierarchical relationships between words and linking with existing

Table 19
List of nodes – NOK_FNOK.

PRG	ORD	NODE_ID	~~NODE~~	~~QUEST~~	PARENT_ID	KB_PAR_WRD_ID	~~KB_PAR_WRD~~	KB_NOD_WRD_ID	KB_NOD_QST_ID
333	1	1	drives	_	0			102	
333	1	2	Daniel	who	1	102	drives	13	12
333	2	1	drives	_	0			102	
333	2	2	Jacob	who	1	102	drives	22	12
333	2	4	a	art	3	247	truck	47	1
333	2	3	truck	what	1	102	drives	247	7
333	3	4	a	art	3	81	car	47	1
333	3	1	drives	_	0			102	
333	3	2	Thomas	who	1	102	drives	44	12
333	3	3	car	what	1	102	drives	81	7
333	4	1	drives	_	0			102	
333	4	2	John	who	1	102	drives	23	12
333	4	3	automobile	what	1	102	drives	58	7
333	4	4	an	art	3	58	automobile	53	1
333	5	2	Sean	who	1	102	drives	42	12
333	5	3	friend	who	2	42	Sean	119	12
333	5	5	car	what	1	102	drives	81	7
333	5	6	a	art	5	81	car	47	1
333	5	4	my	whose	3	119	friend	165	14
333	5	1	drives	_	0			102	
333	6	2	Catherine	who	1	102	drives	11	12
333	6	3	car	what	1	102	drives	81	7
333	6	4	a	art	3	81	car	47	1
333	6	5	red	what	3	81	car	193	7
333	6	6	highway	where	1	102	drives	135	10
333	6	8	the	art	6	135	highway	234	1
333	6	1	drives	_	0			102	
333	6	7	on	where	6	135	highway	177	10

Table 20
List of links – KB_TEXT_QUEST.

ID	PRG	ORD	PAR_WRD_ID	NOD_WRD_ID	NOD_QST_ID	~~PAR_WRD~~	~~NOD_WRD~~	~~NOD_QST~~
1	333	1	102	13	12	drives	Daniel	who
2	333	2	102	22	12	drives	Jacob	who
3	333	2	102	247	7	drives	truck	what
4	333	2	247	47	1	truck	a	art
5	333	3	102	44	12	drives	Thomas	who
6	333	3	102	81	7	drives	car	what
7	333	3	81	47	1	car	a	art
8	333	4	102	23	12	drives	John	who
9	333	4	102	58	7	drives	automobile	what
10	333	4	58	53	1	automobile	an	art
11	333	5	102	42	12	drives	Sean	who
12	333	5	42	119	12	Sean	friend	who
13	333	5	119	165	14	friend	my	whose
14	333	5	102	81	7	drives	car	what
15	333	5	81	47	1	car	a	art
16	333	6	102	11	12	drives	Catherine	who
17	333	6	102	81	7	drives	car	what
18	333	6	81	47	1	car	a	art
19	333	6	81	193	7	car	red	what
20	333	6	102	135	10	drives	highway	where
21	333	6	135	177	10	highway	on	where
22	333	6	135	234	1	highway	the	art

dictionaries. Finally, it has been shown how to transform and write the sentence as a set of separate n-tuples representing individual statements contained in the initial sentence, all of which represent the initial sentence.

In this way, besides the graphical representation of the sentence in the *Diagram Node of Knowledge* (DNOK) form, and in addition to the *Formalized Node of Knowledge* (FNOK) form, a third form of the sentence representation is obtained through n-tuples in the relational database.

Limitation of this research is also in rather small number of simple sentences that use only some word types. The following word types are missing: conjunctions, interjections, two types of pronouns (relative and interrogative pronouns). Complex sentences, sentences that have homonyms and synonyms were not included in the research.

Diagram Node of Knowledge and *Formalized Node of Knowledge*, and the proposed transformation system were applied to Croatian language as well. This method can be applied to different languages and the proposed database structure and the transformation algorithm may be used independently of the language. The problem for different languages is in translating sentences into formalized record. The translation depends on the language.

