Metagraph Approach to the Information-Analytical Systems Development

Yuriy E. Gapanyuk
candidate of technical sciences, associate professor
Bauman Moscow State Technical University,
Baumanskaya 2-ya, 5, postcode 105005, Moscow, Russia
gapyu@bmstu.ru

Abstract: The paper discusses the peculiarities of using the metagraph approach for the development of the information-analytical systems. The metagraph information space is used for the unified semantic description of the information-analytical system. The metagraph multidimensional data model allows storing in hypercube facts and aggregates not only numerical values but any complex data structures that may be represented in the form of a metagraph. The metagraph process aimed for metagraph multidimensional data handling may be considered as a metagraph metaedge based on active metagraph.

Keywords: Information-Analytical System, Metagraph Information Space, Metaprogramming, Metagraph, Metavertex, Metaedge, Active Metagraph, Metagraph Agent, Metagraph Multidimensional Data Model, Metagraph Process.

1 Introduction

Modern information systems traditionally use heterogeneous information models of data, knowledge, and processes. This situation has developed historically, because previously, the power of computing systems was low, and the processing power of the information model was put at the forefront. The issues of integration of information systems and the unification of information models remained in the background.

The situation has now changed. The emergence and active development of big data processing technologies, the expansion of the range of information, and analytical tasks have led to the fact that the processed data may well be knowledge, situations, processes. This requires new approaches to system integration and information models.

Service-oriented approach (including its modern microservice version), to a certain extent, solves the problem of integration of information systems, but each system functions as a black box. Integration is possible only at the level of elements that are placed in the service interface.

The multiagent approach can be considered as an extension of the service-oriented approach. This approach is proposed to be used for industry 4.0 [1], product life cycle management [2, 3], robotics systems [4], including social robotics [5]. The multiagent approach suggests that we should use an agent knowledge model that can be based on the ontological approach, but at the same time, be OLAP-like.

For example, if we want to use an OLAP-like model to store and aggregate not numbers, but situations or processes, in order to solve informational-analytical tasks, using a service-oriented approach will not help, integration at the level of a unified information model is required.

In this article, the metagraph model is considered as such a model, which allows describing complex data and processes within the unified approach.

This article develops the ideas of the article [6], in which it was proposed to use the metagraph approach for multidimensional-ontological data model. However, in this article, we will try to give holistic description of the proposed approach, which will use the concept of the metagraph information space.

The article is organized as follows. The formal definition of information-analytical system is given. The metaprogramming peculiarities in the information-analytical systems are discussed. A brief description of the metagraph model is given. The unified semantic description of the information-analytical system is proposed based on the metagraph approach.

2 The Information-Analytical System

According to [7], the "Information-Analytical System" is "an automated system which carries out the storage, processing, analysis, and provision of information in a user-friendly form."

Based on this verbal definition, we can offer the following formal definition of an information-analytical system:

$$IAS = \langle IAS_{DATA}, IAS_{PROC}, IAS_{INTR} \rangle, \quad (1)$$

where IAS – information-analytical system; IAS_{DATA} – the data and knowledge component of information-analytical system, responsible for the "storage of information"; IAS_{PROC} – the information processes component of information-analytical system, responsible for "processing and analysis of information"; IAS_{INTR} – the user interface component of information-analytical system, responsible for "input and provision of information in a user-friendly form."

Currently, a considerable number of technologies have been developed that allow implementing these components:

- For the IAS_{DATA} component, there are relational and non-relational DBMS, RDF-based ontology storage tools. The most commonly used tool for the information-analytical system is OLAP technology [8], based on multidimensional data representation.
- For the IAS_{PROC} component, there are programming languages, including specialized stored procedure languages for relational, non-relational, and multidimensional data. Nowadays, workflow and rule-based programming approaches are gaining popularity. For example, in the Drools system, it is possible to operate combined processing using the workflow technology approach and a rule-based programming approach. The advantages of a rules-based approach include flexibility. In this case, the program is not hardcoded but forward chained with rules based on the data. The disadvantages include the possibility of rules cycling and the complexity of processing a large set of rules. Nowadays, for the processing of a large set of rules, the Rete algorithm and its modifications are used.
- For the IAS_{INTR} component, there are a considerable number of user interface technologies, based on desktop, web, and mobile approaches. Tools for dashboards and reports building are most commonly used in information-analytical systems.

Such a technology zoo creates serious problems in the development and especially with the support of information-analytical systems. Each technology is dealt with by a separate group of specialists, for the successful implementation of the project between the groups should be established successful cooperation.

One of the ways to reduce the technology zoo is the use of metaprogramming.

3 The Metaprogramming in the Information-Analytical Systems

Currently, the metaprogramming facilities in information systems development are used enough in isolation. For example, using web frameworks, it is possible to automatically generate simple forms of data input using the scaffolding mechanism. However, if it is necessary to create complex data entry forms for a Single Page Application (SPA), then such scaffolding tools will have to be created manually.

Modern languages and programming tools, with rare exceptions, almost do not use homoiconicity in the sense of Lisp. Due to homoiconicity, in Lisp, a program fragment can be considered as a data set for another program fragment.

According to [9]: "In homoiconic languages, all code can be accessed and transformed as data, using the same representation. This property is often summarized by saying that the language treats code as data principle."

However, it should be noted that homoiconicity in Lisp is a tool built into the programming language. Gradually, the idea began to emerge in the developer community that a mechanism like homoiconicity should not be built into a particular programming language, but rather a broader one. It is possible to create a language for describing data or structures with a homoiconic syntax in order to build on it different variants of semantics and apply in different technologies.

The first version of this language was XML, which appeared in the late 1990s. Currently, XML is still widely used in many areas but is gradually being superseded by the use of JSON. The emergence of XML was a huge step forward in terms of not language-oriented, but technological and architectural homoiconicity:

1. First of all, a large number of markup languages appeared to describe various subject areas. Examples of such languages are MathML (Mathematical Markup Language), CML (Chemical Markup Language). The XML namespaces were added to the XML standard to distinguish elements and attributes with the same name but different semantics.
2. The Native XML Databases (NXD) [10] intended for storing XML documents may be considered as a renaissance of hierarchical DBMS. This technology may be considered as part of the IAS_{DATA} component.
3. An XML schema technology [11] allows to describe and validate the structure of XML documents. This technology is homoiconic. The XML schema is an XML document, which elements and attributes are defined in the specific XML namespace. This technology may be considered as part of IAS_{DATA} component.
4. An XSLT technology [12] allow to transform XML document in either XML document, HTML document or text document. This technology is homoiconic. The XSLT transformation is an XML document, which elements and attributes are defined in the specific XML namespace. This technology may be considered as part of IAS_{PROC} component.

