Building Subject Domain Ontology for a Corporate Web Application

A. Gusenkov[1][0000-0003-4019-7322], N. Bukharaev[2][0000-0002-4997-2121] and E. Birialtsev[3][0000-0002-5193-8627]

[1,2] Kazan Federal University, Kazan, Russia
[3] Institute of Applied Research of the Academy of Sciences of the Republic of Tatarstan, Kazan, Russia
[1]gusenkov.a.m@gmail.com, [2]boukharay@gmail.com, [3]igenbir@yandex.ru

Abstract. The technology of automated construction of the subject domain ontology, based on information extracted from the comments of the TATNEFT oil company relational databases, is considered. The technology is based on building a converter (compiler) translating the logical data model of Epicenter Petrotechnical Open Software Corporation (POSC), presented in the form of ER diagrams and a set of the EXPRESS object-oriented language descriptions, into the OWL ontology description language, recommended by the W3C consortium. The basic syntactic and semantic aspects of the transformation are described.

Keywords: Subject Domain Ontology, Relational Databases, POSC, OWL

1 Introduction

Creating of an ontology, i.e. formal model of semantics of some subject domain usually is an extremely laborious process. It requires participation of highly qualified specialists both in the specific subject field and in the field of computer linguistics. One of the methods widely used here is the conceptualization technique [1]. Using it, we actually build an object model of some subdomain of the real world by defining objects, their attributes and relationships. A similar technique for basic entities and relationships extraction is used to define logical models in the relational databases design [2]. The methodological proximity of the techniques used in the ontologies and database models development suggests possibility to use already existing logical database models as a formalized prototype of the subject domain ontology.

In the article such technique for automated construction of the subject domain ontology from its relational model is described on the real world example of development of an intelligent search system for a large oil producing company.

If at all possible, the most natural way to select a prototype of the ontology is to choose one among the logical models of the subject area, having an industry standard status. In our case the most evident choice was the Epicentre data model of the Petrotechnical Open Software Corporation (POSC [3]). It is presented in the form of

ER-diagrams [2] considered together with the set of text files in the EXPRESS object-oriented language (ISO 10303, part 11). The definition also has visual presentation focused on the effective generation of database structures according to its logical model by IT specialists.

To describe the ontology, the Web Ontology Language (OWL) [4, 5], developed by the Semantic Web Activity working group and recommended by the W3C international consortium [6], was chosen. With the prospect of further move towards a logical inference system development, the actual implementation of the ontology was performed in the OWL DL language dialect, corresponding to the rules of descriptive logic. The article describes the scheme for converting the Epicentre logical model into the OWL DL language, which apart the industry-wide standards also takes into account the local specifics of the Tatneft oil and gas company.

2 Epicentre Model

The Epicentre data model defines over 1000 real-world technical and business concepts related to the oil exploration and production. In the terminology of POSC data modeling the corresponding objects are called entities. The model defines also characteristics of entities called attributes. The most important of these are attributes defining relationships between entities.

Here one of the important architectural principles is distinction between definitions of objects, object properties and types of activities. Such separation meets practical requirements, as in fact object properties may have multiple versions of description. Thus each property can be uniquely identified by with its own description history.

Each entity, represented in the model, is determined by such parameters as a set of attributes, local rules and chains of supertypes. An attribute is a list of characteristics describing the given entity. In its own turn an attribute may have several parameters, such as an attribute name, a list of options (key, required, external, etc.), and relationships types. Entities are also associated with entity rules, describing extended data integrity restrictions defining the domain of possible values of attributes and relationships.

In the Epicentre model, specification of entities has been expanded to include so called reference entities. They differ from other entities because they have some standard set of values predefined by POSC. Their presence of such entities is required for compatibility with previous POSC specifications. There are three types of reference entities in the model:

- ***POSC Fixed*** entity has a fixed number of instances predefined by POSC;
- ***POSC Open*** entity also has fixed instances, but it's also possible to create additional instances not predefined by POSC;
- ***Local*** entity has no fixed predefined instances, but it's possible to introduce user defined instances.

