ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2012, Vol.41, No.2

Preserving Semantics of Owl 2 Ontologies in Relational Databases Using Hybrid Approach

Ernestas Vyšniauskas, Lina Nemuraitė, Bronius Paradauskas

Kaunas University of Technology, Department of Information Systems
Studentų st. 50-308, LT-51368 Kaunas, Lithuania
e-mail: vernest@email.lt, lina.nemuraite@ktu.lt, bronius.paradauskas@ktu.lt

crossref http://dx.doi.org/10.5755/j01.itc.41.2.833

Abstract. The goal of the paper is to define requirements to OWL 2 ontologies, under which their semantics may be preserved in a relational database, and to demonstrate that the hybrid approach for transforming OWL 2 ontologies into relational databases possesses such capability. The hybrid approach maps part of ontology concepts to relational database concepts on the base of their common semantics; ontology constructs having no direct equivalents in databases are stored in metatables. The paper defines requirements for ontologies under transformation as ontology normalization and integrity rules, and presents a set of SQL queries for extracting rich data, covering semantics of source ontology, from the resulting database. The capability of the hybrid approach to preserve semantics of OWL 2 ontologies in relational databases is demonstrated with a representative example of a Vehicle ontology.

Keywords: Ontology; relational database; OWL 2; RDB; semantics preserving transformation; SPARQL; SQL.

1. Introduction

Storing ontologies in relational databases becomes one of ordinary needs of Semantic Web and networked enterprises where knowledge models are emerging in various new fields e.g. [1, 2]. While a lot of methods and approaches for transforming ontologies into relational databases (and vice versa) are proposed, the shared comprehension about suitable ways for linking the two technologies still does not exist. One of painful aspects of such transformations is preserving semantics of ontology when it is stored in relational database (RDB). Though storing Web Ontology Language OWL 2 ontology [3, 4] in one vertical table [5] or ontology metamodel based schema [6, 7] guarantee preservation of ontological semantics, obtained relational schemas are suitable for processing ontologies only as they are meaningless from the conceptual point of view; they do not avail advantages of relational databases, and are not applicable for applications of information systems. There are arguments for having different storing models [8, 9] reflecting common semantics of ontologies and conceptual models familiar for domain experts and application developers. We have proposed a reversible OWL2ToRDB transformation pursuing a hybrid approach according which part of ontology concepts is directly mapped to relational schema on the base of their common semantics; ontology constructs having no direct equivalents in database schema are stored in metadata tables [1].

Transformation is semantics preserving if the meaning of the two models is the same, even though it is represented in a different technical space or using a different abstract syntax [11]. A reversible transformation is semantics preserving transformation [12] that does not lose semantics when it executes from a source model to a target model and backwards – from the obtained target model to the source model; but the reverse may not be true if that transformation executes from whatever model of the target type to a model of the source type and backwards. It means that the OWL2ToRDB transformation could be applied starting from ontology and converting it into a database. A converted ontology could be fully restored from the database; however, it does not mean that the reverse OWL2ToRDB transformation could be applied for any database that was not created by a direct one.

Transformation starting from any relational database into ontology may be considered as a reverse engineering problem [13]. Though such a transformation may be reversible, semantics preserving, and conforming to the semantics common for ontologies and information systems (as described in Section 3), it would start from considering different constructs – tables, constraints and dependencies, implemented via functionality of relational database systems. Therefore, it would be a quite different task relevant to exposing contents of existing databases on the Semantic Web (e.g. [14, 15]). For developing new databases, it is desirable to pre-establish their

connection to ontologies by creating the required ontology and transforming it into relational database via the proposed OWL2ToRDB approach.

We apply the following quality criteria to OWL2ToRDB transformation: transformation completeness – coverage of ontology constructs; transformation reversibility – preservation of semantic information; semantic suitability of database schema – relevance of its conceptual data model to semantics of problem domain; schema explicitness – declarative versus procedural representation of ontology constraints in a database; schema stability – schema is stable if it does not change when ontology changes; automation – whether transformation is fully automatic or not.

In the current paper, we focus on requirements that ontology under transformation should fulfil. For being stored in a relational database without loss of information, ontology should conform to rules of ontology design. Also, this ontology should fulfil quality criteria of conceptual models – it should be syntactically correct, precise, and semantically suitable description of a problem domain. Conceptual model is precise if it unambiguously describes the domain. Conceptual model is suitable if it 1) conforms to sets of objects (instances) of the domain including at least one complete instance of the model; 2) allows representing all feasible states of the domain and disallows representing infeasible ones. We illustrate fulfilling of these requirements with Vehicle ontology example and use it for validating completeness and semantics preserving features of OWL2ToRDB transformation.

The rest of the paper is organized as follows. In section 2, the overview of related works is presented. Section 3 describes requirements for ontologies intended for storing in relational databases. Section 4 presents an ontology example; section 5 − a database schema, obtained by OWL2ToRDB transformation, along with SQL queries capable for obtaining the same results as SPARQL queries to ontology. Section 6 gives conclusions and highlights the future work.

2. Related work

2.1. Ontology design methodologies

For representing ontology in a relational database, ontology should meet certain quality requirements. However, existing methods for transforming OWL ontologies into relational databases rarely analyse such requirements. First, ontology under transformation should be well formed in accordance with rules for ontology design. There are many research works on methodologies for developing ontologies e.g. [16−18] that mainly focus on processes for developing ontologies but not on the essence of domain conceptualization. Consequently, these methods also give no knowledge useful for ensuring quality of ontologies. Fundamentals for conceptualizing the problem domain were the subject of early research on ontologies e.g. [19] but such research works are very generic. They had been conducted much time before the Web Ontology Language had emerged. More recently, quality and principles of ontology design and their relation to conceptual models were considered in [8, 20−25] sources, which seem most relevant for our purpose.

The key requirement for quality of relational databases is their normalisation. The normalisation of information models for relational databases is widely accepted and applied in practise of designing databases and information systems. The analogous normalisation for ontologies was discussed in [22, 23], where goals for normalisation of ontology implementations were described including 1) domain correctness that means that the interpretation of the classification inferred by the ontology reasoner corresponds to the desirable model of the domain; 2) modularity – that ontologies are constructed from independently evolving, explicitly described components. Ontologies assume conceptualisation of the open and fractal world, which is often changing, so such requirements are important for their reuse, maintainability and evolution.

According to [22], ontology consists of primitive concepts, composite concepts, roles, descriptions and axioms. Primitive concepts that are not inferable from other concepts are described by necessary conditions expressed as Boolean combinations of other primitives, descriptions and defined concepts, and they can participate in subsumption hierarchy of other primitive concepts. Composite (defined) concepts that may be derived from other concepts are defined by necessary and sufficient conditions. Roles (object properties) relate concepts and can also participate in subsumption hierarchy; they can be functional, transitive, symmetric, or inverse to other roles. Descriptions define constraints on role-concept pairs. Axioms declare constraints on instances of concepts and their combinations. The goal of normalization is to constrain these constructs so that the ontology meets the reusability, maintainability and evolvability criteria.

The essence of Rector‘s proposal for normalization [22] is that the primitive taxonomy of the domain ontology should consist of disjoint homogeneous trees. Principles proposed for ontology normalization in [23, 24] were used in our research for formulating requirements for ontologies intended to transforming into relational databases.

For representing ontology in a relational database, the ontology must satisfy certain integrity constraints inherent for relational databases. Otherwise, the structure of relational database should become too complex, or semantic information could be lost because it would be impossible to store it in the obtained database schema. Ontology axioms define such constraints but ontology development tools use them for inference and do not support ensuring

integrity of ontology. Ouyang et al. [26] have proposed to represent integrity constraints as OWL axioms and to conduct manual improvement of ontology before transforming it into RDB. However, the set of constraints Ouyang et al. have attributed to manual improvement of ontology is incomplete (e.g. it lacks exact constraints; symmetric, asymmetric, reflexive properties); from the other side, it includes some constraints that are inferable by ontology reasoners (e.g. transitive, inverse functional properties). Though OWL2ToRDB transformation principles are different from the ones proposed by Ouyang et al. [26], we state similar integrity rules for ontology under transformation on the base of OWL 2 specification [3], but we 1) use the reasoner for reducing the manual work; 2) take into consideration other required constraints that were not involved in [26].

2.2. Common semantics of ontologies and information systems

Semantics based mapping of ontology to database concepts can provide better flexibility and practical applicability of obtained relational database schema to applications of information systems. The target database schema should not only store ontology concepts but also should allow efficient querying and manipulating in Semantic Web or Enterprise Information systems, integrating ontological and relational data models etc. Consequently, it is desirable that relational database schemas, obtained from ontology, could represent semantics common to ontological and conceptual data models.

