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. In this paper, we show how triple store data compliant with the Unified Foundational Ontology (UFO) can be efficiently queried. We also present foundational patterns benchmark that helps us choosing the most efficient triple store and its layout. We evaluate the foundational benchmark on both generated and real-world datasets for state-of-art triple stores.

Keywords: Foundational Pattern, UFO, SPARQL

1. Introduction

Recently, there has been a growing interest in using ontologies as a tool 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 answering user queries in computer science, conceptual modeling has become widespread in the context of cognitive science [15, 41]. 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 [41]. While the descriptions that arise from conceptual modeling activities were originally intended to be used by humans, not machines, recent works show how to create computable conceptual models, e.g. [34], that can be published as linked data and used as a source of truth for data modelling tasks.

While the aim of a conceptual model is to express the meaning of concepts used in a specific domain to discuss the problem and to find appropriate relationships between different concepts, Web Ontology Language (OWL 2) [53] is too low-level to capture key modeling decisions like concept rigidity (Student vs. Person). This might lead to semantically incorrect representation (yet consistent in OWL 2 sense) ending up in irrelevant query results. For example, events, objects, or their properties are modeled in each domain ontology differently. 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.

In this paper, we show what is the benefit of using foundational ontologies for improving efficiency of domain queries on existing triple stores. For this purpose, we stick to Unified Foundational Ontology (UFO) [16] among other foundational ontologies because of (i) our experience with using UFO in various domains [32, 35], (ii) maturity of UFO for conceptual modeling [15] using OntoUML, a formal UFO-based modeling language based on UML.

1.1. Scenarios

We extend our previous work [3] on a new Resource Description Framework (RDF) [31] indexing

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

approach based on UFO. The different querying scenarios are depicted in Fig. 1. The leftmost diagram shows standard querying scenario – a knowledge base (data) compliant with a domain ontology are queried by users. Typically, large knowledge bases are stored inside triple stores and the user queries are formulated in SPARQL[30]. The middle diagram shows one of the benefits of using foundational ontologies – it unifies the types of queries to be executed on data. Here, the types of queries are expressed by *UFO query patterns*, which are then instantiated as user queries in terms of domain ontologies. Executing queries compliant with these query patterns then can be optimized by the triple store.

The rightmost diagram shows the scenario we use in this paper – we define a benchmark, consisting of a set of UFO Query Patterns and UFO-based data generator. Based on the created foundational-based domain ontology, the generator provides instances of the ontology, stores the generated data in triple stores, indexes the stored data with foundational index and evaluates these data on generated foundational patterns. This benchmark can be reused not only for our foundational generated data but also for all datasets compliant with the unified foundational ontology such as INBAS[1], STAMP Hazard and Risk Ontology[2] and TermIT[3].

As a result, the main goal of this benchmark is to provide insight into how performing existing triple stores are w.r.t. recurring patterns of SPARQL queries posed to ontology-backed large datasets. For this purpose, we analysed recurring patterns of queries based on the foundational ontology (e.g., Who participate in an event? This event can be marriage, party, football game, business operations, etc) which are then used to evaluate the performance of query executions on UFO-indexed data in triple stores.

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

2. Related Work

Recently, different benchmarks have been proposed to compare a query execution performance for triple stores. These benchmarks focus on different aspects of SPARQL query evaluations. E.g. a comprehensive overview of the main features of benchmarks is shown in [44]. Authors describe a benchmark as a combination of a dataset, a set of queries and a set of performance metrics and evaluate existing SPARQL benchmarks accordingly. We select here only some of the representative SPARQL benchmarks together with their main characteristics.

FEASIBLE [43] is a cluster-based SPARQL benchmark generator able to synthesize customized benchmarks from the query logs of SPARQL endpoints. The *Berlin SPARQL Benchmark (BSBM)* [7] 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)* [28] is a widely used benchmark for comparing the performance, completeness and soundness of OWL reasoning engines. It is based on a customizable deterministic generator of synthetic OWL data based on the Univ-Bench ontology – an ontology of the university domain that includes universities, their departments, their professors, employees, courses, publications and their relations in the OWL language and offers necessary features for evaluation purposes. 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)[4] is a more expressive new version of LUBM involving additional OWL constructs (like disjunctive axioms, or negation). The *DBpedia SPARQL Benchmark DBPSB* [39] 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 benchmark queries