The described system can be used as a basis for development of a question-answering system for any application domain, for example: Educational tutoring system (generating questions for testing students and/or giving answers to students' questions), Customer support system (giving answers to customer's inquiry), Tourist guidance system (assisting visitors offering required information), etc. Since it is adaptable to different languages, it can be useful for systems, which support learning and teaching of foreign languages.

Further research is directed towards the development of algorithms for the transformation of interrogative sentences into a relational database, expansion of the dictionary with all word classes, expansion of the dictionary with synonyms, solution to the problem of homonyms, expansion of storing contextual terms for all word types, introduction of complex sentence modeling, introduction of semantics of sentences (duplication processing, related sentences).

Acknowledgment

The research has been conducted under the projects "Extending the information system development methodology with artificial intelligence methods" (reference number 13.13.1.2.01.) and "Development of a QA system on a relational database" (reference number 17.14.2.2.03) supported by University of Rijeka (Croatia).

Apendix A. Examples of input sentences

RBR	TENG	FNOK
1	Daniel drives.	drives ("who?" Daniel)
2	Jacob drives a truck.	drives ("who?" Jacob, "what?" truck ("art?" a))
3	Thomas drives a car.	drives ("who?" Thomas, "what?" car ("art?" a))
4	John drives an automobile.	drives ("who?" John, "what?" automobile ("art?" an))
5	My friend Sean drives a car.	drives ("who?" Sean ("who?" friend ("whose?" my)), "what?" car ("art?" a))
6	Catherine drives a red car on the highway.	drives ("who?" Catherine, "what?" car ("art?" a, "what?" red), "where?" highway ("where?" on, "art?" the))

(*continued on next page*)

(*continued*)

RBR	TENG	FNOK
7	Catherine drives a green car in the race.	drives ("who?" Catherine, "what?" car ("art?" a, "what?" green), "where?" race ("where?" in, "art?" the))
8	Peter drives his brother's car.	drives ("who?" Peter, "what?" car ("whose?" brother's ("whose?" his)))
9	Joseph draws with wooden crayons.	draws ("who?" Joseph, "what?" crayons ("what?" with, "what?" wooden))
10	The car is black.	is ("what?" car ("art?" the), "what?" black)
11	It is a stone table.	is ("what?" it, "what?" table ("art?" a, "what?" stone))
12	The table is made of stone.	is ("what?" made ("what?" table ("art?" the), "what?" stone ("what?" of)))
13	The table is wooden.	is ("what?" table ("art?" the), "what?" wooden)
14	Croatia is a country.	is ("what?" Croatia, "what?" country ("art?" a))
15	Peter is a lonely man.	is ("who?" Peter, "what?" man ("art?" a, "what?" lonely))
16	This clock is German.	is ("what?" clock ("what?" this), "what?" German)
17	The knife isn't sharp.	isn't ("what?" knife ("art?" the), "what?" sharp)
18	Grammar is complicated.	is ("what?" complicated ("what?" grammar))
19	The tasks are complicated.	are ("what?" complicated ("what?" tasks ("art?" the)))
20	Life is sometimes complicated.	is ("what?" complicated ("what?" life, "when?" sometimes))
21	The teacher writes with student's pencil.	writes ("who?" teacher ("art?" the), "what?" pencil ("what?" with, "whose?" student's))
22	Student's pencil is on the table.	is ("what?" pencil ("whose?" student's), "where?" table ("where?" on, "art?" the))
23	I need a lot of money.	need ("who?" I, "what?" money ("how much?" lot ("art?" a), "what?" of))
24	Andrea, buy any book for Harry.	buy ("who?" Andrea, "whom?" Harry ("whom?" for), "what?" book ("which?" any))
25	Every day is the same.	is ("what?" day ("what?" every), "what?" same ("art?" the))
26	On either side of the world people work hard every day.	work ("who?" people, "how?" hard, "when?" day ("what?" every), "where?" world ("what?" side ("what?" either ("what?" on)),"where?" of, "art?" the))
27	All my friends are good boys.	are ("who?" friends ("what?" all, "whose?" my), "what?" boys ("what?" good))
28	Somebody is in the house.	is ("who?" somebody, "where?" house ("where?" in, "art?" the))
29	She is a teacher.	is ("who?" she, "what?" teacher ("art?" a))
30	It is snowing.	is ("what?" snowing ("what?" it))
31	It is Angela's bicycle.	is ("what?" it, "what?" bicycle ("whose?" Angela's))
32	Those are stars.	are ("what?" stars, "what?" those)
33	That is doctor's car.	is ("what?" that, "what?" car ("whose?" doctor's))
34	This is your cat.	is ("what?" this, "what?" cat ("whose?" your))
35	I brush myself.	brush ("who?" I, "who?" myself)
36	He speaks about himself.	speaks ("who?" he, "who?" himself ("who?" about))
37	Jack is in the house.	is ("who?" Jack, "where?" house ("where?" in, "art?" the))
38	We don't sleep in the hotel Intercontinental.	don't ("what?" sleep ("who?" we, "where?" hotel ("where?" in, "art?" the) ("which?" Intercontinental)))
39	You can buy anything.	can ("what?" buy ("who?" you, "what?" anything))
40	Thomas buys 10 pounds of potatoes in the store.	buys ("who?" Thomas, "what?" potatoes ("how many?" pounds ("how many?" 10), "what?" of), "where?" store ("where?" in, "art?" the))
41	Nobody came.	came ("who?" nobody)
42	There was no-one on the beach.	was ("where?" there, "who?" no-one, "where?" beach ("where?" on, "art?" the))
43	There is nothing left.	is ("where?" there, "what?" left ("what?" nothing))
44	Everybody is afraid of him.	is ("what?" afraid, "who?" everybody, "whom?" him ("whom?" of))