5. An XQuery [13] is a query language allowing to query and process XML data stored in Native XML Databases. This query language itself is not homoiconic but is has the homoiconic variant of syntax representation, which is called XQueryX [14]. This technology may be considered as part of the IAS_{PROC} component.
6. An XML Process Definition Language (XPDL) [15] allows defining workflow processes and interchange business process definitions between different workflow products. This technology may be considered as part of the IAS_{PROC} component.
7. An XForms technology [16] allows describing input forms for user input. Historically, XForms technology was one of the first attempts of the SPA approach. This technology is very complicated and is not "rich" enough compared to modern JavaScript frameworks and libraries such as Angular, Vue.js, React, and others. Therefore, XForms technology is currently used rarely. Nevertheless, this technology is also homoiconic. Its elements and attributes are defined in the specific XML namespace. This technology may be considered as part of the IAS_{INTR} component.

As an example of architectural homoiconicity, we can consider the XRX architecture [17]. The consideration of XRX architecture is partly based on our article [18]. XRX stands for "XForms–REST–XQuery." XRX is an example of "zero-translation architecture." It means that there is no translation (data conversion) between server and client data format. XML used both on server and client sides and for data transfer between server and client. Thus, XRX architecture is simple and friendly for the developer.

Consider the classical three-tier architecture of web application represented in Figure 1. This architecture contains three tiers: presentation tier, application tier, and data tier.

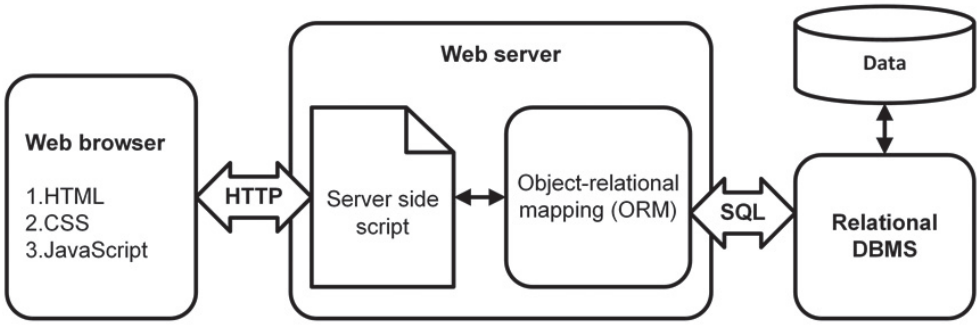

Figure 1 – The classical three-tier architecture of Web application

The main problem with this architecture is that it is required to use many different technologies at the same time to develop a web application, which increases the complexity of the development.

At the presentation tier can be used: HTML, cascading style sheets (CSS), JavaScript. In the case of modern JavaScript frameworks, a significant portion of the web application is developed in JavaScript, and server-side scripts are mainly used to exchange data with the presentation tier.

At the application tier for server-side scripting, a procedural or object-oriented programming language can be used, such as Python, C#, or Java. The DBMS is accessed directly using SQL or using object-relational mapping libraries such as SQLAlchemy (Python), NHibernate (C#), or Hibernate (Java).

The data tier is a relational database. Some kind of stored procedure language can be used in the data tier.

This architecture causes an "object-relational impedance mismatch" problem. According to [19]: "The object-relational impedance mismatch is a set of conceptual and technical difficulties that are often encountered when a relational database management system (RDBMS) is being served by an application program written in an object-oriented programming language or style, particularly because objects or class definitions must be mapped to database tables defined by a relational schema." Obviously, this problem is caused by the fact that the server-side script and DBMS use different data representations and different approaches to data processing.

One of the solutions to this problem is an approach in which a single data model is used at all tiers, and the same data model is used in the translation of information between the tiers. This approach is called "zero-translation architecture". Figure 2 shows the generalized three-tier architecture of the XRX web application, which is an example of "zero-translation architecture."

The presentation tier is XForms-processor. XForms-processor receives XForms-document from Web-server, converts XForms-document to HTML+CSS+Javascript, provides user input, makes XML-fragment from inputted values and sends result XML-fragment back to Web-server.

The application tier is XQuery-script that runs on Web-server. This is the main tier that communicates with XForms-processor and Native XML Database.

The data tier is a Native XML Database that stores data in XML format.

Data transfer between Application tier and Presentation tier is XML, because XQuery-script sends to XForms-processor XForms-documents (that's XML), default XML data and receives inputted XML data. Data transfer between the application tier and the data tier is XML too because Native XML Database sends and receives data in XML format.

