Optimization of Queries Based on Foundational Ontologies

Jana Ahmad(✉), Petr Křemen, and Martin Ledvinka

Czech Technical University, Prague, Czech Republic
{jana.ahmad,petr.kremen,martin.ledvinka}@fel.cvut.cz

Abstract. Using ontologies for enterprise data integration brings an opportunity to use them for efficient data query evaluation. The rationale is that ontologies represent common-sense structures that are reflected also in queries posed by data users and thus can be used for query optimization. This paper presents how proper foundational-ontology-based knowledge can be used to design a generic index, which can help in answering a wide range of queries compliant with the foundational ontology. We discuss several indexing techniques and evaluate our proposal for different UFO-based queries extracted from different query sets.

Keywords: Foundational ontologies · SPARQL · Indexing

1 Introduction

A conceptual model is a model of the world where we live in, created within a human mind. It can be understood as the result of an activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication [23]. The aim of a conceptual model is to express concepts' meaning which can be used to discuss problems in a specific domain and to find the correct relationships between different concepts. Therefore, the aspect of conceptual modeling has become widespread in the context of cognitive science [10,23]. Formal ontologies [10] produced by conceptual modeling activities represent shared common vocabularies of concepts together with formal relationships among them valid in the particular domain. As such, they are suitable abstractions for data access and integration.

In this paper, we will deal with RDF [17] data integration. Principles behind RDF have been adopted by technology leaders for collecting and accessing knowledge, like Facebook[1], Google[2], or Microsoft[3]. There are many RDF management and storage systems[4] e.g., Sesame and virtuoso. These systems are based on the

[1] http://ogp.me, accessed 2018-07-16.
[2] https://tinyurl.com/g-knowledge-graph, accessed 2018-07-16.
[3] https://tinyurl.com/ms-concept-graph, accessed 2018-07-16.
[4] E.g. http://rdf4j.org/, cit. 16.7.2018, or https://virtuoso.openlinksw.com/, cit. 16.7. 2018, to list some.

H. Panetto et al. (Eds.): OTM 2018 Conferences, LNCS 11230, pp. 351–367, 2018.
https://doi.org/10.1007/978-3-030-02671-4_22

RDF model, using generic RDF indexing techniques for improving query performance. However, generic RDF indices do not take into account the semantics of the data. Thus, ontological patterns which can be presented in data and reflected in queries are not optimized in any way.

From previous points, this paper optimizes RDF query evaluation by means of an index based on actual knowledge structure provided by the conceptual model-based ontology. Since we use a *foundational ontology* [10], we defend that such index is reusable for a wide range of queries, corresponding to *foundational query patterns*, i.e., query patterns expressed in a foundational ontology and compliant with many queries in the actual domain.

1.1 Contribution

The main contribution of this paper is foundational query patterns analysis and foundational-ontology-based index design to make query evaluation more efficient, tailored to RDF data, see Fig. 1.

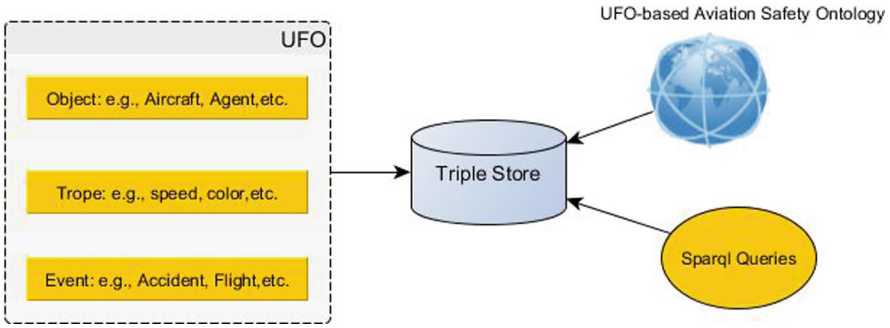

Fig. 1. UFO-based aviation safety domain queries architecture.

In particular, this paper proposes: (i) a new RDF indexing approach based on Unified Foundational Ontology (UFO) which is top-level ontology, aimed at answering queries compliant with common query patterns. (ii) Benchmark of queries for evaluation (applied in aviation safety domain ontology as a use case).

The paper is organized as follows. Section 2 reviews the necessary background of RDF and querying, in Sect. 2.2 we briefly define the notion of the Unified Foundational Ontology (UFO). Section 3 presents related work. UFO index design is presented in Sect. 4 including motivating scenario, query patterns, analyzing UFO query patterns in different data sources and UFO-based index tables. The evaluation of UFO indexing techniques and query results are given in Sect. 5, with the description of our use-case. Conclusion is in Sect. 6.

2 Background

First, we introduce a fragment of OWL 2-DL [1] in a simplified manner, as a knowledge-representation language together with simple conjunctive queries over this fragment. Next, we give an overview of UFO, as one of the foundational ontologies. Last, we show an example of RDF representation of an OWL-based UFO fragment and queries.

2.1 OWL 2-DL

An OWL 2-DL *ontology* $\mathcal{O} = \{\alpha_i\}, i \in \{1, \ldots, N_\mathcal{O}\}$, where α_i is an axiom is either

- a class assertion $\mathsf{A}(\mathsf{a})$, saying that "a is an instance of A", e.g. $\mathsf{Person}(\mathsf{Frank})$.
- a object property assertion $\mathsf{P}(\mathsf{a}, \mathsf{b})$, saying that "$\mathsf{a}$ is related to b through P", e.g. $\mathsf{hasFriend}(\mathsf{Frank}, \mathsf{John})$.
- a terminological axiom of the form $C_1 \sqsubseteq C_2$.