[1] https://www.inbas.cz/
[2] http://onto.fel.cvut.cz/ontologies/stamp-hazard-profile/
[3] https://kbss.felk.cvut.cz/termit-demo/
[4] https://www.cs.ox.ac.uk/isg/tools/UOBMGenerator/

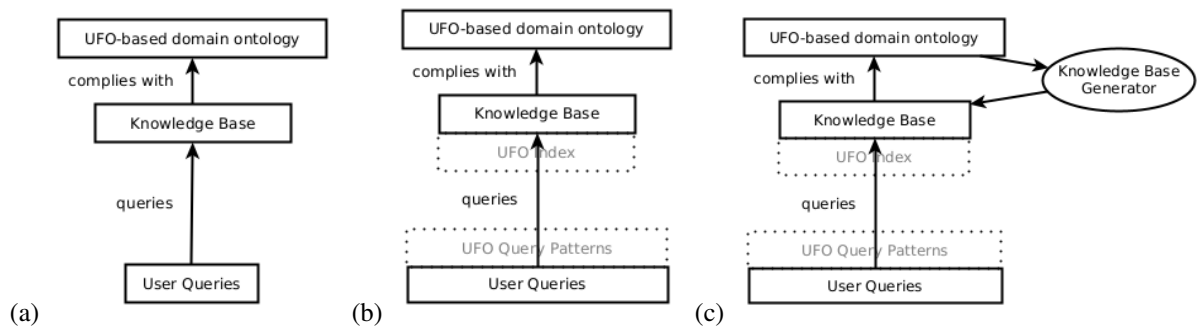

Fig. 1. (a) querying scenario (b) UFO-based querying scenario (c) UFO-based querying of generated data

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)* [45] 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. The aforementioned benchmarks focus on different characteristics and performance of SPARQL evaluation scenario (size of data, expressiveness of ontology, diversity of queries). However, no attention is paid to the shape of data. For example, although most top-level ontologies distinguish between objects and events, these benchmarks are agnostic to this distinction and do not reflect top-level semantics of data and queries. Although a triple store with UFO index might perform poorly e.g. on star graph patterns in general, particular graph patterns with specific ontological types of nodes (e.g. object, or event) might be more efficient. We study this difference in this paper and propose a benchmark of foundational patterns that are generated using Unified foundational ontology (UFO) [16]. We claim that a top-level ontology gives a common and recurring shape to both data and queries – our benchmark shows how an index based on OWL representation of UFO[5] implemented in various triple stores can help making SPARQL evaluation more efficient over such data.

3. Background

First, we introduce a fragment of OWL 2-DL [53] 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. $\mathsf{Person}(\mathsf{Frank})$.
- an object property assertion $\mathsf{P}(\mathsf{a}, \mathsf{b})$, saying that "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, $(\exists hasChild \cdot Man)$ 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,

[5] http://onto.fel.cvut.cz/ontologies/ufo

– 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 [53].

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 named 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 [33]. 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 $Q(?x_1, ?x_2) = \mathsf{object}(?x_1), \mathsf{performs}(?x_1, ?x_2)$ 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 (agent $\sqsubseteq$ 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 [16].

Based on Description Logic formalization of basic UFO concepts introduced in [4] we publish Unified Foundational Ontology with dereferencable identifiers in OWL at . 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) [15], perdurants/events (UFO-B) [24] and social agents (UFO-C) built on top of UFO-A and UFO-B [19]. 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 (endurant $\sqsubseteq$ Individual), (perdurant $\sqsubseteq$ individual). Endurants can be observed as complete concepts in a given time snapshot, and they can be any object (e.g. an agent, aircraft) (object $\sqsubseteq$ endurant), or its tropes or tropes (e.g. speed, location, colors, etc.) (trope $\sqsubseteq$ endurant), that exist as long as an object they inhere in exists (trope $\sqsubseteq$ (= 1 inheres-in·object)), and situations (situation $\sqsubseteq$ endurant).

Perdurants only partially exist in a given time snapshot. They involve events (event $\sqsubseteq$ perdurant) and object snapshots (object-snapshot $\sqsubseteq$ perdurant).