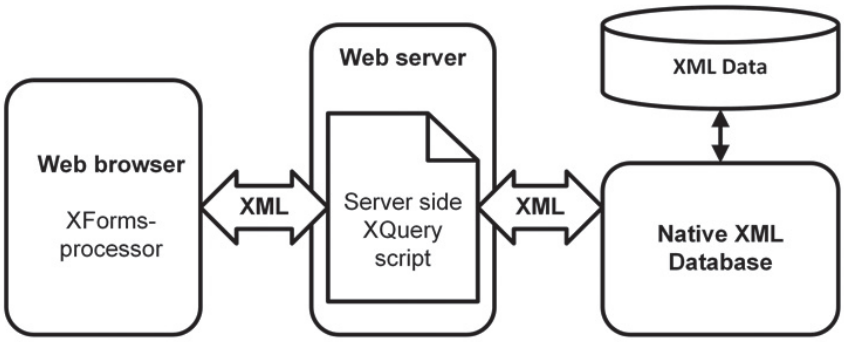

Figure 2 – The generalized XRX-architecture of Web application

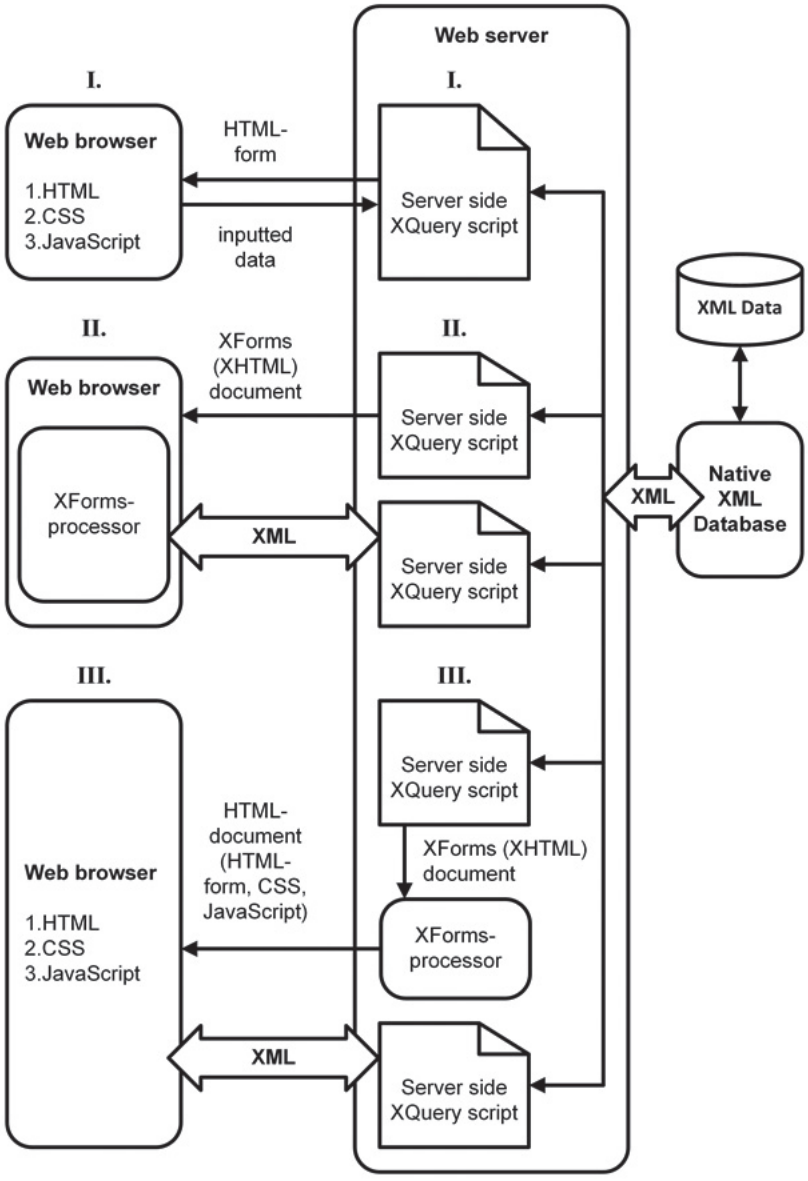

Figure 3 – The three variants of detailed XRX-architectures of Web application

Now consider the detailed XRX-architecture of web application represented in Figure 3. There are three variants of the detailed architecture. Variant I is an example of a "shallow XRX" architecture. This is the case where XRX stack is simplified; for example, traditional HTML forms are used instead of XForms.

Variants II and III are examples of "deep XRX" architecture when full XRX stack is used. Variant II is the case when XForms-processor is a client solution. Variant III is the case when XForms-processor is a web server application solution.

Because in the variant II XForms-processor is a client solution, it is very easy to create XForms-documents dynamically and transfer them to the client-side XForms-processor.

There are variants II and III that may be considered as examples of "zero-translation architecture." Comparing to the classical three-tier architecture of web-application there is no "object-relational impedance mismatch" problem in XRX architecture because XML is a single data model used in all tiers. Thus, the XML approach provides not only technological but also architectural homoiconicity.

XML is currently criticized for its redundant syntax, and JSON is increasingly used instead. However, some XML technologies are ported for use with JSON, or ideologically similar tools are created for JSON:

- JSON-based document-oriented databases are used instead of Native XML Databases. The examples of such DBMS are MongoDB, CouchDB.
- The JSON Schema technology [20] is ideologically based on XML schema technology.
- The JSONiq query language [21] is a ported XQuery language, adopted for JSON syntax.

Summing up, we can say that XML and JSON are two variants of the homoiconic syntax allowing to build on it different variants of semantics. Thus, these technologies make it possible to get rid of the "syntactic zoo" in information-analytical systems.

It should be noted that the "fight against the zoo" in practice is carried out exclusively by trial and error. Nobody can theoretically prove that any approach will completely solve the problem of the zoo in practice. For example, a very conceptual and unified XRX approach has not taken deep roots in practice. Thus, we can only propose an approach, in the hope that it will take root in practice.

This article attempts to overcome the "semantic zoo" in information-analytical systems. For this purpose, it is proposed to use a unified semantic model for *IAS*. We call this unified model "metagraph information space." The basis for unification is the metagraph approach, which is briefly described in the following section.

4 The Brief Description of the Metagraph Model

In this section, the metagraph model is briefly described, which is necessary for further understanding. Metagraph is a kind of complex network model, proposed by A. Basu and R. Blanning in their book [22] and then adapted for information systems description in our papers [6, 23].

4.1 The Metagraph Data Model Definitions

According to [6, 23] metagraph model may be described as follows:

$$MG = \langle V, MV, E, ME \rangle, \tag{2}$$

where MG – metagraph; V – set of metagraph vertices; MV – set of metagraph metavertices; E – set of metagraph edges; ME – set of metagraph metaedges.

Metaedge is an optional element of the metagraph model aimed for process description. In the case of the information-analytical system description, this element is used for IAS_{PROC} component description.

Metagraph vertex is described by a set of attributes:

$$v_i = \{atr_k\}, v_i \in V, \tag{3}$$

where v_i – metagraph vertex; atr_k – attribute.

Metagraph edge is described by a set of attributes, the source, and destination vertices and edge direction flag:

$$e_i = \langle v_S, v_E, eo, \{atr_k\} \rangle, e_i \in E, eo = true \mid false, \tag{4}$$

where e_i – metagraph edge; v_S – source vertex (metavertex) of the edge; v_E – destination vertex (metavertex) of the edge; eo – edge direction flag ($eo=true$ – directed edge, $eo=false$ – undirected edge); atr_k – attribute.

The metagraph fragment:

$$MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV \cup ME), \tag{5}$$

where MG_i – metagraph fragment; ev_j – an element that belongs to the union of vertices, edges, metavertices, and metaedges.

