Foundational Patterns Benchmark

Jana Ahmad [a,*] and Petr Křemen [a]

[a] *Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic*
E-mails: jana.ahmad@fel.cvut.cz, Petr.Kremen@fel.cvut.cz

Abstract. Recently, there has been a growing interest in using ontology as a fundamental methodology to represent domain-specific conceptual models in order to improve the semantics, accuracy and relevancy of the domain user query results. However, the volume of data has grown steadily over the past decade. Therefore, managing, answering user's queries and retrieving data from multiple data sources could be a significant challenge for any enterprise. Thus, in this paper, we describe the foundational queries benchmark using the unified foundational ontology (UFO) and discuss how foundational queries help in optimizing the query answering results. For evaluation, we tested the foundational benchmark in different data sets – generated and real world – and on different triple stores.

Keywords: Foundational Pattern, UFO, SPARQL

1. Introduction

Recently, there has been a growing interest in using Ontology as a fundamental methodology to represent domain-specific conceptual models in order to improve the semantics, accuracy and relevancy of the domain users queries results. However, the volume of data has grown steadily over the past decade; therefore, managing, answering user's queries and retrieving data from multiple data sources could be a significant challenge for any enterprise. Therefore, for describing real-world phenomena and retrieving user queries in computer science, the aspect of conceptual modeling has became widespread in the context of cognitive science [23, 56]. Conceptual modeling is defined as the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication [56]. The descriptions that arise from conceptual modeling activities are intended to be used by humans, not machines. While the aim of a conceptual model is to express the meaning of terms and concepts used in a specific domain to discuss the problem and to find the correct relationships among different concepts, the simple conceptual schema that is represented by ontologies that are specified in formalized knowledge representation languages such as the Web Ontology Language (OWL 2) [67] causes modeling problems which hinder interoperability and lead to wrong relations so that cause not relevant answers. For example, common concepts and relationships such as events, objects, features and roles are modeled in each domain ontology differently and/or. Foundational ontologies aim to tackle this problem by extending the basic conceptual schema (i.e. the conceptual model that doesn't have foundational concepts that we can use with any domain) with additional constructs. Previous points and problems motivate us to defend how using foundational patterns in semantic systems and tools lead to more relevant and efficient query answering results for domain users.

The goal of this work is to optimize ontological queries by using commonsense knowledge, i.e., foundational Ontology which can analyze their query requests and intentions by semantics. For this work, we selected Unified Foundational Ontology (UFO) among other foundational ontologies because of, (i) our experience with using UFO in various conceptual model-based domains [47, 50], (ii) UFO is addressing many essential aspects of conceptual modeling, which have not received sufficiently detailed attention in other foundational ontologies [23], (iii) the availability of its formal translation to OWL [67] and (iv) the avail-

*Corresponding author. E-mail: jana.ahmad@fel.cvut.cz.

ability of OntoUML, an ontology modeling language that could be used to create ontology-driven conceptual models and domain ontologies in a variety of existing UML tools. OntoUML aims to design a language for structural conceptual modeling [23].

This paper is an extended version of a previous work [3] which described a new Resource Description Framework (RDF)[1] indexing approach based on UFO. For this version, we present as our main contribution the benchmark of foundational Patterns that are generated based on the UFO [24], and evaluate these foundational Patterns on UFO-based indexed big data sets generated by our UFO-based-model data generator, i.e., build foundational-based domain models, generate big data as instances from this foundational model, store the generated data in triple stores, index the stored data with foundational index and evaluate these data on generated fondational query patterns, see figure 1.

The paper is organized as follows. The motivation scenario section 2. Section 3 reviews the necessary background on RDF and querying. In Section 3.2 we briefly define the notion of the Unified Foundational Ontology. Section 4 presents related work. Data generator is explained in section 5 including the UFO model and UFO index technique. Foundational patterns benchmark is presented in section 6. The evaluation of benchmark query results is given in Section 7, with the description of our use-case. The discussion of the experiments results is on section 8. Finally, we conclude our paper in Section 9.

2. Motivation

In recent years, human-ontology interaction has become an increasingly important subject for computational and information systems developers. Human information consumers and web agents need to use and query ontologies using their web applications, that could understand common sense knowledge. Thus the need for developing ontological foundations for conceptual modeling arises. Despite the improvements in query answering systems technology in recent decades, currently there are many problems, such as searching, extracting, maintaining, uncovering and viewing information [4]. Also, query answering systems suffer from: inconsistencies in terminology, keywords do not provide user with the results he wants or understands, weakly structured collections of documents and slow retrieval of the results. The aim of the semantic web is to allow much more advanced knowledge management system, thus information needs to be organized in conceptual models according to its meaning, and have conceptual models based answering system.

The aforementioned points motivate us to create a foundational conceptual benchmark and index the data by using our UFO-based index [3]. This benchmark can be reused not only for our foundational generated data but also for all data sets compliant with the unified foundational ontology.

[1] https://www.w3.org/RDF/

3. Background

First, we introduce a fragment of OWL 2-DL [67] in a simplified manner, as a knowledge-representation language together with simple conjunctive queries over this fragment. Next, we overview UFO, as one of the foundational ontologies. Then, we show an example of RDF representation of OWL-based UFO fragment and queries. Last, we introduce the JOPA library that we use to access our UFO-based domain ontologies.

3.1. 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

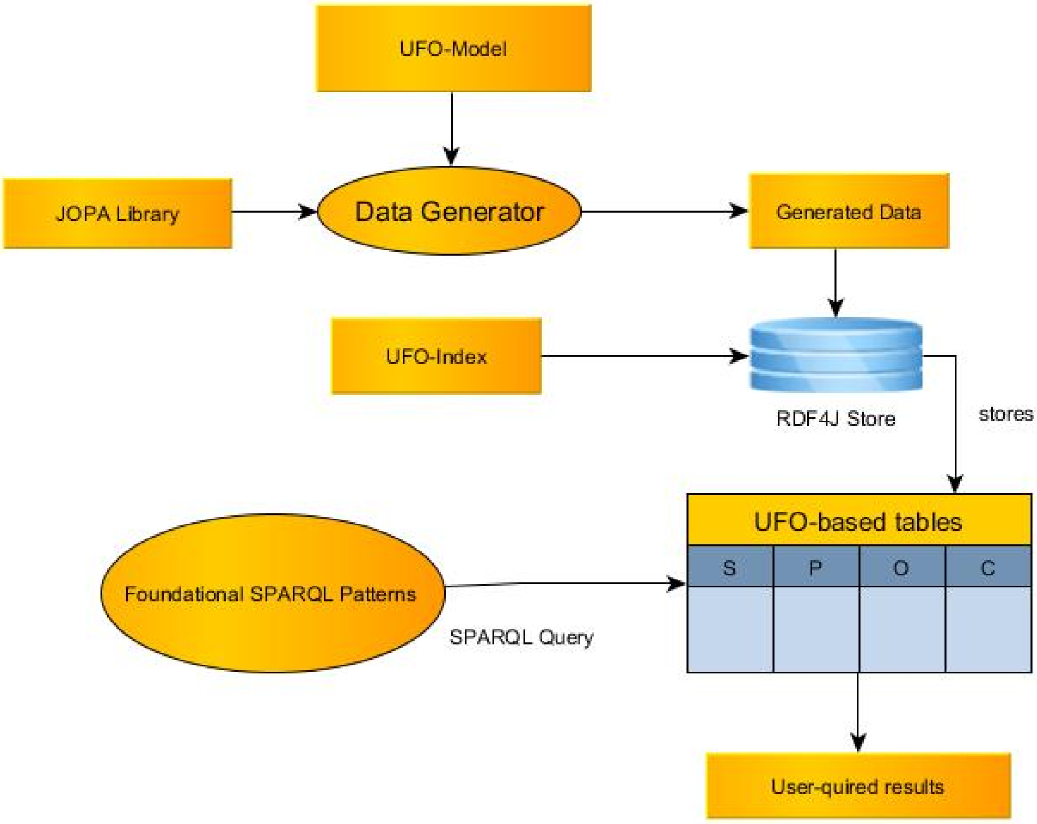

Fig. 1. UFO-Based Data Generator Model

least one instance of C. For example, $(\exists hasChild \cdot Man)$ denotes the 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, $(\forall hasChild \cdot Man)$ denotes the class of all individuals having only sons as children, if any,
- qualified cardinality restrictions $(\leqslant n \, \mathsf{P} \cdot C)$, or $(\geqslant n \, \mathsf{P} \cdot C)$, $(= n \, \mathsf{P} \cdot C)$, denoting a class, elements of which are related through P to at least-/at most/exactly n individuals through P. For example, $(\geqslant 4 \, \mathsf{hasChild} \cdot Man)$ denotes a class, elements of which have at least four sons (and possibly some daughters).

Full OWL 2-DL syntax as well as its formal semantics can be found in [67].

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 a an OWL individual. Intuitively, queries match the class/property assertion axioms, possibly extended by inferencing from other axioms. Full query syntax and semantics of distinguished conjunctive queries can be found in [48]. Let's show the notions on an example.

Example: Having an OWL 2-DL ontology $\mathcal{O} = \{\mathsf{Agent} \sqsubseteq \mathsf{Object}, \mathsf{Agent}(\mathsf{a}), \mathsf{performs}(\mathsf{a}, \mathsf{b})\}$, the query $Q(?x_1, ?x_2) = Object(?x_1), performs(?x_1, ?x_2)$ 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 agent ($\mathsf{Agent} \sqsubseteq \mathsf{Object}$).