All reference entities may have additional characteristics that allow to specify the source and bibliography, that describe origin of information, contained in the in-

stance. To denote the reference entities and their types, the distinctive prefix *Ref_* to the entity names is used.

The object-oriented concept of class inheritance is also an important part of the Epicentre model architecture. Since the data model contains really large volume of information, this concept provides an efficient way to organize all entities definitions into a logically related structure.

Another fundamental part of Epicentre architecture comes from understanding that many entities can be characterized by their spatial representation (such as geographical coordinates of oil well). Each of these geometric objects of activity can be connected through some relations with one or more similar objects (such as the “to be located on the territory” relationship).

The Epicentre data model is defined in terms of the EXPRESS language and uses its basic concepts such as:

- *Entity;*
- *Supertype and subtype;*
- *Attribute, explicit and inverse;*
- *User defined data type;*
- *Simple type;*
- *Aggregate type;*
- *Consistency constraint, or "where" rule;*
- *Uniqueness rule;*
- *Schema.*

Thus, the Epicentre model entity description actually is a definition of a class, on the basis of which class instances or objects can be created.

An entity may be a subtype of some supertype entity, inheriting its attributes, rules and uniqueness constraints. In other words specification of supertype opens the way to define type properties, common to all its subtypes.

A supertype may be abstract, which means that all instances must be specified. Otherwise, its instances may be either specialized or not.

An attribute is a specific characteristic of an entity. To each attribute a name and a representation type are assigned. It can be explicit, inverse, or derived from other attributes.

Explicit attribute is an attribute that is not derived from any other attribute in the model. Inverse attribute is used to express the opposite direction of a relationship appeared in some explicit attribute specification.

A schema definition is a container that includes definitions of all entities, types, and constraints visible in some particular EXPRESS language schema (Fig. 1).

Here attribute definitions are as follows:

- ***name*** stands for the schema identifier;
- ***types*** denotes a set of entities and specific types, declared in the schema;
- ***global_rules*** denotes a set of global constraints declared in the schema.

```
ENTITY schema_definition;
        name : STRING;
        types : SET OF named_type;
        global_rules : SET OF global_rule;
        UNIQUE
        Url : name;
END_ENTITY;
```

Fig. 1. EXPRESS language schema definition.

UNIQUE Url is a formal statement, expressing the "schema name must be unique" constraint.

The uniqueness constraint points here to combination of key attributes whose values in the aggregate must uniquely identify a specific instance of the entity. Consistency constraint in general defines condition imposed on attributes of all instances of the entity.

3 OWL Structure

Web Ontology Language supports:

- formal definition of classes and their properties;
- definitions of individuals (class instances) and their properties;
- refining the definitions of classes and objects in logical terms.

OWL is largely compatible with the RDF [7] and RDF Schema [8] languages. The XML [9] and RDF formats actually are part of the OWL standard.

The main structural units of an OWL ontology are classes, properties, objects (i.e. class instances or individuals) and relationships between them.

The most fundamental concepts of the subject domain should correspond to the classes, located at the root of various taxonomic trees. Each individual here is an instance of the *owl:Thing* class. Thus, each user-defined class is automatically a subclass of the *owl:Thing* class. The root classes, specific to the given subject domain, are defined simply by declaration of a named class. The fundamental taxonomic constructor is represented by the *rdfs:subClassOf* class; it defines the “to be a subclass” class relation.

OWL properties allow to state some general facts about class members and specific facts about individuals. In fact property here is a binary relation. There are two types of properties:

- *value-property* is a relation between class instances and RDF literals or data types, defined by XML Schema;
- *object-property* is a relation between instances of two classes.

There are many ways to specify such relationship. One can define the domain and range. A property can be defined as the specialization (*sub-property*) of some already existing property. Like classes, properties can be organized into a hierarchy.