The metagraph metavertex:

$$mv_i = \langle \{atr_k\}, MG_j \rangle, mv_i \in MV, \tag{6}$$

where mv_i – metagraph metavertex belongs to set of metagraph metavertices MV; atr_k – attribute, MG_j – metagraph fragment.

Thus, metavertex, in addition to the attributes, includes a fragment of the metagraph. The presence of private attributes and connections for metavertex is a distinguishing feature of the metagraph. It makes the definition of metagraph holonic – metavertex may include a number of lower-level elements and in turn, may be included in a number of higher-level elements.

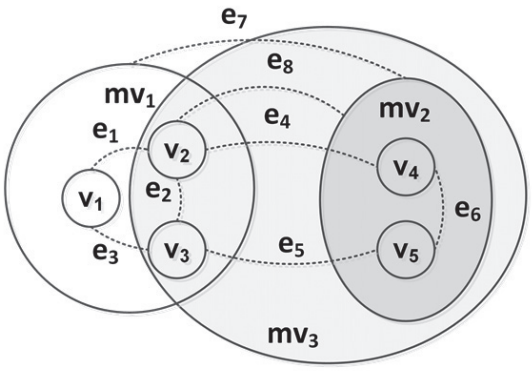

Figure 4 – The example of data metagraph

The example of data metagraph (shown in Figure 4) contains three metavertices: mv_1, mv_2, and mv_3. Metavertex mv_1 contains vertices v_1, v_2, v_3 and connecting them edges e_1, e_2, e_3. Metavertex mv_2 contains vertices v_4, v_5, and connecting them edge e_6. Edges e_4, e_5 are examples of edges connecting vertices v_2-v_4 and v_3-v_5 are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of the edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of the edge connecting vertex v_2 and metavertex mv_2. Metavertex mv_3 contains metavertex mv_2, vertices v_2, v_3, and edge e_2 from metavertex mv_1 and also edges e_4, e_5, e_8 showing holonic nature of the metagraph structure.

The vertices, edges, and metavertices are used for data description while the metaedges are used for process description. The metagraph metaedge:

$$me_i = \left\langle v_S, v_E, \{atr_k\}, MG_j \right\rangle, me_i \in ME, \tag{7}$$

where me_i – metagraph metaedge belongs to set of metagraph metaedges ME; v_S – source vertex (metavertex) of the metaedge; v_E – destination vertex (metavertex) of the metaedge; atr_k – attribute, MG_j – metagraph fragment.

It is assumed that a metagraph fragment contains vertices (or metavertices) as process nodes and connecting them edges. A metagraph fragment can also contain nested metaedges, which makes the description of the metaedge recursive.

The example of a directed metaedge is shown in Figure 5.

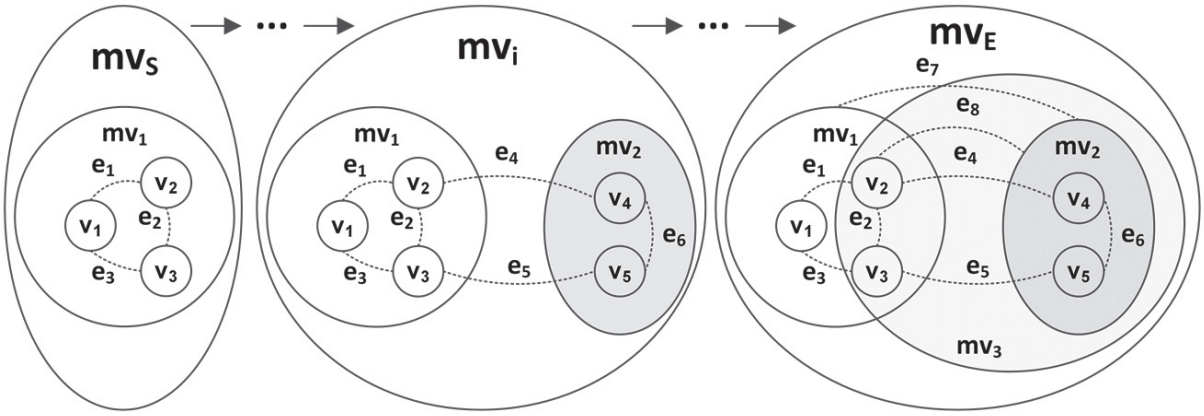

Figure 5 – The example of the directed metaedge

The directed metaedge contains metavertices mv_S, … mv_i, … mv_E and connecting them edges. The source metavertex contains a nested metagraph fragment. During the transition to the destination metavertex, the nested metagraph fragment became more complex, new vertices, edges, and inner metavertices are added. Thus, metaedge allows binding the stages of nested metagraph fragment development to the steps of the process described with metaedge.

4.2 The Metagraph Agents Definitions

The metagraph model is aimed for complex data descriptions. However, it is not aimed for data transformation. To solve this issue, the metagraph agent (ag^{MG}) aimed for data transformation is proposed. There are two kinds of metagraph agents: the metagraph function agent (ag^F) and the metagraph rule agent (ag^R). Thus $ag^{MG} = ag^F \mid ag^R$.

The metagraph function agent serves as a function with input and output parameter in the form of metagraph:

$$ag^F = \left\langle MG_{IN}, MG_{OUT}, AST \right\rangle, \tag{8}$$

where ag^F – metagraph function agent; MG_{IN} – input parameter metagraph; MG_{OUT} – output parameter metagraph; AST – abstract syntax tree of metagraph function agent in the form of metagraph.

The metagraph rule agent is rule-based:

$$ag^R = \left\langle MG, R, AG^{ST} \right\rangle, R = \{r_i\}, r_i : MG_j \to OP^{MG}, \quad (9)$$

where ag^R – metagraph rule agent; MG – working metagraph, a metagraph on the basis of which the rules of agent are performed; R – set of rules r_i; AG^{ST} – start condition (metagraph fragment for start rule check or start rule); MG_j – a metagraph fragment on the basis of which the rule is performed; OP^{MG} – set of actions performed on metagraph.

The antecedent of the rule is a condition over the metagraph fragment; the consequent of the rule is a set of actions performed on metagraph. Rules can be divided into open and closed.

The consequent of the open rule is not permitted to change the metagraph fragment occurring in rule antecedent. In this case, the input and output metagraph fragments may be separated. The open rule is similar to the template that generates the output metagraph based on the input metagraph.