Where A is an OWL atomic class, $C_{(i)}$ are OWL class expressions (discussed later), a is an OWL individual and P is an OWL object property. Typical OWL class expressions could be constructed from atomic classes as follows

- each atomic class A is a class expression.
- boolean operators $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$, or $(\neg C_2)$, for class intersection/union and complement. For example, $(\mathsf{Person} \sqcap \mathsf{Male})$ denotes the concept of men.
- existential restriction $(\exists \mathsf{P} \cdot C)$, denoting a class, elements of which are related through P to at least one instance of C. For example, $(\exists \mathsf{hasChild} \cdot \mathsf{Man})$ denotes a class of all individuals having at least one son.
- universal restriction $(\forall \mathsf{P} \cdot C)$, denoting a class, elements of which are related through P only to instances of C. For example, $(\forall \mathsf{hasChild} \cdot \mathsf{Man})$ denotes a class of all individuals having only sons as children, if any,
- qualified cardinality restrictions $(\leq n\, \mathsf{P} \cdot C)$, or $(\geq n\, \mathsf{P} \cdot C)$, $(= n\, \mathsf{P} \cdot C)$, denoting a class, elements of which are related through P to at least/at most/exactly n individuals through P. For example, $(\geq 4\, \mathsf{hasChild} \cdot \mathsf{Man})$ denotes a class, elements of which have at least four sons (and possibly some other daughters).

We omitted parts of OWL 2-DL that we do not use in this paper, namely inverse properties, nominals, role chains and data types. Full OWL 2-DL syntax as well as its formal semantics can be found in [1].

Having an OWL 2-DL ontology $\mathcal{O}$, we define a *distinguished conjunctive query* as $Q(?x_1, \ldots, ?x_n) = \mu_1, \ldots, \mu_M$, where $?x_i$ is a variable occurring in some μ_i, μ_i is an atom of the form $\mathsf{A}(y)$ or $\mathsf{P}(y, z)$, where A is an atomic OWL class, P is an OWL object property and y, resp. z is either a variable $?x_i$, or an OWL individual. Intuitively, queries match the class/property assertion axioms, possibly extended by inferencing from other axioms. Let's show the notions on an example. Full query syntax and semantics of distinguished conjunctive queries can be found in [21].

Example. Having an OWL 2-DL ontology $\mathcal{O} = \{\mathsf{Agent} \sqsubseteq \mathsf{Object},\ \mathsf{Agent(a)},\ \mathsf{performs(a, b)}\}$, the query $Q(?x_1, ?x_2) = \mathsf{Object}(?x_1), \mathsf{performs}(?x_1, ?x_2)$ asks for all object and actions they perform. In our case, the query returns a single result binding $\{(?x_1, ?x_2) \rightarrow (\mathsf{a}, \mathsf{b})\}$, because a is inferred to be an object ($\mathsf{Agent} \sqsubseteq \mathsf{Object}$).

2.2 Unified Foundational Ontology

Ontologies are used to specify the meaning of terms of some domain, categorizing it and setting relationships among these terms. They define basic concepts like objects, relations, events, processes, situations, and other high-level domain-independent categories. A *top-level ontology* is an ontology that describes general concepts that are common to multiple domains. There are many well-known foundational ontologies, like (DOLCE) [23], Unified Foundation Ontology (UFO) [10], and Basic Formal Ontology (BFO) [7].

For the purpose of this study, we have chosen to build our proposal on top of UFO. Because of (i) our experience in using UFO in various conceptual model-based domains [20], (ii) UFO addresses many essential aspects for conceptual modeling, which have not received a sufficiently detailed attention in other foundational ontologies [10], (iii) the availability of its formal translation to OWL [4], (iv) availability of OntoUML, an ontology modeling language based on UFO [10].

UFO is a top-level ontology. Its main concepts fundamental for this study are sketched in the UML class diagram[5] in Fig. 2. UFO describes static objects (UFO-A) [10], events (UFO-B) [13] and social agents (UFO-C) [12]. UFO splits entities into endurants and perdurants ($\mathsf{Endurant} \sqsubseteq \mathsf{Individual}$) ($\mathsf{Perdurant} \sqsubseteq \mathsf{Individual}$)[6]. Endurants can be observed as complete concepts in a given time snapshot. Endurants can be any object (an agent, an aircraft) ($\mathsf{Object} \sqsubseteq \mathsf{Endurant}$), or its tropes (e.g. speed, location) ($\mathsf{Trope} \sqsubseteq \mathsf{Endurant}$), that exist as long as an object they inhere in exists ($\mathsf{Trope} \sqsubseteq (= 1\ \mathsf{inheresIn} \cdot \mathsf{Object})$).

Perdurants only partially exist in a given time snapshot. They involve events ($\mathsf{Event} \sqsubseteq \mathsf{Perdurant}$), situations ($\mathsf{Situation} \sqsubseteq \mathsf{Perdurant}$) and object snapshots ($\mathsf{ObjectSnapshot} \sqsubseteq \mathsf{Perdurant}$).