3.2. Unified Foundational Ontology

UFO is a top-level ontology that has been developed based on a number of theories from Formal Ontology, Philosophical Logic, Philosophy of Language, Linguistics and Cognitive Psychology [24]. Its main concepts fundamental for this work are sketched in the UML class diagram in Fig. 2. UFO describes endurants that are static objects (UFO-A) [23], perdurants/events (UFO-B) [31] and social agents (UFO-C) built on top of UFO-A and UFO-B [27]. UFO splits entities into endurants and perdurants which are both individuals, i.e. entities that exist in reality and possess an identity that is unique ($\mathsf{Endurant} \sqsubseteq \mathsf{Individual}$), ($\mathsf{Perdurant} \sqsubseteq \mathsf{Individual}$)[2]. Endurants can be observed as complete concepts in a given time snapshot, and they can be any object (e.g. an agent, aircraft) ($\mathsf{Object} \sqsubseteq \mathsf{Endurant}$), or its tropes or moments (e.g. speed, location, colors, etc.) ($\mathsf{Moment} \sqsubseteq \mathsf{Endurant}$), that exist as long as an object they inhere in exists ($\mathsf{Moment} \sqsubseteq (= 1\ \mathsf{inheresIn}{\cdot}\mathsf{Object})$), and situations ($\mathsf{Situation} \sqsubseteq \mathsf{Endurant}$).

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

Events happen in time and cannot undergo nonrelational changes, e.g., death can't die. They can be either atomic or complex ($\mathsf{Event} \sqsubseteq (\mathsf{AtomicEvent} \sqcup \mathsf{ComplexEvent})$). complex events have temporal branchings, occurring over incomparable TimePoints, and have participants ($\mathsf{Event} \sqsubseteq (\geqslant 1\ \mathsf{hasParticipant} \cdot \mathsf{Object})$) and complex events have parts ($\exists\ \mathsf{hasEventPart} \cdot \top \sqsubseteq \mathsf{ComplexEvent}$) [31]. An event occurs in a certain situation at a certain point in time, and transforms it to another situation, they may change reality by changing the state of affairs from one (pre-state) situation to a (post-state) situation [29]. 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.

Moreover, UFO defines Dispositions which are Intrinsic Moments ($\mathsf{IntrinsicMoments} \sqsubseteq \mathsf{Moment}$), i.e. existentially dependent entities that are realizable through the occurrence of an Event ($\mathsf{Dispositions} \sqsubseteq \mathsf{Moment}$). This occurrence brings about a Situation [28]. In other words, UFO considers dispositions as properties that are only manifested in particular situations or the occurrence of certain triggering events, and that can also fail to be manifested ($\mathsf{Dispositions} \sqsubseteq (= 1\ \mathsf{isManifestedBy}{\cdot}\mathsf{Event})$). Dispositions inhere in particular objects ($\mathsf{Dispositions} \sqsubseteq (= 1\ \mathsf{inheresIn}{\cdot}\mathsf{Object})$). For example, security flaw in an information system is manifested by event of stealing sensitive data that brings about non-safe situation.

Additionally, UFO introduces the notion of agents ($\mathsf{Agent} \sqsubseteq \mathsf{Substational}$), i.e. proactive objects with an intention, the propositional content of intention is a Goal. Intentions cause the agent to perform actions ($\exists\ \mathsf{performs} \cdot \top \sqsubseteq \mathsf{Object}$) [25]. Finally, UFO also defines services [20], and powertypes, i.e. universal types whose instances are individuals in the subject domain [11, 32].

Representation language: UFO-A is expressed in a quantified modal logic (QML) that allows the expression of the alethic modalities of truth (viz., necessity and contingency), and UFO-B is defined in first-order logic (FOL) with the Method of Temporal Arguments (MTA) [62], But [6] defines a method for rewriting UFO-A in FOL, with no loss of content, and consistently with a revisited UFO-B. Also, to represent UFO using Description Logic (DL), authors in [5] proposed a number of alternative translations from UFO-B's original axiomatization in first-order logic to the description logic SROIQ, which is the formal underpinning of OWL 2 DL. UFO is used in domains such Geology, Biodiversity Management, Petroleum Reservoir Modeling, Disaster Management, Datawarehousing, Telecommunications, Petroleum and Gas, Logistics, among many others [33].

3.3. 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 section 4. 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 [38], a query language for RDF.

Consider a *triple pattern*, an ordered tuple $t^v = (s^v \quad p^v \quad o^v)$, 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*[3] C.

Having an OWL ontology $\mathcal{O}$, its RDF serialization is given directly by OWL specification [67]. For distinguished conjunctive queries, we translate each atom of the form $\mathsf{A}(y)$ into an RDF triple pattern

[2] We reuse Description Logic formalization of basic UFO concepts introduced in [5]

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

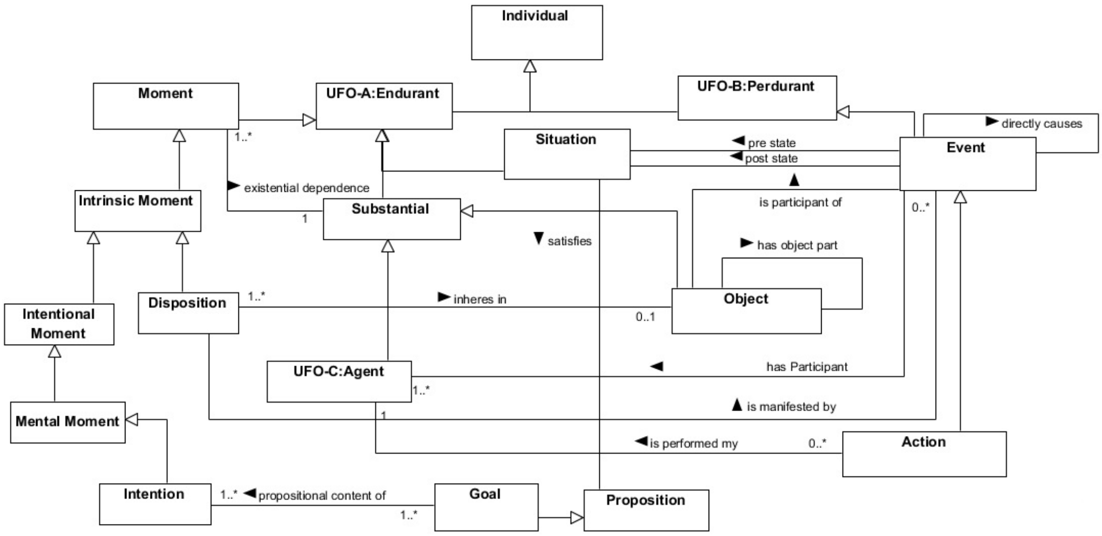

Fig. 2. Main concepts of UFO

$(y$ rdf:type $\mathsf{A})$[4] 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 3.1, its RDF serialization would be[5]

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

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

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

4. Related Work

Recently, different benchmarks have been proposed to compare query execution performance for triple stores. *FEASIBLE* [58] which is a cluster-based SPARQL benchmark generator, which is able to synthesize customizable benchmarks from the query logs of SPARQL endpoints. It generates customized benchmarks from a set of queries for given use cases or needs of an application. The *Berlin SPARQL Benchmark (BSBM)* [8] is designed to compare query performance of native RDF stores with the performance of SPARQL-to-SQL rewriters across architectures. It is applied to various triple stores, such as Sesame (now RDF4J), Virtuoso, and Jena-TDB. The BSBM benchmark is settled in an e-commerce use case in which a set of products is offered by different vendors and consumers have posted reviews about these products on various review sites. The *Lehigh University Benchmark (LUBM)* [35] is a widely used benchmark for comparing the performance, completeness and soundness of OWL reasoning engines. It is based on a customizable and deterministic generator of synthetic data. In the LUBM, the Univ-Bench ontology models the university domain that include universities, their departments, their professors, employees, courses, publications and their relations in the OWL language and offers necessary features for evaluation purposes. The OWL datasets are synthetically created over the ontology. The data generated are random and repeatable and can scale to an arbitrary size, and it uses plain SPARQL queries. The University Ontology Benchmark (UOBM)[6] is a more expressive new version of LUBM with a more complex ontology, which also contains disjunctive axioms and negation. The *DBpedia SPARQL Benchmark DBPSB* [54] is a benchmark for evaluating the performance of triple stores based on non-artificial data and queries, it is settled in the DBLP bibliographic database. It generates bench-

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

[5] 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>.

[6] https://www.cs.ox.ac.uk/isg/tools/UOBMGenerator/

mark queries from DBpedia dataset and tests them with 4 different triple stores, namely Virtuoso, Sesame, Jena-TDB, and BigOWLIM (now it is GraphDB). *A SPARQL Performance Benchmark (SP2Bench)* [59] is a benchmark to assess SPARQL performance. It proposes a methodical approach for testing the performance of SPARQL engines w.r.t. different operator constellations, RDF access paths, typical RDF constructs, and a variety of possible optimization approaches.

Relation to our approach. Although each of aforementioned benchmarks has its own proposal and criteria to compare SPARQL query execution and performance for triples stores, none of them propose a foundational query benchmark, i.e., a benchmark that can be used for all datasets compliant with the foundational ontology. Thus, in this paper, we propose a benchmark of foundational patterns that are generated using Unified foundational ontology (UFO) [24] to optimize SPARQL queries execution of triples stores.