The consequent of the closed rule is permitted to change the metagraph fragment occurring in rule antecedent. The metagraph fragment changing in rule consequent cause to trigger the antecedents of other rules bound to the same metagraph fragment. But incorrectly designed closed rules system can cause an infinite loop of metagraph rule agent.

Thus, the metagraph rule agent can generate the output metagraph based on the input metagraph (using open rules) or can modify the single metagraph (using closed rules).

4.3 The Active Metagraph Definition

In order to combine the data metagraph model and metagraph agent model, we propose the concept of "active metagraph":

$$MG^{ACTIVE} = \left\langle MG^D, AG^{MG} \right\rangle, AG^{MG} = \{ag_i\}, \quad (10)$$

where MG^{ACTIVE} – an active metagraph; MG^D – data metagraph; AG^{MG} – set of metagraph agents ag_i, attached to the data metagraph.

Thus, active metagraph allows combining data and processing tools for the metagraph approach. Similar structures are often used in computer science. As an example, we can consider a class of object-oriented programming language, which contains data and methods of their processing. Another example is a relational DBMS table with an associated set of triggers for processing table entries.

The main difference between an active metagraph and a single metagraph agent is that an active metagraph contains a set of metagraph agents that can use both closed and open rules. For example, one agent may change the structure of active metagraph using closed rules while the other may send metagraph data another active metagraph using open rules. Agents work independently and can be started and stopped without affecting each other.

5 The Unified Semantic Description of the Information-Analytical System

Based on the metagraph approach, in this section, we propose the unified semantic description of the information-analytical system. We call this unified model "metagraph information space." This model is based on the information-analytical system formal definition, formula (1). In this section, we consider how components of *IAS* are translated into the "metagraph information space."

5.1 The User Interface Component

It should be noted that in this article, we will not talk in detail about the user interface component of the information-analytical system (IAS_{INTR}). Input form definition can be based on any homoiconic technology similar to XForms.

We assume that for the generation of input forms based on metagraphs, is considered can be used metagraph function agents and metagraph rule agents. The AST (in case of function agent) or rules (in case of rule agent) of these agents are similar to XSLT transformations that generate XForms-forms based on the description of the data model in XML.

In further explanation, we consider such operations as "generating an input form based on a data description" and "entering data into a form" as atomic operations. At the same time, we omit entirely such issues as combining input forms with wizards, creating complex reports based on data, building dashboards. Undoubtedly, such complex forms of the user interface can be generated on the basis of metagraphs using metagraph agents, but this issue is the subject of separate research.

5.2 The Data and Knowledge Component

The most commonly used tool for the information-analytical system is OLAP technology, based on multidimensional data representation. This section develops the ideas of the article [6], in which it was proposed to use the metagraph approach for the multidimensional-ontological data model.

5.2.1 The Multidimensional Data Model

The classical multidimensional data model, proposed by Edgar F. Codd, allows working with numerical data (measures) binding them to the hierarchical taxonomies (dimensions) [8]. The multidimensional data model is a core for OLAP (online analytical processing) information systems.

Many variants of formalization of such a model were proposed, for example [24]. In this section, we use our own simplified version of the formalization according to [6], which will help to describe the proposed model. Multidimensional hypercube may be described as follows:

$$\begin{gathered} HC = \langle MSR, HCD, HCF, HCR \rangle, \\ MSR = \{msr_i\}, HCD = \{hcd_i\}, HCF = \{hcf_i\}, HCR = \{hcr_i\}, \end{gathered} \tag{11}$$

where HC – hypercube; MSR – set of hypercube measures (msr_i – measure); HCD – set of hypercube dimensions (hcd_i – dimension); HCF– set of hypercube facts (hcf_i – fact); HCR – set of hypercube aggregation rules (hcr_i – rule).

In the case of the classical multidimensional data model, it is assumed that a measure can store only numerical values.

Hypercube dimension:

$$hcd_i = \langle \{hcd_i^k\}, \prec \rangle, \tag{12}$$

where hcd_i^k – hypercube dimension element; $\prec$ – a partial order on the set of hypercube dimension elements.

In most cases, the hypercube dimension element is organized in a tree structure, in case of the time dimension, e.g., year → month →week → day. However, partial order organization is more suitable then tree structure organization because partial order organization allows describing ragged hierarchies, in case of the time dimension, e.g. the month → week → day and month → decade → day hierarchies are allowed to exist simultaneously in one dimension.

Hypercube fact:

$$hcf_j = \langle \{hcd_i^{ref}\}, \{msr_n\} \rangle, \tag{13}$$

where hcd_i^{ref} – a reference to the dimension element; msr_n – measure.

In the case of low-level hypercube fact, the set $\{hcd_i^{ref}\}$ contains references to low-level elements of all hypercube dimensions. In the case of aggregated hypercube fact $\{hcd_i^{ref}\} \in P(HCD)$, the set $\{hcd_i^{ref}\}$ belongs to the powerset of all hypercube dimensions because aggregation rules may exclude dimensions during the aggregation process. Simultaneously, during aggregation, elements hcd_i^{ref} roll up upon their hierarchies, providing data aggregation on higher levels of hierarchies.

Hypercube aggregation rule:

$$hcr_k : \{hcf_{OUT}\} = agf(\{hcf_{IN}\}, HCD^{ag}), HCD^{ag} \subset HCD, \tag{14}$$

where hcf_{OUT} – output (aggregated) facts; agf – aggregation function; hcf_{IN} – input (non-aggregated) facts; HCD^{ag} – a subset of hypercube dimensions used in aggregation.

Aggregation rules allow calculating aggregated facts on the base of non-aggregated or low-level aggregated facts and hypercube dimensions. The typical aggregation functions are *count*, *sum*, *min*, *max,* and other numerical functions. Depending on the multidimensional system implementation, aggregation rules may be bound to the particular dimensions or the whole hypercube.

Today the multidimensional model is used in a significant number of information-analytical system, and its advantages are worldwide recognized. However, multidimensional model is oriented for numerical measures usage. Textual or object-oriented information is not considered for use as measures. This may be noted as a limitation of a classical multidimensional model.