(*continued on next page*)

(*continued*)

RBR	TENG	FNOK
45	These actors are Rob’s friends.	are ("who?" actors ("which?" these), "what?" friends ("whose?" Rob’s))
46	That car is red.	is ("what?" car ("which?" that), "what?" red)
47	This boy is Joseph’s brother.	is ("who?" boy ("which?" this), "what?" brother ("whose?" Joseph’s))
48	Tom has two brothers.	has ("who?" Tom, "what?" brothers ("how many?" two))
49	There are seventy-two students in the hotel.	are ("where?" there, "who?" students ("how many?" seventy-two), "where?" hotel ("where?" in, "art?" the))
50	Vanessa was the fifteenth person to win the award since 1999.	was ("who?" Vanessa, "what?" person ("what?" fifteenth ("art?" the)), "what?" award ("what?" to, "what?" win, "art?" the) ("when?" 1999 ("when?" since)))
51	Monica went to Italy two times this year.	went ("who?" Monica, "where?" Italy ("where?" to), "how many?" times ("how many?" two), "when?" year ("which?" this))
52	The car moves slowly.	moves ("what?" car ("art?" the), "how?" slowly)
53	The bears eat greedily.	eat ("who?" bears ("art?" the), "how?" greedily)
54	Mark drives a car very quickly today.	drives ("who?" Mark, "what?" car ("art?" a), "when?" today, "how?" quickly ("how?" very))
55	You look absolutely fabulous!	look ("who?" you, "how?" fabulous ("how?" absolutely))
56	Mary plays the violin extremely well.	plays ("who?" Mary, "what?" violin ("art?" the), "how?" well ("how?" extremely))
57	The book is on the table.	is ("what?" book ("art?" the), "where?" table ("where?" on, "art?" the))
58	Students talk about the solution.	talk ("who?" students, "what?" solution ("what?" about, "art?" the))
59	I see a teacher.	see ("who?" I, "who?" teacher ("art?" a))
60	The train to New York is fast.	is ("what?" train ("art?" the) ("where?" New York ("where?" to)), "what?" fast)
61	The boy is very young.	is ("who?" boy ("art?" the), "what?" young ("how?" very))
62	Peter has an office in New York.	has ("who?" Peter, "what?" office ("art?" an), "where?" New York ("where?" in))
63	Rob has an office 200 meters from the sea in Dubrovnik.	has ("who?" Rob, "what?" office ("art?" an), "where?" sea ("how many?" meters ("how many?" 200), "where?" from, "art?" the), "where?" Dubrovnik ("where?" in))
64	Tom is quite a good boy.	is ("who?" Tom, "what?" boy ("what?" good ("how?" quite, "art?" a)))
65	I work hard.	work ("who?" I, "how?" hard)
66	Noah gave this book to a friend.	gave ("who?" Noah, "what?" book ("which?" this), "who?" friend ("who?" to, "art?" a))
67	Maya has some interesting books.	has ("who?" Maya, "what?" books ("how many?" some, "what?" interesting))
68	I often go to the cinema.	go ("who?" I, "when?" often, "where?" cinema ("where?" to, "art?" the))
69	London is the capital of Great Britain.	is ("what?" London, "what?" capital ("art?" the) ("what?" Great Britain ("what?" of)))
70	Mary is Lucy’s friend.	is ("who?" Mary, "what?" friend ("whose?" Lucy’s))
71	Father is a good man.	is ("who?" father, "what?" man ("art?" a, "what?" good))
72	Beauty is not forever.	is not ("what?" beauty, "what?" forever)
73	Honesty is the best policy.	is ("what?" honesty, "what?" policy ("what?" best ("art?" the)))
74	Mary’s father is old.	is ("who?" father ("whose?" Mary’s), "what?" old)
75	It is six o’clock.	is ("what?" it, "what time?" o’clock ("what time?" six))
76	Lucy goes to the gym three times a week.	goes ("who?" Lucy, "how often?" week ("how many?" times ("how many?" three), "art?" a), "where?" gym ("where?" to, "art?" the))
77	School is 50 meters from our house.	is ("what?" school, "where?" house ("how far?" meters ("how far?" 50), "where?" from, "which?" our))
78	John saw me on the street yesterday.	saw ("who?" John, "whom?" me, "where?" street ("where?" on, "art?" the), "when?" yesterday)
79	Tom is younger than Peter.	is ("who?" Tom, "what?" younger ("who?" Peter ("who?" than)))