5. Data Generator

UFO-based Data Generator (UDG)[7] is a RDF triples generator. It generates data based on a foundational ontology model, the generated data is stored in a triple store and is indexed using UFO index [3], we can access this data by JOPA. Thus, in this section, we, first, describe the UFO Model. Next, we show the JOPA application. Then, we present the UFO indexing technique.

5.1. UFO Model

UDG generates a number of persons (Agents), Actions and Tropes. Regarding UFO, Each event has Participants of Endurants. In the benchmark each Action has random Agents as participants in this event. Every Action has a balanced binary tree of sub-events. Each participant has numbers of different properties (or Tropes) are persisted with it. All attributes of all entities are set, none is left empty Figure 3 shows the main entities in the model as following:

– Trope (or Moment): Typical examples of tropes are: a color, a connection, a gender, a social commitment. An important feature that characterizes all moments is that they can only exist in other particulars (for example, color can exist only in some particular such as the color of an apple, color does not exists without the existence of apple). The relation of inherence is a special type of existential dependence relation that holds between a moment x and the particular y on which x depends. Thus, for a particular x to be a moment of another particular y, the relation i(x,y) must hold among the two. For example, inherence your ability to walk to your legs. Also, moments can inhere in other moments. For example, the graveness of a particular symptom. The infinite regress in the inherence chain is prevented by the fact that there are individuals that cannot inhere in other individuals, namely, objects.
– Object: Is an Endurant. Objects are particulars that possess (direct) spatial-temporal qualities, and those are founded on matter. Examples of objects include ordinary entities of everyday experience such as an individual person, a dog, a house, a hammer, a car, Alan Turing and The Rolling Stones but also the so-called Fiat Objects such as the North-Sea and its proper-parts, postal districts and a non-smoking area of a restaurant. In contrast with moments, objects do not inhere in anything and, as a consequence, they enjoy a higher degree of independence.
– Agent which is proactive object, it has its own beliefs, intentions, and goals that are sets of intended states of affairs of an agent. An agent role is defined by the set of commitments and claims implied by the its role to achieve his goals. The category of agents further specializes in Physical Agents (e.g., a person) and Social Agents (e.g., an organization, a society). Agents can also be further specialized into Human Agent, Artificial Agent and Institutional Agent, which can be represented, respectively, by human beings, computationally-based agents and organization or organizational unit (departments, areas and divisions). Institutional Agents are composed by a number of other agents, which can themselves be Human Agents, Artificial Agents or other Institutional Agents.
– Actions are intentional events, i.e., events that agents perform in order to satisfy their goals. As events, actions can be atomic or Complex Action. While an Atomic Action is an action event that is not composed by other action events, a Complex Action is a composition of at least two basic

[7]https://github.com/ahmadjana/ufomodel: It is the GitHub link for the source code.

- actions. Each event has Participants of Endurant types, i.e., all objects that participate in events. For example, a football player participates in a football match.

5.2. JOPA Library

Once the UFO-based model is designed, then the task now is to generate and access to the big data. JOPA is a persistence library primarily designed for accessing OWL ontologies. It is aimed at efficient programmatic access to OWL2 ontologies and RDF graphs in Java [51]. It is used here to create instances of the model entities, i.e., create an object graph and then persist it into repository as follows:

- persist Agent instances and assign data properties to all of them;
- all Agent instances have tropes (OWL object property) or moments (OWL datatype property) that specify each agent;
- assign to each generated Action (an event type) a random Agent (Objects) that participated in this event (has participant: object property) and persist them in a separate transaction;
- generate and persist sub events (event parts) that comprise that whole event.

section

5.3. UFO Index

Once having the generated data in the repository, the task now is to index the generated data using UFO-index script; In [3], we presented our novel approach to improve the efficiency of SPARQL[8] queries by using UFO-based indexing techniques. Note, we use our UFO index not only for generated data but also, for all UFO grounded data sets. We created UFO-based physical design index tables that store RDF data according to the main concepts of UFO, Perdurant and Endurant. As following:

- UFO Triple Tables that store triples physically into two tables instead of one triple table as in general design [1, 16]; one for Perdurant category and the other for Endurant.

[8]common SPARQL prefixex 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/.

Table 1
Perdurant Table, depicted from [3]

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 2
Endurant Table, depicted from [3]

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

- UFO Property Table that builds a UFO property table for endurants and another table for perdurants, that will reduce Null values in each property table [1, 16], but we will still have them.
- UFO Vertical Partitioning that applies vertical partitioning approach where 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 the property, and the second column contains the object values for those subjects [1, 2, 37].

For this version, we use UFO triple table technique to index the generated and existing data. Figures 1 and 2 explain how this technique works by dividing one triple table into two UFO-based categories tables.

6. Foundational Patterns Benchmark

After having the data, the job now is to generate a benchmark of queries the users interest in, in other words, queries that match people thoughts and languages. Users are interested in searching and have answers for specific physical objects (e.g., person, man, woman, car, animal, etc.), tropes or properties (e.g., weight, height, color, etc.) and events (e.g., accident, party, fight, war, sales, etc.). For example, some people search for individuals who attended Celine Dion birthday events. Therefore, they are looking for concepts such invited, going, or attended. Other example, one of the most important factors in creating a successful e-commerce shop is answering the question: What to sell online (objects)? When will the black Friday start?

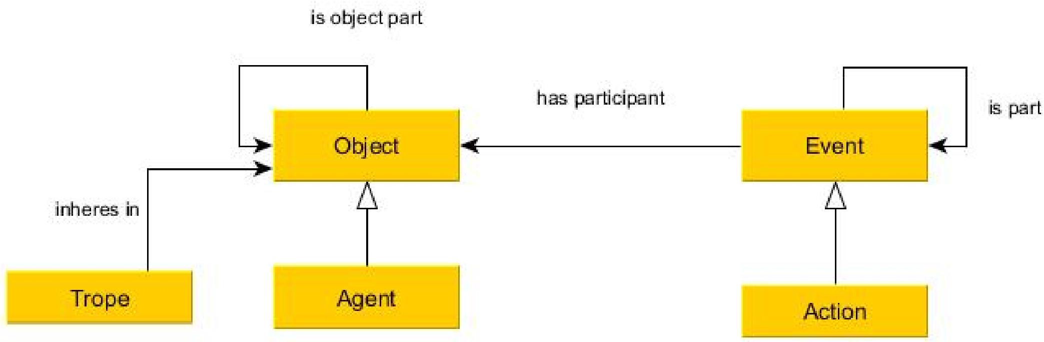

Fig. 3. UFO Model Entities

Thus, the meanings of the variety of words such as: *red, John, Jana, marriage, accident, ball, process, attend, happen, party, hot, warm, play, situation, tasks*, etc. reflect 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 word *("how" versus "what")* is often invoked to check the different nature of these elements [30]. Moreover, UFO-B suggests a discrete linear ordering of TimePoints to answer question word *("when")* [6].

Therefore, 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.

How this benchmark is created: Conceptually, we created a benchmark of all possible foundational queries that could be created between Perdurants-Endurants, Pendurant-Pendurant or Endurant-Endrant. i.e., foundational patterns between structural (objects, tropes, agents, situations, etc.) and dynamic aspects (events, actions, etc) of reality, thus, it must be able to characterize ontological aspects of endurants, perdurants, as well as their interplay. In the table 6.1 are examples of foundational patterns of the generated benchmark. Technically, we created these benchmark automatically by executing SPARQL queries over UFO-based-indexed triple tables [3], Each query selects all relations that has Perdurant or Endurant as its domain or range and vice-versa.

Query 1, selects all relations from Endurant table have object as a domain, such as, the participation relation (ufo:is participant of) between events and their participations. So, user can ask questions such as, Who participated in the Joker film? Or, inheritance relation that expresses the properties or moments that inhere in objects, for example, What is the color of Barcelona's team jerseys?

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g {?term rdfs:domain
   ufo:Object} }
```

Listing 1: SPARQL query

Query 2 retrieves all relations that have event as a domain, i,e, the dynamic aspects of reality. Then, the user can have answers to questions such as, when did the second world war start? What are the parts of Jana's wedding party?, etc.

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {?term rdfs:domain
   ufo:Event} }
```

Listing 2: SPARQL query

6.1. Description of the Benchmark Patterns

In this section, we describe the created foundational patterns. As we mentioned in section 3.2, UFO mainly

Table 3

Foundational query patterns and their formal representation.

