Bringing the Subject Domain Ontology to Optimal Canonical Form

Anatoli Karpuk
Belarusian State Academy of Communications
Minsk, Belarus
Email: a_ karpuk@mail.ru

Abstract—A formal definition of the subject domain ontology is given. The concept of the canonical form of a subject domain ontology is considered, which is built on the basis of an analysis of functional dependencies between concepts and properties of the subject domain ontology concepts. An algorithm for bringing the subject domain ontology to a canonical form is described. The concept of the optimal canonical form of a subject domain ontology is introduced, containing the minimum number of classes and the minimum number of attributes in the classes. A method for bringing the subject domain ontology to the optimal canonical form is roposed.

Keywords—ontology, domain, canonical form, functional dependence, optimal canonical form

I. Introduction

In computer science, ontology is a comprehensive and detailed formalization of a certain area of knowledge in the form of a conceptual diagram. A conceptual schema is a set of concepts and information about concepts, which includes properties, relationships, restrictions, axioms and statements about concepts necessary to describe the processes of solving problems in a selected subject domain. For each knowledge area, an applied ontology is built, which consists of a top-level ontology, a subject domain ontology, and a task ontology. Subject domain ontology are simultaneously developed and used by many users. For this reason, in subject domain ontology, the same property of a concept can be represented in different ways. Such ambiguity can lead to difficulties when solving problems using subject domain ontology. To eliminate this drawback, works [1], [2] propose to bring the subject domain ontology to the so-called canonical form, which is based on the analysis of functional dependencies between the concepts and properties of the ontology. However, the developed algorithm for bringing the ontology to a canonical form also does not provide a unique solution. Depending on the order in which the functional dependencies between attributes are analyzed, it is possible to obtain a different number of classes with different numbers of attributes in them. This article introduces the concept of an optimal canonical form of a subject domain ontology, containing a minimum number of classes and a minimum number of attributes in classes, and proposes a method for bringing a subject domain ontology to an optimal canonical form.

II. Formal Definition the Subject Domain Ontology

Let us define the subject domain ontology in the form of a quadruple $O = <K, R, F, I>$ [3], [4] where K is a finite set of concepts of the subject domain ontology; R – a finite set of relations between concepts; F – a finite set of interpretation functions defined on concepts and relationships; I – a finite set of axioms, each of which is always a true statement on concepts and relations. The set of concepts has the form $K = <D, A, Q>$, where D is a finite set of domains; A — a finite set of attributes; Q — a finite set of classes in subject domain ontology. Domains are used as sets of possible attribute values. Each domain's data has one of the data types allowed in the XML language [5]. Based on the number of data elements in the value, domains are divided into atomic, union, and list. An atomic domain consists of indivisible data elements of a specific type and format. The value of a federated domain is a data aggregate (structure) consisting of other aggregates and data elements. The value of any merged domain can be represented as a union of the values of its constituent atomic domains. A list domain value is a list (repeating group) of atomic or concatenated domain values. The number of list elements can be any. The value of any list domain can be represented as a repeating group of values from one or more atomic domains. Based on value restrictions, atomic domains are divided into primitive, built-in, and constructed. Primitive data types of the XML language are used as primitive atomic domains. Built-in domains are derived from primitive domains by applying fixed constraints to them. For example, the primitive domain decimal produces the built-in domains integer, long, int, short, byte, nonNegativeInteger, positiveInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, nonPositiveInteger, negativeInteger. Derived domains are derived from primitive and built-in domains by applying various facets to them. For example, the constraints length, minLength, maxLength, pattern, enumeration, whiteSpace, assertions can be applied to the primitive atomic domain string. The constraints totalDigits, fractionDigits, pattern, whiteSpace, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive, assertions can be applied to the primitive atomic domain decimal. Depending on the identification method, domains can be unnamed or named. Unnamed domains do not carry semantic load and are used only as data types. Unnamed domains cannot be merged or list domains. Named domains are distinguished from unnamed domains by having a user-defined domain name and can be atomic primitives, inline and derived domains, as well as federated and list domains.

Concepts from set A represent properties (attributes) of subject domain classes. Each attribute is specified by its unique name and the domain to which the attribute values belong. Depending on what domain the attribute is defined on, it can be atomic, aggregated, or list.

Concepts from set Q represent classes (objects, entities) of the subject area. One class includes real or abstract people, objects, phenomena, events, processes that have the same or similar set of properties (attributes), knowledge about which is stored in the ontology and used when solving problems from a given subject domain. When constructing a subject domain ontology, classes are first described, and then knowledge about the individuals of each class is recorded in the ontology. In Russianlanguage literature, individuals of classes are often called instances of classes or objects. Each class is given its own unique name and its own set of attributes.