Events happen in time and cannot undergo non-relational changes, e.g., death can't die. They can be either atomic or complex (event $\sqsubseteq$ (atomic-event $\sqcup$ complex-event)). complex events have temporal branchings, occurring over incomparable time-points, and have participants (event $\sqsubseteq$ ($\geqslant$ 1 has-participant $\cdot$ object)) and complex events have parts ($\exists$ has-event-part $\cdot \top \sqsubseteq$ complex-event) [24]. 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 [22]. object-snapshot 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 Tropes (intrinsic-trope $\sqsubseteq$ Trope), i.e. existentially dependent entities that are realizable through the occurrence of an event (disposition $\sqsubseteq$ trope). This occurrence brings about a situation [21]. 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 (disposition $\sqsubseteq$ ($=$ 1 is-manifested-by·event)). Dispositions inhere in particular objects (disposition $\sqsubseteq$ ($=$ 1 inheres-in·object)). For example, a security flaw in an information system is manifested by the event of stealing sensitive data that brings about unsafe situation.

Additionally, UFO introduces the notion of agents (agent $\sqsubseteq$ endurant), i.e. proactive objects with an intention, the propositional content of intention is a goal. Intentions cause the agent to perform actions ($\exists$ performs $\cdot \top \sqsubseteq$ object) [17]. Finally, UFO also defines services [12], and powertypes, i.e. universal types whose instances are individuals in the subject domain [8, 25].

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) [48], But [5] 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 [4] pro-

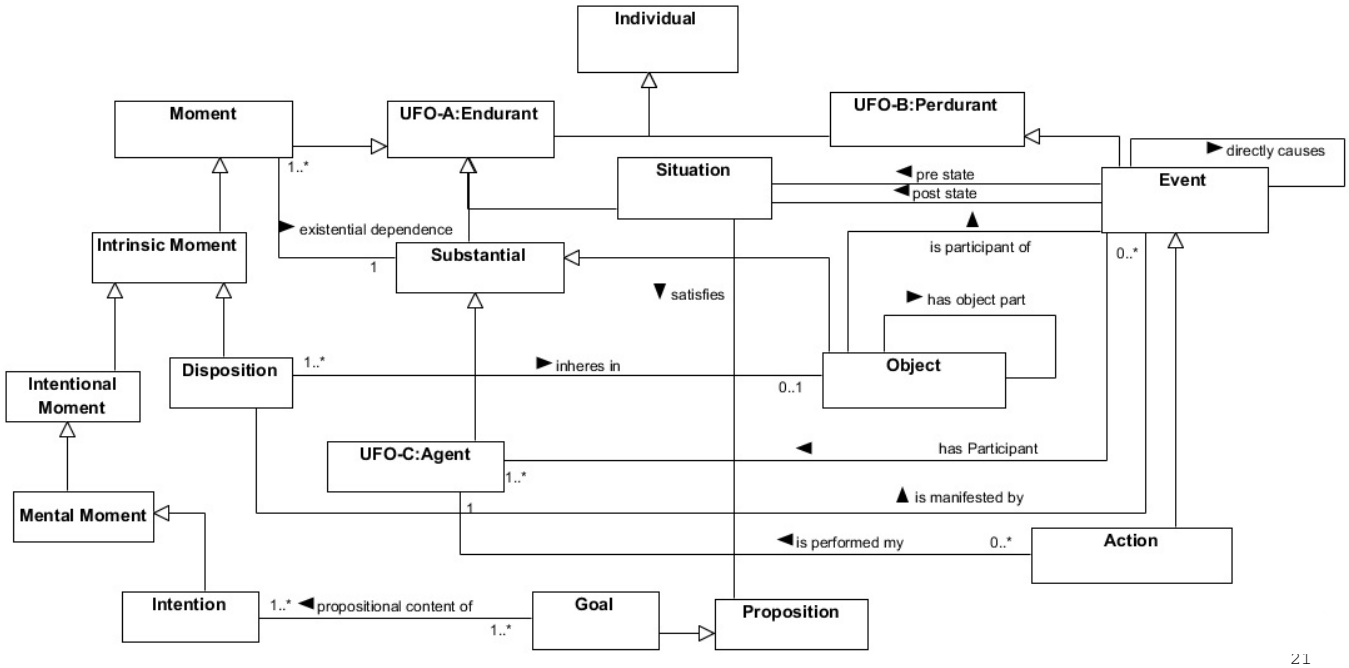

Fig. 2. Main concepts of UFO.

posed 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 [26].

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 2. 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 [30], 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*[6] C.