(*continued on next page*)

(*continued*)

RBR	TENG	FNOK
80	Joseph is the youngest.	is ("who?" Joseph, "what?" youngest ("art?" the))
81	Competitions are in the last week of April.	are ("what?" competitions, "when?" April ("when?" week ("when?" in, "what?" last ("art?" the)), "what?" of))
82	I do my homework every day at five o'clock.	do ("who?" I, "what?" homework ("whose?" my), "when?" day ("what?" every), "when?" o'clock ("when?" at, "when?" five))
83	Lucas does his homework every day at seven o'clock.	does ("who?" Lucas, "what?" homework ("whose?" his), "when?" day ("what?" every), "when?" o'clock ("when?" at, "when?" seven))
84	He can read.	can ("what?" read ("who?" he))
85	You can touch sister's notebook.	can ("what?" touch ("who?" you, "what?" notebook ("whose?" sister's)))
86	We could be friends.	could ("what?" be ("who?" we, "what?" friends))
87	I will work tomorrow.	will ("what?" work ("who?" I, "when?" tomorrow))
88	Lucas eats a sandwich.	eats ("who?" Lucas, "what?" sandwich ("art?" a))
89	Students eat chocolate.	eat ("who?" students, "what?" chocolate)
90	Our ship crossed the Atlantic in 7 days.	crossed ("what?" ship ("whose?" our), "what?" Atlantic ("art?" the), "what time?" days ("what time?" in, "how many?" 7))
91	Dinner is Lucas's favorite meal.	is ("what?" dinner, "what?" meal ("what?" favorite, "whose?" Lukas's))
92	His father's car was stolen.	was ("what?" stolen ("what?" car ("whose?" father's ("whose?" his))))
93	You want a good book.	want ("who?" you, "what?" book ("art?" a, "which?" good))
94	It rains.	rains ("what?" it)
95	There are 30 students in the classroom.	are ("where?" there, "who?" students ("how many?" 30), "where?" classroom ("where?" in, "art?" the))
96	Students play basketball every Saturday.	play ("who?" students, "what?" basketball, "when?" Saturday ("what?" every))
97	A man is going on a journey	is ("what?" going ("who?" man ("art?" a), "where?" journey ("where?" on, "art?" a)))
98	Students individually write a seminar paper during the whole semester.	write ("who?" students, "how?" individually, "what?" paper ("art?" a, "what?" seminar), "when?" semester ("what?" during, " art?" the, "what?" whole))
99	Scientists talk about global warming on the TV show.	talk ("who?" scientists, "what?" warming ("what?" about, "what?" global), "where?" show ("where?" on, "art?" the, "what?" TV))
100	Mary doesn't look tired.	doesn't ("what?" look ("who?" Mary, "how?" tired))