	Patterns	Pattern formalization
P_1	What are the tropes (properties) of an object?	$(?p) \rightarrow$ ufo:has-trope$(?p, o1)$
P_2	What are the objects that participate in a given event e1 ?	$(?o) \rightarrow$ ufo:has-participant$(?o,$ e1$)$
P_3	What are the parts of an object?	$(?o1, ?o2) \rightarrow$ ufo:has-object-part$(?o1, ?o2)$
P_4	What are the parts of a given event e1?	$(?e) \rightarrow$ ufo:has-event-part(e1$, ?e)$
P_5	What are factors of an event?	$(?f) \rightarrow$ ufo:is-manifestation-of$(?f, e1)$
P_6	What is the situation that a given event changed it?	$(?s) \rightarrow$ ufo:pre-state$(?s, e1)$
P_7	What is the resulting situation of a given event?	$(?s) \rightarrow$ ufo:post-state$(?s, e1)$
P_8	What does an agent perform?	$(?s) \rightarrow$ ufo:performs$(?s, a1)$
P_9	What are the actions that agents perform?	$(?s, ?a) \rightarrow$ ufo:performs$(?s, ?a)$
P_10	What is directly cause a given event?	$(?e) \rightarrow$ ufo:directly-causes$(?e, e1)$
P_11	when did a given event start ?	$(?t) \rightarrow$ ufo:has-begin-point$(?t, e1)$
P_12	when did a given event finish ?	$(?t) \rightarrow$ ufo:has-end-point$(?t, e1)$
P_13	What is an entity that a specific property inheres in it?	$(?e) \rightarrow$ ufo:inheres-in$(?e1, p1)$
P_14	What is the situation triggers a given event ?	$(?s) \rightarrow$ ufo:triggers$(?s, e1)$
P_15	What is the situation a given event bingsAbout ?	$(?s) \rightarrow$ ufo:brings-about$(e1, ?s)$
P_16	how a specific disposition that inheres in an object is activated?	$(?s) \rightarrow$ ufo:activates$(?s, d1)$

distinguishes between events and objects. Thus, the foundational benchmark consists of all the patterns between UFO categories, i.e., the interplay between endurants and the dynamic aspects of reality (e.g., events, processes, causation, dispositions, situations, moments). Given the objective of characterizing this interplay between endurants and perdurants, these two ontologies are meant to form an integral whole. Thus, let's discuss some examples or queries of benchmark patterns from table 6.1:

who participates in an event? Events are mapping of statements or occurrence in the reality, in which objects (things and people) participate, playing certain tasks (Event $\sqsubseteq$ ($\geqslant$ 1 has Participant$\cdot$Object)). E.g., what are the objects who participate in the department meeting?, Who attends the Christmas party? and etc.

What are the object's parts? Endurants are entities that, whenever they exist, they exist with all their parts, while maintaining their identity, i.e., we can refer to Jana's arm, leg and head as the same entity (Object $\sqsubseteq$ ($\geqslant$ 1 isObjectPartOf$\cdot$Object)), e.g., What are the parts of Jana's body? What are the parts of the car?

What are the event's parts? This pattern describes how events relate to its parts, where according UFO every complex event consists of parts which accumulate together to have the end event (ComplexEvent $\sqsubseteq$ ($\geqslant$ 2 hasEventPart$\cdot$Event). E,g., What are all temporal precedence involved in an event?

What does an event bring about? How is an event triggered? An event occurs in a certain situation at a certain point in time, and transforms it to another situation, they may change reality by changing the state of affairs from one (pre-state) situation ufo:pre-situation to a (post-state) situation ufo:post-situation [29]. An Event *bringsAbout* exactly one Situation (Event(e) $\rightarrow \exists!$s $(bringsAbout(e, s)))$, which holds in all *endPoints* of the Event. Also, an Event is triggered by exactly one Situation (triggers(s, e) $\rightarrow$ Situation(s) Event(e)), which holds in all *beginPoints* of the Event, e.g. The Event car's Accident *bringsAbout* the Situation that *driverIsinjured*, which triggers the event *ambulance'sCall*.

What does a specific situation activate? A Situation triggers an event if and only if (iff) there is a Disposition (e.g skills, abilities, disabilities, weak points, etc.) that is activated by the Situation ($\exists$activates $\cdot$ $\top$ $\sqsubseteq$ (Situation)) and is manifested by an event, e.g., having a written exam (Situation) activates the ability of writing (Disposition) of a student to write (Event).

What are the factors of a given event? In UFO, events existentially depend on the objects that participate in them and an event is a manifestation of a disposition of an object, then an event occurs due to the dispositions of its participants, where dispositions are defined as properties that inhere in particular objects and are only manifested in particular situations of the occurrence of certain triggering events, and that can also fail to be manifested ($\exists$manifestedBy $\cdot \top \sqsubseteq$ (Disposition)) [28]. When manifested, they are manifested through the occurrence of resulting events and state changes (ufo:isManifestedBy), e,g., what are the factors of a cancer disease?

What does an agent perform? Agent has its own beliefs, intentions, and goals that are sets of intended states of affairs of an agent. He performs actions to achieve their goals (ufo:performs), e.g., a doctor performs surgery operations in order to satisfy his intentions in saving peoples' life.

Table 6.1 contains the SPARQL representation of the benchmark queries.

SPARQL Representation Of the Foundational Queries

Query 1: who participates in an event ?

```
SELECT DISTINCT ?particioations FROM
    NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {benchmark:
   givenEvent ufo:has_participant ?
   particioations} }
```

Query 2: What are the object's parts?

```
SELECT DISTINCT ?parts FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g {benchmark:
   givenObject ufo:has_object-part
   ?parts} }
```

Query 3: What are the event's parts?

```
SELECT DISTINCT ?parts FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {benchmark:
   givenEvent ufo:has_event-part ?
   parts} }
```

Query 4: What does an event bring about?

```
SELECT DISTINCT ?situation FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {benchmark:
   givenEvent ufo:bringsAbout ?
   situation} }
```

Query 5: how an event is triggered?

```
SELECT DISTINCT ?situation FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {?situation ufo:
   triggers benchmark:givenEvent }
   }
```

Query 6: What does a specific situation activate?

```
SELECT DISTINCT ?disposition FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g { benchmark:
   givenSituation ufo:activates ?
   disposition } }
```

Query 7: What are the factors of a given event?

```
SELECT DISTINCT ?disposition FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:is_manifestation-
   of ?disposition } }
```

Query 8: What does an agent perform?

```
SELECT DISTINCT ?Action FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g { benchmark:
   givenAgent ufo:performs ?Action
   } }
```

Query 9: What are the tropes (properties) of an object?

```
SELECT DISTINCT ?trope FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g { benchmark:
   givenObject ufo:has_trope ?trope
    } }
```

Query 10: What is the situation that a given event changed it?

```
SELECT DISTINCT ?situation FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:pre_state ?
   situation } }
```

Query 11: What is the resulting situation of a given event?

```
SELECT DISTINCT ?situation FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:post_state ?
   situation } }
```

Query 12: when did a given event start?

```
SELECT DISTINCT ?point FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:has_begin_point ?
   point } }
```

Query 13: when did a given event finish?

```
SELECT DISTINCT ?point FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:has_end_point ?
   point } }
```

Query 14: What is the entity that a specific property inheres in it?

```
SELECT DISTINCT ?entity FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g { benchmark:
   givenProperty ufo:inheresIn ?
   entity} }
```

Query 15: What is directly cause a given event?

```
SELECT DISTINCT ?event FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:directly_causes ?
   event } }
```

Query 16:What are the actions that agents perform?

```
SELECT DISTINCT ?action ?agent FROM
   NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g { ?agent ufo:
   performs ?action } }
```

7. Benchmark Experiment

For evaluation, we tested the foundational patterns in two different use cases, generated data using UDG and existing real world data. The comparison is done in different triple stores. We run the Foundational SPARQL Benchmark against three popular RDF stores (Sesame (or RDF4J[9]), Fuseki Jena with JenaTDB and GraphDB).

Sesame (or RDF4J[10]): Version 2.5.2+0dedb9c with Tomcat Version 8.0.48, Operating System Windows 10 10.0 (amd64), Java Runtime Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8.0-151). It is physically designed bases on B-Tree indexing triple tables with context. It allows the user to choose between three storage engines (in-memory, native, DBMS-backend).

Fuseke Jena [11]: Version 3.13.1 with Tomcat Version 8.0.48, Operating System Windows10 10.0 (amd64). It provides the SPARQL 1.1 protocols for query and update as well as the SPARQL Graph Store protocol. Fuseki is tightly integrated with TDB to provide a robust, transactional persistent storage layer, and incorporates Jena text query. It can be used to provide the protocol engine for other RDF query and storage systems.

[9]http://rdf4j.org/

GraphDB free[12]**:** Version 9.1 with Tomcat Version 8.0.48, Operating System Windows10 10.0 (amd64). It is the free standalone edition of GraphDB. It is implemented in Java and packaged as a Storage and Inference Layer (SAIL) for the RDF4J RDF framework. GraphDB Free is a native RDF rule-entailment and storage engine. The supported semantics can be configured through rule-set definition and selection. Included are rule-sets for OWL-Horst, unconstrained RDFS with OWL Lite and the OWL2 profiles RL and QL. Custom rule-sets allow tuning for optimal performance and expressiveness.

The experiment was conducted on a Lenovo, Intel® Core™i5-7200U CPU @2.5GHz 2.71 GHz processor, installed memory is 8.00 GB and 64-bit operating system. The average execution time results and standard deviation of pattern instances are specified, where the given results are averages from executing each query five times against the different triple stores.

We run the queries in different data sets to compare their execution time (performance), number of results and correctness w.r.t. each triple store. The correctness of results is evaluated by domain expert in real data only. Each query will be executed three times either on the Perdurant table (named graph) or on the Endurant table after indexing the data by running UFO indexing script on a triple's repository. As we proposed in [3], this script automatically group all Perdurant statements together through a single group identifier (Named Graph), i.e., in one Perdurant table. And all Endurant triples in another Endurant table.

7.1. SPARQL Features Selection

To use the foundational benchmark on different data sets by running multiple queries against triple stores, we select a number of frequently executed queries that cover most SPARQL features that allow us to assess the performance of foundtional queries with SPARQL features. Note, all the executed queries are instances of our foundational patterns. The SPARQL features we consider are:

- the overall number of triple patterns
- SPARQL pattern constructors (UNION or OPTIONAL)
- the solution sequences and modifiers (DISTINCT)
- filter conditions and operators (FILTER, LANG, REGEX and STR)

7.2. Generated Data Experiments

We instantiated the foundational patterns w.r.t. the generated data benchmark and w.r.t. SPARQL features; we tested the instances of UFO pattern for three generated data-set sizes (200000, 500000 and million triples). Following are samples of the foundational patterns instances with their SPARQL queries.
Q1: What are the tropes (properties) for a (Person-1000344628) and Person-1009237217? (Instance of P1).