An approach called "Graph OLAP" is currently being developed [25, 26]. This idea is somewhat similar to ROLAP (Relational OLAP) approach, but instead of the relational model, the graph model is used. Instead of the hypercube aggregation operation, the graph aggregation operation (aggregate network) is used [26].

Thus, the "Graph OLAP" approach is an attempt to adapt the standard multidimensional model to graph data. This approach inherits the main problem of the classical multidimensional model. It does not allow changing the type and structure of the data in the aggregation process. To overcome this problem, it is proposed to use the metagraph approach.

5.2.2 The Metagraph Representation of the Multidimensional Data Model

In this subsection, several definitions of the multidimensional data model will be further redefined to match the metagraph information space. Let us call this model the "metagraph multidimensional data model" (HC^{MG}).

In the case of the metagraph information space, the measure is a metagraph fragment:

$$msr_i \equiv MG_j, \tag{15}$$

where msr_i – measure; MG_j – metagraph fragment.

This means that a hypercube cell can contain not only a numeric value but any complex data structure described by the metagraph.

According to formula (12), the hypercube dimension:

$$hcd_i = \left\langle \left\{ hcd_i^k \right\}, \prec \right\rangle, hcd_i \in MV, hcd_i^k \in (V \cup MV), \tag{16}$$

where hcd_i – hypercube dimension; hcd_i^k – hypercube dimension element; $\prec$ – a partial order on the set of hypercube dimension elements; MV – set of metagraph metavertices; V – set of metagraph vertices.

The hypercube dimension may be represented in the form of a hierarchically organized metavertex. The hypercube dimension elements that correspond to leaves of the tree can be represented as vertices, while the elements of the higher levels as metavertices.

According to formulae (13, 15, 16), the hypercube fact:

$$hcf_j = \left\langle \left\{ hcd_i^{ref} \right\}, \left\{ msr_n \right\} \right\rangle, hcd_i^{ref} \in (V \cup MV), msr_n \equiv MG_j, \tag{17}$$

where hcd_i^{ref} – reference to the dimension element; msr_n – measure; MV – set of metagraph metavertices; V – set of metagraph vertices; MG_j – metagraph fragment.

The hypercube aggregation rule in the metagraph information space corresponds to formula (14). However, instead of aggregation function, the metagraph agent is used for aggregation:

$$hcr_k : \left\{ hcf_{OUT} \right\} = ag^{MG} \left(\left\{ hcf_{IN} \right\}, HCD^{ag} \right), HCD^{ag} \subset HCD, \tag{18}$$

where hcf_{OUT} – output (aggregated) facts; ag^{MG} – the metagraph agent used for aggregation; hcf_{IN} – input (non-aggregated) facts; HCD^{ag} – the subset of hypercube dimensions used in aggregation.

The aggregation example is represented in Figure 6. There is a simple hypercube with two dimensions hcd_1 and hcd_2. The hypercube facts corresponds to the hypercube dimension elements combinations hcd_1^{11}-hcd_2^{11}, hcd_1^{11}-hcd_2^{12}, hcd_1^{12}-hcd_2^{11}, hcd_1^{12}-hcd_2^{12} are lower-level hypercube facts. The combination hcd_1^{1}-hcd_2^{1} corresponds to the aggregated hypercube fact. In the process of aggregation, not only quantitative characteristics change but also the metagraph structure of cells corresponding to the facts of the hypercube.

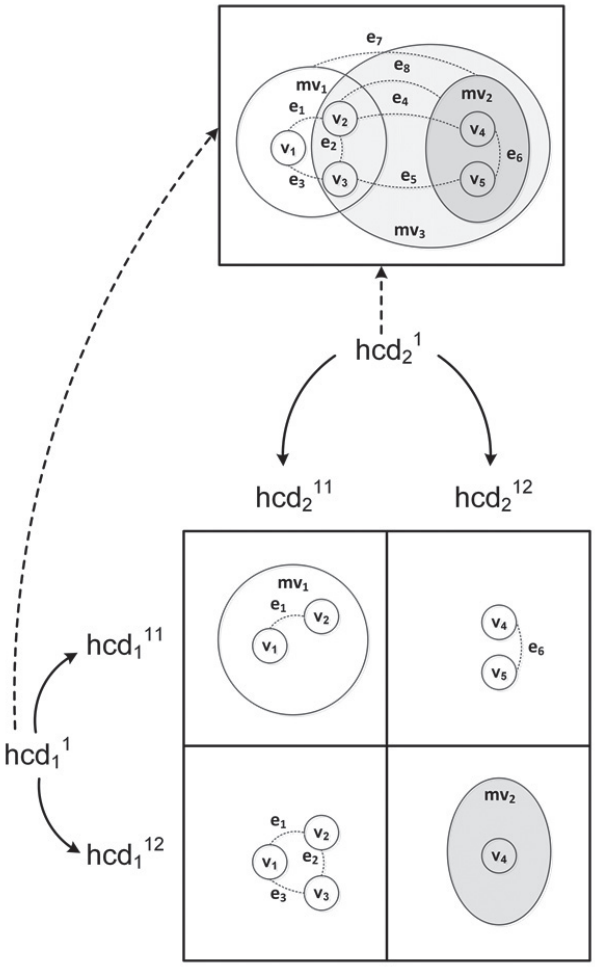

Figure 6 – The example of the aggregation

It should be noted that according to formula (10), the combination of input hypercube fact and set of corresponding agents from the hypercube aggregation rules may be considered as an active metagraph. Let us call this combination "active fact":

$$hcf^{ACTIVE} = \left\langle hcf_{IN}, \left\{ ag^{MG} \right\} \right\rangle \tag{19}$$

In formula (18), the ag^{MG} agent may aggregate many input facts. The formula (19) considers the situation that one input fact may be used in many aggregation rules.

The three essential conclusions can be drawn from this model:

1. The proposed approach allows storing in hypercube facts and aggregate not only numerical values but any complex data structures. This allows working with data, knowledge, situations, processes descriptions represented in the form of metagraph.
2. In one data metagraph, it is possible to distinguish an arbitrary number of hypercubes. A hypercube may not be the entire data metagraph, but a fragment of it.
3. In the proposed approach, agents used in aggregation rules may be considered as somewhat similar to database triggers. However, these agents perform an aggregation function and cannot fully implement the data processing in the information-analytical system.

5.3 The Information Processes Component

To implement data processing, we propose to use the "metagraph processes" ($PROC^{MG}$), which is the set of processes ($PROC_i$): $PROC^{MG} = \{PROC_i\}$.

