Data Base Processing Programs with Using Extended Base Semantic Hypergraph

Alibek Barlybayev(✉), Talgat Sabyrov, Altynbek Sharipbay, and Assel Omarbekova

L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
frank-ab@mail.ru

Abstract. For any intelligent system it is necessary to have knowledge base and method of processing. In this paper we will look at globals as a way of storing knowledge in a structural way. As a structure, we chose the base advanced semantic hypergraph frames that are networked. We will conduct an experiment on the performance of our approach.

Keywords: Knowledge base · OOP · Semantic hypergraph · Extended base semantic hypergraph

1 Introduction

One of the most important and complex issues of artificial intelligence is the issue of representation and processing of knowledge. The basic models of knowledge representation are: productional, frame-based, logical and semantic networks. Using any of these models can be developed intelligent system. Therefore, before starting to develop we must choose the proper model [1].

The first production model was offered by Post in 1943. It is based on rules that allows us to represent knowledge in the form of sentences like "If (condition) then (action)". The production model has the disadvantage that the accumulation of a sufficiently large number (several hundreds) of productions they begin to contradict each other [2]. Frame - a method of knowledge representation in artificial intelligence, which is a scheme of action in a real time. Originally, the term "frame" Marvin Minsky introduced in the 70-ies of XX century to refer to the knowledge structure for the perception of spatial scenes. It is an abstract model of the image, the minimum possible description of the essence of any object, phenomenon, events, situations, process [3]. The main idea of the approach in constructing logical models of knowledge representation is all the information needed for applications viewed as a collection of facts and statements which are presented as a formula to some logic. Knowledge is displayed as a set of such formulas, and the generation of new knowledge is reduced to the implementation of inference procedures. A formal theory is based on logical models of representation [5]. Semantic web is the information domain model that has the form of a directed graph. In semantic network the role of vertexes is executed by meaning in database and arcs (directed) explains the relation between these meanings. Thus, the semantic network reflects the semantics of the domain in the form of meanings and relationships [5].

Á. Rocha et al. (eds.), *Recent Advances in Information Systems and Technologies*,
Advances in Intelligent Systems and Computing 569, DOI 10.1007/978-3-319-56535-4_3

In the treatment of knowledge commonly used search methods based on predicate calculus decisions (rules modus-ponus etc., conjunction and disjunction negation, etc.) [6]. Also it is used forward and reverse output in expert systems of production type (search strategy in depth, the width of the search strategy, splitting into sub-tasks, a-b-algorithm, etc.) [7]. Increasingly, they began to be used the knowledge in intelligent processing method with frame-systems (demons attached procedures, inheritance) [8].

Very interesting way would be to use multi-dimensional arrays as the repository for the knowledge base. Multi-dimensional arrays are arrays whose elements are arrays. Defining a multidimensional array must contain information about the type, dimensions and number of elements of each dimension. The elements of a multidimensional array are arranged in the memory in ascending order of the right index. Variables are both local and global can be as simple or indexed structures. Global variables or globals, as stored data, form the basis of so-called direct access [9]. Globals this data structure usually multidimensional stored in a database and can be processed by different processes in a multiuser environment. Multidimensionality of data is realized through the indexes, that is why we talk about an indexed variable.

Global data is stored in the B *tree. A tree that has the same number of sub-levels in each of its sub-tree, is called balanced (balanced tree, hence the B-tree). Tree, in which each key points to the data block containing the required entry, called B *tree. It enables the integration of the area pointers and the data area. B *tree can be thought of as a network consisting of graphs. According to hypergraph, H (V, E) has a pair, where V - the set of vertices $V = \{v\_i\}$, $i \in I = \{1, 2, \ldots, n\}$, and E - the set of edges $E = \{e\_j\}$, $j \in J = \{1, 2, \ldots, m\}$; each edge is a subset of V. The vertexx v and the edge e are called incident, if $v \in e$. For $v \in V$ throught d(v) is it defined the count edges, that are incident to v; d(v) is called the degree of vertex v. Degree of edge e–the count of vertexes that are incident to this edge marked as r(e). Hypergraph H is r-homogenious, if all the edges have the same degree r. Ontology–a comprehensive and detailed formalized idea of a specific domain using conceptual frameworks consisting of concepts copies (classes), attributes (properties), functions (operations), axioms (facts) and links. To construct the ontological model it will be used extended base semantic hypergraph (XBSH). The nodes in these graphs represent the semantic attributes (properties and functions) of objects or entities, and the arcs represent the relationships between them. Note that XBSH structure is similar to the paradigm of object-oriented programming Therefore XBSH can be used to describe application software that can answer customer questions about the knowledge base. Concepts in the hypergraph structure described in trees that are converted into mathematical formulas. There are a number of papers that describes the semantic graph. Zhen L, Jiang Z. describes a model of semantic hypergraph as "hypergraph based on semantic web" that can provide more sophisticated semantic networks and more efficient data structure for storing knowledge in the repositories.