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Endurant>
WHERE {GRAPH ?g {{benchmark:Person
   -1000344628 ufo:has_trope?term}
UNION {benchmark:Person-1009237217
   ufo:has_trope?term}}}
```

Q2: Select all participants of all events (Instance of P2)

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {{?event ufo:
   has_participant ?term}
OPTIONAL { ?term rdf:label ?label.}
}}
```

Q3: What are the parts of an Event-1453752566? Instance of P4

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/ontologies/
   ufo/Perdurant>
WHERE {GRAPH ?g {benchmark:
   Event1670269156 ufo:has_part ?
   term} }
```

Figures 4, 6, 8, 5, 7 and 9 show the results of running execution time of the instantiated queries on the different triple stores.

7.3. Real Word Data Sets Experiments: Aviation Safety Data Set

The ontology that we used to evaluate the benchmark is the Aviation Safety ontology. We designed

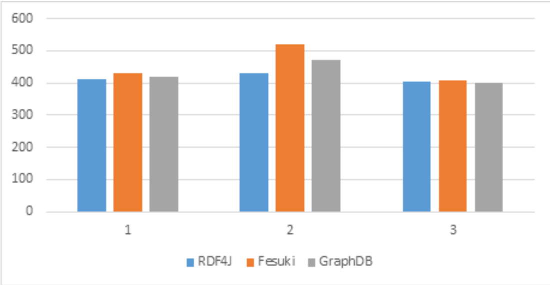

Fig. 4. Mean Value ϕ of Execution Query Time for *200000* triples

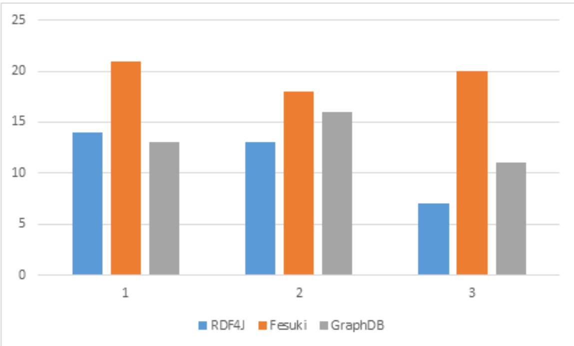

Fig. 5. Standard Deviation σ of Execution Query Time for *200000* Triples

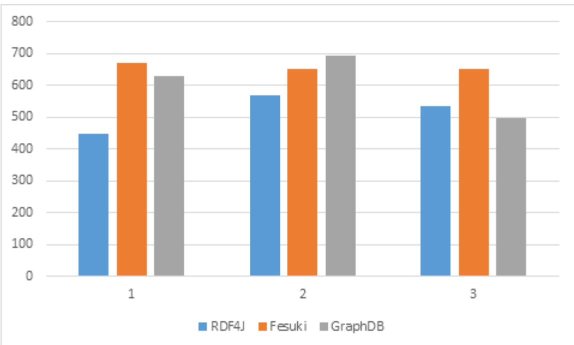

Fig. 6. Mean Value ϕ of Execution Query Time for *500000* triples

the Aviation Safety Ontology[13] 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 aviation domain [47]. Our strategy is to analyze safety events

[13]https://www.inbas.cz/aviation-safety-ontology

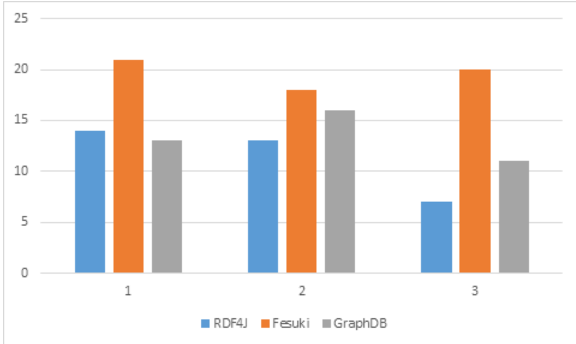

Fig. 7. Standard Deviation σ of Execution Query Time for *500000* Triples

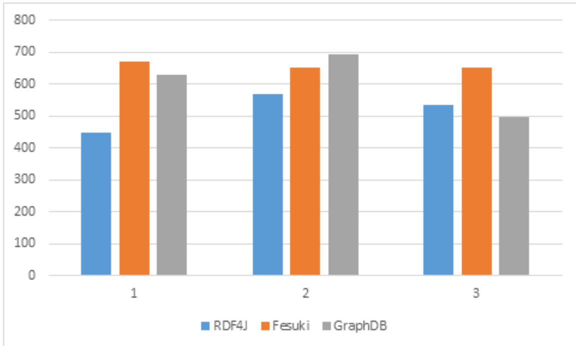

Fig. 8. Mean Value ϕ of Execution Query Time for *million* triples

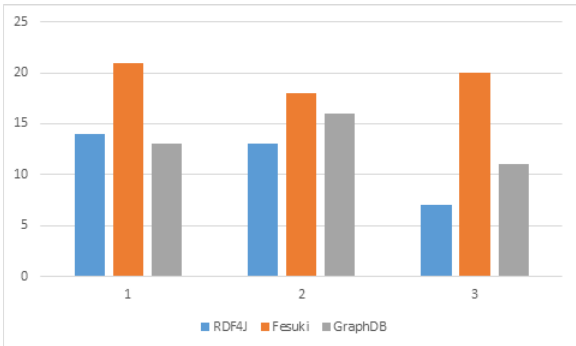

Fig. 9. Standard Deviation σ of Execution Query Time for *million* Triples

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) and all safety reports in aviation safety domain, i.e., all safety reports that are created to inform about all accidents or incidents in aviation domain [47]. The ontology consists of *19421* axioms, *6895* logical axioms, *1725* classes and *129* Object Properties. We built Aviation Safety Ontology on top of the Unified Foundational Ontology (UFO)[14] [23, 56]. Figure 10 depicts basic concepts in Aviation Safety Ontology that are represented in UFO.

For evaluation, we answer examples of foundational patterns instances by running the following queries against selected triple stores. The data set consists of *25000* triples.

- Q1': What are the tropes (properties) that inhere in the air traffic control agent? (instance pf P1)

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/
   ontologies/ufo/Endurant>
WHERE {GRAPH ?g {{ ?term ufo:
   inheresIn aviation-safety:
   air_traffic_control_agent}
}}
```

- Q2: what are the Participants of a Damage event? (instance pf P2)

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

- Q3: What are the parts of a specific Flight? (instance pf P4)

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/
   ontologies/ufo/Perdurant>
WHERE {GRAPH ?g {?term ufo:
   is_part_of aviation-safety:
   Fligt-i} }
```

- Q4: Who performs Ground handling operation-i? (instance pf P8)

```
SELECT DISTINCT ?term FROM NAMED
<http://onto.fel.cvut.cz/
   ontologies/ufo/Endurant>
WHERE {GRAPH ?g {?term ufo:
   performs aviation-safety:
   Ground_handling_operation-i}
   }
```

- Q5: Select everyone that performs actions in aviation domain and filter all participating relation? (instance pf P8 and P2)