More complex restrictions are also possible. OWL provides powerful mechanism to express various characteristics of properties. Let's mention some traditional characteristics of binary relations important to explain methodology of converting the Epicentre logical model into the OWL ontology description language.

Transitive property. Property P can be marked as transitive, if for any x, y and z P (x, y) and P (y, z) implies P (x, z).

Symmetric property. Property P is marked as symmetric, if for any x and y P (x, y) implies P (y, x).

Functional property. Property P is marked as functional, if for any x, y and z P (x, y) and P (x, z) implies y = z.

Inverse property. If property P1 is marked as *owl:inverseOf* P2, then for all x and y P1 (x, y) implies P2 (y, x).

Inverse functional property. Property P is marked as inverse functional, if for all x, y and z: P (y, x) and P (z, x) implies y = z.

In addition to designating property characteristics, it is also possible in specific contexts to limit explicitly the range of a property. This can be done by using the following property restrictions local to the class containing them.

allValuesFrom. This restriction requires, for each instance of a class with this property, all property values to be instances of the class, specified in the *owl:allValuesFrom* clause.

someValuesFrom. Similarly, this constraint requires, for each instance of a class with this property, at least one property value to be a representative of the class specified in the *owl:someValuesFrom* clause.

Cardinality. The *owl:cardinality* parameter allows to specify the number of elements in the relationship. In OWL DL, in particular, *owl:maxCardinality* can be used to set an upper limit, and *owl:minCardinality* can be used to set a lower limit. In combination, they can be used to limit the cardinality of a property within some numerical range. Fig. 2 shows an example of defining a single-valued relationship for the class called *Name_entity*.

```
<owl:Class rdf:ID=" Name_entity">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#UNIQUE"/>
                        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
                        </owl:cardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Fig. 2. Example of OWL cardinality definition.

4 Epicentre to OWL DL Conversion.

To construct the OWL ontology from the Epicentre data model the following main approaches were used:

- each Epicentre model entity corresponds to a primary class of the OWL ontology; all these classes are located at the root of the taxonomy tree; specific name prefixes, that allow to identify entity-properties and reference entities, are incorporated to the class names;
- entities relation types (such as *one-to-one*, *one-to-many* and *many-to-many*) are expressed by OWL definition of simple attribute properties, if the related entity is not a data type, or by definition of value properties otherwise; indicating the type of relation between classes is implemented using the OWL concept of cardinality.

OWL contains no structural elements to define explicitly the notion of uniqueness of the Epicentre model. Therefore, a new predefined property, containing the list of all the unique key attributes, has been added to the definition of each OWL class. Similarly, the problem of expressing the Epicentre constraint conditions was solved by constructing specific OWL classes for each Epicentre data category.

The formal LR (1) grammar [10] of the Epicentre model was created to serve the base of automatic semantically equivalent transformation of the Epicentre model into the OWL DL dialect syntax. The descriptive logic dialect was chosen to allow in perspective to complement the ontology with inference system engine. Complete Russification of descriptions of the Epicentre model entities and attributes, as well as all corresponding OWL classes and properties was fulfilled.

The software implementation of the Epicentre to OWL DL converter was performed in the Java programming language, using the flex lexical analyzer's generator [11] and the CUP parser generator [12].

To verify semantic correctness by the subject domain professionals, the OakOwlProject 1.0 program visual interface, providing navigation and manipulation with the OWL ontologies has also been developed.

The well-known open-source Java project Protégé [13] was chosen as a prototype of the OakOwlProject environment implementation. In the process of development, its functionality was essentially expanded. The final appearance of the OakOwlProject interface, reflecting the capabilities of the system for working with ontologies is shown in Fig. 3.

Syntactically, the Epicentre model is a sequence of entity descriptions in EXPRESS language. To convert the model, it was necessary to translate the description of each entity from EXPRESS into OWL language. The syntactic structure of the Epicentre model is shown in Fig. 4.

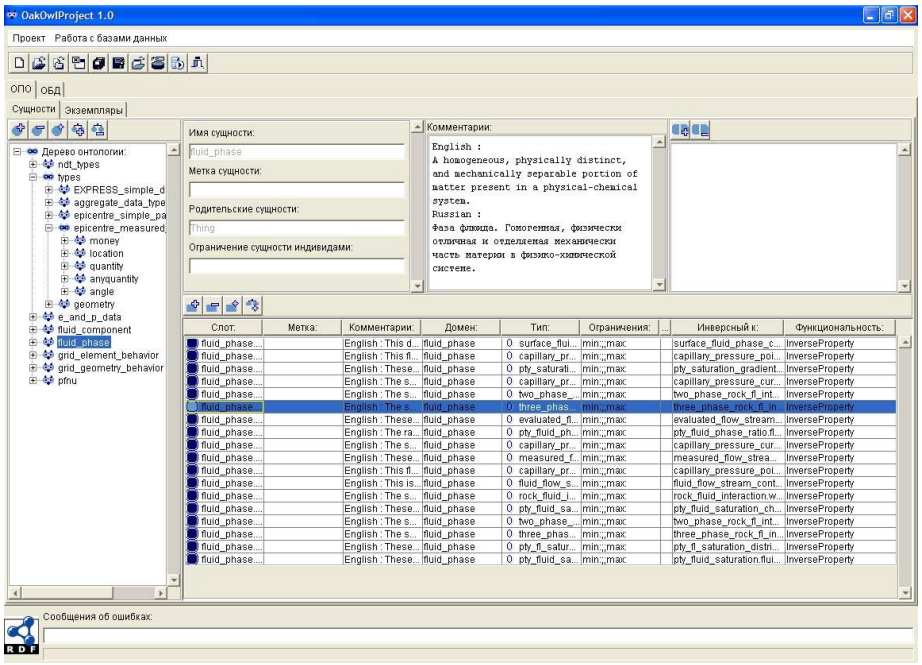

Fig. 3. General view of the OakOwlProject system user interface.