Each attribute within a class can have cardinality and functional properties. Cardinality values indicate the minimum and maximum number of attribute values that one individual of a class can have. By default, a class individual can have any number of attribute values. If an attribute value may not be present in an individual of a class, then the ontology must indicate a minimum cardinality equal to 0. The functionality sign shows that any individual of a class can have no more than one attribute value. If the functionality attribute is set, then the maximum cardinality of this attribute should not be specified, or should be equal to 1. The set of class attributes, the values of which uniquely determine an individual of the class, is declared as the key of the class. A class can have more than one key.

The set of relations between concepts R includes relations between domains for constructing derived domains, relations between attributes and domains for determining the scope of attributes, relations between classes and attributes for determining the composition of the attributes of each class, and relations between classes. Relationships between classes reflect “wholepart”, “genus-type” connections, as well as hierarchical and other connections between classes that exist in the subject domain. Each relationship between classes can also have cardinality and functionality properties. In addition, relationships between classes can be inverse, inverse functional, transitive, symmetric, asymmetric, reflexive, and irreflexive. The set of functions F consists of $n - ary$ relations between classes or attributes in which the value of an element with a number n is uniquely determined by the values of previous $(n - 1)$ elements. Using functions, you can describe class keys, hierarchical relationships between classes and attributes, and any other functional dependencies between classes and attributes that exist in the subject domain. The set of axioms I serves to represent in the ontology statements about classes, attributes, domains and relations that are always true. Each axiom is formulated in the form “if <condition on the values of domains for given attributes of given classes or relations> then <statement about the values of domains for given attributes of given classes or relations>”. Axioms are included in the ontology to check restrictions on the values of attributes, to check the correctness of the description of the ontology, to derive new true statements about classes, attributes, domains and relationships.

III. Functional Dependencies between Attributes of the

Subject Domain Ontology Let $X \subset A$ be a subset of attributes of the subject domain ontology, $Z \in A$ — some attribute. We will say that in the subject domain ontology there is a functional dependence (FD) $X \to Z$, if any combination of attribute values from X always corresponds to a single value of the attribute Z.

The FD structure on a set of attributes A satisfies Armstrong's axioms [6]:

if $X \subseteq A$ then $A \to X$ (reflexivity axiom); if $X \to Y$ and $YC \to D$, then $XC \to D$ (axiom of pseudotransitivity).

If there is a FD $X \to Y$, then they say that X functionally determines Y or Y functionally depends on X. From the given axioms, one can derive a number of properties of the FD structure, which in the literature (for example, [7]) are often also called axioms, although it is more accurate to call them rules of inference. The most important are the following inference rules: if $X \to YC$, then $X \to Y$ and $X \to C$ (decomposition rule), indeed, by the axiom of reflexivity we have $XYC \to Y$ and $XYC \to C$, then by the axiom of pseudotransitivity we obtain $X \to) Y$ and $X \to C$; if $X \to Y$, then $XC \to YC$ (replenishment rule), indeed, by the axiom of reflexivity we have $XYC \to YC$, then by the axiom of pseudotransitivity we obtain $XC \to YC$; if $X \to Y$ and $X \to C$, then $X \to YC$ (the union rule), indeed, by the completion rule we have $X \to XY$ and $XY \to YC$, then by the axiom of pseudotransitivity we obtain $X \to YC$.

Obviously, the inclusion relation $\subseteq$ determines the FD structure on the set A, which is called the trivial structure of the FD, and the FDs included in it are called trivial FDs. To specify a FD structure that differs from the trivial one, it is necessary to postulate a finite set of FD $F = Fj = Xj \rightarrow Yj | Xj \subset A, Yj \subseteq A, j = \overline{1, \mathrm{m}}$, which in the article [8] was called a system of generators of the FD structure on the set of attributes A. The FD structure, specified by the system of generators F, will be denoted by S(F). It is obvious that from $Z \in X$ it follows that $X \rightarrow Z$. Such an FD, in which the dependent attribute is part of the left side of the FD, is called trivial. In what follows, we will consider only non-trivial FDs between attributes. In the subject domain ontology, the following non-trivial FDs between attributes can be distinguished:

- each functional attribute of a class that is not a subclass of another class, that is not a subordinate attribute of another class attribute, functionally depends on each class key;
- each functional attribute of a subclass that is not a subordinate attribute of another attribute of a subclass is functionally dependent on each subset of attributes obtained by combining each key of the parent class with each key of the subclass;
- each functional attribute of a class that is not a subclass of another class that is a subordinate attribute of another class attribute functionally depends on each subset of attributes obtained by combining each class key with a parent attribute;
- each functional attribute of a subclass, which is a sub-attribute of another attribute of a subclass, functionally depends on each subset of attributes obtained by combining each key of the parent class with each key of the subclass and the parent attribute;
- each functional relationship between classes from the set R specifies the FD of attributes of each key of the parent class from each key of the subordinate class;
- each function from the set F, defined on the attributes of the subject domain ontology, sets the FD of the last attribute of the relation from the previous attributes of this relation;
- each function from the set F, defined on the classes of the subject domain ontology, specifies the FD of the attributes of each key of the last class of the relation from each subset of attributes containing any one key of the previous classes of this relation.

When defining the FD between the attributes of the subject domain ontology, it is possible to write more than one attribute on the right side of the FD, since for the FD between attributes the property of the cluster decomposition of the right side is valid, namely, if Y1 $\in$ A, Y2 $\in$ A and Y = Y1 $\cup$ Y2, then the record X $\rightarrow$ Y corresponds to the simultaneous presence of FD X $\rightarrow$ Y1 and X $\rightarrow$ Y2. Moreover, instead of the last two FD, you can write X $\rightarrow$ Y1Y2. Each FD between attributes in the subject domain ontology can be considered as the simplest rule for deriving new knowledge from the knowledge available in the ontology. Indeed, if the values of the attributes of the left side of the FD are known, then either the ontology already contains uniquely corresponding values of the attributes of the right side of the FD, or the values of the attributes of the right side of the FD can be obtained by solving the problem from the ontology of tasks. The input data of this problem are the values of the attributes of the left side of the FD, and the output data are the values of the attributes of the right side of the FD.

Let us single out in the subject domain ontology all FDs between attributes and represent them as a set of FDs P = Pj = Xj $\rightarrow$ Yj — Xj $\subset$ A, Yj $\subseteq$ A, j = $\overline{1, \mathrm{m}}$, which is called the system of forming FD structures on the set of attributes of the ontology. The structure of the FD, given by the system of generators P, will be denoted S(P).

The closure of the set of attributes X $\subset$ A concerning to the structure of FD S(P) is a set X+(P) $\subseteq$ A such that for any Y $\subseteq$ A from X $\rightarrow$ Y follows Y $\subseteq$ X+(P). In other words, the closure of the set of attributes X includes all the attributes, the values of which can be obtained from the known values of the attributes of set X, using the FD derivation from the set P. The algorithm for constructing the closure X+(P) consists of the following steps [8].

1. Put X+(P) = X and $Pj = 0$, $j = \overline{1, \mathrm{m}}$
2. Put $q = 0$ and for each $j = 1$, m perform step 3.
3. If $Pj = 0$ and Xj $\subseteq$ X+(P) then put X+(P) = X+(P) $\cup$ Yj , $q = 1$ and $Pj = 1$.
4. If $q = 1$, then go to step 2, otherwise finish the job.

The structures of FD $S(P^1)$ and $S(P^2)$ on the set of attributes A with systems of generators $P^1 = X_i^1 \rightarrow Y_i^1 | X_i^1 \subset A, Y_i^1 \subseteq A, i = 1, \overline{1, \mathrm{m}}$ and $P^2 = X_j^2 \rightarrow Y_j^2$ — $X_j^2 \subset A, Y_j^2 \subseteq A, j = \overline{1, \mathrm{m}_2}$, respectively, are called equivalent if for any $X \subset A$ the equality $X + (P^1) = X + (P^2)$. In article [9] it is proven that necessary and sufficient conditions for the equivalence of structures FD $S(P^1)$ and $S(P^2)$ on the set of attributes A are the fulfillment of equalities $X_i^1 + (P^1) = X_i^1 + (P^2)$ and $X_j^2 + (P^1) = X_j^2 + (P^2)$ for all $i = \overline{\pm 1, m_1}, j = \overline{\pm 1, m2}$. The system of generators $E = H_j \rightarrow T_j | H_j \subset A, T_j \subseteq A, j = \overline{\pm 1, m}$ is called an elementary basis of the structure of FD S(E) if the removal of any attribute from the left or right side of any FD from E leads to the structure of FD that is not equivalent to $S(E)$