```
SELECT ?term
FROM NAMED
<http://onto.fel.cvut.cz/
   ontologies/ufo/Perdurant>
 WHERE {GRAPH ?g {
 ?x ufo:is_performed_by ?name
FILTER (
   NOT EXISTS {
     ?x ufo:has_participant ?name
       }
  )
     }}
```

The result are presented in the figures 11 and 12.

Moreover, we run the previous queries (Q1, Q2, Q3, Q4) on the aviation ontology without applying UFO index (the UFO indexing technique is described in details in [3]) in order to compare the performance of triple stores with and without UFO. Where, in figures 13 and 14, the left set of bars is UFO with UFO index, while the right three bars are without UFO index. These figures indicate that using UFO-indexing approach makes the search process easier and faster, as we demonstrated in [3].

7.4. *UDG Data VS Aviation Safety Existing Data*

In this section, we optimize the foundational patterns by running the same following foundational queries on the same size of both generated and safety data (around 26000 triples). Our goal is to show how these foundational patterns are applicable for any data set based on a unified foundational ontology, i.e., we can run these fondational pattern for different UFO based data sets.

- Q1: who are the *participants* in all event in each data set?

[14] http://onto.fel.cvut.cz/ontologies/ufo/current/index-en.html

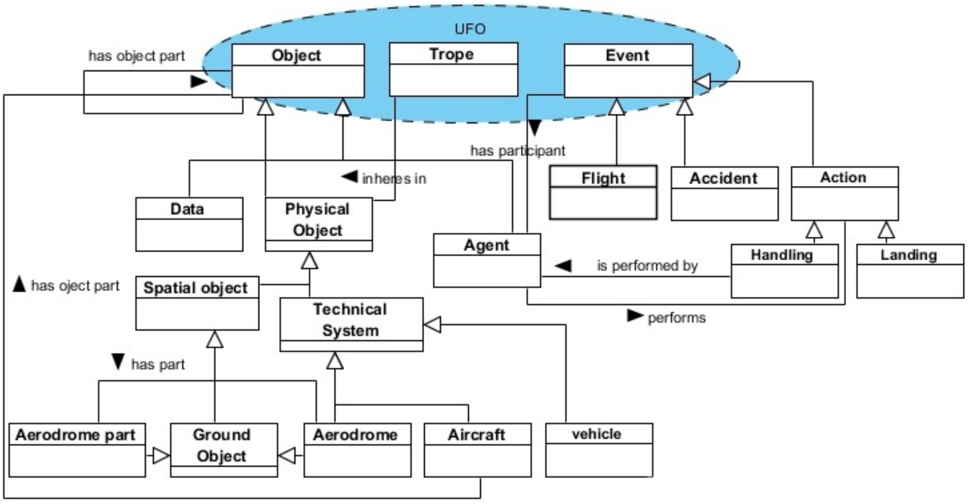

Fig. 10. Aviation Safety Ontology

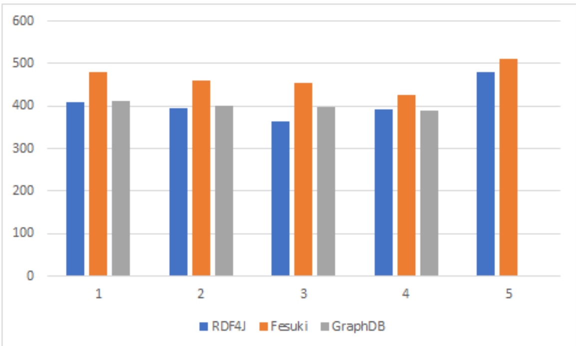

Fig. 11. Mean Value ϕ of Execution Query Time for *Real Data*

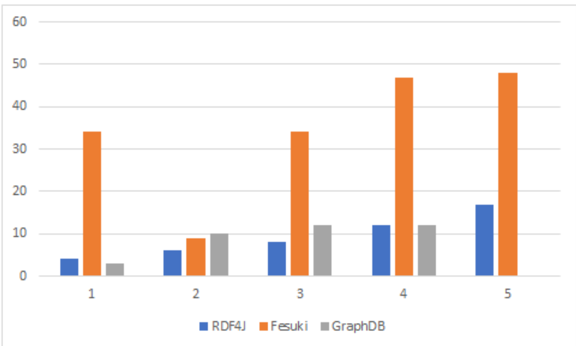

Fig. 12. Standard Deviation σ of Execution Query Time for *Real Data*

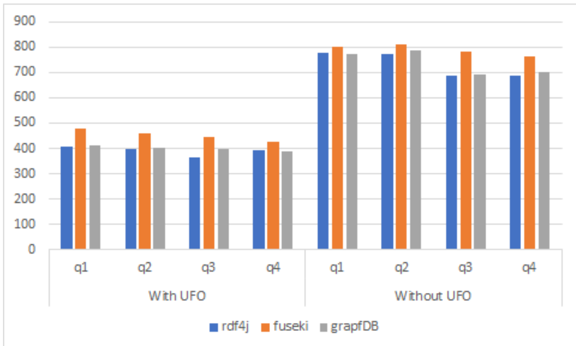

Fig. 13. Mean Value ϕ of Execution Query Time with and without UFO Index

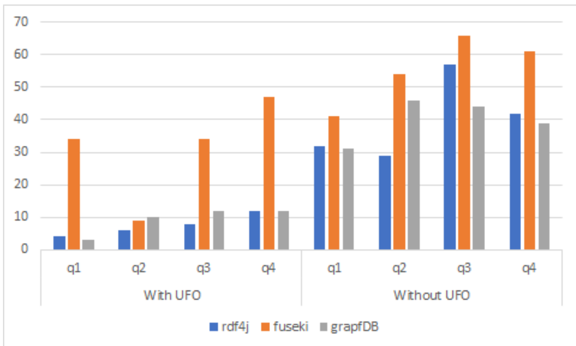

Fig. 14. Standard Deviation σ of Execution Query with and without UFO Index

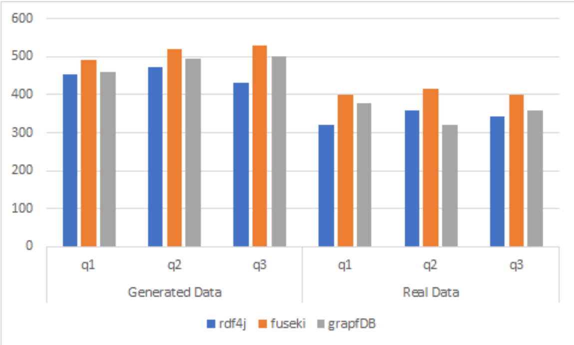

Fig. 15. Mean Value ϕ of Execution Query Time on Generated vs Real Data

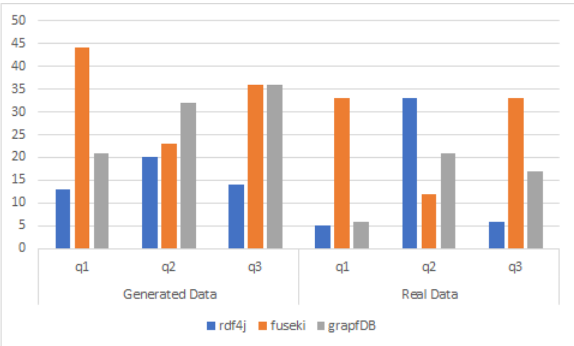

Fig. 16. Standard Deviation σ of Execution Query on n Generated vs Real Data

```
SELECT DISTINCT ?object ?event
    FROM NAMED
<http://onto.fel.cvut.cz/
    ontologies/ufo/Perdurant>
WHERE {GRAPH ?g {?event ufo:
    has_participnt ?object} }
```

– Q2: What are all *properties* in each data set and in which entity inhere?

```
SELECT DISTINCT ?trope ?entity
    FROM NAMED
<http://onto.fel.cvut.cz/
    ontologies/ufo/Endurant>
WHERE {GRAPH ?g ?entity ufo:
    has_trope ?trope} }
```

– Q3: What are all *Actions* that happened in both data sets with their parts?