```
ENTITY name_entity
        ABSTRACT SUPERTYPE OF (
                ONEOF(name_entity1, ..., name_entityN)
        )
        SUBTYPE OF (sub_name1, ..., sub_nameN);
        name_attribute: OPTIONAL SET[0 :?] OF type_name
        ..................
        INVERSE
                invers_attribute_name: SET[0 :?] OF invers_name_entity
                                                FOR entity_invers_type
        ..................
        UNIQUE
        si: name_attribute1, ..., name_attributeN
        WHERE условие
        ..................
END_ENTITY;
```

Fig. 4. EXPRESS entity descriptions.

The following notation is used here:

name_entity denotes the name of the entity; it may have the *Pty_* or *Ref_* prefix, indicating respectively an property entity or reference entity;

name_entity1, ..., name_entityN is the list of the named entities;

name_attribute denotes the explicit attribute; it may also have the *Ref_* prefix;

sub_name is the name of the parent entity, which can a regular entity (no prefix) or a property entity (*Pty_* prefix);

sub_name1, ..., sub_nameN denotes a list of named parent entities;

type_name is the name of the direct attribute type, it can be a reference entity (identified by *Ref_* prefix), a regular entity (which name with no prefix) or a named data type (identified by *Ndt_* prefix);

inverse_attribute_name is obviously the name of an inverse attribute; it may denote a regular entity (no prefix) or a property entity (there is a *Pty_* prefix);

inverse_name_entity is the name of an entity, inversely related to this entity; it may be a reference entity (marked by *Ref_* prefix*)*, a regular entity (no prefix) or a property entity (marked by *Pty_* prefix);

entity_inverse_type is the name of an entity, which is the type of direct attribute, corresponding to the given inverse attribute; it may be a regular or reference entity;

name_attribute1, ..., name_attributeN denotes a list of attributes.

The meaning of capitalized keywords is described below.

Fig. 5 shows the way to convert Epicentre entities into OWL classes.