The most fundamental understanding of conceptual schemas was defined by Van Griethuysen in [27] where they are understood as schemas of state and behaviour of information systems. Conceptual schema describes all storable states of problem domain; all causes of changes of these states; how these states are changing; what states are consistent, and what are derivation rules for deriving new states of problem domain.

A problem domain is described by object types and their relation types that are classified to concepts. Conceptual model includes collections of instances that correspond to conceptual schema (a conceptual schema does not include instances and this makes it different from a conceptual model). A concept is an abstract idea that generalizes individual instances (objects). The concept has its intention (definition), extension (a set of objects (instances)) and representation (symbol) [28]. Entity type is a concept, which instances are individual identifiable objects (entities, e. g. locations, persons, goods). Concepts, whose instances are links, are called relationship types. A set of entity and relationship types, representing states of a problem domain, is called a conceptual schema of state.

Relational database schemas mostly are developed on the base of conceptual data models, which usually are presented in modelling languages (ER, ORM or UML). Semantics of conceptual models should involve (in UML terms) object types, relationships, properties, instances and constraints of a problem domain. This semantics is equivalent to semantics of ontology and can be defined on the base of ontological analysis provided by El-Ghalayini et al. [9] where authors state that ontologies and conceptual models have much in common. Ontology as well as a conceptual model consists of concepts, properties, individuals, and constraints (restrictions and axioms). For semantic mapping between ontologies and conceptual models, El-Ghalayini et al. propose rules based on semantics of the Bunge-Wand-Weber (BWW) ontology used for modelling information systems:

Rule 1. An individual corresponds to an entity or object in conceptual model.

Rule 2. Every named class maps to an entity type under corresponding constraints.

Rule 3. Every data property maps to attribute of the class along with constraints associated with this property.

Rule 4. Every object property corresponds to relationship in conceptual model. `SubClassOf` relation in ontology can be mapped to a generalization /specialization relation between super/sub-entity types. In contrast with conceptual models, properties are considered first-class elements in ontology and can exist without specifying classes related by that property. Such properties are senseless in databases and conceptual models [9]. El-Ghalayini et al. propose to map only properties that are related to classes satisfying constraints in the ontology under consideration.

Rule 5: Every property constraint used in ontology class corresponds to a relationship constraint that restricts the kind of the relation, number of entities or entity type of this relationship. `ObjectIntersectionOf` and `ObjectUnionOf` are mapped to {and} and {or} constraints. An existential and universal quantifier restrict the minimum cardinality of the target entity type to 1 and its maximum cardinality to n. Cardinality restricts the number of objects of the target entity type that can participate in a role of a relationship.

Rule 6: A composition relation corresponds to a property relating a composite and its components where the existence of the component depends on the composite, i.e. any individual of a component-class, which is connected to an individual of a whole-class, must not be connected to any other individual. A composition relation has no direct representation in ontology but it can be defined by ontology constraints.

Our proposed OWL2ToRDB transformation [29, 30] supports this common semantics [9]. It maps ontology classes to relational tables; individuals – to rows; functional object properties (or having cardinality restricted to 1) – to foreign keys; object properties with cardinality greater than 1 – to tables

having foreign key relations from tables representing classes of property domain and range; `SubClassOf` relation is mapped to a foreign key from a parent table to a child table, for which that foreign key, a primary key of a parent table, is also a primary key; data properties are mapped to columns; OWL data types – to RDB data types. Such a relational schema satisfies mappings of conceptual models to relational schemas and corresponds to practised schemas of relational databases well understandable for analysts and application developers. Besides, that schema has metatables for storing axioms and properties of OWL 2 constructs – i.e. explicit (versus implicit [31]) representation of ontology constraints that may be used for ensuring integrity of a database or reasoning about its contents.

2.3. Retrieving ontology data from RDB

There are several possibilities for querying ontologies from relational databases. Semantic query language SPARQL [32, 33] is used for querying ontologies on the Web. When ontology data are stored in a relational database, queries may be executed in several ways: 1) on ontology layer when ontology and its instances are recovered from a database into ontology processing environment; 2) both on ontology model layer (for finding classes, properties and restrictions), and relational database layer for finding individuals and assertions; 3) on database layer when ontology concepts and instances are retrieved via SQL queries.

The first case is a traditional approach to querying ontologies; it suffers from problems related with handling large ontologies. The second case for OWL2RDB method was investigated in [10]. For searching concepts of ontology model, Pellet OWL Reasoner was used here to manipulate the ontology model restored from OWL 2 metaschema in relational database. This recovered ontology model does not contain information about individuals and assertions; they are retrieved from the tables of relational schema by executing SQL queries. The querying algorithm firstly executes a part of SPARQL query responsible of obtaining information about ontology concepts, and then separately searches individuals and assertions by using SQL query language. The experimental investigation of the algorithm has shown its semantic equivalence to traditional approach (i.e. querying ontologies and instances entirely on ontology layer) along with better capabilities for handling large ontologies.

In the third case, when ontology concepts and instances are retrieved exclusively via SQL queries, querying depends on ontology storage model and the mode of storing ontology. If obtained database stores ontology and its instances together with inference results, inferred values can be retrieved by SQL queries in the same manner, as other data. If inferred data are not stored, as in the case of OWL2ToRDB transformation, SQL queries are needed for extracting information about ontology concepts and axioms as well as for finding instances.

Semantics preserving SPARQL to SQL translation was investigated in [34] where a proof is given for equivalence of SPARQL and SQL interpretations of RDF data and their representation in relational database. The same proof is valid for OWL ontologies if we analyse them w.r.t. RDF-based semantics. Regarding direct OWL 2 semantics [35], transformation [34] is necessary but not sufficient as OWL 2 gives additional sense beyond RDF. This factor requires additional SQL queries for understanding the inquired ontology and formulating further queries allowing obtaining data, relevant to source ontology, from RDB. We present such query patterns in Section 5.

3. Requirements for ontology under transformation

For ensuring that OWL2ToRDB transformation is semantics preserving, source ontology should conform to **Ontology Normalization Rules** (ONR1–ONR6) summarized from [9, 22, 23]. In [24], additional advices are given how to ensure domain correctness. We refer to these advices as to additional ontology normalization rules (ONR7–ONR10) (Table 1).

For ensuring consistency of ontology under transformation, we supplement ontology normalization rules with **Ontology Integrity Rules** (OIR1–OIR18), formulating them on the base of OWL 2 specification [1] (Table 2).

Table 1. Ontology normalization rules

Rule	Desciption
ONR1	No primitive domain concept should have more than one primitive parent.
ONR2	Primitive taxonomy of the domain ontology should comprise homogeneous disjoint trees, specialised by subsumption based on the same or gradually narrower criteria.
ONR3	Self-standing concepts (types and roles) of primitive taxonomy should form open disjoint taxonomies where all the primitive children of each primitive concept should be disjoint, but not necessarily covering the parent. Value types and values should form closed taxonomies where the primitive values are disjoint, but primitive value types may be disjoint or overlapping.
ONR4	Any named individual must be an instance of exactly one most specific self-standing concept; axioms should be defined in such a way that inferences should never result in subsumption of one primitive concept by another, since this would

Rule	Desciption
	denormalise the ontology.
ONR5	Primitive concepts should be described by conjunctions of one primitive and zero or more descriptors; every primitive open concept or value should be disjoint with its siblings; every set of primitive values of a concept should be covering.
ONR6	The primitive taxonomy of domain concepts should be untangled, i.e. if a primitive concept is disjoint from its siblings, its children must also be disjoint, and if a primitive concept is part of a partition, its children must also form a partition.
ONR7	To avoid trivially satisfiable restrictions, an existential restriction `someValuesFrom` should supplement every universal restriction `allValuesFrom` in the class or one of its superclasses.
ONR8	For ensuring desired inferences, classes should be defined.
ONR9	To avoid the open world assumption, closure restrictions should be defined for covering subclasses.
ONR10	To be able to represent ontology in relational database, each `ObjectProperty` or its parent must have exactly one domain and exactly one range; the same holds for each `DataProperty`.