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

Having an OWL ontology $\mathcal{O}$, its RDF serialization is given directly by OWL specification [53]. For distinguished conjunctive queries, we translate each atom of the form $\mathsf{A}(y)$ into an RDF triple pattern $(y \quad \text{rdf:type} \quad \mathsf{A})$[7] and each atom of the form $\mathsf{P}(y, z)$ into an RDF triple pattern $(y \quad \mathsf{P} \quad z)$. 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[8]

```
: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 .
```

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

[8]We use the prefix "ufo:" to denote the namespace <http://onto.fel.cvut.cz/ontologies/ufo/>, thus a translation of agent into its RDF representation would be an IRI <http://onto.fel.cvut.cz/ontologies/ufo/agent>.

4. Data Generator

UFO-based Data Generator (UDG)[9] 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.

4.1. 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 exist 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.
- Person or Agent which is a 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 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 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.

4.2. 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 [36]. It is used here to create instances of the model entities, i.e., create an object graph and then persist it into a 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;

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

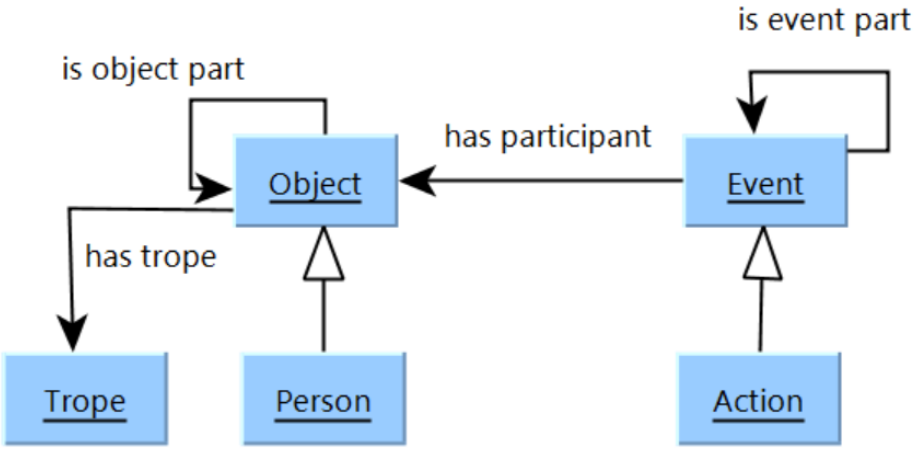

Fig. 3. UFO Model Entities

- 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.

4.3. Validator

The previous sections showed how the synthetic UFO-compliant RDF data can be generated. In order to use our benchmark for existing real-world datasets, we provide a validator based on SHACL [54] rules. The validator checks for cardinality and domain/range constraints of RDF data on the input to check whether they are usable for the benchmark queries presented in section 5. For example, the RDF graph

```
aviation-safety:number-of-engines-of
    -aircraft-123
  a ufo:intrinsic-trope .
```

will not pass validation due to missing inheres-in link to the respective aircraft instance. While the RDF graph

```
aviation-safety:number-of-engines-of
    -aircraft-123
  a ufo:intrinsic-trope .
aviation-safety:number-of-engines-of
    -aircraft-123
  ufo:inheres-in aviation-safety:
    aircraft-123 .
```

will pass. The validator can be found at https://github.com/kbss-cvut/ufo-validator, together with example usage.

4.4. 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[10] queries by using UFO-based indexing techniques. Note, we use our UFO index not only for generated data but also, for all UFO grounded datasets. 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, 10]; one for Perdurant category and the other for Endurant.
- UFO Property Table that builds a UFO property table for endurants and another table for perdu-

[10] 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/.

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

rants, that will reduce Null values in each property table [1, 10], 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, 29].

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.

5. Foundational Patterns Benchmark

After having the data, the job now is to generate a benchmark of queries the user 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, flight, war, sales, etc.), i.e., immaterial entities that only exist in the mind of the user or a community of users of a language [20]. For example, some people search for individuals who attended Celine Dion birthday events. Therefore, they are looking for concepts such as invited, going, or attended. Another 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 (event)?

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 [23]. Moreover, UFO-B suggests a discrete linear ordering of time points to answer question word *("when")* [5].

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, i.e., the patterns based on UFO concepts and entities 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 table 3 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 have Perdurant or Endurant as its domain or range and vice-versa.

Query 1, selects all relations from Endurant table that 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?

```
PREFIX ufo: <http://onto.fel.cvut.cz
    /ontologies/ufo/>
SELECT DISTINCT ?term FROM NAMED ufo
    :endurant
```

Table 3
Foundational query patterns and their formal representations.