```
ENTITY name_entity          <owl:Class rdf:ID=" name_entity ">
...                   =>    ...
END_ENTITY;                 </owl:Class
```

Fig. 5. Epicentre entities to OWL Classes conversion.

The base classes of the Epicentre model, such as E_AND_P_DATA, FLUID_COMPONENT, FLUID_PHASE, GRID_ELEMENT_BEHAVIOR, GRID_GEOMETRY_BEHAVIOR, PFNU, are converted to the subclasses of the *owl:Thing* class, which are located in the root of the taxonomic tree. In the OakOwl-Project_1.0 shell, the base classes are also located at the root of the hierarchy shown in Fig. 6.

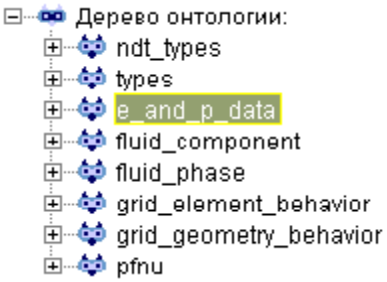

Fig. 6. Base classes of the data model.

Conversion of the inheritance relationship is shown in Fig. 7. Here, for the given entity the parent entities are listed in the **SUBTYPE OF** construct. Such relation uniquely corresponds to the OWL **subclass** construct.

```
SUBTYPE OF (                          <owl:Class rdf:ID="name_entity">
    sub_name1, ..., sub_nameN =>      <rdfs:subClassOf rdf:resource="# sub_name1" />
);                                            ...
                                              <rdfs:subClassOf rdf:resource="# sub_nameN" />
                                              ...
                                      </owl:Class>
```

Fig. 7. Inheritance relationship conversion.

Here definition of the entity name and its place in the Epicentre model hierarchy is followed by a list of the entities attributes, both direct and inverse. The definition of a direct attribute is shown in Fig. 8.

```
name_attribute: OPTIONAL SET[0 :?] OF type_name
```

Fig. 8. Direct attribute definition.

Here, the **OPTIONAL** and **SET [0:?] OF** syntax constructs may be absent. If present, the **OPTIONAL** keyword denotes the optionality of the attribute value and the **SET [0:?] OF** construct determines the degree of connectivity (i.e. *one-to-one*, *one-to-many* and *many-to-many* options) for the aggregate attribute. Such syntactic unit corresponds to the OWL definition of simple attribute properties, if the related entity is not a data type, and value properties otherwise. Hence, it can be represented in OWL in the way shown in Fig.9.

```
<owl:ObjectProperty rdf:ID="name_attribute">
        <rdfs:domain rdf:resource="#name_entity"/>
        <rdfs:range rdf:resource="#type_name"/>
</owl:ObjectProperty>
```

Fig. 9. OWL definitions of attribute-properties and value-properties.

Here, the property name corresponds to the attribute name and the relation binds the original entity and some class. Potential name collision is solved in traditional way by introducing dot notation. For example, in the case of match of the property name and the class name, the property name can be specified as *ontology_entity_name.attribute_name*.

If an attribute has values of some specific data type, then, by sense, there should be represented by a value property. But since the Epicentre data types have more complex structure, we actually get the same object properties.

An indication of the degree of class relations, as well as **OPTIONAL** construct can be implemented using the OWL concept of cardinality.

Here, **OPTIONAL** construct means that the degree of relationship may be equal to 0, so the corresponding property constraint can be added to the class description. For example, the description of the *well_test_open_period_recovery* entity (see Fig. 10) will be expressed as shown in Fig. 11. The OWL definition of the property itself is shown in Fig. 12.

```
ENTITY well_test_open_period_recovery
        SUBTYPE OF ( well_test_recovery );
        time_to_surface: OPTIONAL ndt_time;
END_ENTITY;
```

Fig. 10. Implementation of EXPRESS property constraint.