Table 2. Ontology integrity rules

Rule	Desciption
OIR1	For each functional object property `OPE` and for each individual `x`, there can be at most one distinct individual `y` such that `x` is connected by `OPE` to `y`. The same rule holds if `ObjectMinCardinality` restriction on `OPE` is less than 1 and `ObjectMaxCardinality` equals 1.
OIR2	For each inverse functional object property `OPE` and for each individual `x`, there can be at most one distinct individual `y` such that `y` is connected by `OPE` with `x`.
OIR3	For each object property `OPE` and for each individual `x`, there must be exactly one distinct individual `y` such that `x` is connected by `OPE` to `y` if `ObjectMinCardinality` and `ObjectMaxCardinality` restriction on `OPE` is equal to 1, or `ObjectExactCardinality` equals 1.
OIR4	For each object property `OPE` and for each individual `x`, there must be exactly `n` distinct individuals `y` such that `x` is connected by `OPE` to `y` if `ObjectExactCardinality` restriction on `OPE` equals `n`.
OIR5	For each object property `OPE` and for each individual `x`, there must be at least `n` distinct individuals `y` such that `x` is connected by `OPE` to `y` if `ObjectMinCardinality` equals `n`.
OIR6	For each object property `OPE` and for each individual `x` that is an instance of class `C` having existential class expression `ObjectSomeValuesFrom (OPE C)` there must be at least one distinct individual `y` such that `x` is connected by `OPE` to `y`.
OIR7	For each object property `OPE` and individual `x`, there must be at most `n` distinct individuals `y` such that `x` is connected by `OPE` to `y` if `ObjectMaxCardinality` equals `n`.
OIR8	For each object property `OPE` and for each individual `x`, the value of `OPE` must equal individuals $y_1, \ldots, y_n$ that are connected to `x` by class expressions `ObjectHasValue (OPE` y_1`), …. ,ObjectHasValue (OPE` y_n`)`.
OIR9	For each functional data property `DPE` and for each individual `x`, there can be at most one distinct literal `y` such that `x` is connected by `DPE` to `y`. The same rule holds if `DataMinCardinality` is less than `1` and `DataMaxCardinality is` equal `1`.
OIR10	For each data property `DPE` and for each individual `x`, there must be exactly one distinct literal `y` such that `x` is connected by `DPE` to `y` if `DataMinCardinality` and `DataMaxCardinality` equal `1`, or `DataExactCardinality` equals `1`.
OIR11	For each object property `DPE` and for each individual `x`, there must be exactly `n` distinct literals `y` such that `x` is connected by `DPE` to `y` if `DataExactCardinality` equals `n`.
OIR12	For each data property `DPE` and for each individual `x`, there must be at least `n` distinct literals `y` such that `x` is connected by `DPE` to `y` if `DataMinCardinality` equals `n`.
OIR13	For each data property `DPE` and for each individual `x`, there must be at most `n` distinct literals `y` such that `x` is connected by `DPE` to `y` if `DataMaxCardinality` equals `n`.
OIR14	For each data property `DPE` and for each individual `x` that is an instance of class `C` having existential class expression `DataSomeValuesFrom (DPE C)` there must be at least one distinct literal `y` such that `x` is connected by `DPE` to `y`.
OIR15	For each data property `DPE` and for each individual `x`, the value of `DPE` must equal literals $y_1, \ldots, y_n$ that are connected to `x` by class expressions `DataHasValue(DPE` y_1`), …, DataHasValue (DPE` y_n`)`.
OIR16	For ensuring ontology consistency and integrity rules, ontology must have enough number of instances – at least one individual for each class and at least `n` individuals required for each integrity rule. For example, for ensuring `ObjectMinCardinality(5 OPE C)` we will need to have `5` instances of class `C`.
OIR17	All inferable concepts and properties should not have values, properties or restrictions, and should not participate in axioms because it could make ontology inconsistent.
OIR18	Inference should be made before transforming ontology for ensuring consistency of ontology and retrieving all relevant data from the obtained database via `SQL` queries. Ontological data are inferred in accordance with symmetric, reflexive, transitive, inverse, equivalent object properties, object property chains, object has value, and other axioms. Another set of axioms and restrictions (e.g. functional properties, existential class expressions, etc) serve for ensuring consistency of ontology.

4. An ontology example

Ontology example, constructed according ONR1−ONR10 and OIR1−OIR18 rules, is represented in Figure 1. For brevity, we do not use full UML OWL 2 profile [36] as the mapping of OWL 2 constructs to UML is self-explaining. In Figure 1, UML classes represent OWL 2 classes; associations represent object properties; generalizations stand for `SubClassOf` axioms; attributes represent data properties; UML constraints correspond to OWL 2 restrictions; 0:0..1 cardinality corresponds to functional property or inverse of inverse functional object property (except settings when explicit cardinality restrictions exist); cardinality of 1 represent exact (object or data) cardinality; 0:n cardinality corresponds to non-functional property or inverse of functional property. Vehicle ontology involves at least one representative of main OWL 2 constructs (examples are given in Tables 1−2) thus it can serve as a representative example for validating OWL2RDB capability to preserve OWL 2 semantics in a relational database. Types and examples of OWL 2 constructs having impact on semantic OWL2 to RDB mapping (i. e. structure of schema tables) are presented in Table 3; constructs having impact on semantic querying are listed in Table 4.

For ensuring semantic suitability of ontology model I, we must describe at least one interpretation of ontology (a set of named individuals) satisfying all its axioms and having at least one interpretation for each construct of the ontology [35]. Such an interpretation is also a model of ontology.

Ontology model may be constructed w.r.t. principles of Formal Concept Analysis (FCA) [37] applied in formal and relational contexts [38]. In FCA, a formal context is defined as a triple $C = (U, A, B)$, where U is a universe (a finite set of objects), A is a finite set of attributes, and B is a binary relation between U and A. Concept structure of Vehicle ontology is presented in Figure 2. Such a structure comprises a complete lattice as every concept has its greatest lower and smallest upper bound [37]. FCA allows creating right taxonomies and validating completeness of concept lattices but it does not directly help to construct a set of individuals comprising ontology model because it does not take into account relations between objects.

Table 3. Ontology concepts having impact on semantic mapping

OWL 2 construct	Example
SubClassOf	SubclassOf(a:Organization a:Party)
ObjectInverseOf	ObjectInverseOf(a:isAssembledFrom)
DisjointClasses	DisjointClasses(a:Organization a:Person))
DisjointUnion	DisjointUnion(a:Party(a:Organization a:Person))
ObjectUnionOf	ObjectUnionOf(a:Organization a:Person)
ObjectHasSelf	ObjectHasSelf(a:believesIn)
ObjectSomeValuesFrom	ObjectSomeValuesFrom(a:hasInsurance a:Insurance)
ObjectAllValuesFrom	ObjectAllValuesFrom(a:hasInsurance a:Insurance)
ObjectHasValue	ObjectHasValue(a:isSuppliedBy a:Company3)
FunctionalObjectProperty	FunctionalObjectProperty(a:hasFather)
InverseFunctionalObject Property	InverseFunctionalObjectProperty(a:hasInsurance)
SubObjectPropertyOf	SubObjectPropertyOf(a:isProducedBy a:hasMaker)
DisjointObjectProperties	DisjointObjectProperties(a:hasMother a:hasFather)
ObjectMinCardinality	ObjectMinCardinality(5 a:isAssembledFrom a:AutomobilePart))
ObjectMaxCardinality	ObjectMaxCardinality(1 ObjectHasSelf(a:hasChief a:Assurer))
ObjectExactCardinality	ObjectExactCardinality(1 a:insuredBy a:Assurer))
FunctionalDataProperty	FunctionalDataProperty(a:birthDate xsd:dateTime)
SubDataPropertyOf	SubDataPropertyOf(a:personCode a:partyCode).
DataMinCardinality	DataMinCardinality(a:AutomobileModelVersionTitle xsd:string)
DataMaxCardinality	DataMaxCardinality(a:ownedTillDate xsd:dateTime)
DataExactCardinality	DataExactCardinality(1 a:automobileModelTitle xsd:string))
DataHasValue	DataHasValue(a:automobileCountryCode "LT"^^xsd:string)
HasKey	HasKey(a:AutomobileModelVersion(a:isOfAutomobileModel) (a:AutomobileModelVersionNumber))

Table 4. Ontology concepts having impact on semantic querying

OWL 2 construct	Example
SubClassOf	SubclassOf(a:Organization a:Party)
EquivalentClasses	EquivalentClasses(a:Automobile a:Car)
ObjectInverseOf	ObjectInverseOf(a:isAssembledFrom)
TransitiveObjectProperty	TransitiveObjectProperty(a:consistsIn)
SymmetricObjectProperty	SymmetricObjectProperty(a:isColleagueOf)
EquivalentObjectProperties	EquivalentObjectProperties(a:isColleagueOf a:isPartnerOf)
EquivalentDataProperties	EquivalentDataProperties(a:certificationNumber a:assurerLicenceNumber)
SubObjectPropertyOf	SubObjectPropertyOf(a:isProducedBy a:hasMaker).
SubDataPropertyOf	SubDataPropertyOf(a:companyCode a:partyCode).
ObjectPropertyChain	SubObjectPropertyOf(ObjectPropertyChain(a:isInsuredBy a:isEmployedBy) a:isVerifiedBy)
HasKey	HasKey(a:AutomobileModelVersion(a:isOfAutomobileModel) (a:AutomobileModelVersionNumber))