	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:has-pre-situation$(?s, e1)$
P_7	What is the resulting situation of a given event?	$(?s) \rightarrow$ ufo:has-post-situation$(?s, e1)$
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?	$(?e1) \rightarrow$ 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$(?s, d1)$

```
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.

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

Listing 2: SPARQL query

However, the generic generated patterns in table 3 are simple and mostly contain a single triple pattern, we combine foundational patterns to generate more complex pattern and more numbers of triples, we selected these patterns from our previous experiences of UFO based systems such as INBAS [11], STAMP Hazard and Risk Ontology[12] and TermIT[13]. The extended patterns are in table 4

5.1. Description of the Benchmark Patterns

In this section, we describe the created foundational patterns. As we mentioned in section 3.2, UFO mainly 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 3, we don't describe here the extended patterns because they consist of the generic ones (combined of foundational patterns):

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

[11] https://www.inbas.cz/

[12] http://onto.fel.cvut.cz/ontologies/stamp-hazard-profile/

[13] https://kbss.felk.cvut.cz/termit-demo/

Table 4

Extended Foundational query patterns and their formal representations.

	Patterns	Pattern formalization
P'_1	What are the properties of an object and in which event participates?	$(?p, ?e) \leftarrow$ ufo:has-trope$(?p, o1)$, ufo:participates-in(?e,o1)
P'_2	What are the actions that a given agent performs and what are the properties of this agent?	$(?a, ?p) \leftarrow$ ufo:performs$(?a, a1)$, ufo:has-trope(?p,a1)
P'_2	What are the actions that a given agent performs and what are the properties of this agent?	$(?a, ?p) \leftarrow$ ufo:performs$(?a, a1)$, ufo:has-trope(?p,a1)
P'_3	What are the parts of an object with their properties?	$(?o2, ?p) \leftarrow$ ufo:has-object-part$(?o2, o1)$, ufo:has-trope(?p,o1)
P'_4	What are factors of an event and in which object these factors inhere?	$(?f, ?o) \leftarrow$ ufo:is-manifestation-of$(?f, e1)$ ufo:inheres(?o,?f)
P'_5	What are the pre and post situations of a given event?	$(?s1, ?s2) \leftarrow$ ufo:pre-state$(?s1, e1)$, ufo:post-state$(?s2, e1)$
P'_6	What are the parts of an object and in which events participate this object?	$(?a, ?p) \leftarrow$ ufo:performs$(?a, a1)$, ufo:has-trope(?p,a1)

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 is-object-part-of·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 their parts. According to UFO every complex event consists of parts which accumulate together to have the end event (complex-event $\sqsubseteq$ ($\geqslant$ 2 has-event-part·event). E,g., What are all temporal precedence involved in an event?

How is an event triggered? An event occurs in a certain situation at a certain point in time, and transforms it into another situation, they may change reality by changing the state of affairs from one (pre-state) situation ufo:has-pre-situation to a (post-state) situation ufo:has-post-situation [22]. An event *ufo:brings-about* exactly one situation (event(e) $\rightarrow \exists$!s(brings-about(e,s))), which holds in all *end points* of the event. Also, an event is triggered by exactly one situation (triggers(s, e) $\rightarrow$ situation(s) event(e)), which holds in all *begin points* of the event, e.g. The event car's Accident brings-about 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 the 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 ($\exists$is-manifested-by $\cdot\top \sqsubseteq$ (disposition)) [21]. When manifested, they are manifested through the occurrence of resulting events and state changes (is-manifested-by), 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 performs, e.g., a doctor performs surgery operations in order to satisfy his intentions in saving peoples' lives.

Table 5.1 contains the SPARQL representations of the benchmark queries.

SPARQL Representations Of the Foundational Queries[14]

Query 1: who participates in an event ?

```
SELECT DISTINCT ?participations
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {
  benchmark:givenEvent
  ufo:has-participant ?
      participations}
```

[14]Initial prefix declaration `PREFIX ufo: <http://onto.fel.cvut.cz/ontologies/ufo/>` is ommited from the queries.

```
}
```

Query 2: What are the object's parts?

```
SELECT DISTINCT ?parts
FROM NAMED 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 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 ufo:perdurant
WHERE {GRAPH ?g {benchmark:
   givenEvent ufo:bringsAbout ?
   situation} }
```

Query 5: how an event is triggered?

```
SELECT DISTINCT ?situation
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {?situation ufo:
   triggers benchmark:givenEvent }
   }
```

Query 6: What does a specific situation activate?