References

Abulaish, M., & Dey, L. (2007). Biological relation extraction and query answering from medline abstracts using ontology-based text mining. *Data & Knowledge Engineering, 61*(2), 228–262.

Ašenbrener Katić, M., Čandrlić, S., & Pavlić, M. (2017). Comparison of two versions of formalization method for text expressed knowledge. In *Communications in computer and information science 716, CCIS* (pp. 55–66). Springer International Publishing. https://doi.org/10.1007/978-3-319-58274-0_5.

Ašenbrener Katić, M., Čandrlić, S., & Pavlić, M. (2018). Modeling of Verbs Using the Node of Knowledge Conceptual Framework. In *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1192–1197). IEEE.

Ašenbrener Katić, M., Pavlić, M., & Čandrlić, S. (2015). The representation of database content and structure using the NOK method. *Procedia Engineering, 100*, 1075–1081. https://doi.org/10.1016/j.proeng.2015.01.469.

Biber, D., Johansson, S., Leech, G., Conrad, S., & Finegan, E. (1999). *Longman grammar of spoken and written english*. London: Longman.

Brinton, L. J., & Brinton, D. M. (2010). *The linguistic structure of modern english*. Philadelphia: John Benjamins B.V.

Chein, M., & Mugnier, M.-L. (2009). *Graph-based knowledge representation, computational foundations of conceptual graphs*. Springer.

Chen, Z. Y., Yao, S., Lin, J. Q., Zeng, Y., & Eberlein, A. (2007). Formalisation of product requirements: From natural language descriptions to formal specifications. *International Journal of Manufacturing Research, 2*(3), 362–387.

Cinková, S., Hajič, J., Mikulová, M., Mladová, L., Nedolužko, A., et al. (2008). *Annotation of English on the tectogrammatical level*. Universitas Carolina Pragensis Online, Available: https://ufal.mff.cuni.cz/pedt1.0/papers/TR_En.pdf Accessed: 21-May-2018.

Erjavec, T.. *MULTEXT-East - Morphosyntactic Specifications (Version 4)* from http://nl.ijs.si/ME/V4/msd/html/index.html Retrieved January 1, 2010 .

Fellbaum, C. (1998). *WordNet*. Blackwell Publishing Ltd Blackwell publishing ltd..

Fougères, A. J., & Trigano, P. (1996). The formalisation of specifications from specifications written in natural language. *Expersys', 96*, 369–374.

Hallett, C. (2006). Generic querying of relational databases using natural language generation techniques. In *Proceedings of the fourth international natural language generation conference* (pp. 95–102). Association for Computational Linguistics.

Helbig, H. (2006). *Knowledge representation and the semantics of natural language*. Springer.

Jakupović, A., Pavlić, M., Meštrović, A., & Jovanovic, V. (2013). Comparison of the nodes of knowledge method with other graphical methods for knowledge representation. In *2013 36th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1276–1280). IEEE.

Jakupović, A., Pavlić, M., & Dovedan Han, Z. (2014). Formalisation method for the text expressed knowledge. *Expert Systems Applications, 41*(11), 5308–5322. https://doi.org/10.1016/j.eswa.2014.03.006.

Jespersen, O. (1964). *Essentials of English grammar*. Tuscaloosa: The University of Alabama Press.

Kanhe, S., Udawant, V., Bodke, P., & Chikhale, A. (2015). SQL generation and PL/SQL execution from natural language processing. *International Journal of Engineering Research & Technology (IJERT), 4*(2), 886–889.

Kovács, L. (2011). SQL generation for natural language interface. *Computer Technology and Computer Programming: Research and Strategies*, 19–22.