The "metagraph process" may be considered as a metagraph metaedge based on active metagraph. Based on formula (7) we use an active metagraph instead of data metagraph as a process node:

$$PROC_i = \left\langle v_S, v_E, \{atr_k\}, MG_j \left[mv_{node} \equiv MG_{node}^{ACTIVE} \right] \right\rangle, mv_{node} \in MV, \tag{20}$$

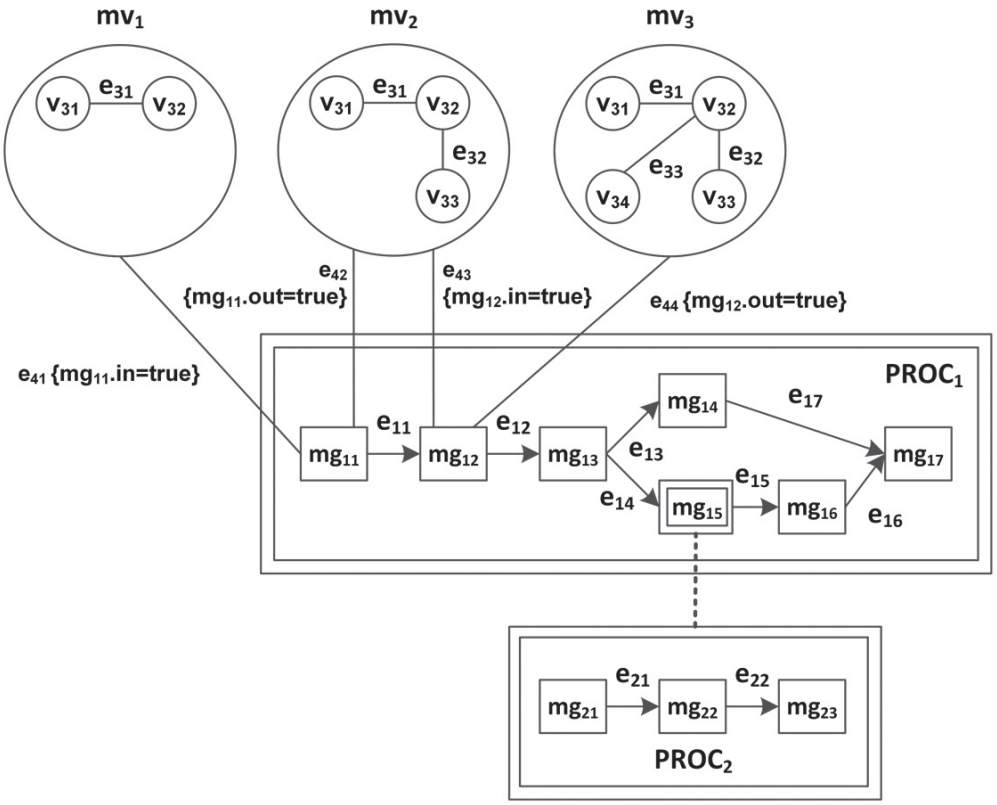

Figure 7 – The example of the metagraph processes description

The example of the metagraph processes description is represented in Figure 7. The vertices and metavertices used for data descriptions are shown with circles. Active metagraphs mg_* corresponding to the metagraph process elements are shown with rectangles. The metagraph processes $PROC_1$ and $PROC_2$ are shown with double rectangles. The dashed link shows the call of the nested metagraph process $PROC_2$ from the active metagraph element mg_{15}. The directed edges show the relationship between metagraph process elements. The undirected edges show the relationship between data elements or the relationship between data elements and active metagraphs.

The input data for the mg_{11} element are shown as metavertex mv_1, which contains vertices v_{31}, v_{32}, and connecting them edge e_{31}. The connection between the metavertex mv_1 and the process element mg_{11} is performed using the edge e_{41}. The attribute mg_{11}.in = true means the semantic of input data. Similarly, using the edges e_{42}, e_{43}, e_{44}, the metavertices mv_2 and mv_3 are connected with active metagraph elements mg_{11} and mg_{12} as input-output data.

It should be noted that the description of the metagraph process is homoiconic. It is represented in the form of a metagraph and can be modified by a higher-level metagraph agent. It also may be stored in "metagraph multidimensional data model."

The operations of metagraph agents associated with the process elements can perform the following classes of actions:

- DDL (Data Definition Language) operations. Definition and modification of the elements of the "metagraph multidimensional data model" (HC^{MG}). Definition of dimensions, measures, aggregation rules.
- DML (Data Manipulation Language) operations. Metagraph data transfer between different parts of the metagraph hypercube. Perform non-standard aggregation operations.
- User interface operations such as "generating an input form based on a data description" or "entering data into a form."
- Homoiconic operations related to metagraph processes modification.

5.4 The Metagraph Information Space

To sum up this section, let us formally define the "metagraph information space" (*MIS*):

$$MIS = \left\langle HC^{MG}, PROC^{MG}, MIS_{INTR} \right\rangle, HC^{MG} \leftrightarrow IAS_{DATA}, PROC^{MG} \leftrightarrow IAS_{PROC}, MIS_{INTR} \leftrightarrow IAS_{INTR} \quad (21)$$

The definition of "metagraph information space" corresponds to the definition of the "information-analytical system," formula (1). The data and knowledge component of information-analytical system (IAS_{DATA}) is implemented as "metagraph multidimensional data model" (HC^{MG}), while the information processes component (IAS_{PROC}) is implemented as "metagraph processes" ($PROC^{MG}$). The metagraph implementation (MIS_{INTR}) of the user interface component of the information-analytical system (IAS_{INTR}) is not considered in details in this article.

6 Conclusions

The information-analytical system consists of three components: the data and knowledge component, the information processes component, and the user interface component.

One of the ways to reduce the technology zoo during the information-analytical system is the use of metaprogramming, especially homoiconicity. Languages like XML or JSON allows describing structures with a homoiconic syntax in order to build on it different variants of semantics and apply in different technologies. These technologies make it possible to get rid of the "syntactic zoo" in information-analytical systems. This article attempts to also overcome the "semantic zoo" in information-analytical systems. For this purpose, it is proposed to use a unified semantic model based on the metagraph approach.