Events can be either atomic or complex ($\mathsf{Event} \sqsubseteq (\mathsf{AtomicEvent} \sqcup \mathsf{ComplexEvent})$). Events occur in time, have participants ($\mathsf{Event} \sqsubseteq (\geq 1\ \mathsf{hasParticipant} \cdot \mathsf{Object})$). complex events have parts ($\exists\ \mathsf{hasEventPart} \cdot \top \sqsubseteq \mathsf{ComplexEvent}$). Events themselves can be temporally related, i.e. one happening before another, during another, etc. [13]. ObjectSnapshot: is an immutable state description of an object within a situation. Situation: is a snapshot of object states valid in the given temporal range.

[5] The figure is a straightforward visualization of a fragment of UFO axiomatization in OWL, e.g. $\mathsf{Trope} \sqsubseteq \forall \mathsf{inheresIn} \cdot \mathsf{Individual} \sqcap \exists \mathsf{inheresIn} \cdot \mathsf{Individual}$ denotes the edge from Trope to Individual. See [4] for details.

[6] We reuse Description Logic formalization of basic UFO concepts introduced in [4].

Additionally, UFO introduces the notion of agents ($\mathsf{Agent} \sqsubseteq \mathsf{Object}$), i.e., proactive objects with an intention (e.g. a person, or a company), their intentions, commitments and actions they perform ($\exists\ \mathsf{performs} \cdot \top \sqsubseteq \mathsf{Object}$), and services [20,22].

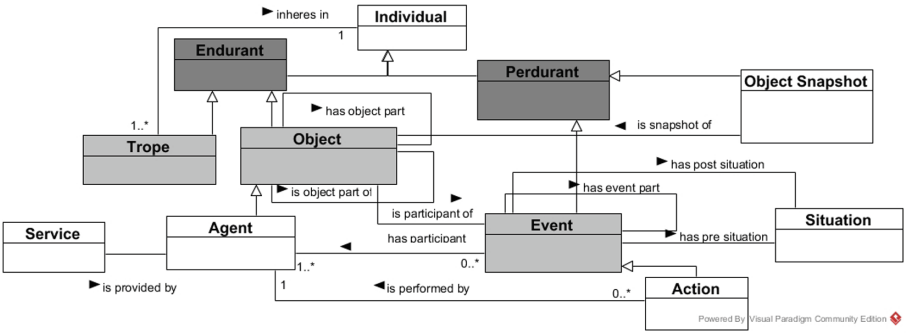

Fig. 2. Main concepts of UFO

2.3 RDF Representation of UFO Models

To use a wide-spread technology for UFO index representation, we will consider RDF triple stores. Indexing techniques over RDF are discussed in Sect. 3. At this point, we show how to represent common OWL axioms, representing an OWL ontology, in RDF and a distinguished conjunctive query over the ontology as basic graph patterns of SPARQL [15], a query language for RDF.

Consider a *triple pattern*, an ordered tuple $t^v = (s^v, p^v, o^v)$, where s^v is its *subject term*, p^v is its *predicate term* and o^v is its *object term*. Each subject is either a *variable* V, or a *constant*[7] C.

Having an OWL ontology $\mathcal{O}$, its RDF serialization is given directly by OWL specification [1]. For distinguished conjunctive queries, we translate each atom of the form $\mathsf{A}(y)$ into an RDF triple pattern (y rdf:type A)[8] and each atom of the form $\mathsf{P}(y, z)$ into an RDF triple pattern (y P z). Note that all constants (A, P) and possibly (y, z) are represented by the corresponding IRIs.

Example. Having an OWL 2-DL ontology $\mathcal{O}$ from example in Sect. 2.1, its RDF serialization would be[9]

[7] For the purpose of this paper, we consider constant to be URIs only.

[8] rdf:type is a special predicate of RDF denoting instantiation.

[9] We use the prefix ":" to denote the namespace <http://example.org/>, thus a translation of Agent into its RDF representation would be an IRI <http://example.org/Agent>.

```
:Agent rdfs:subClassOf :Object.
:a :performs :b.
```

and the SPARQL representation of the query Q would be a SPARQL basic graph pattern

```
?x1 rdf:type :Object.
?x1 :performs ?x2.
```

3 Related Work

In this section, we discuss two research directions relevant to our work. RDF Indexing Techniques that explains the basic types of the physical design for each triple store. And Ontology-based Query-answering systems that query triple stores to retrieve data.

3.1 RDF Indexing Approaches

There has been a significant amount of research done on developing systems for storing and managing RDF data. Triple stores use different physical organization techniques to store RDF data, such as a triple table, property table, vertical partitioning approaches [2,6,14,18].

In the *Triple Table* (TT) approach, each RDF statement of the form (s p o) is stored as a triple in one table with three columns. Indexes are then added for each of the columns to make joins less expensive. Since all triples are stored in a single RDF table, very large data and complex queries cause the execution time to be prohibitive [2,6]. An example of a triple store that uses this approach is 3store [14].

When using the *Property Table* (PT) technique, each table contains a column for the subject of the triple and separate columns for several fixed properties. The main idea is to discover clusters of subjects often appearing with the same set of properties. However, the disadvantage of the property table technique is that it generates many NULL values since, for the given cluster, not all properties will match all subjects and multi-valued attributes. Another disadvantage of this approach is that adding properties requires also to add a new table [2,6]. This approach has been used in Jena 2 [25].