Lami, G., Gnesi, S., Fabbrini, F., Fusani, M., & Trentanni, G. (2004). An automatic tool for the analysis of natural language requirements. *Informe Técnico, CNR information science and technology institute, Pisa, Italia, Setiembre*.

Li, F., & Jagadish, H. V. (2014). NaLIR: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on management of data* (pp. 709–712). ACM.

Li, H., & Shi, Y. (2010). A WordNet-based natural language interface to relational databases. In *The 2nd International conference on computer and automation engineering: 1* (pp. 514–518). IEEE.

Mansuri, I. R., & Sarawagi, S. (2006). Integrating unstructured data into relational databases. In *Proceedings of the 22nd international conference on data engineering, ICDE'06*. IEEE.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications ACM, 38*(11), 39–41.

Minsky, M. A (1974). *Framework for representing knowledge* http://web.media.mit.edu/~minsky/papers/Frames/frames.html Available: 1-Jun-2018.

Nihalani, N., Motwani, M., & Silakari, S. (2010). An intelligent interface for relational databases. *International Journal of Simulation: Systems, Science & Technology, 11*(1), 29–34.

Oracle. (2015). *SQL- the natural language for analysis Oracle White Paper*.

Pavlić, M., Dovedan Han, Z., Jakupović, A., Ašenbrener Katić, M., & Čandrlić, S. (2017). Adjective representation with the method Nodes of Knowledge. In *2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1221–1226). IEEE. https://doi.org/10.23919/MIPRO.2017.7973610.

Pavlić, M. (2011). *Oblikovanje baza podataka*. Rijeka: Odjel za informatiku, Sveučilište u Rijeci.

Pavlić, M., Dovedan Han, Z., & Jakupović, A. (2015). Question answering with a conceptual framework for knowledge-based system development "Node of Knowledge". *Expert Systems with Applications, 42*(12), 5264–5286. https://doi.org/10.1016/j.eswa.2015.02.024.

Pavlić, M., Jakupović, A., & Čandrlić, S. (2014). *Modeliranje procesa*. Rijeka: Odjel za informatiku Sveučilište u Rijeci.

Pavlić, M., Jakupović, A., & Meštrović, A. (2013a). Nodes of knowledge method for knowledge representation. *Informatologia, 46*(3), 206–214.

Pavlić, M., Meštrović, A., & Jakupović, A. (2013b). Graph-based formalisms for knowledge representation. In *Proceedings of the 17th world multi- conference on systemic cybernetics and informatics (WMSCI 2013), Orlando* (pp. 200–204).

Peroperovvšek, M., Vavpetic, A., & Lavrac, N. (2012). A wordification approach to relational data mining: Early results. In *Late breaking papers of the 22nd international conference on inductive logic programming* (pp. 56–61).

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science, 12*, 410–430.

Quirk, R., Greenbaum, S., Leech, G., & Svartvik, J. (1985). *A comprehensive grammar of the english language*. New York: Longman Inc.

Sarawagi, S. (2005). Models and indices for integrating unstructured data with a relational database. In *Knowledge discovery in inductive databases* (pp. 1–10). BerlinHeidelberg: Springer.

Sgall, P., Panevová, J., & Hajičová, E. (2004). Deep syntactic annotation: Tectogrammatical representation and beyond. In *Proceedings of the workshop frontiers in corpus annotation at HLT-NAACL 2004* Online, Available: https://ufal.mff.cuni.cz/pdt2.0/publications/SgallPanevovaHajicova2004.pdf. Accessed: 3-Jun-2018.

Stanojević, M., & Vraneš, S. (2007). Knowledge representation with SOUL. *Expert Systems with Applications, 33*, 122–134.

Tolle, K.. *Introduction to RDF* Retrieved from http://www.dbis.informatik.uni-frankfurt.de/~tolle/RDF/DBISResources/RDFIntro.html.

Yang, S., Bhowmick, S. S., & Madria, S. (2005). Bio2X: A rule-based approach for semi-automatic transformation of semi-structured biological data to XML. *Data & Knowledge Engineering, 52*(2), 249–271.