The key element of the metagraph data model is metavertex. The metavertex, in addition to the attributes, includes a fragment of the metagraph. The presence of private attributes and connections for metavertex is a distinguishing feature of the metagraph model. It makes the definition of metagraph holonic – metavertex may include a number of lower-level elements and in turn, may be included in a number of higher-level elements. Metaedges are purposed for the process description. Using metagraph agents, it is possible either to generate the output metagraph based on the input metagraph (using open rules) or to modify the metagraph (using closed rules). An active metagraph combines the data metagraph and metagraph agents.

The metagraph information space consists of the metagraph multidimensional data model and the metagraph process model.

The metagraph multidimensional data model allows storing in hypercube facts and aggregate not only numerical values but any complex data structures. This allows working with data, knowledge, situations, processes descriptions represented in the form of metagraph.

The metagraph process may be considered as a metagraph metaedge based on active metagraph. The description of the metagraph process is homoiconic. It is represented in the form of a metagraph and can be modified by a higher-level metagraph agent. It also may be stored in the metagraph multidimensional data model.

The proposed metagraph approach allows representing the main components of the information-analytical system model in a homoiconic way.

References

[1] V.B. Tarassov. Enterprise total agentification as a way to industry 4.0: Forming artificial societies via goal-resource networks. In: Intelligent Information Technologies for Industry 2018, AISC, vol. 874, pp. 26-40. Springer, 2018.

[2] V. Taratukhin, Y. Yadgarova. Towards a socio-inspired multiagent approach for new generation of product life cycle management. Procedia Computer Science 123:479-487, August 2017.

[3] V.O. Karasev, V.A. Sukhanov. Product Lifecycle Management Using Multi-agent Systems Models. Procedia Computer Science 103:142-147, October 2016.

[4] A.V. Nazarova, M. Zhai. Distributed Solution of Problems in Multi Agent Robotic Systems. Studies in Systems, Decision and Control 174: 107-124, 2019.
[5] V.E. Karpov, V.B. Tarassov. Synergetic artificial intelligence and social robotics. In Intelligent Information Technologies for Industry 2017, AISC, vol. 679, pp. 3-15. Springer, 2017.
[6] V.M. Chernenkiy, Yu.E. Gapanyuk, A.N. Nardid, A.V. Gushcha, Yu.S. Fedorenko. The Hybrid Multidimensional-Ontological Data Model Based on Metagraph Approach. In: A.K. Petrenko, A. Voronkov (eds.) Perspectives of systems informatics. 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers / edited by Alexander K. Petrenko, Andrei Voronkov, vol. 10742. Lecture notes in computer science, 0302-9743, vol. 10742, pp. 72–87. Springer (2018). doi: 10.1007/978-3-319-74313-4_6
[7] T.I. Buldakova, S.I. Suyatinov. The Significance of Interdisciplinary Projects in Becoming a Research Engineer. In: E.V.Smirnova, R.P.Clark (eds.) Handbook of Research on Engineering Education in a Global Context. pp. 243-253. Hershey, PA: IGI Global (2019). doi:10.4018/978-1-5225-3395-5
[8] Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical report, E.F.Codd & Associates, 1993.
[9] Homoiconicity. [Online]. Available: https://en.wikipedia.org/wiki/Homoiconicity
[10] G.M. Pavlovic-Lazetic. Native XML databases vs. relational databases in dealing with XML documents. Kragujevac Journal of Mathematics 30:181-199, January 2007.
[11] W3C XML Schema Definition Language. W3C Recommendation 5 April 2012. [Online]. Available: https://www.w3.org/TR/xmlschema11-1/
[12] XSL Transformations (XSLT) Version 3.0. W3C Recommendation 8 June 2017. [Online]. Available: https://www.w3.org/TR/xslt-30/
[13] XQuery 3.1: An XML Query Language. W3C Recommendation 21 March 2017. [Online]. Available: https://www.w3.org/TR/xquery-31/
[14] XQueryX 3.1. W3C Recommendation 21 March 2017. [Online]. Available: https://www.w3.org/TR/xqueryx-31/
[15] Process Definition Interface - XML Process Definition Language, Version 2.2. The Workflow Management Coalition Specification 30 August 2012. [Online]. Available: http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf
[16] XForms 2.0. W3C Working Draft 7 August 2012. [Online]. Available: https://www.w3.org/TR/xforms20/
[17] XRX (web application architecture). [Online]. Available: https://en.wikipedia.org/wiki/XRX (web application architecture)
[18] Yu. Gapanyuk, E. Lakomkin, S. Ionkin, M. Davtyan. MVC web framework based on eXist application server and XRX architecture. In: SYRCoDIS 2011. Proceedings of the Seventh Spring Researchers Colloquium on Databases and Information Systems Moscow, Russia, June 2-3, 2011. [Online]. Available: http://ceur-ws.org/Vol-735/paper4.pdf
[19] Object-relational impedance mismatch. [Online]. Available: https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch
[20] JSON Schema Specification. [Online]. Available: https://json-schema.org/specification.html
[21] The JSON Query Language [Online]. Available: http://jsoniq.org/
[22] A. Basu, R.W. Blanning. Metagraphs and Their Applications. Springer, 2007.
[23] V.M. Chernenkiy, Yu.E. Gapanyuk, G.I. Revunkov, Yu.T. Kaganov, Yu.S. Fedorenko, S.V. Minakova. Using metagraph approach for complex domains description. In: Selected Papers of the XIX International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2017). Moscow, Russia, October 9-13, 2017. [Online]. Available: http://ceur-ws.org/Vol-2022/paper52.pdf
[24] S. Mansmann, M.H. Scholl. Extending the Multidimensional Data Model to Handle Complex Data. Journal of Computing Science and Engineering, 1(2):125-160, December 2007.
[25] A. Ghrab, O. Romero, S. Skhiri, A. Vaisman, E. Zimanyi. A framework for building OLAP cubes on graphs. In: T. Morzy, V. Patrick, B. Ladjel (eds.) Advances in Databases and Information Systems. Proceedings of the 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015. Lecture notes in computer science, vol. 9282, pp. 92-105. Springer (2015). doi: 10.1007/978-3-319-23135-8_7
[26] P. Zhao, X. Li, D. Xin, J. Han. Graph Cube: On Warehousing and OLAP Multidimensional Networks. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, pp. 853-864. ACM New York (2011). doi: 10.1145/1989323.1989413