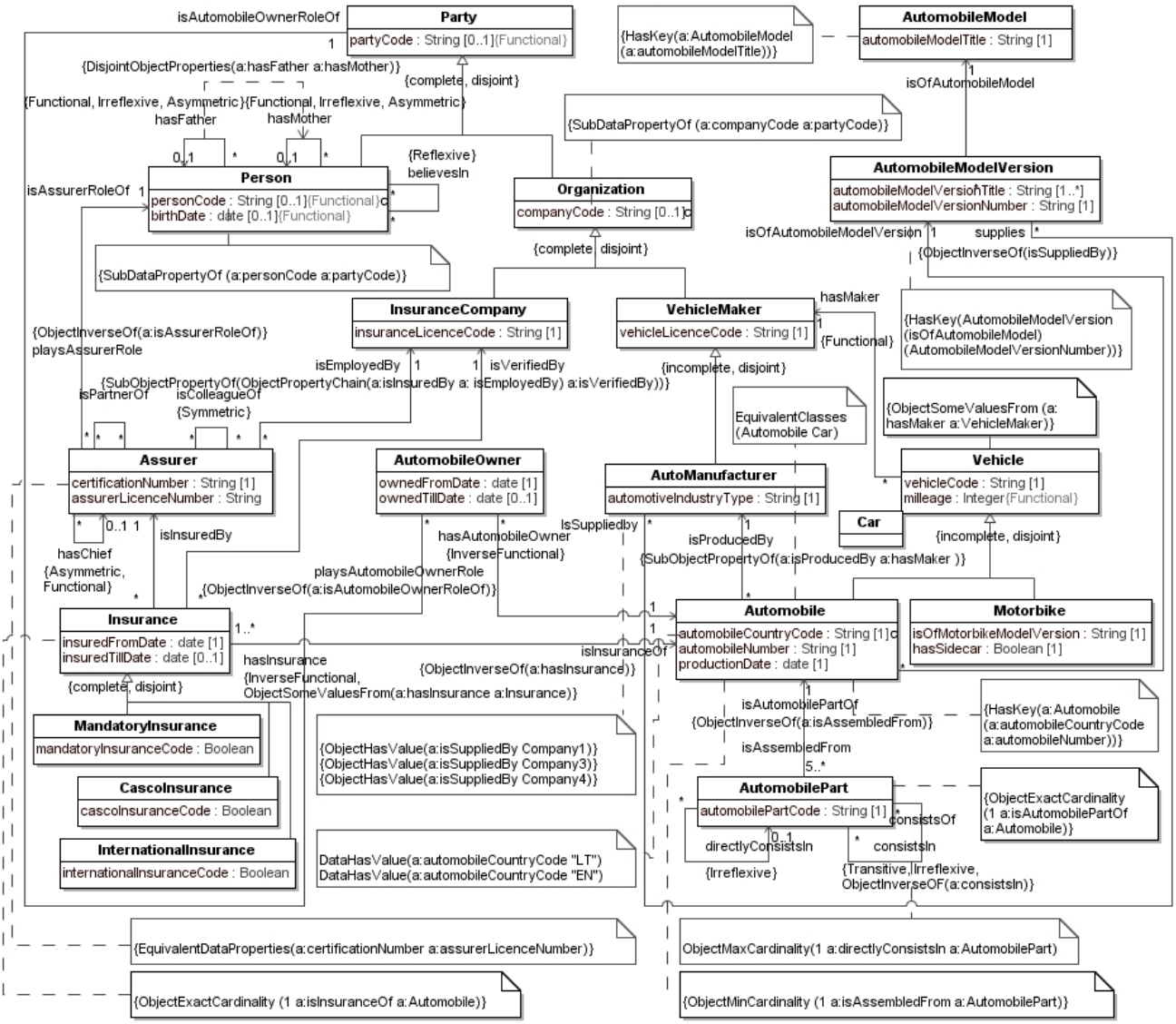


Figure 1. Vehicle ontology represented by UML class diagram (annotations are not visualized due to space limits)

A relational context is defined as a pair $R = (U, r)$, where r is a binary relation in U and describes links between objects of U. In [38], the formal proof is given that C is a corresponding formal context of relational context R, if B is such that for any $x, y \in U$, $a_y \in A \Rightarrow xBa_y$ if $r(x, y)$. Further, Jiang et al. [38] introduce reflexive, transitive and symmetric relations r.

Analysis of a relational context can help to ensure the partial order w.r.t. functional dependencies and identify a number of individuals required for having a complete set of them comprising ontology model. Concept hierarchy extended with functional dependencies of Vehicle ontology is represented in Figure 3 (note that UML diagram of Vehicle ontology in Figure 1 also reflects ordering based both on `SubClassOf` relation and functional object properties). One object is required for representing each most specific concept in `SubClassOf` hierarchy if all siblings in that hierarchy are disjoint and covering their parents; open disjoint hierarchies require one object for each concept. Bijective functional dependencies between two concepts require one object for each concept; functional dependencies (i.e. `FunctionalObjectProperty or Inverse of Inverse Functional Object Property`) require two objects for a domain concept and one object for a range to analyse cases when the relation exists and when it does not.

For representing reflexive, irreflexive, symmetric, and assymetric relations between objects, we need two objects for representing one concept from the corresponding formal context, and three objects for transitive relations; constraints on a number of objects participating in relations also should be taken into account. One value or value object is required for each value from closed disjoint value sets. The number of required objects may be reduced as the same object can (or even must) participate in several relations if this participation is not forbidden by axioms and does not cause inconssistency of ontology.