```
SELECT DISTINCT ?event ?subEvent
     FROM NAMED
<http://onto.fel.cvut.cz/
    ontologies/ufo/Perdurant>
WHERE {GRAPH ?g {?event ufo:
    has_part ?subEvent} }
```

Figures 15 and 16 present the mean values and standard deviations of the execution queries time on both data sets.

8. Experiments Discussion

In this section, we discuss the performance of triple stores after running the above different foundational SPARQL queries and the SPARQL features that we used within those queries against them. From the experiments that we did on different triple stores with different data sets sizes and types (i.e., generated data and existing real data). It is clear that the performance of Fuseki Jena-TDB is the lowest of all triple stores and for all data set sizes. However, RDF4J (Sesame) is better than GraphDB, taking into the consideration that in many cases, RDF4J is almost equal to GraphDB performance. Also, in our experiment, we have shown a significant performance increase on a relatively small data sample for all foundational queries, i.e., the size of data set plays an important role in a triple store performance.

Regarding the number of results, all of the three triple stores return the same number of results. However, GraphDB triple store didn't return any results with respect to query 5 in aviation safety data sets (real data). The problem that, this query involves the feature *FILTER* with *NOT EXIST* which seem not supported by GraphDB free.

Moreover, the results of the real data set validation were checked by a domain expert, who confirmed their correctness and the usability of foundational ontologies in developing safety domain ontologies.

It is interesting to note that foundational patterns allowed us to run the same queries in different data sets as we demonstrated in section 7.4. However, the performance of triple stores w.r.t. real word data was better than generated data.

We did not compare our foundational benchmark with other benchmark in this work, because our benchmark is aimed to be used on foundational-based ontology, and there is no such benchmark which brings the novelty to our approach. But, we compare triples stores with and without ufo index which is the most interesting thing. The results show that using UFO-indexing approaches make the results retrieval process faster, see figures 13 and 14. Also, the results indicates that performance of Fuseki Jena-TDB is the lowest and RDF4J (Sesame) is the best.

9. Conclusion

In this paper, we proposed a foundational benchmark that optimizes SPARQL queries on foundational based domain ontologies. We used this benchmark for evaluating the performance of different triple stores on both real world and generated data. For this purpose, we created a foundational data generator that generates a big data based on a UFO model.

Furthermore, we indexed all data sets using our foundational indexing technique which shows faster results. The benchmark is applicable in any foundational grounded domain ontology, i.e., we can run the same queries in different domains.

Several improvements can be planned for future work to cover more SPARQL features with OWL entailment regimes. Also, for future work, we wil do more evaluation for our UFO indexing approach by generating larger data and we will compare more triple stores with bigger sizes of UFO based indexed data sets.

References

[1] Abadi, D.J., Madden, S.R. & Hollenbach, K. (2007). Scalable Semantic Web Data Management Using Vertical Partitioning. *Proceedings of the 33rd International Conference on Very Large Data Bases*, **VLDB '07**, 411—422. doi:http://doi.ieeecomputersociety.org/10.1109/WKDD.2010.121. http://dl.acm.org/citation.cfm?id=1325900.

[2] Abadi, D.J., Marcus, A., Madden, S.R. & Hollenbach, K. (2009). SW-Store: A vertically partitioned DBMS for semantic web data management. *VLDB Journal*, **18**(2), 385–406. doi:10.1007/s00778-008-0125-y.

[3] Ahmad, J., Kremen, P. & Ledvinka, M. (2018). Optimization of Queries Based on Foundational Ontologies. In *OTM Conferences*.

[4] Antoniou, G. & van Harmelen, F. (2004). A semantic web primer.

[5] Benevides, A.B., Bourguet, J.-R., Guizzardi, G. & Peñaloza, R. Representing the UFO-B Foundational Ontology of Events in SROIQ. In *Proceedings of the Joint Ontology Workshops 2017 Episode 3.*.

[6] Benevides, A., João, P., Almeida, J. & Guizzardi, G. (2019). Towards a Unified Theory of Endurants and Perdurants: UFO-AB.

[7] Bienvenu, M., Bourhis, P., Mugnier, M.L., Tison, S. & Ulliana, F. (2017). Ontology-mediated query answering for key-value stores. In *IJCAI International Joint Conference on Artificial Intelligence* (pp. 844–851). doi:10.24963/ijcai.2017/117.

[8] Bizer, C. & Schultz, A. (2009). The Berlin SPARQL benchmark. *Int. J. Semantic Web Inf. Syst.*, **5**, 1–24. doi:10.4018/jswis.2009040101.

[9] Borgo, S., Kutz, O., Loebe, F., Neuhaus, F., Adrian, K., Antovic, M., Basile, V., Boeker, M., Calvanese, D., Caselli, T., Colombo, G., Confalonieri, R., Daniele, L., Euzenat, J., Galton, A., Gromann, D., Hedblom, M.M., Herre, H., Hinterwaldner, I., Janes, A., Jansen, L., Krois, K., Lieto, A., Masolo, C., Peñaloza, R., Porello, D., Radicioni, D.P., Sanfilippo, E.M., Schober, D., Stufano, R. & Vizedom, A. (Eds.) (2018). *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano, Italy, September 21-23, 2017*. {CEUR} Workshop Proceedings (Vol. 2050). CEUR-WS.org. http://ceur-ws.org/Vol-2050.

[10] Brickley, D. (2003). Basic Geo (WGS84 lat/long) Vocabulary. https://www.w3.org/2003/01/geo/{#}vocabulary.

[11] Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M. & Guizzardi, G. (2017a). Multi-level ontology-based conceptual modeling. *Data Knowledge Engineering*, **109**, 3–24. Special issue on conceptual modeling — 34th International Conference on Conceptual Modeling. doi:https://doi.org/10.1016/j.datak.2017.03.002. http://www.sciencedirect.com/science/article/pii/S0169023X17301052.

[12] Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M. & Guizzardi, G. (2017b). Multi-level ontology-based conceptual modeling. *Data and Knowledge Engineering*, **109**, 3–24. doi:10.1016/j.datak.2017.03.002.

[13] Chiu, P.-H., Lo, C.-C. & Chao, K.-M. (2009). Integrating Semantic Web and Object-Oriented Programming for Cooperative Design. *J. UCS*, **15**, 1970–1990.

[14] Costa, P.D.P.D.P.D.P., Guizzardi, G., Almeida, J.P.A., Pires, L.F.L.F.L.F., Sinderen, M.V., van Sinderen, M., Sinderen, M.V., A. Almeida, J., Pires, L.F.L.F.L.F. & Sinderen, M.V. (2006). Situations in Conceptual Modeling of Context. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)* (pp. 0–9). IEEE. doi:10.1109/EDOCW.2006.62. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031266.

[15] COSTA, P.D., ALMEIDA, J.P.A., PIRES, L.F., GUIZZARDI, G. & SINDEREN, M.V.A.N. (2006). Towards Conceptual Foundations for Context-Aware Applications. In *Modeling and Retrieval of Context. Papers from the 2006 AAAI Workshop* (pp. 54–58). Boston, United States: AAAI Press. http://doc.utwente.nl/63200/1/mrc06-pdockhorn-etal.pdf.

[16] Faye, D.C., Cure, O. & Blin, G. (2012). A survey of RDF storage approaches. *ARIMA Journal*, **15**(2012), 11–35. http://www.citeulike.org/user/paarnio/article/11477528.

[17] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A. & Schneider, L. (2002). Sweetening Ontologies with DOLCE. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web* (pp. 166–181). Springer-Verlag.

[18] Grenon, P. & Smith, B. (2004). SNAP and SPAN: Towards dynamic spatial ontology. *Spatial Cognition and Computation*, **4**(1), 69–104. doi:10.1207/s15427633scc0401$_5$.

[19] Griffo, C., Almeida, J.P.A. & Guizzardi, G. (2015). A systematic mapping of the literature on legal core ontologies. *CEUR Workshop Proceedings*, *1442*.

[20] Griffo, C., A João, P., Almeida, J., Guizzardi, G. & Nardi, J. (2017). From an Ontology of Service Contracts to Contract Modeling in Enterprise Architecture.

[21] Grüninger, M., Fox, M.S. & Gruninger, M. (1995). Methodology for the Design and Evaluation of Ontologies. *International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing*, 1–10. doi:citeulike-article-id:1273832. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.8723.

[22] Guarino, N., Masolo, C. & Vetere, G. (1999). OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems and their Applications*, **14**(3), 70–80. doi:10.1109/5254.769887.

[23] Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Model. PhD thesis. doi:10.1007/978-3-642-31095-9-45. http://doc.utwente.nl/50826.

[24] Guizzardi, G. & Wagner, G. *Towards Ontological Foundations for Agent Modelling Concepts Using the Unified Fundational Ontology (UFO). Lecture Notes in Computer Science* (pp. 110–124). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/b136434.

[25] Guizzardi, G. & Wagner, G. (2005). Towards Ontological Foundations for Agent Modelling Concepts Using the Unified Fundational Ontology (UFO). In *Proceedings of the 6th International Conference on Agent-Oriented Information Systems II*. AOIS'04 (pp. 110–124). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11426714$_8$.

[26] Guizzardi, G. & Wagner, G. (2010a). Towards an ontological foundation of discrete event simulation. *Simulation Conference (WSC), Proceedings of the 2010 Winter*, 652–664. http://dl.acm.org/citation.cfm?id=2433508.2433585.

[27] Guizzardi, G. & Wagner, G. (2010b). Using the Unified Foundational Ontology (UFO) as a foundation for general conceptual modeling languages. In *Theory and Applications of Ontology: Computer Applications* (pp. 175–196). doi:10.1007/978-90-481-8847-5$_8$.

[28] Guizzardi, G. & Wagner, G. (2014). Dispositions and causal laws as the ontological foundation of transition rules in simulation models. In *Simulation Conference (WSC), 2013 Winter* (pp. 1335–1346,).

[29] Guizzardi, G., Falbo, R. & Guizzardi, R.S.S. (2008). Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology. In *In 1th Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'2008*.