Weights of vertexes extended base semantic hypergraph: $K = \{k_a\}$, $a \in A = \{1,2,\ldots b\}$, where $k_a = \{S_a, V_a, E_a\}$, $S_a = \bigcup_{j=1,2,\ldots}^{ks} s_j^a$ – *set of properties of class*, $V_a = \bigcup_{j=1,2,\ldots}^{kv} v_j^a \in k_a$ –*set of*

class instanses, $E_a = \bigcup_{j=1,2,\ldots}^{ke} e_j^a \in k_a$ *– set of semantic arcs that are incident to class ks –count of properties of class kv – count of class instances, ke – set of semantic arcs that are incident to class. Vertext-instance can be presented as a three* of $v_i = \{k_i, S_i, E_i\}$, *где* k_i *– parent class, that means* $v_i \in k_i$, *where* $S_i = \bigcup_{j=1,2,\ldots}^{ks} s_j^i$ *is a set of instace,s* $E_i = \bigcup_{j=1,2,\ldots}^{ke} e_j^i \in v_i$ *– is a set of semantic arcs that are incident to instance ks – count of class instances, ke – count of semantic arcs that incident to class instance*. This three, $k_a = \{S_a, V_a, E_a\}$, we have changed to five $k_a = \{S_a, F_a, I_a, V_a, E_a\}$, where F_a *– a set of class function,* I_a *– a set of incapsulations in class. Vertext-instance of class can be presented as a five of* $v_i = \{k_i, S_i, F_a, I_a, E_i\}$, *where* F_i *– is a set of instance functions,* I_a *– a set of instance.*

The extention of semantic hypergraph has the same meaning and extention of context-free grammar to attribute grammar. That means if *G = (N, T, P, S) is a context-free grammar, the attribute grammars defined as AG = (N, T, P, S, AS, AI, R), where N - set of nonterminals, T - a plurality of terminals (disjoint from N), P - the set of rules, S - the initial nonterminal, AS - a finite set of synthesized attributes, AI - a finite set of hereditary attributes (non-overlapping with AS), R - a finite set of semantic rules.*

Here, attribute grammars entirely new mathematical tool, which allows to describe not only the structure of language units, but their attributes (semantic features). It is a well-known classical scientific result Donald Knuth. That's how it should be understood the extention of semantic hypergraph, we have added two new characters that allow you to describe the set of functions of the class and a set of class encapsulation. But we do not have a name for the extended semantic hypergraph, so let's call it an extended base semantic hypergraph. Knowledge stored in globals, we will handle both structured and ontology, which is based on XBSH.

2 Knowledge Presentation

Globals a data structure in an OODB Intersystems Cache`, usually multidimensional stored in a database and can be processed in a multiuser environment, the various processes. To evaluate all globals values, develop a catalog of electronic devices. For this we introduce the Global ^*device*, which is on the first level has the article ^*device (article)*, and Global is the level of equipment properties ^*device (article, property)* on the second level, the level of properties of devices may comprise n-number of sublevels Device Properties ^*device(article, property, subproperty,…, n)*. Also on the third level of article can be a function ^*device (article, * func *)*. This level is determined by a key string **func**. All sublevels **func** contain a function object (in this case the device) ^*device(article, *func*, function)*. Similarly, the level of device characteristics, the level of the functions may include n-number of sublevels function devices ^*device(article,*func*, function, subfunction,…,n)*. Thus, we used multivariate data provided globals.

```
^device(111587,"*func*","run2") =    "w 'Hello 2'"
^device(111587,"frequence")    = 2.6
^device(111587,"name") = "ASUS GL752VW "
^device(111587,"os") = "Windows 10"
^device(111587,"processor")    = "Intel Core i7"
^device(111587,"ram") = "12gb"
^device(111587,"type") = "pc" etc.
```