```
<owl:Class rdf:ID="well_test_open_period_recovery">
        <rdfs:subClassOf rdf:resource="#well_test_recovery"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:resource=
                                "#well_test_open_period_recovery.time_to_surface"/>
                        </owl:onProperty>
                        <owl:minCardinality>0</owl:minCardinality>
                        <owl:maxCardinality>1</owl:maxCardinality>
                        <owl:cardinality>0</owl:cardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Fig. 11. Cardinality concept implementation.

```
<owl:ObjectProperty>
        <rdf:about>#well_test_open_period_recovery.time_to_surface
        </rdf:about>
        <rdfs:domain>
                <owl:Class>
                        <rdf:about>#well_test_open_period_recovery</rdf:about>
                </owl:Class>
        </rdfs:domain>
        <rdfs:range>
                <owl:Class>
                        <rdf:about>#ndt_time</rdf:about>
                </owl:Class>
        </rdfs:range>
</owl:ObjectProperty>
```

Fig. 12. OWL property definition.

The following syntax construction corresponds to the inverse attribute definition (Fig. 13):

```
INVERSE
invers_attribute_name: SET[0 :?] OF invers_name_entity
        FOR entity_invers_type;
```

Fig. 13. Inverse attribute definition.

There is no concept of an inverse attribute in the OWL language. This problem is solved by adding the *Type:InverseProperty* property, indicating to the entity to which

the given attribute is inverse. For example, let an *activity* entity has an *activity_alias* attribute, which is inverse to an *aliased_object* entity. A description of this attribute is provided in Fig. 14

```
<owl:ObjectProperty>
        <rdf:about>#activity.activity_alias</rdf:about>
        <owl:type>
                <rdf:about>#InverseProperty</rdf:about>
        </owl:type>
        <owl:inverseOf>
                <owl:ObjectProperty>
                        <rdf:about>#activity_alias.aliased_object</rdf:about>
                </owl:ObjectProperty>
        </owl:inverseOf>
</owl:ObjectProperty>
```

Fig. 14. OWL definition of inverse attribute.

The following syntactic construction corresponds to the definition of entity attributes uniqueness and **WHERE**-constraints (Fig. 15).

```
UNIQUE
si: name_attribute1, ..., name_attributeN;
WHERE условие
```

Fig. 15. OWL definition of entity attributes uniqueness and WHERE-constraints.

Here, **UNIQUE** keyword is followed by the list of unique attributes forming the key of this entity.

The OWL language lacks structural elements to express Epicentre's notion of uniqueness. Therefore, a new property called *name_entity.UNIQUE* has been added to the definition of each OWL class to contain a list of all key attributes. Similarly, a new property called *name_entity.WHERE* has been added to the definition of each OWL class to contain constraint conditions.

All data types in the Epicentre model are classified into the following categories:

- Simple Data Types;
- Simple Pattern Types;
- Measured Quantity Types;
- Geometry Types.

To define these categories in terms of the OWL predefined data types, separate classes have been added to the OWL ontology.

5 Conclusion

To define POSC's Epicentre 3.0 model syntax, a formal LR (1) grammar was constructed; it was used as the input file for the Java Cup parser generator. Based on this grammar and the Epicentre model conversion scheme described above, an OWL ontology of the subject domain of natural-technical objects was constructed.

Not all Epicentre model constructs can be explicitly expressed in the OWL language. To convert information adequately from the Epicentre model, special OWL classes and reserved properties were defined.

The total volume of LR(1) grammar, together with the built-in implementation of the conversion semantics, contains about 30 pages. The EXPRESS Epicentre model file contains about 500 pages. The resulting OWL ontology definition has a volume of about 3,500 pages. Thus the Epicentre model has been completely translated into OWL ontology with no loss of information. Visualization of the constructed ontology allows the subject domain experts to validate its correctness.

Semantic search web application for TATNEFT oil corporation based on the described subject domain ontology, unified data ontology integrating various corporation heterogeneous relational databases and linguistic thesaurus of professional terminology is described in [14–18].