[30] Guizzardi, G., Guarino, N. & Almeida, J.P.A. (2016). Ontological considerations about the representation of events and endurants in business models. In *Lecture Notes in Computer Science* (Vol. 9850 LNCS, pp. 20–36). doi:10.1007/978-3-319-45348-4$_2$.

[31] Guizzardi, G., Wagner, G., de Almeida Falbo, R., S.S. Guizzardi, R. & Almeida, J.P.A. (2013). Towards ontological foundations for the conceptual modeling of events. In *Conceptual Modeling* (pp. 327–341). http://link.springer.com/chapter/10.1007/978-3-642-41924-9{_}27.

[32] Guizzardi, G., Almeida, J.P.A., Guarino, N. & Carvalho, V.A.D.E. (2015a). Towards an Ontological Analysis of Powertypes. In *The Joint Ontology Workshops at the International Join Conference on Artificial Intelligence*.

[33] Guizzardi, G., Wagner, G., Almeida, J. & Guizzardi, R. (2015b). Towards Ontological Foundations for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story. *Applied ontology*, *10*. doi:10.3233/AO-150157.

[34] Guizzardi, R., Franch, X., Guizzardi, G. & Wieringa, R. (2013). Using a foundational ontology to investigate the semantics behind the concepts of the i* language. *CEUR Workshop Proceedings*, **978**(iStar), 13–18.

[35] Guo, Y., Pan, Z. & Heflin, J. (2005). LUBM: a benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, **3**, 158–182. doi:10.1016/j.websem.2005.06.005.

[36] Hanke, H. & Knees, D. (2017). A phase-field damage model based on evolving microstructure. *Asymptotic Analysis*, **101**, 149–180.

[37] Harris, S. & Shadbolt, N. (2005). SPARQL Query Processing with Conventional Relational Database Systems. *Web Information Systems Engineering – WISE 2005 Workshops, New York, New York*, **3807**(C), 235–244. doi:10.1007/11581116-25. http://eprints.ecs.soton.ac.uk/11126/.

[38] Harris, S. & Seaborne, A. (2013). SPARQL 1.1 Query Language. Technical report, W3C Consoritum. doi:citeulike-article-id:2620569.

[39] Heller, B. & Herre, H. (2004). Ontological categories in GOL. *Axiomathes*, **14**(1), 57–76. doi:10.1023/B:AXIO.0000006788.44025.49.

[40] Horridge, M. & Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic Web*, **2**, 11–21. doi:10.3233/SW-2011-0025.

[41] Jarke, M., Oberweis, A., van den Heuvel, W.-J., Papazoglou, M. & Jeusfeld, M. (1999). Advanced Information Systems Engineering, *1626*, 41–56–56. doi:10.1007/3-540-48738-7. http://www.springerlink.com/content/87fjm6xr6a08xtyj/.

[42] J.Leigh (2007). AliBaba,. https://bitbucket.org/openrdf/alibaba/.

[43] Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., Bilidas, D., Giese, M., Haase, P., Horrocks, I., Kllapi, H., Koubarakis, M., Özçep, Ö., Rodríguez-Muro, M., Rosati, R., Schmidt, M., Schlatte, R., Soylu, A. & Waaler, A. (2013). Optique: Towards OBDA systems for industry. In *Lecture Notes in Computer Science* (Vol. 7955 LNCS, pp. 125–140). doi:10.1007/978-3-642-41242-4$_1$1.

[44] Klyne, G. & Carroll, J.J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation*, **10**(October), 1—20. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[45] Kolas, D., Emmons, I. & Dean, M. (2009). Efficient linked-list RDF indexing in Parliament. In *CEUR Workshop Proceedings* (Vol. 517, pp. 17–32).

[46] Kontchakov, R., Rodríguez-Muro, M. & Zakharyaschev, M. (2013). Ontology-based data access with databases: A short course. In *Lecture Notes in Computer Science* (Vol. 8067 LNAI, pp. 194–229). doi:10.1007/978-3-642-39784-4$_5$.

[47] Kostov, B., Ahmad, J. & Křemen, P. (2017). *Towards ontology-based safety information management in the aviation industry* (Vol. 10034 LNCS). doi:10.1007/978-3-319-55961-2$_2$5.

[48] Křemen, P. & Kouba, Z. (2011). Conjunctive Query Optimization in OWL2-DL. In *Lecture Notes in Computer Science* (pp. 188–202). Springer Berlin Heidelberg. doi:10.1007/978-3-642-23091-2-18. https://doi.org/10.1007%2F978-3-642-23091-2_18.
[49] Kremen, P. & Kouba, Z. (2012). Ontology-Driven Information System Design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **42**, 334–344.
[50] Křemen, P., Kostov, B., Blaško, M., Ahmad, J., Plos, V., Lališ, A., Stojić, S. & Vittek, P. (2017). Ontological foundations of european coordination centre for accident and incident reporting systems. *Journal of Aerospace Information Systems*. doi:10.2514/1.I010441.
[51] Ledvinka, M. & Kremen, P. (2015). JOPA: Accessing Ontologies in an Object-oriented Way. In *ICEIS*.
[52] Lefever, E. (2016). A hybrid approach to domain-independent taxonomy learning. *Applied Ontology*, **11**(3), 255–278.
[53] Meltzer, P.S., Kallioniemi, A. & Trent, J.M. (2002). Chromosome alterations in human solid tumors. In B. Vogelstein and K.W. Kinzler (Eds.), *The Genetic Basis of Human Cancer* (pp. 93–113). New York: McGraw-Hill.
[54] Morsey, M., Lehmann, J., Auer, S. & Ngomo, A.-C.N. (2011). DBpedia SPARQL Benchmark: Performance Assessment with Real Queries on Real Data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*. ISWC'11 (pp. 454–469). Berlin, Heidelberg: Springer-Verlag. http://dl.acm.org/citation.cfm?id=2063016.2063046.
[55] Murray, P.R., Rosenthal, K.S., Kobayashi, G.S. & Pfaller, M.A. (2002). *Medical Microbiology* (4th ed.). St. Louis: Mosby.
[56] Mylopoulos, J. (1992). Conceptual modelling and Telos. *Conceptual Modeling, Databases, and Case An integrated view of information systems development.*, 49–68.
[57] Nardi, J.C., Falbo, R.d.A., Almeida, J.P.A., Guizzardi, G., Pires, L.F., van Sinderen, M.J., Guarino, N., de Almeida Falbo, R., Almeida, J.P.A., Guizzardi, G., Pires, L.F., van Sinderen, M.J. & Guarino, N. (2013). Towards a Commitment-Based Reference Ontology for Services. In *2013 17th IEEE International Enterprise Distributed Object Computing Conference* (pp. 175–184). IEEE. doi:10.1109/EDOC.2013.28. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6658277.
[58] Saleem, M., Mehmood, Q. & Ngonga Ngomo, A.-C. (2015). FEASIBLE: A Featured-Based SPARQL Benchmark Generation Framework. doi:10.1007/978-3-319-25007-6_4.
[59] Schmidt, M., Schallhorn, T., Lausen, G. & Pinkel, C. (2009). SP2Bench: A SPARQL performance benchmark (pp. 222–233). doi:10.1109/ICDE.2009.28.
[60] van der Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C., Guerson, J., Almeida, J.P.A. & Guizzardi, G. Support for Domain Constraints in the Validation of Ontologically Well-Founded Conceptual Models, 302–316.
[61] Vargas-Vera, M. & Motta, E. (2004). AQUA - Ontology-based question answering system. *Micai 2004: Advances in Artificial Intelligence*, **2972**, 468–477.
[62] Vila, H. Lluis; Reichgelt (1996). The token reification approach to temporal reasoning.
[63] von Malottki (2008). Architecture for OWL Binding (JAOB),.
[64] Wilkinson, K., Sayers, C., Kuno, H. & Reynolds, D. (2003). Efficient RDF storage and retrieval in Jena2. *Proceedings 1th International Workshop on Semantic Web and Databases*, 35–43. doi:citeulike-article-id:926609. https://www.cs.uic.edu/{~}ifc/SWDB/papers/Wilkinson{_}etal.pdf.
[65] Wilson, E. (1991). Active vibration analysis of thin-walled beams. PhD thesis, University of Virginia.
[66] Zamborlini, V. & Guizzardi, G. (2010). On the representation of temporally changing information in OWL. In *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC* (pp. 283–292).
[67] (2012). OWL 2 Web Ontology Language Document Overview. Technical report December, W3C Consoritum. http://www.w3.org/TR/owl2-overview/.

Appendix A.
The Experiments Results

We present here the results in numbers (mean value and standard deviation of execution query time) of the benchmark experiments on triples stores:

Table 4
Mean Value ϕ of Execution Query Time for *200000* triples

	Q1	**Q2**	**Q3**
RDF4J	413ms	431ms	405 ms
JenaTDB	431ms	521ms	409 ms
GrapgDB	420ms	470ms	401 ms

Table 5
Standard Deviation σ of Execution Query Time for *200000* Triples

	Q1	**Q2**	**Q3**
RDF4J	14 ms	13 ms	7 ms
FusekiJena	21ms	18ms	20 ms
GrapfDB	13ms	16ms	11 ms

Table 6
Mean Value ϕ of Execution Query Time for *500000* triples

	Q1	**Q2**	**Q3**
RDF4J	438ms	444ms	393 ms
FusekiJena	582ms	649ms	585 ms
GraphDB	565 ms	472ms	378 ms

Table 7
Standard Deviation σ of Execution Query Time for *500000* Triples

	Q1	**Q2**	**Q3**
RDF4J	16 ms	13 ms	8 ms
FusekiJena	27ms	ms 27	24 ms
GrapfDB	16 ms	16 ms	19 ms

Table 8
Mean Value ϕ of Execution Query Time for *Million* triples

	Q1	**Q2**	**Q3**
RDF4J	449ms	576ms	536 ms
FusekiJena	671ms	650ms	652 ms
GrapfDB	628ms	693ms	459 ms

Table 9
Standard Deviation σ of Execution Query Time for *Million* triples

	Q1	**Q2**	**Q3**
RDF4J	11 ms	12 ms	8 ms
FusekiJena	24 ms	23ms	32 ms
GrapfDB	18 ms	11 ms	20 ms

Table 10
Mean Value ϕ of Execution Query Time for Real Data

	Q1	Q2	Q3	Q4	Q5
RDF4J	408ms	396ms	364 ms	393 ms	480 ms
FusekiJena	480ms	460ms	445 ms	425 ms	510 ms
GraphDB	412ms	401ms	397 ms	389 ms	-

Table 11
Standard Deviation σ of Execution Query Time on Real Data

	Q1	Q2	Q3	Q4	Q5
RDF4J	4 ms	6 ms	8 ms	12 ms	17 ms
FusekiJena	34ms	9ms	34 ms	47 ms	48 ms
GraphDB	3ms	10ms	12 ms	12 ms	-

Table 12
Mean Value ϕ of Execution Query Time with and without UFO s

	Q1 (With UFO / Without UFO)	Q2 Execution Query Time with and without UFO	Q3 Execution Query Time with and without UFO	Q4 Execution Query Time with and without UFO
RDF4J	408/778 ms	396/772 ms	364/688 ms	393/685 ms
FusekiJena	480/803ms	460/811ms	445/782 ms	425/763 ms
GraphDB	412/772 ms	401/785 ms	397/358/692 ms	389/701 ms

Table 13
Standard Deviation σ of Execution Query Time with and without UFO s

	Q1 (With UFO / Without UFO)	Q2 Execution Query Time with and without UFO	Q3 Execution Query Time with and without UFO	Q4 Execution Query Time with and without UFO
RDF4J	4/32 ms	6/29 ms	8/57 ms	12/42 ms
FusekiJena	34/413ms	9/54s	34/66 ms	47/61 ms
GraphDB	3/31 ms	10/49 ms	12/44 ms	12/39 ms

Table 14
Mean Value ϕ of Execution Query Time for both Data Sets

	Q1 (generated / real)	Q2 (generated / real)	Q3 (generated / real)
RDF4J	453/322 ms	472/375 ms	432/344 ms
FusekiJena	491/399ms	521/414ms	530/401 ms
GraphDB	460/377 ms	498/320 ms	501/358 ms

Table 15
Standard Deviation σ of Execution Query Time for both Data Sets

	Q1 (generated / real)	Q2 (generated / real)	Q3 (generated / real)
RDF4J	13 / 5 ms	20/12 ms	14/10 ms
FusekiJena	44/33ms	23/12ms	36/32 ms
GraphDB	21/6 ms	32/21ms	36/17 ms