After we pointed out the properties and functions of the devices, we need to connect these devices for part numbers. For this purpose, the key line "rel" *^device(rel,article1,article2)*, here given direction from article1 to article2, so arc-relations have directionality. To set the bidirectional arc-relations need to create an entry in the form of "rel" *^device(rel,article2, article1).* The levels of this line contains articles related devices, and in the values of these levels will contain the names of contacts, arcs, separated by a "/".

For example, consider a $^{device}(''rel'', 111587, 111588) =''$ *SameCompany/MadeOfIron''*, further prepared adjacency matrix and drawn-arc connection Fig. 1.

relations	111587	111588	111589	111590
111587		SameCompany/ MadeOfIron		
111588				
111589				
111590				

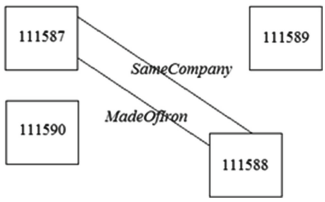

Fig. 1. Adjacency matrix and relations between nodes.

Functions in global are written like:

^device(111587,"*func*","run") = "w 'Hello'"

Thus, we have created globals that store of knowledge about the devices. We will collect these globals into structured frames in a semantic network in our knowledge base about the devices. In Fig. 2 we gather our globals network.

Built semantic network based on frame is very similar to the structure of the objects, which is used in object-oriented programming paradigm. The semantic network with a complex structure where the properties and functions of multidimensionality there, it is difficult to do semantic search. Therefore, it is necessary to define the mathematical model for it, and then build an algorithm of semantic search based on it. A hypergraph is a pair of H(V, E), but as we use the knowledge in addition to the properties and relations of functions and also has its own internal nesting hypergraph this type does not suit us. Therefore, we propose to extend the hypergraph by adding functions to the expanded core semantic hypergraph (XBSH). We use $H(A, P, F, R)$, where H – XBSH, A – Id concept, P – property, F – functions, R – relations, semantic arcs. $A = \{A_1, A_2, \ldots, A_n\}, P = \{P_1, P_2, \ldots, P_n\}$, $F = \{F_1, F_2, \ldots, F_n\}$, $R = \{R_1, R_2, \ldots, R_n\}$, where $P_1 = \{P_{11}, P_{12}, \ldots, P_{1k}\}$, $P_2 = \{P_{21}, P_{22}, \ldots, P_{2k}\}$, and $P_n = \{P_{n1}, P_{n2}, \ldots, P_{nk}\}$, also $F_1 = \{F_{11}, F_{12}, \ldots, F_{1k}\}$,

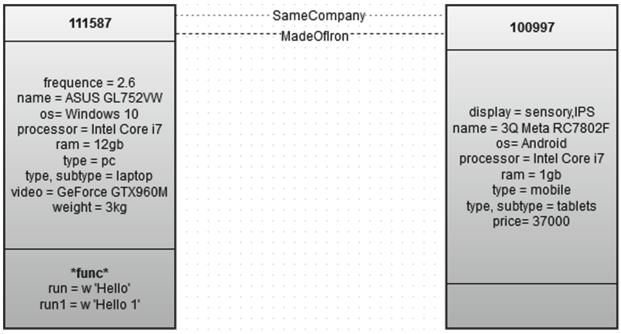

Fig. 2. Representation of knowledge about the devices in the structural form.