Using the *Vertical Partitioning* (VP) approach, each triple table includes n-two column tables where n is the number of unique properties in the data. In each of these tables, the first column contains the subjects that match property and the second column contains the object values for those subjects [2,3,14]. In this indexing technique, each table is sorted by subject, thus particular subjects can be located quickly. And this approach supports multi-valued attributes.

Compressed Indexes use six B+ tree indices to store RDF triples tables or quads of a subject, predicate, object, and a "context". In each B+ tree, the key is a concatenation of the subject, predicate, object, and context [18].

Relation to Our Approach. In our approach, we build on these existing techniques, extend them and use them together with UFO patterns to improve the efficiency of queries based on foundational ontologies (UFO-based query answering).

3.2 Ontology-Based Query-Answering Systems

Next research direction to mention are ontological query answering systems that use triple stores to query data.

Ontology-Based Data Access (ODA) [19] is a paradigm for accessing data through a conceptual layer. In the OBDA, an ontology defines a high-level global schema of data sources and provides a vocabulary for user queries. End-users formulate queries using the ontological terms and thus they are not required to understand the structure of the data sources, trading it for limitations in expressiveness [16].

In Ontoseek [9], queries are represented as conceptual graphs, then the query answering problem is reduced to ontology-driven graph matching where individual nodes and arcs match if one subsumes the other.

AQUA [24] is a question answering system which integrates NLP, logic, information retrieval and ontologies. It transfers the English question into logical queries, expressed in QLL language.

Ontology-mediated query answering (OMQA) [5] is a paradigm for accessing legacy data while taking into account knowledge expressed by a domain ontology.

Relation to Our Approach. Although each of the related works above presents a unique solution for data storage and query answering, none of them proposes a foundational Ontology as a solution for query processing. Thus, this paper discusses, how proper foundational-ontology-based index design can help to optimize query answering.

4 UFO Index

The meanings of the variety of words such: red, John, Jana, marriage, accident, ball, process, attend, happen, party, hot, warm, play, situation, tasks, etc. reflects the essential differences between things that happen and who performs these things, i.e., the distinction between behavioral elements and structural elements. UFO distinguishes between these two categories with the behavioral elements referred to as "events" and the structural referred to as "objects". The question words *("how")* versus *("what")* is often invoked to check the different nature of these elements [11].

For more comprehensive representation of any ontological domain, it is important to focus on the representation of endurants (e.g., objects, their parts, their properties, etc.) and perdurant (e.g., events, their parts, etc.), and that is exactly what UFO considers. Based on them, we constructed eight competency questions [8] shown in Table 1. The questions show foundational patterns that

are used in real queries. Let's discuss the questions in more details and show their realistic exemplifications in two domains: the aviation safety domain and the social relation domain.

Table 1. Foundational query patterns and their DCQ representation.

	Question	DCQ formalization
Q_1	Who participates in an event?	$Q_1(?o, ?e) \rightarrow \mathsf{ufo{:}has\text{-}participant}(?e, ?o)$
Q_2	Which objects participate in a specific event e1 ?	$Q_2(?o) \rightarrow \mathsf{ufo{:}has\text{-}participant}(\mathsf{e1}, ?o)$
Q_3	What are the properties of (inhere in) a certain object o1?	$Q_3(?t) \rightarrow \mathsf{ufo{:}inheres\text{-}in}(?t, \mathsf{o1})$
Q_4	what are object's parts ?	$Q_4(?o1, ?o2) \rightarrow \mathsf{ufo{:}has\text{-}object\text{-}part}(?o1, ?o2)$
Q_5	What are the snapshots of a specific object?	$Q_5(?s) \rightarrow \mathsf{ufo{:}is\text{-}snapshot\text{-}of}(?s, \mathsf{o1})$
Q_6	What caused an event?	$Q_6(?e1, ?e2) \rightarrow \mathsf{ufo{:}is\text{-}caused\text{-}by}(?e1, ?e2)$
Q_7	How is a particular process structured?	$Q_7(?e) \rightarrow \mathsf{ufo{:}has\text{-}event\text{-}part}(\mathsf{e1}, ?e)$
Q_8	What are snapshots of any object?	$Q_8(?s, ?o) \rightarrow \mathsf{ufo{:}is\text{-}snapshot\text{-}of}(?s, ?o)$

Q_1 and Q_2 show the relation between an event and an object. Events are mapping of statements or occurrence in the reality, in which objects (things and people) participate playing certain tasks ($(\mathsf{Event} \sqsubseteq (\geq 1\ \mathsf{hasParticipant} \cdot \mathsf{Object})$). E.g., who participates in Jana's birthday party?

Q_3 presents inherence relation between an object and its qualities that are existentially dependent on the object itself ($\mathsf{Trope} \sqsubseteq (= 1\ \mathsf{inheresIn} \cdot \mathsf{Object})$) [10]. I.e., a trope is a property of some object or another trope that cannot exist without the object it inheres in. E.g., what are the tropes or properties vehicle-i has?

In UFO, endurants are entities that, whenever they exist, they exist with all their parts, while maintaining their identity ($\mathsf{Object} \sqsubseteq (= 1\ \mathsf{isObjectPartOf} \cdot \mathsf{Object})$). This is what Q_4 presents. E.g., what are the parts of aircraft-i?