The way of the corporation data integration essentially uses the methods of computer linguistics to extract information from natural language texts contained in databases comments. The corresponding knowledge extraction algorithm is described in more detail in the above-mentioned works.

The core algorithm of the intellectual search system is the algorithm for constructing SQL queries from the end user queries, written in professional dialect. Optimization methods for implementing the most resource-intensive part of this algorithm, associated with the need to enumerate table joins, are considered in [19].

Acknowledgments. This work was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, grant agreement 1.2368.2019 and subsidy of the Russian Fund of Fundamental Research, grant agreement 18-07-00964.

References

1. Gavrilova, T.A., Khoroshevskii, T.A.: Bazy znanii intellektualnykh system. SPb., Piter (2001).
2. Date, C.J.: Vvedenie v sistemy baz dannykh. M., Izd. Dom Viliams (2001).
3. Epicentre v3.0, http://www.energistics.org/energistics-standards-directory/epicentre-archive, last accessed 2019/12/09.
4. OWL Web Ontology Language, https://www.w3.org/TR/2004/REC-owl-features-20040210, last accessed 2019/12/09.
5. Towards the Semantic Web: Ontology-Driven Knowledge Management. Chicester, UK, John Wiley & Sons (2003).
6. The World Wide Web Consortium (W3C), http://www.w3c.org, last accessed 2019/12/09.

7. RDF 1.1 Concepts and Abstract Syntax, https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225, last accessed 2019/12/09.
8. RDF Schema 1.1, https://www.w3.org/TR/rdf-schema, last accessed 2019/12/09.
9. Extensible Markup Language (XML), https://www.w3.org/XML, last accessed 2019/12/09.
10. Lewis, P., Rosenkrantz, D., Stearns, R.: Teoreticheskie osnovy proektirovaniia kompiliatorov. M., Mir (1979).
11. Allmon, B.J., Anderson, J.: Flex on Java. Manning Publications Co. Greenwich, CT, USA, ISBN: 1933988797 (2010).
12. CUP Parser Generator for Java, https://www.cs.princeton.edu/~appel/modern/java/CUP, last accessed 2019/12/09.
13. Protégé, http://protege.stanford.edu, last accessed 2019/12/09.
14. Birialtsev, E., Bukharaev, N., Gusenkov A.: Intelligent search in Big Data. Journal of Physics: Conference Series, vol. 913, conf. 1. Published online: 25 October 2017 (2017).
15. Gusenkov, A.M.: Intellektualnyi poisk slozhnykh obieektov v massivakh bolshikh dannykh. Elektronnye biblioteki, vol. 19, book 1, pp. 3–39 (2016).
16. Gusenkov, A., Birialtsev, E., Zhibrik, O. Intellektualnyi poisk v strukturirovannykh massivakh informatsii. LAP LAMBERT Academic Publishing, Deutschland: OmniScriptum Marceting DEU GmbH, ISBN 978-3-659-76919-1 (2015).
17. Gusenkov, A.M., Birialtsev, E.V.: Integratsiia reliatsionnykh baz dannykh na osnove ontologii. Uchenye zapiski Kazanskogo gosudarstvennogo universiteta. Seriia Fiziko-matematicheskie nauki, vol. 149, book. 2, pp. 13–34 (2007).
18. Gusenkov, A., Bukharaev, N., Birialtsev, E.: On ontology based data integration: problems and solutions. Journal of Physics: Conference Series, vol. 1203, conf. 1, 012059 (2019), https://iopscience.iop.org/article/10.1088/1742-6596/1203/1/012059/meta, last accessed 2019/12/09.
19. Gusenkov, A., Bukharaev, N.: On Semantic Search Algorithm Optimization. New Knowledge in Information Systems and Technologies. WorldCIST'19. Advances in Intelligent Systems and Computing, vol. 930. Springer, Cham (2019), https://link.springer.com/chapter/10.1007/978-3-030-16181-1_45, last accessed 2019/12/09.