$F_2 = \{F_{21}, F_{22}, \ldots, F_{2k}\}$, a $F_n = \{F_{n1}, F_{n2}, \ldots, F_{nk}\}$. The composition would be $A(x) \circ P(y) \circ F(z) \circ R(w)$. For example, data processing (to find an example in the graphs), we used the Dijkstra algorithm. Definition of Dijkstra algorithm: *A – a set of vertexes in graph. R – a set edges in graph. w – weight of all edges. In this case constant 1. a - the distance where it is searched from. U – a set of visited vertexes. d[u] - is equal to the closest way from a to vertex u when the algorithm is finished. l[u] - consists to the closest way from a to vertex u when the algorithm is finished.* Algorithm:

Assign d[a] ← 0, p[a] ← 0 ┘ For all u ∈ A different from a ┘ assign d[u] ← ∞ ┘ while ∃v ∉ U ┘ let v ∉ U – vertex with minimum d[u] ┘ write v in U ┘ For all u ∉ U that vu ∈ R ┘ if d[u] > d[v] + w then ┘ change d[u] ← d[v] + w ┘ change l[u] ← l[v], u

3 Processing Data

For a software implementation knowledge base we recorded test data about the devices in the Global ^*device*. Displays can be seen in the management portal system indicated in Fig. 3.


```
{
  "^device(58783)": {
    "^device(58783,\"name\")": "Data switch KVM D-Link DKVM-4K",
    "^device(58783,\"price\")": "10460",
    "^device(58783,\"type\")": "network"
  },
  "^device(75009)": {
    "^device(75009,\"interface\")": "usb",
    "^device(75009,\"name\")": "Defender Accent 930, Black, USB",
    "^device(75009,\"price\")": "2059",
    "^device(75009,\"type\")": "keyboard"
  },
  "^device(77115)": {
    "^device(77115,\"frequence\")": "20-16000h",
    "^device(77115,\"length\")": "1.37m",
    "^device(77115,\"name\")": "ACME MK-200",
    "^device(77115,\"price\")": "803",
    "^device(77115,\"type\")": "earphone"
  }
}
```


Fig. 3. Display the globals in the Management Portal.

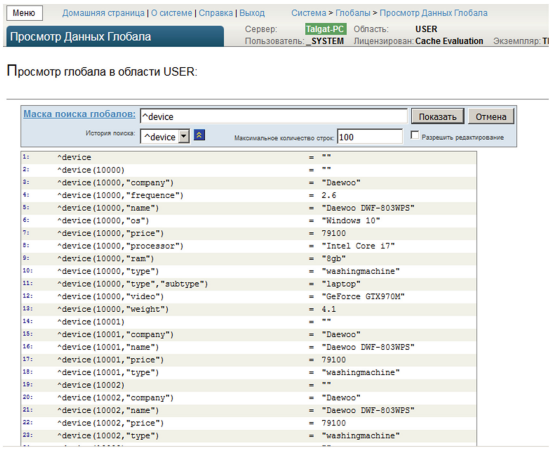

Fig. 4. Detail of JSON data.

With the data we have gathered in the global with frames and translated them into JavaScript data format Object Notation (JSON) Caché Object Script (COS) programming language to output the data to the client part, JSON is described in Fig. 4.

Further data in the form of frames in JSON format are displayed in the client browser. To do this, use the JavaScript programming language with a framework AngularJS, is described in Fig. 5.

Next, we carry out a test process according to the algorithm described above data. The algorithms for processing the input data are written as the name of the vertex. The program finds the shortest way to every vertex connected to the above algorithm, the result of the algorithm is described in Fig. 6.

As a rule, knowledge base processing algorithm should solve problems in which questions like put “List all the possible options…”, “How many ways are there to…”, “Is there any way…”, “whether the object exists…”, etc. Next, we plan to improve the knowledge-processing algorithm to the above-mentioned level.

The task of filling the knowledge base and find it is closely associated with the responses nlp problems; it is necessary to convert the text in a natural language into the language of the Knowledge Base. The first algorithm is needed normalizing simple sentences. Further necessary syntactic and morphological analyzers. Also we need an algorithm able to identify concepts in the text, attributes, and actions of these concepts, as well as revealing the relationships between concepts. Names tops Knowledge Base - nouns. Properties tops – articles, adjectives, pronouns, numerals, adverbs, prepositions. Unions – will be used for the normalization of the proposals. Relations and functions – Verbs (interaction between concepts, XBSH top). Functions can change the property values, if something happens, such as a condition, and they can work closely with different properties and functions of the other peaks. Work on nlp described below.