Q_5 and Q_8 present the notion of an object snapshot which is an immutable state description of an object within a situation. Regarding to UFO, object snapshot and object share the attribute types and relation types of their identifying objects ($\mathsf{Object\ Snapshot} \sqsubseteq (=1\ \mathsf{isSnapshotOf} \cdot \mathsf{Object})$). Where the object serves only for identification purpose. E.g., What are all revisions of reports of specific occurrence? And What is the last snapshot of a specific aircraft just before take-off?

What are the changes from and to a certain situation in the world? An answer is an event, which is the answer of Q_6. E.g., How the accident-i happened? UFO defines full meaning of events, describing how events relate to their parts ($\mathsf{ComplexEvent} \sqsubseteq (\geq 2\ \mathsf{hasEventPart} \cdot \mathsf{Event})$. Moreover, it prescribes all temporal precedence involving events. This notion could translate to Q_7. E.g., what happened before the accident-i?

The previous questions motivate us to create UFO-based physical design index tables that store RDF data according to the main concepts of UFO, Perdurant and Endurant.

Use of the Questions in Enterprise Information Systems. The above-defined questions represent generic foundational query patterns that are inherent to all information systems. The advent of harmonization in enterprise data management gave rise to unified data models.

For example, *customer master data*[10] model customers (endurants), their properties like the name (trope). Particular writes into the master data base corresponding to the individual snapshots of the name, resp. customer. Thus, by instantiating Q_3, we can easily retrieve all properties a master data base contains about a customer.

Another example might be an information system for customer care[11]. It gathers activities (events), as well as their responsible organizations (event participants). Thus, by instantiating Q_1, we can get all organizations which are responsible for performing some activity.

Analyze UFO Queries in Different Data Sources. The question now is: how can we represent UFO query patterns in different data sources, e.g., Wikidata? How could different data source queries be analyzed with UFO? Or how data source queries could be translated to UFO?

For this purpose, in this section, we discuss the representation of UFO-based queries and how $Q_1 \ldots Q_8$ competence questions match different queries in two different data sources Wikidata[12] and Reporting Tool[13]. Reporting Tool is an ontology-based information system allowing to manage safety data in the aviation industry. Most of the queries in the Reporting Tool are centered around events, especially safety events, that can match Q_1, Q_6 and Q_7 of competence questions. Or around the snapshot of objects in specific time which match Q_5 and Q_8. *For example*: what is the last revision of specific occurrence reports? According to UFO we can map this query to Q_5 in competence questions (what is the last snapshot of a specific object), where the last version could be represented as the snapshot of occurrence report.

Wikidata which is a collaboratively edited knowledge base under development by the Wikimedia Foundation. There are different sets of queries: (i) simple questions: about objects, properties, and events which match $Q_1..Q_8$ (Such as Humans without children (Q_1, Q_2, Q_4), Recent Events (Q_5, Q_6), Popular eye colors (Q_3), etc.). (ii) Showcase Queries: that are centered around objects and their properties or tropes(such as: Whose birthday is today? that match Q_1) (iii)

[10] https://wiki.scn.sap.com/wiki/display/SD/Customer+master+data, cit. 16.07.2018.

[11] E.g. http://www.adrm.com/ba-customer-service.shtml, cit. 16.07.2018.

[12] https://www.wikidata.org/wiki/Wikidata:SPARQL-query-service/queries/examples Showcase-Queries, cit. 16-07.2018.

[13] https://github.com/kbss-cvut/reporting-tool, cit. 16.07.2018.

Samples with coordinates to illustrate maps queries: about objects and trope (such as Locations of universities in Germany that match Q_3) and others. So, we can noticed that the most queries of Wikidata about endurant (object and trope). Let's analyze a complete query: e.g., select people whose gender we know or we don't know?

With respect to UFO, Human is an Object that has tropes that inhere in it, which is here (gender) therefore, this query match Q_3 ($\mathsf{Trope} \sqsubseteq (= 1\ \mathsf{inheresIn} \cdot \mathsf{Object})$) of competence questions.

4.1 UFO Index Tables Approaches Discussion

Due to the nature of RDF that represents data as triples (subject, predicate, object), the goal is to handle variant types of queries such as: triples having the same property, triples having the same subject and finally, list of subjects or properties related to a given object. Thus, there is a real need of efficient tools for storing and querying knowledge using the ontologies and the related resources.

In this section, we discuss technical challenges for the indexing and query processing regarding to our novel approach in designing index tables for top-level ontology concepts (namely UFO). We discussed the indexing techniques without UFO in Sect. 2.

Triple Tables. As we mentioned before UFO is divided mainly into two categories, Perdurant (event) and Endurant (object), $((\mathsf{Event} \sqcup \mathsf{Endurant}) \sqsubseteq \top)$. Applying triple table technique in UFO will store all instances of these two categories in one table which will lead not only to slow in query execution but also the insufficient use of the UFO conceptualization (see Table 2). Therefore, our proposal in this approach is to divide Table 2 physically into two tables; one for each category (Perdurant and Endurant). See Tables 3 and 4.