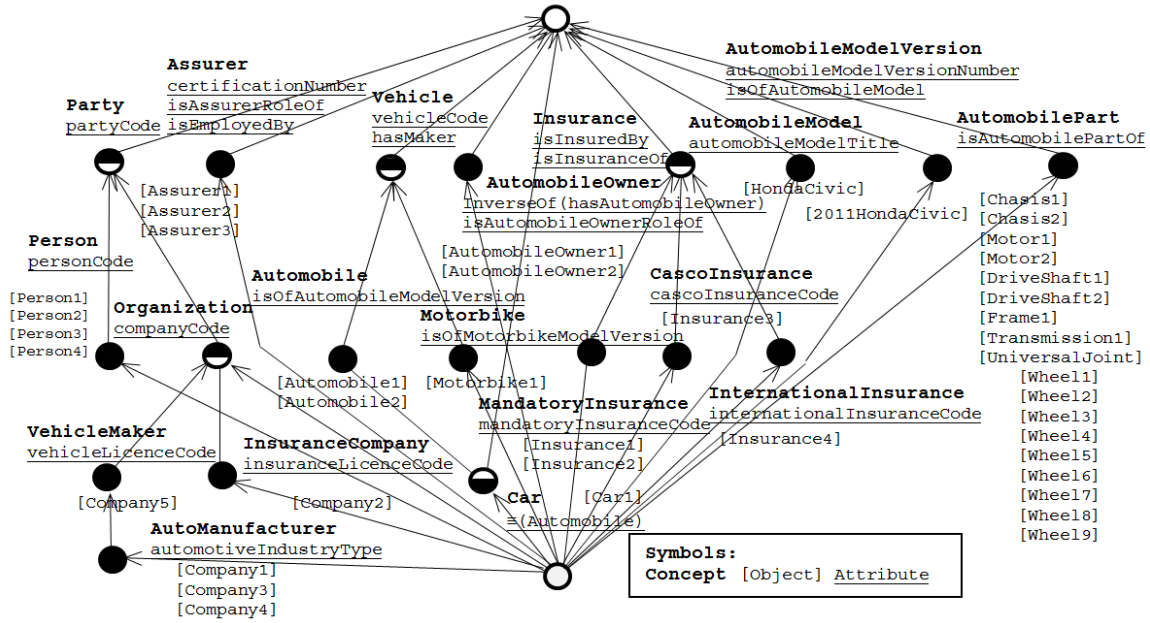


Figure 2. Concept lattice of the Vehicle ontology

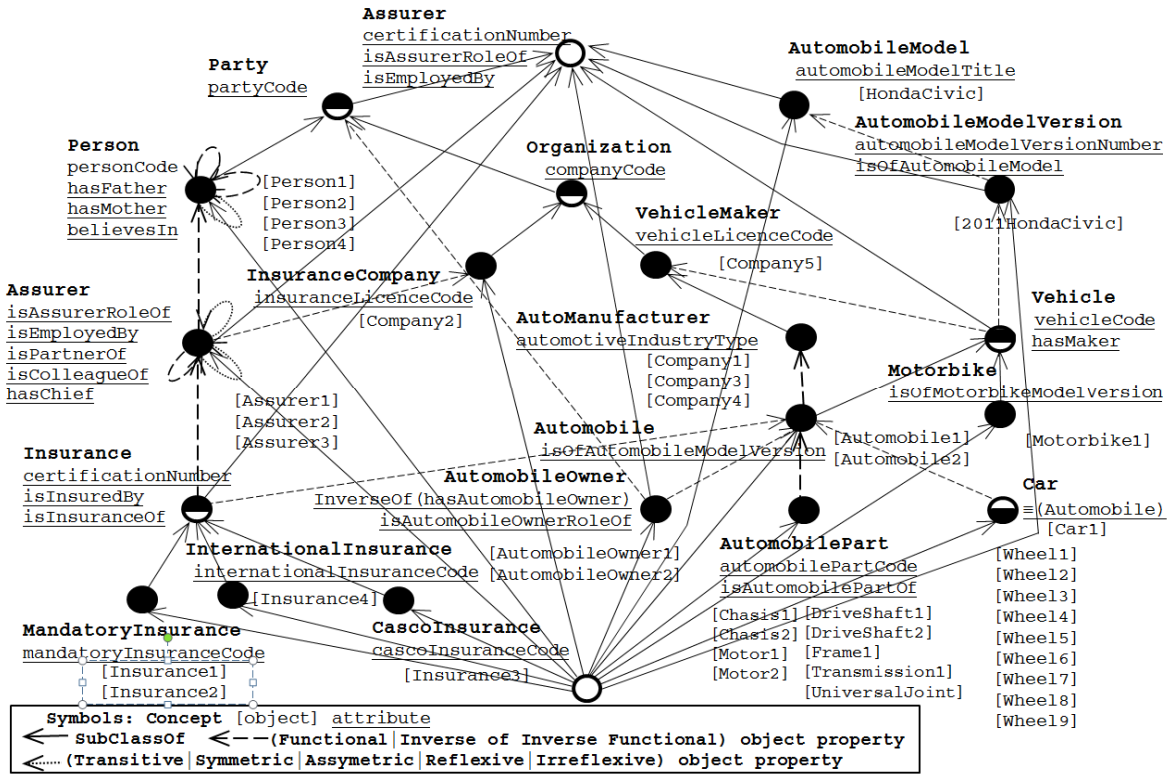


Figure 3. Concept structure of the Vehicle ontology ordered w.r.t. `SubclassOf` and functional dependencies

Following these principles, the set of 42 objects for the Vehicle ontology was identified allowing us to define a partial order w.r.t. `SubClassOf` relation and functional dependencies on a set of objects of the Vehicle domain. Note that in Figures 2 and 3, only those attributes (from Figure 1), which correspond to essential characteristics and determine existence of objects, were used. The set of these 42 individuals is a model I satisfying the Vehicle ontology O (this was verified by HermiT reasoner in Protégé) and it is suitable for validating semantics preserving properties of OWL2ToRDB transformation.

5. Representing and querying ontology in database

Relational database schema obtained from Vehicle ontology by applying OWL2ToRDB transformation,

implemented in Protégé plug-in, is presented in Figure 4. The overall transformation and metaschema are described in detail in [29, 30]; here we concentrate on retrieving the semantics of ontology when it is stored in a database.

Querying ontology axioms is required for knowing how to retrieve additional, richer data that ontology contains beyond those stored in relational tables. For example, we can find instances of equivalent classes, values of inverse or other inferable object properties etc.

For querying transitive relations, equivalent recursive SQL queries are needed while queries of symmetric, equivalent and inverse relations are much simpler. In the hybrid method for storing ontologies in a database method, transitive subsumption relations between classes and properties are discovered by SQL queries on metatables. A special attention is needed for querying OWL 2 transitive, symmetric, equivalent, inverse properties and object property chains.

Transitivity, symmetry, equivalence, or inversion of relational table records is not supported by native functionality of relational databases. For example, the transitive closure of the relation *consistsOf* (Figure 1) is a relation that contains all automobile part pairs (*i, j*) such that *j* is direct or indirect part of *i*. The previous SQL standard, SQL'92, did not support computation of transitive closure; therefore, relational databases were not suitable for using them for storing knowledge data. Current SQL standard, SQL'99 supports iterative or recursive SQL queries and allows to compute transitive closure.

SPARQL queries for graph patterns presented in Figure 5 would find subclasses (Q1) and subproperties; instances by types (Q2), supertypes (Q3) and key values (Q4); annotations, e.g. comments (Q5), labels (Q6); property values by properties (Q7) and subproperties; property paths (Q8), inverse properties (Q9); object property chains (Q10); equivalent (Q11), transitive (Q12) and symmetric (Q13) properties. Answers to these queries (Q1a–Q13a) are given in Figure 8.