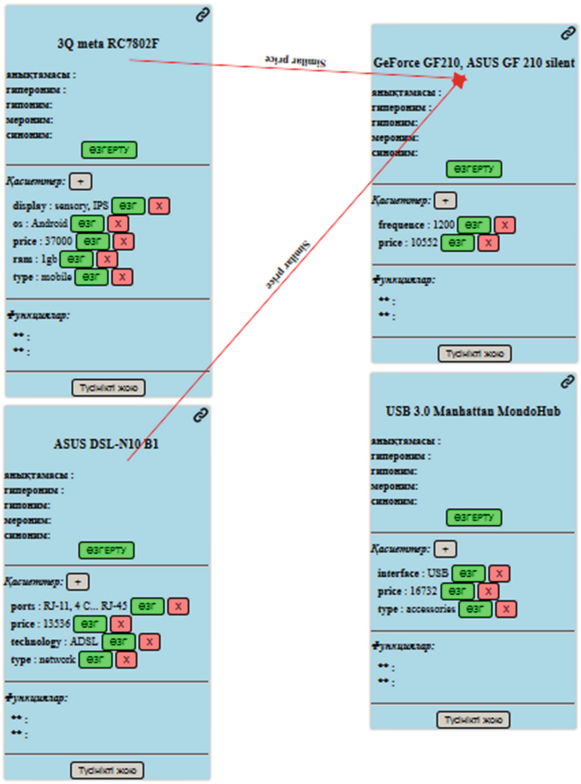

Fig. 5. Display data as XBSH.

```
^a(10000): ^a(10000)=0
^a(10000): ^a(10001)=1
^a(10000): ^a(10002)=2
^a(10000): ^a(10003)=3
^a(10000): ^a(10004)=4
^a(10000): ^a(10005)=5
^a(10000): ^a(10006)=6
^a(10000): ^a(10007)=7
^a(10000): ^a(10008)=8
^a(10000): ^a(10009)=9
нет пути до вершины ^a(77150)
^a(10000): ^a(10010)=10
```

Fig. 6. The result of the algorithm bypass (there is no way to the vertex ^a).

4 Evaluation Results

To evaluate the work of the results, we have increased the number of vertices (with properties and functions), the number of links between them and calculated the time for which will be set aside for all the vertices Dijkstra algorithm described above, the results

described in Table 1 and Figs. 7 and 8. Computer Options which conducted experiments Intel (R) Pentium (R) CPU B960 2.20 GHz 4 GB.

Table 1. The results of the experiment.

Experiment number	Vertexes count	Relations count	Time (seconds)
1	100	10	0,004955
2	1000	100	0,241899
3	5000	200	0,875635
4	10000	300	1,679333
5	15000	400	1,996542
6	20000	500	5,714703
7	40000	500	5,439382
8	60000	500	5,053411
9	80000	500	5,228277
10	100000	500	5,718899
11	100000	600	5,699353
12	100000	700	9,828479
13	100000	800	11,127078
14	100000	900	9,00501
15	100000	1000	10,054957

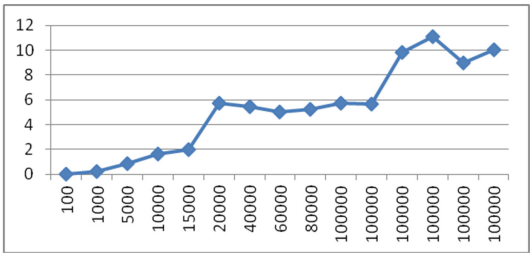

Fig. 7. Schedule of execution of the algorithm, the X-axis - the number of vertices, Y-axis - time to complete in seconds.

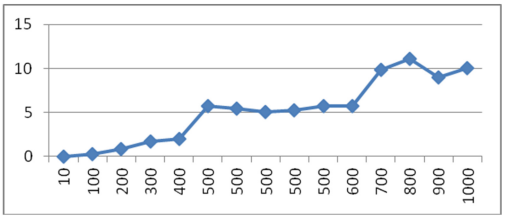

Fig. 8. Schedule of execution of the algorithm, the X-axis - the number of relations between nodes, Y-axis - time to complete in seconds.