Table 2. Triples table

Subject	Predicate	Object
Event-i	has-participant	Agent-i
Person-i	is-participant-of	Event-i
Agent-i	performs	Action-i
Process-i	is-event-part-of	Event-i
Action-i	is-performed-by	Agent-i
E_n	P_n	O_n

Based on UFO-tables, we answer Q2: what objects participate in a specific event (e.g., Action-i?) $((\mathsf{Action} \sqsubseteq \mathsf{Event}) \rightarrow (\mathsf{Action} \sqsubseteq (= 1\ \mathsf{hasParticipant} \cdot \mathsf{Object}))$. Instead of searching in the whole triple table, the query will be executed

Table 3. Perdurant table

Subject	Predicate	Object
Event-i	has-participant	Agent-i
Process-i	is-event-part-of	Event-i
Action-i	is-performed-by	Agent-i

Table 4. Endurant table

Subject	Predicate	Object
Person-i	is-participant-of	Event-i
Agent-i	performs	Action-i

in the endurant table. And this will be more efficient because we use the conceptualization of UFO, and faster because it searches only in endurant table (logically it needs half of TT time).

Property Table. UFO describes the relationship between two categories: behavioral elements (events) and structural elements (objects), by specifying conceptual properties that help to distinguish between event and object. For example, each event has an object participant who participates and performs a specific task in a particular event (e.g., What objects participate in a specific event? (Q_2)). Therefore, our proposal is also to build a UFO property table for endurant and another table for event and snapshot perdurant (see Tables 5 and 6), that will reduce Null values in each table, but we will still have them.

Table 5. Endurant property table

Subject	is-object-part-of	is-participant	Performs	has-trope
Object-i	Object-ii	Event-i	Event-i	Trope-i

Vertical Partitioning. Applying vertical partitioning (see Sect. 3) as UFO-indexing technique would gain the benefit from UFO predicates (i.e., Object Properties). For example: what are objects that are parts of another object (Q_4)? To answer this question, the query will be constructed such that: select all objects with property (has-object-part) of another object ($\mathsf{Object} \sqsubseteq (=1\ \mathsf{isObjectPartOf} \cdot \mathsf{Object})$) see Table 7.

5 Evaluation

For evaluation, we constructed a benchmark of SPARQL queries on 30000 triples. Then analyzed these queries to evaluate our proposal when using one or more of

Table 6. Perdurant property table

Subject	is-event-part-of	has-participant	is-performed-by	is-snapshot-of
Event-i	Event-i	Person-i	Agent-i	
Object-Snapshot				Object-i

Table 7. Has-object-part predicate table

Object-i	Object-ii
Object-i	Object-iii

indexing techniques that any triple store uses for storing UFO-based ontological data. We use the aviation safety ontology as a use case.

5.1 Aviation Safety Ontology

We built our Aviation Safety Ontology on top of the Unified Foundational Ontology (UFO) [10,23]. We designed Aviation Safety Ontology for describing safety issues in aviation organizations, and to increase the awareness of analytical methods and tools in the aviation community for safety analysis in the aviation domain [20]. Our strategy is to analyze safety events that lead to incidents or accidents and explain factors, that contribute to these safety events. Thus, Aviation Safety Ontology consists of the common aviation domain concepts, such as objects (e.g., aircraft, crew, aerodrome) and events (e.g. flight, accident) [20]. Figure 3 depicts basic concepts in Aviation Safety Ontology that are represented in UFO.

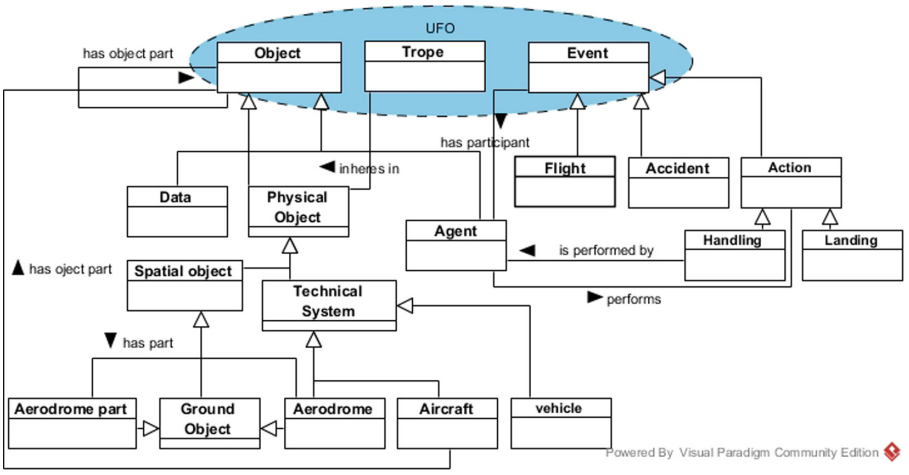

Fig. 3. Aviation safety ontology

5.2 Benchmark of Queries

To optimize the evaluation of UFO-based queries, consider a domain ontology (Aviation Safety Domain), we instantiated the queries $Q_1..Q_8$ in the aviation domain:

- Q_1': What objects participate in specific flight (flight-i)?
- Q_2': What are aircraft-i's parts?
- Q_3': What events are performed by Cabin-crew?
- Q_4': What are the tropes of a person in aviation safety domain?
- Q_5': What are the events that contribute in a specific flight (flight-i)?

These questions should be analyzed to evaluate the expressiveness of the ontology that is required to represent the competency questions and to characterize their solutions. Users in Aviation Safety Domain are interested in having answers to domain-specific questions like: knowing who is responsible for an aviation safety event. Who and what participate in aviation safety events (e.g. flight, process). And what are the properties (tropes) of the domain?