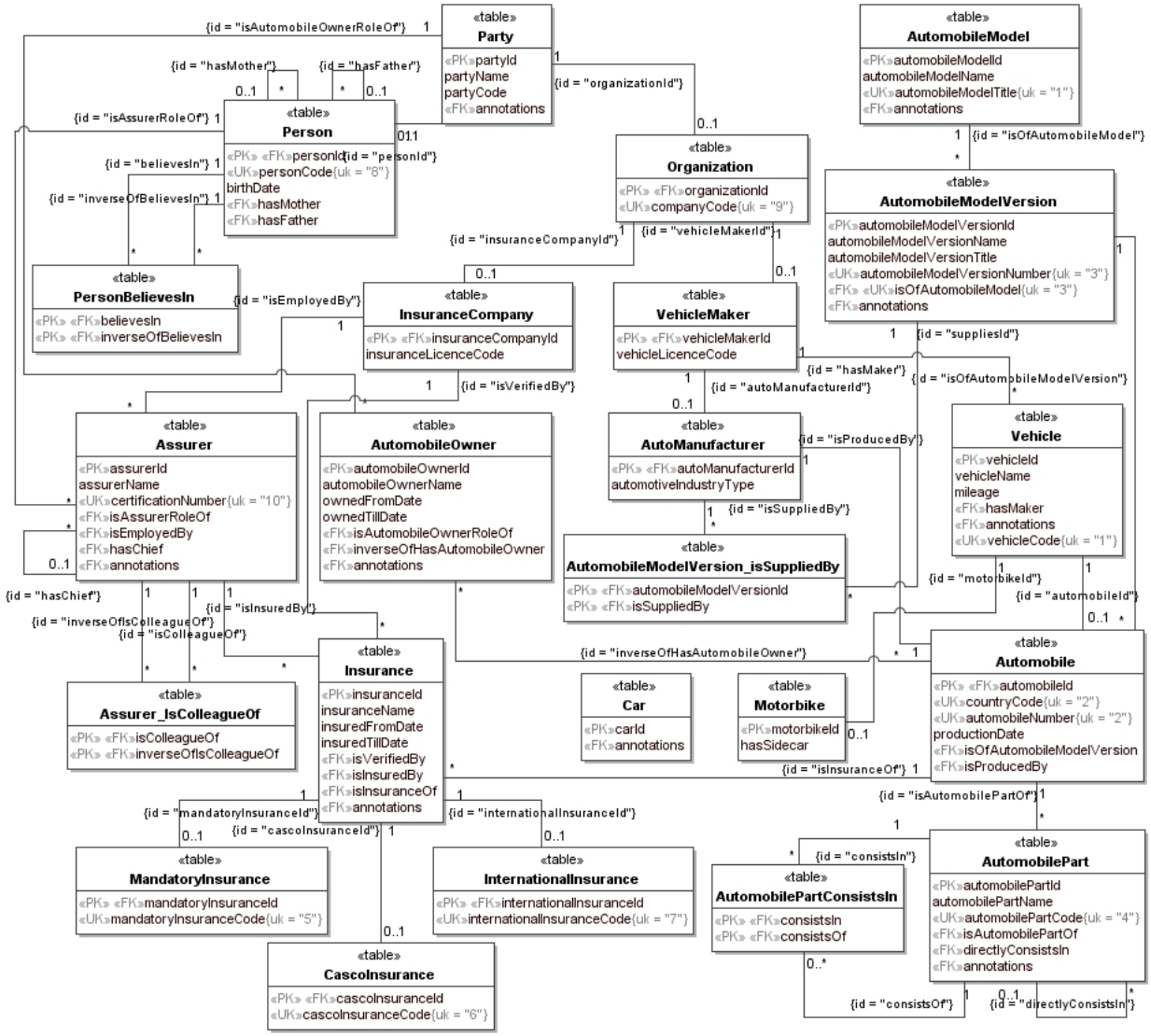


Figure 4. Relational database schema for Vehicle ontology

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cars: <http://www.semanticweb.org/ontologies/2011/7/Ontology1313497711109.owl#>
Q1: select ?subclasses {?subclasses rdfs:subClassOf cars:Vehicle}
Q2: select ?instances {?instances rdf:type cars:Automobile}
Q3: select ?instances {?instances rdf:type cars:Party}
Q4: select ?automobile {?automobile cars:automobileNumber "136"^^xsd:string .
   ?automobile cars:countryCode "LT"^^xsd:string}
Q5: select ?x ?y {?x rdfs:comment ?y}
Q6: select ?x ?y {?x rdfs:label ?y}
Q7: select ?automobile ?maker ?insurance ?production_date {?automobile cars:isProducedBy
    ?maker . ?insurance cars:isInsurranceOf ?automobile .
    ?automobile cars:productionDate ?production_date}
Q8: select ?automobile ?insurance ?insurance_company{?insurance cars:isInsurranceOf ?automobile .
    ?insurance cars:isInsuredBy ?assurer . ?assurer cars:isEmployedBy ?insurance_company}
Q9: select ?parts {cars:Automobile1 cars:isAssembledFrom ?parts}
Q10:select ?insurance ?is_verified_by {?insurance cars:isInsuredBy ?assurer .
    ?assurer cars:isEmployedBy ?is_verified_by}
Q11: select ?person ?is_partner_of {cars:isColleagueOf owl:equivalentProperty ?eq .
     ?person ?eq ?is_partner_of}
Q12: select ?part1 ?part2 {?part1 cars:consistsIn+ ?part2}
Q13: select ?person ?colleague {{?person cars:isColleagueOf ?colleague}
     Union {?colleague cars:isColleagueOf ?person}}
```

Figure 5. SPARQL queries for graph patterns of Vehicle ontology

More complex SPARQL queries may be formulated from graph patterns, obtained by queries in Figure 5, by using SPARQL conjunction, OPTIONAL, UNION, FILTER constructs [32] or more advanced features of SPARQL 1.1 [33]. FILTER expressions may include IRIs, variables, literals, constants, logical connectives (¬, ∧, ∨), inequality symbols (<, ≤, >, ≥), the equality symbol (=), and other features.

For querying ontology from database and retrieving all semantics obtainable by SPARQL, we should know structure of ontology. Therefore, first we should extract ontology constructs. SQL queries providing information about ontology from metatables, should find all classes; subclass – class pairs; object properties, domains and ranges; inverse object properties, their domains and ranges; equivalent object and data properties, their domains and ranges; properties of properties − symmetric, reflexive and transitive object properties, sub object properties and sub data properties; keys and object property chains (e.g. in Figure 6). This knowledge should be used for formulating queries allowing finding rich information equivalent to source ontology.

SQL queries on metatables are simple ones; some examples for finding transitive (SM1) and inverse (SM2) object properties along with `OWLObjectProperties` metatable and results of these queries (SM1a, SM2a) are presented in Figure 6.

SQL queries, retrieving ontology data from relational database, are presented in Figure 7. They provide results S1a−S13a equivalent to SPARQL queries Q1a−Q13a (Figure 8) and may be applied as query patterns for constructing SQL queries about ontologies of other domains.

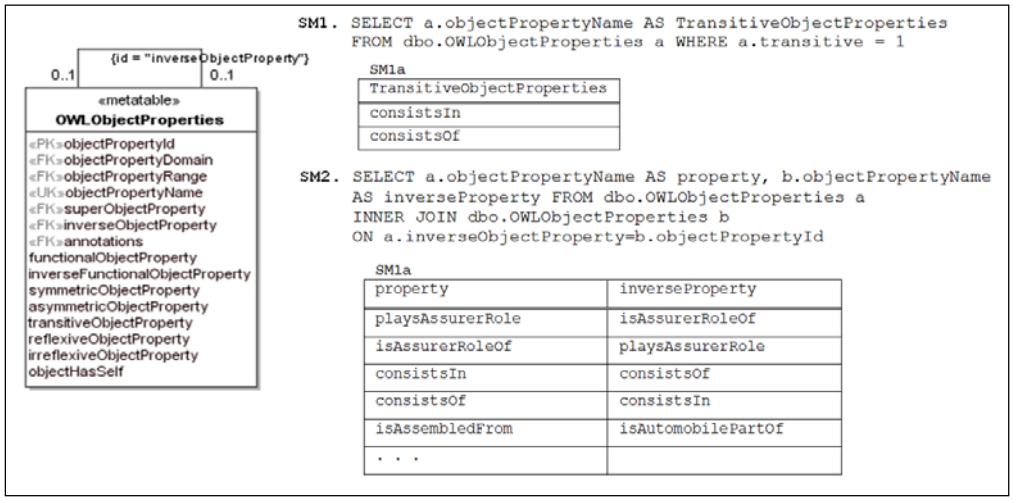


Figure 6. Example of SQL queries on metatable `OWLObjectProperties` (more metatables are presented in [29])

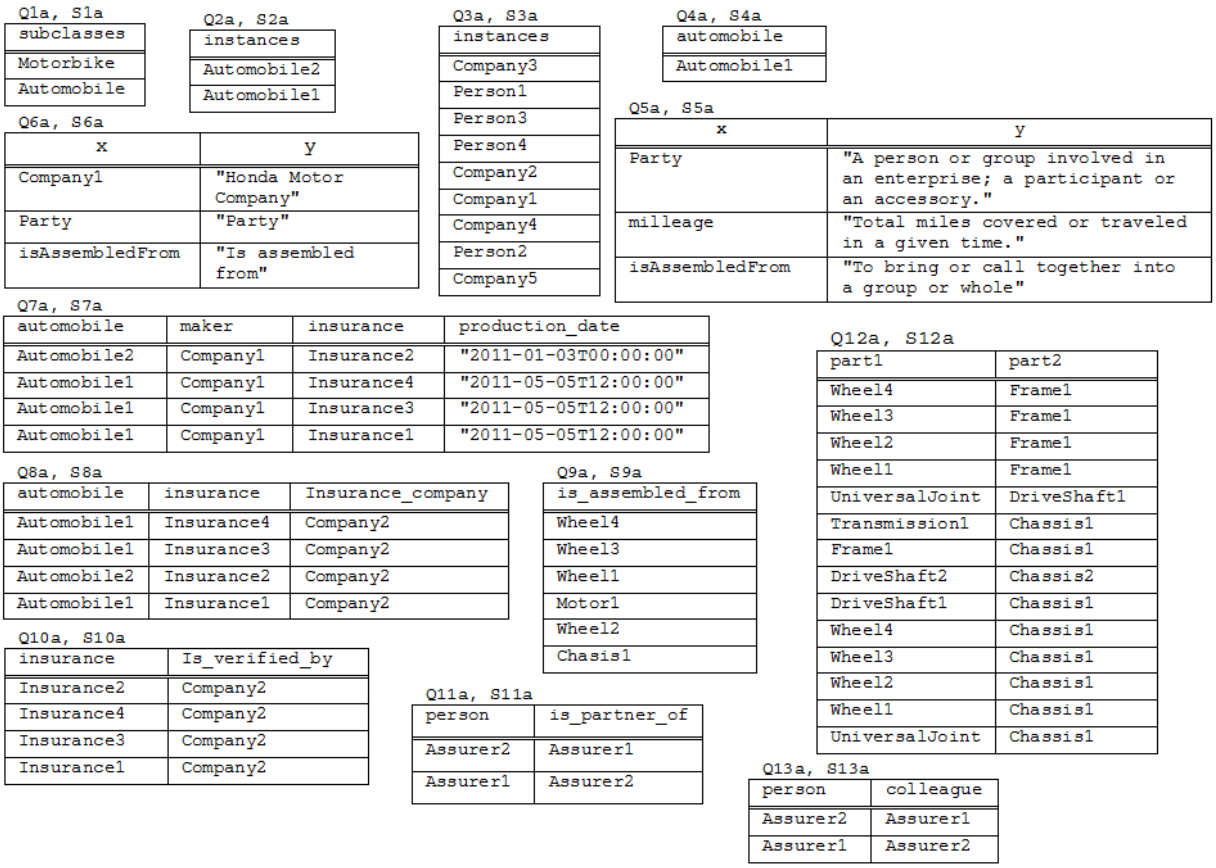
```
S1: SELECT b.className as subclasses FROM dbo.OWLClasses a
    INNER JOIN dbo.OWLClasses b ON a.classId = b.superClass WHERE a.className = 'Vehicle'
S2: SELECT b.vehicleName as instances FROM dbo.Automobile a
    INNER JOIN dbo.Vehicle b ON a.automobileId = b.vehicleId
S3: SELECT a.PartyName as instances FROM dbo.Party a
S4: SELECT b.vehicleName AS auotomobile FROM dbo.Automobile a INNER JOIN dbo.Vehicle b ON
    a.automobileId = b.vehicleId WHERE a.automobileNumber = '136' AND a.countryCode = 'LT'
S5: SELECT c.className as x, a.annotationValue as y FROM dbo.OWLAnnotation a
    INNER JOIN dbo.OWLAnnotationProperties b ON a.annotationProperty = b.annotationPropertyId
    INNER JOIN dbo.OWLClasses c ON c.annotations = a.annotationOf WHERE b.annotationPropertyName = 'comment'
    UNION ALL SELECT d.dataPropertyName as x, a.annotationValue as y FROM dbo.OWLAnnotation a
    INNER JOIN dbo.OWLAnnotationProperties b ON a.annotationProperty = b.annotationPropertyId
    INNER JOIN dbo.OWLDataProperty d ON d.annotations = a.annotationOf
    WHERE b.annotationPropertyName ='comment'
    UNION ALL SELECT o.objectPropertyName as x, a.annotationValue as y FROM dbo.OWLAnnotation a
    INNER JOIN dbo.OWLAnnotationProperties b ON a.annotationProperty = b.annotationPropertyId
    INNER JOIN dbo.OWLObjectProperties o ON o.annotations = a.annotationOf
    WHERE b.annotationPropertyName = 'comment'
S6: SELECT a.annotationValue as y FROM dbo.OWLAnnotation a INNER JOIN dbo.OWLAnnotationProperties b ON
    a.annotationProperty = b.annotationPropertyId WHERE b.annotationPropertyName = 'label'
S7: SELECT c.vehicleName AS auotomobile, d.partyName AS maker, b.insuranceName AS insurance,
    a.productionDate AS production_date FROM dbo.Automobile a INNER JOIN dbo.Insurance b
    ON a.automobileId = b.isInsurranceOf INNER JOIN dbo.Vehicle c ON a.automobileId = c.vehicleId
    INNER JOIN dbo.Party d ON d.partyId = a.isProducedBy
S8: SELECT g.vehicleName AS auotomobile, b.insuranceName AS insurance, f.partyName AS insurance_company
    FROM dbo.Automobile a INNER JOIN dbo.Insurance b ON a.automobileId = b.isInsurranceOf
    INNER JOIN dbo.Assurer c ON b.isInsuredBy = c.assurerId INNER JOIN dbo.InsuranceCompany d
    ON c.isEmployedBy = d.insurancecompanyId INNER JOIN dbo.Party f ON d.insurancecompanyId = f.partyId
    INNER JOIN dbo.Vehicle g ON a.automobileId = g.vehicleId
S9: SELECT c.vehicleName as Automobile, a.automobilepartName as AutomobilePart FROM dbo.AutomobilePart a
    INNER JOIN dbo.Automobile b ON a.isAutomobilePartOf = b.automobileId
    INNER JOIN dbo.Vehicle c ON b.automobileId = c.vehicleId WHERE c.vehicleName = 'Automobile1'
S10: SELECT a.insuranceName AS insurance, d.partyName AS is_verified_by FROM dbo.Insurance a
     INNER JOIN dbo.Assurer b ON a.isInsuredBy = b.assurerId
     INNER JOIN dbo.InsuranceCompany c ON b.isEmployedBy = c.insurancecompanyId