As we can see the first graph (change the number of vertices) and the second graph (change the number of relations between nodes) have a similar circuit. That is, increasing the number of vertices and an increase in the number of relations between nodes have a similar circuit time interval changes to the algorithm, the changes here and there almost lonely affect turnaround time. For example, in experiments 10-13 the number of vertices is not changed, but the number of relations between them has grown, so has increased the time to process the data by the algorithm. Also, there is another example in 6-10 experiments increased the number of vertices and the number of relations between them are constant, it is sometimes even resulting in lower operating time algorithm. Then spend the calculation of the dependencies between the changes in the number of vertices, the number of connections between nodes and time for data processing algorithm. The most widely known Pearson's correlation coefficient, of the degree of linear correlation between variables. It is defined as: $r_{xy} = \frac{\sum(x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \times \sum (y_i - \bar{y})^2}}$, where x_i – variable value X; y_i – variable value Y; $\bar{x}$ – the arithmetic mean of the variable X; $\bar{y}$ – the arithmetic mean of the variable *Y*. Then try to make a prediction on the performance. To calculate the correlation coefficients 1 and 2 of the schedule were selected (Figs. 7 and 8). Pearson's correlation coefficient for the first schedule: 0.8737. Pearson correlation coefficient of the second schedule: 0.9458. With 100 000 000 (100 million) of vertices, the algorithm is similar to the car to complete its work ≈ 7500 s. At 1 000 000 (one million) links between nodes, an algorithm similar to the car to complete its work ≈ 14,000 s. For Dijkstra algorithm is critical is not the number of vertices in the graph, and the number of relations between the nodes. The figure of 14,000 s, an excellent result with a small performance computing. This was achieved by structuring knowledge in frames, thus the number of nodes has been reduced, and the semantics of the left and had a great expressive power. For each vertex in XBSH it is necessary create a thesaurus. It will be used in a smart learning and smart control of knowledge. These data hyperons, hyponym, synonym Meron and will give us in the future latent semantic links between nodes and their properties. They need to use advanced methods of processing knowledge bases.

5 Conclusion

The proposed method of storage (XBSH) will reduce the number of vertices in a semantic network, thereby increasing computing performance. Usually there is a certain subject area of are not more than 1,000 objects and subjects. That is, too much into the depth and width of 1000 peaks will take a little time from seconds to 0.241899 10.054957 s. In future work, we propose a new data search method will improve the client part of the input data to the knowledge base, will improve knowledge search method (search by name, by the properties, on relations between the nodes), connect nlp tools for self-learning knowledge base (autocomplete knowledge through text analysis natural language), we will try to enhance the expressive power of XBSH (set the weight on the ratio and the properties using the mathematical apparatus of fuzzy logic).

References

1. Giudici, P., Heckerman, D., Whittaker, J.: Statistical models for data mining. Data Min. Knowl. Disc. **5**(3), 163–165 (2001)
2. Ishida, T.: An optimization algorithm for production systems. IEEE Trans. Know. Data Eng. **6**(4), 549–558 (1994)
3. Strobbe, M., Van Laere, O., Dhoedt, B., De Turck, F., Demeester, P.: Hybrid reasoning technique for improving context-aware applications. Knowl. Inf. Syst. **31**(3), 581–616 (2012)
4. Chan, C.W.: From knowledge modeling to ontology construction. Int. J. Softw. Eng. Knowl. Eng. **14**(06), 603–624 (2004)
5. Lera, I., Juiz, C., Puigjaner, R.: Performance-related ontologies and semantic web applications for on-line performance assessment of intelligent systems. Sci. Comput. Program. **61**(1), 27–37 (2006)
6. Uckan, Y.: Knowledge representation using views in relational deductive data bases. J. Syst. Softw. **15**(3), 217–232 (1991)
7. Vogel-Heuser, B., Fay, A., Schaefer, I., Tichy, M.: Evolution of software in automated production systems: Challenges and research directions. J. Syst. Softw. **110**, 54–84 (2015)
8. Karimi, J., Zand, M.K.: Asset-based system and software system development – a frame-based approach. Inf. Softw. Technol. **40**(2), 69–78 (1998)
9. Ivancheva, N.A, Ivancheva, T.A.: Postrelational DBMS Caché. In: Novosibirsk 2004, Novosibirsk State University, Higher College of Informatics, pp. 83–93 (2004)