Thus, we transferred the previous constructed competence questions to formal representation (SPARQL queries[14]) with their conceptual models that may help to defend our suggestion of using UFO, then we discussed the answers of these queries in the UFO indexing techniques proposal. Finally, we tested one of the native triple stores (RDF4J[15]) that physically designed bases on B-Tree indexing triple tables with context, by comparing the execution time of these queries with and without UFO index tables. We use RDF4J context mechanism [15] to group all perdurant statements together through a single group identifier (Named Graph), i.e., in one perdurant table that contains around 15000 UFO-aviation safety data triples. And all endurant triples in another endurant table also around 15000 triples. This grouping process is executed automatically, by iterating all perdurants (events) from RDF and adding them to the perdurant table or objects to the endurant table.

Foundational Query Patterns Evaluation. To answer Q_1': Which is instantiated from Q_2 of competence questions, as shown in Fig. 4:

```
SELECT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/ufo/perdurant>
WHERE {GRAPH ?g
{ aviation-safety:flight-i ufo:has_participant ?term} }
```

[14] Common SPARQL prefixes include rdfs: to denote http://www.w3.org/2000/01/rdf-schema, rdf: to denote http://www.w3.org/1999/02/22-rdf-syntax-ns, ufo: to denote http://onto.fel.cvut.cz/ontologies/ufo/ and aviation-safety: to denote http://onto.fel.cvut.cz/ontologies/aviation-safety/.

[15] E.g. http://rdf4j.org/, cit. 16.7.2018.

– In *UFO Triple Table*: each event has objects as participants (Event $\sqsubseteq$ (= 1 hasParticipant · Object)), thus the query will be executed on perdurant table. In *Property Table*: the system will search for (has-participant) property in perdurant property table. By using UFO-based *Vertical Partitioning*: each table is built regarding to unique UFO-predicate, so that, the search process will look for has-participant UFO-predicate.

To evaluate Q_2' that is instantiated from Q_4, as shown in Fig. 4:

```
SELECT ?term
FROM NAMED <http://onto.fel.cvut.cz/ontologies/ufo/endurant>
where{ GRAPH ?g
{ ?term ufo:is_Object_part_of aviation-safety:aircraft-i.}}
```

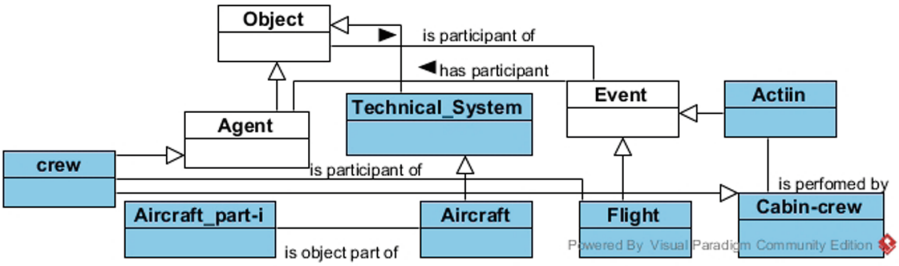

Fig. 4. Conceptual model for Q_1', Q_2' and Q_3'

– In *Triple Table*: this query looks for objects that are part of Aircraft-i which is also an object. Therefore, Q'_2 will be executed on the endurant table. In *UFO-Property Table*: Aircraft-i has many parts, so the UFO-based system will search for (is-object-part-of) property in endurant property tables. In *Vertical Partitioning* Proposal: the search process will look for (is-object-part-of) UFO-predicate.

Q_3' that instantiated from Q_1 (where is-performed-by is sub-property of is-participant-of), this query selects in perdurant table all events or actions that are performed by Cabin-crew (Object), see Fig. 4. (Event $\sqsubseteq$ (= 1 isPerformedBy·Object)). Its SPARQL transcription would be

```
SELECT ?term
FROM NAMED <http://onto.fel.cvut.cz/ontologies/ufo/perdurant>
where { GRAPH ?g
{ ?term ufo:is_performed_by aviation-safety:Cabin-crew-i}}
```

– The same discussion for Q'_4 that is instantiated from Q_3 and Q'_5 is instantiated from Q_7.

Evaluation Results. So, based on our proposal the query will be executed either on the perdurant table (named graph) or on the endurant table. The average execution time results and standard deviation are in Table 8, where the given results are averages from executing each query ten times in RDF4J store. The experiments were performed on a Dell computer with Intel(R) core(TM)i7-2600 CPU (3.40 GHz). The machine was equipped with 16.0 GB installed memory and HDD 1000 GB, running on Windows 7 professional 64-bit. Java version 8 update 66. Storage RDF4J version 2.1.1.

Table 8. Execution query time, σ denotes standard deviation, ϕ denotes mean value, MD denotes difference of the means.

	ϕ UFO TT	ϕ without-UFO	σ UFO TT	σ without-UFO	MD
Q_1'	134.4 ms	152.7 ms	4.9 ms	7.7 ms	18.3 ms
Q_2'	130.7 ms	138.8 ms	7.2 ms	6.2 ms	8.1 ms
Q_3'	132.3 ms	184.5 ms	7.9 ms	17.7 ms	52.2 ms
Q_4'	128.2 ms	142 ms	5.6 ms	9.3 ms	13.8 ms
Q_5'	125.1 ms	141.2 ms	3.9 ms	5.8 ms	16.1 ms

In our experiment, we have shown a performance increase on a relatively small data sample for all foundational queries. The table shows that

– having a foundational ontology, we can design a few index tables. The index tables can be reused for all datasets compliant with the foundational ontology and bring performance boost of evaluation of many queries (represented by foundational query patterns Q_1 to Q_5).