     INNER JOIN dbo.Party d ON c.insurancecompanyId = d.partyId
S11: select ?insurance ?is_verified_by {?insurance cars:isInsuredBy ?assurer .
     ?assurer cars:isEmployedBy ?is_verified_by}
S11: SELECT a.assurerName AS person, c.assurerName AS is_partner_of FROM dbo.Assurer a
     INNER JOIN dbo.AssurerIsColleagueOf b ON a.assurerId = b.assurer1Id
     INNER JOIN dbo.Assurer c ON b.assurer2Id = c.assurerId UNION ALL
     SELECT a.assurerName AS person, c.assurerName AS is_partner_of FROM dbo.Assurer a
     INNER JOIN dbo.AssurerIsColleagueOf b ON a.assurerId = b.assurer2Id
     INNER JOIN dbo.Assurer c ON b.assurer1Id = c.assurerId
S12: WITH AutoParts (Part1, Part2, PartId) AS (
     SELECT a.automobilepartName, b.automobilePartName, a.automobilepartId FROM dbo.AutomobilePart a
     INNER JOIN dbo.AutomobilePartConsistsIn c ON a.automobilepartId = c.automobilepart1Id
     INNER JOIN dbo.AutomobilePart b ON c.automobilepart2Id = b.automobilepartId
     UNION ALL SELECT a.automobilepartName, b.Part2, a.automobilepartId FROM dbo.AutomobilePart a
     INNER JOIN dbo.AutomobilePartConsistsIn c ON a.automobilepartId = c.automobilepart1Id
     INNER JOIN AutoParts b ON c.automobilepart2Id = b.PartId)
     SELECT Part1, Part2 FROM AutoParts ORDER BY Part2 DESC
S13: SELECT a.assurerName AS person, c.assurerName AS colleague FROM dbo.Assurer a
     INNER JOIN dbo.AssurerIsColleagueOf b ON a.assurerId = b.assurer1Id
     INNER JOIN dbo.Assurer c ON b.assurer2Id = c.assurerId UNION ALL
     SELECT a.assurerName AS person, c.assurerName AS colleague FROM dbo.Assurer a
     INNER JOIN dbo.AssurerIsColleagueOf b ON a.assurerId = b.assurer2Id
     INNER JOIN dbo.Assurer c ON b.assurer1Id = c.assurerId
```

Figure 7. SQL queries for retrieving Vehicle ontology data from relational database

6. Conclusions and future works

Analysis provided in the paper has shown that the hybrid method for transforming OWL 2 ontology to relational database allows meeting contrasting requirements of preserving all ontology information and having a meaningful database schema corresponding to semantics of information systems. For this purpose, the ontology under transformation must meet ontology normalization and integrity rules taken from several sources and summarized in the paper. The method was analysed using a representative example – ontology model satisfying Vehicle ontology created according ontology normalisation and integrity rules.

The method is able to cover all OWL 2 constructs by storing information about them in metatables. Metatables explicitly represent ontology constraints that may be used for ensuring integrity of a database or reasoning about its contents. Having knowledge about OWL 2 transitive subsumption relations between classes as well as transitive, symmetric, equivalent, inverse object properties and object property chains, it becomes possible to retrieve more meaningful data on the base of subsumption, transitivity, symmetry, equivalence, or inversion of relational tables. SQL query patterns are presented for retrieving semantics, equivalent to semantics of OWL 2 ontologies, from databases.



Q1a, S1a

subclasses
Motorbike
Automobile

Q2a, S2a

instances
Automobile2
Automobile1

Q3a, S3a

instances
Company3
Person1
Person3
Person4
Company2
Company1
Company4
Person2
Company5

Q4a, S4a

automobile
Automobile1

Q6a, S6a

x	y
Company1	"Honda Motor Company"
Party	"Party"
isAssembledFrom	"Is assembled from"

Q5a, S5a

x	y
Party	"A person or group involved in an enterprise; a participant or an accessory."
milleage	"Total miles covered or traveled in a given time."
isAssembledFrom	"To bring or call together into a group or whole"

Q7a, S7a

automobile	maker	insurance	production_date
Automobile2	Company1	Insurance2	"2011-01-03T00:00:00"
Automobile1	Company1	Insurance4	"2011-05-05T12:00:00"
Automobile1	Company1	Insurance3	"2011-05-05T12:00:00"
Automobile1	Company1	Insurance1	"2011-05-05T12:00:00"

Q12a, S12a

part1	part2
Wheel4	Frame1
Wheel3	Frame1
Wheel2	Frame1
Wheel1	Frame1
UniversalJoint	DriveShaft1
Transmission1	Chassis1
Frame1	Chassis1
DriveShaft2	Chassis2
DriveShaft1	Chassis1
Wheel4	Chassis1
Wheel3	Chassis1
Wheel2	Chassis1
Wheel1	Chassis1
UniversalJoint	Chassis1

Q8a, S8a

automobile	insurance	Insurance_company
Automobile1	Insurance4	Company2
Automobile1	Insurance3	Company2
Automobile2	Insurance2	Company2
Automobile1	Insurance1	Company2

Q9a, S9a

is_assembled_from
Wheel4
Wheel3
Wheel1
Motor1
Wheel2
Chasis1

Q10a, S10a

insurance	Is_verified_by
Insurance2	Company2
Insurance4	Company2
Insurance3	Company2
Insurance1	Company2

Q11a, S11a

person	is_partner_of
Assurer2	Assurer1
Assurer1	Assurer2

Q13a, S13a

person	colleague
Assurer2	Assurer1
Assurer1	Assurer2

Figure 8. Results of SPARQL and SQL queries for Vehicle ontology (prefixes are omitted for brevity)

The method is fully automatic and allows obtaining database schemas meaningful for developers of semantic applications of information systems. Our future research is related with two practical issues: 1) providing support for preparing existing ontologies for storing them in relational databases; 2) maintaining changes in database schema when ontology changes, as a part of the schema is domain-dependent and only metatables have a stable structure. Since changes of constraints are much faster than changes of domain concepts the method partially meets the requirement of schema stability but it is not able to avoid this problem on the whole.

References

[1] **A. Morkevičius, S. Gudas.** Enterprise Knowledge Based Software Requirements Elicitation. In: *Information Technology and Control*, 40(3), 2011, 181−190.

[2] **A. Sasa, O. Vasilecas**. Ontology based support for Complex Events. In: *Electronika ir Elektrotechnika*, 7, 2011, 83−88.

[3] **B. Motik, P.F. Patel-Schneider, B. Parsia**. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Proposed Recommendation 22 September 2009. Available from: http://www.w3.org/TR/2009/PR-owl2-syntax-20090922. [Accessed 10 Jan 2011].

[4] **C. Golbreich, E.K. Wallace, P.F. Patel-Schneider**. OWL 2 Web Ontology Language New Features and Rationale. W3C Proposed Recommendation, 2009, Available from: http://www.w3.org/TR/2009/PR-owl2-new-features-20090922 [Accessed 11 Jan 2011].

[5] **R. Agrawal, A. Somani, Y. Xu.** Storage and querying of e-commerce data. In: *Proceedings of the 27th International Conference on Very Large Databases (VLDB'01)*, Sep 11−14, 2001, Roma, Italy. Roma, Italy: Morgan Kaufmann Publishers, 2001, 149−158.

[6] **J. Zhou, L. Ma, Q. Liu, L. Zhang, Y. Yu, Y. Pan**. Minerva: A Scalable OWL Ontology Storage and Inference System. In: *The Semantic Web – ASWC 2006*, LNCS 4185, 2006, 429–443.

[7] **J. Lu, L. Ma, L. Zhang, J.S. Brunner, C. Wang, Y. Pan, Y. Yu**. SOR: a practical system for ontology storage, reasoning and search. In: *Proceedings of the 33rd International Conference on Very Large Data Bases,* Vienna, Austria, 2007, 1402–1405.

[8] **B. Motik, A. Maedche, R. Volz**. Ontology Representation and Querying for Realizing Semantics-Driven Applications. In: *G. Stamou, S. Kollias (Eds.): Multimedia Content and the Semantic Web: Methods, Standards and Tools*, John Wiley & Sons, Ltd, Chichester, UK, 2005, 45−73.

[9] **H. El-Ghalavini, M. Odeh, R. McClatchey, T. Solomonides.** Reverse Engineering Ontology to Conceptual Data Models. In: *M.H. Hamza (Ed.): IASTED International Conference on Databases and Applications, part of the 23rd Multi-Conference on*

Applied Informatics, Innsbruck, Austria, February 14-16, 2005. IASTED/ACTA Press, 2005, 222–227.

[10] E. Vyšniauskas, L. Nemuraitė, A. Šukys. A hybrid approach for relating OWL 2 ontologies and relational databases. In: *P. Forbrig, H. Gunther (Eds.): Perspektives in Business Informatics Research*. Proceedings of the 9th international conference, BIR 2010, Rostock, Germany, September 29 - October 1, 2010. Berlin-Heidelberg-New York, Springer, 2010, 86–101.

[11] M. Biehl. Literature Study on Model Transformations. Technical Report, Royal Institute of Technology, ISRN/KTH/MMK/R-10/07-SE, Stockholm, Sweden, July 2010.

[12] J.L. Hainaut, C. Tonneau, M. Joris, M. Chandelon. Transformation-based Database Reverse Engineering. In: *Proceedings of the 12th Internationa*l Conference on the Entity-Relationship Approach, LNCS 823, Springer-Verlag, 1993, pp. 364–375.

[13] B. Paradauskas, A. Laurikaitis. Extracting conceptual data specifications from legacy information systems. In: *Electronika ir Elektrotechnika*, 2011, 1(107), 46–50.

[14] R. Ghawi, N. Cullot. Database-to-Ontology Mapping Generation for Semantic Interoperability. In: *VDBL'07 conference, VLDB Endowment ACM*, 2007, 1–8.

[15] M.E. Saleh. Semantic-Based Query in Relational Database Using Ontology. In: *Canadian Journal on Data, Information and Knowledge Engineering*, 2(1), January 2011, 1−16.

[16] M. Fernandez, A. Gomez-Perez, N. Jurist. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. AAAI Technical Report SS-97-06, 1997, AAAI, 33−40.

[17] O. Corcho, M. Fernandez-Lopez, A. Gomez-Perez. Methodologies, tools and languages for building ontologies. Where is their meeting point? In: *Data & Knowledge Engineering*, 46, 2003, 41–64.

[18] A. De Nicola, M. Missikoff, R. Navigli. A software engineering approach to ontology building. In: *Information Systems*, 34(2), 2009, 258–275.

[19] T.R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: *International Journal Human-Computer Studies*, 43, 1993, 907–928.

[20] N. Guarino, C. Welty. Towards a methodology for ontology based model engineering. In: *ECOOP-2000 Workshop on Model Engineering*, Cannes, France, 2000.

[21] G. Guizzardi, G. Wagner, N. Guarino, M. Sinderen. An Ontologically Well-Founded Profile for UML Conceptual Models. In: *A. Person and J. Stirna (Eds.): CAISE 2004*, LNCS 3084, 2004, 112–126.

[22] A.L. Rector. Normalisation of ontology implementations: Towards modularity, re-use, and maintainability. In: *Proceedings Workshop on Ontologies for Multi-agent Systems (OMAS) in conjunction with European Knowledge Acquisition Workshop*, 2002, 1–16.

[23] A.L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In: *J. Genari (Ed.): Proceedings of the 2nd International Conference on Knowledge Capture K−CAP 03*, ACM, 2003, 121–128.

[24] A.L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, C. Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: *Engineering Knowledge in the Age of the Semantic Web*, LNCS 3257, 2004, 63–81.

[25] OMG. Information Management Metamodel (IMM) Specification. OMG document: ad/2011-05-06, 2011.

[26] D. Ouyang, X. Cui, Y. Ye. Mapping integrity constraint ontology to relational databases. In: *The Journal of China Universities of Posts and Telecommunications*, 17(6), December 2010, 113–121.

[27] J.J. Van Griethuysen. Concepts and Terminology for the Conceptual Schema and the Information Base. Publication Number ISO/TC97/SC5 – N 695, New York: ANSI, 1982.

[28] J. Martin, J. Odell. Object-Oriented Methods: A Foundation. Prentice-Hall, 2004.

[29] E. Vyšniauskas, L. Nemuraitė, R. Butleris, B. Paradauskas. Reversible Lossless Transformation from OWL 2 Ontologies into Relational Databases. In: *Information Technology and Control* 40(4), 2011, 293−306.

[30] E. Vyšniauskas, L. Nemuraitė, B. Paradauskas. Hybrid Method for Storing and Querying Ontologies in Databases. In: *Elektronika ir Elektrotechnika,* 9(115), 2011, 67−72.

[31] O. Vasilecas, D. Kalibatiene, G. Guizzardi. Towards a Formal Method for the Transformation of Ontology Axioms to Application Domain Rules. In: *Information Technology and Control* 38(4), 2009, 271−282.

[32] W3C. SPARQL query language for RDF. E. Prud'hommeaux, A. Seaborne (Eds.): W3C Recommendation 15 January 2008.

[33] W3C. SPARQL 1.1 Query Language. S. Harris, A. Seaborne (Eds.), W3C Working Draft 12 May 2011.

[34] A. Chebotko, S. Lu, F. Fotouhi. Semantics preserving SPARQL-to-SQL translation. In: *Data & Knowledge Engineering*, 68(10), 2009, 973−1000.

[35] W3C. OWL 2 Web Ontology Language Direct Semantics. I. Horrocs, B. Parsia, U. Sattler (Eds.). W3C Recommendation 27 October 2009.

[36] OMG, 2009. Ontology Definition Metamodel. Version 1.0. OMG Document Number: formal/2009-05-01, 2009.

[37] R. Wille. Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. In B. Ganter et al. (Eds.): Formal Concept Analysis, LNCS 3626, 2005, 47−70.

[38] F. Jiang, Y. Meng, Y. Liu. Formal Concept Analysis in Relational Contexts. In: *Proceedings of the IEEE International Conference of Granular Computing GRC 2008*, 2008, 326−329.

Received November 2011.