Pragmatically, the performance boost can be easily achieved for existing RDF stores by proper conceptual analysis and modeling of the datasets and queries by means of a foundational ontology.

6 Conclusion and Future Work

In this paper, we presented our novel approach to improve the efficiency of SPARQL queries by using UFO-based indexing techniques, discussed the different techniques using UFO-based index tables, and evaluated benchmark of SPARQL queries regarding to our proposal to explain how UFO-indexing approaches make the search process easier. In the next step, we will evaluate our idea practically in different ontological domains, different triple stores and large data sets with respect to UFO indexing techniques. A limitation of our approach is that it might not work perfectly for noisy data.

Acknowledgments. This work was supported by grant No. GA 16-09713S Efficient Exploration of Linked Data Cloud of the Grant Agency of the Czech Republic and by grant No. SGS16/229/OHK3/3T/13 Supporting ontological data quality in information systems of the Czech Technical University in Prague.

References

1. OWL 2 web ontology language document overview. Technical report, W3C Consoritum (2012). http://www.w3.org/TR/owl2-overview. Accessed 30 Aug 2018
2. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 411–422. VLDB Endowment (2007)
3. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: SW-Store: a vertically partitioned DBMS for semantic web data management. VLDB J. **18**(2), 385–406 (2009)
4. Benevides, A.B., Bourguet, J., Guizzardi, G., Peñaloza, R.: Representing the UFO-B foundational ontology of events in SROIQ. In: JOWO. CEUR Workshop Proceedings, vol. 2050. CEUR-WS.org (2017)
5. Bienvenu, M., Bourhis, P., Mugnier, M.L., Tison, S., Ulliana, F.: Ontology-mediated query answering for key-value stores. In: IJCAI International Joint Conference on Artificial Intelligence, pp. 844–851 (2017)
6. Faye, D.C., Cure, O., Blin, G.: A survey of RDF storage approaches. ARIMA J. **15**, 11–35 (2012)
7. Grenon, P., Smith, B.: SNAP and SPAN: towards dynamic spatial ontology. Spat. Cogn. Comput. **4**(1), 69–104 (2004)
8. Grüninger, M., Fox, M.: Methodology for the design and evaluation of ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI 1995 (1995)
9. Guarino, N., Masolo, C., Vetere, G.: OntoSeek: content-based access to the web. IEEE Intell. Syst. Appl. **14**(3), 70–80 (1999)
10. Guizzardi, G.: Ontological foundations for structural conceptual model. Ph.D. thesis (2005)
11. Guizzardi, G., Guarino, N., Almeida, J.P.A.: Ontological considerations about the representation of events and endurants in business models. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 20–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_2
12. Guizzardi, G., Wagner, G.: Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology: Computer Applications, pp. 175–196. Springer, Dordrecht (2010). https://doi.org/10.1007/978-90-481-8847-5_8
13. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 327–341. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41924-9_27
14. Harris, S., Shadbolt, N.: SPARQL query processing with conventional relational database systems. In: Dean, M., et al. (eds.) WISE 2005. LNCS, vol. 3807, pp. 235–244. Springer, Heidelberg (2005). https://doi.org/10.1007/11581116_25
15. Harris, S., Seaborne, A.: SPARQL 1.1 query language. Technical report, W3C Consoritum (2013)
16. Kharlamov, E., et al.: Optique: towards OBDA systems for industry. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 125–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41242-4_11

17. Klyne, G., Carroll, J.J.: Resource description framework (RDF): concepts and abstract syntax. Technical report, W3C Consoritum (2004)
18. Kolas, D., Emmons, I., Dean, M.: Efficient linked-list RDF indexing in parliament. In: CEUR Workshop Proceedings, vol. 517, pp. 17–32 (2009)
19. Kontchakov, R., Rodríguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: a short course. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Web 2013. LNCS, vol. 8067, pp. 194–229. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39784-4_5
20. Kostov, B., Ahmad, J., Křemen, P.: Towards ontology-based safety information management in the aviation industry. In: Ciuciu, I., et al. (eds.) OTM 2016. LNCS, vol. 10034, pp. 242–251. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55961-2_25
21. Křemen, P., Kouba, Z.: Conjunctive query optimization in OWL2-DL. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011. LNCS, vol. 6861, pp. 188–202. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23091-2_18
22. Křemen, P., et al.: Ontological foundations of European coordination centre for accident and incident reporting systems. J. Aerosp. Inf. Sys. **14**(5), 279–292 (2017)
23. Mylopoulos, J.: Conceptual modelling and Telos. In: Conceptual Modeling, Databases, and Case An integrated view of information systems development, pp. 49–68 (1992)
24. Vargas-Vera, M., Motta, E.: AQUA – ontology-based question answering system. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 468–477. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24694-7_48
25. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient RDF storage and retrieval in Jena2. In: Proceedings of the 1th International Workshop on Semantic Web and Databases, pp. 35–43 (2003)