```
SELECT DISTINCT ?disposition
FROM NAMED 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 ufo:perdurant
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:is-manifestation-
   of ?disposition } }
```

Query 8: What does an agent perform?

```
SELECT DISTINCT ?Action
FROM NAMED 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 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 ufo:perdurant
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:has-pre-situation
    ?situation } }
```

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

```
SELECT DISTINCT ?situation
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g { benchmark:
   givenEvent ufo:has-post-
   situation ?situation } }
```

Query 12: when did a given event start?

```
SELECT DISTINCT ?point
FROM NAMED 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 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 ufo:endurant
WHERE {GRAPH ?g { benchmark:
   givenProperty ufo:inheres-in ?
   entity} }
```

Query 15: What is directly cause a given event?

```
SELECT DISTINCT ?event
FROM NAMED 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 ufo:perdurant
WHERE {GRAPH ?g { ?agent ufo:
   performs ?action } }
```

6. 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 four popular RDF stores (Sesame (or RDF4J[15]), Fuseki Jena with JenaTDB, GraphDB and Virtuso.

Sesame (or RDF4J[16]): 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 [17]: 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.

GraphDB free[18]: 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.

Virtuoso [19] Open-Source Edition v7.02 with Tomcat Version 8.0.48, Operating System Windows10 10.0 (amd64).

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 ten times against the different triple stores.

We run the queries in different datasets 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 wrt real data only. Each query will be executed 10 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.

6.1. SPARQL Features Selection

To use the foundational benchmark on different datasets by running multiple queries against triple stores, we select a number of frequently executed queries based on domain experts in real data and on our previous experience of developing UFO based systems. We selected queries that cover most SPARQL features that allow us to assess the performance of foundational queries with SPARQL features. Note, all the executed queries are instances of our foundational patterns. The SPARQL features we consider are:

[15] http://rdf4j.org/

[19] https://virtuoso.openlinksw.com/

- 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)
- aggregates such as (COUNT, HAVING and GROUP BY).

We used the methodology proposed in [44] to analyse the characteristics of the proposed benchmark. The diversity score of our benchmark queries is approx 1.6, which is rather a low value, similarly to synthetic benchmarks presented in the comparison. This is not surprising as our benchmark focuses on studying evaluation of rather simple foundational queries.

6.2. 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 , million and 10 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 ?trope1 ?trope2
FROM NAMED
ufo:endurant
WHERE {GRAPH ?g {{benchmark:Person
    -1000344628 ufo:has-trope ?
    trope1}
UNION {benchmark:Person-1009237217
    ufo:has-trope ?trope2}}}
```

Q2: Select the number and all participants of all events. (Instance of P2)

```
SELECT (count (?participants) as ?
    num) ?participants FROM NAMED
    ufo:perdurant
WHERE {GRAPH ?g {{?event ufo:has-
    participant participants}
OPTIONAL {participants rdf:label ?
    label.}
}} GROUP BY participants
```

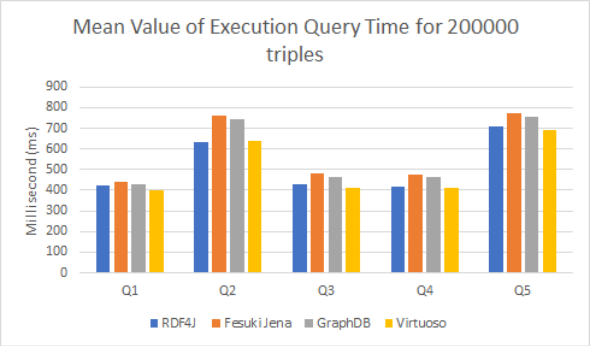

Fig. 4. Mean Value ϕ of Execution Query Time for *200000* triples

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

```
SELECT DISTINCT ?part
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {benchmark:
    Event1670269156 ufo:has-part ?
    part} }
```

Q4: Select all actions that have the agent (Person-444525854) as a participant with all properties of this agent

```
SELECT DISTINCT ?trope ?action
FROM NAMED ufo:perdurant
FROM NAMED ufo:endurant
WHERE { GRAPH ?graph{ {?action ufo:
    has-participant benchmark:Person
    -444525854}
UNION{
benchmark:Person-444525854 ufo:has-
    trope ?trope}}}
```

Q5: Count all agents in the dataset.

```
  SELECT (count (?agents) as ?num)
FROM NAMED ufo:endurant
WHERE { GRAPH ?graph { {?agents rdf:
    type ufo:agent}}
```

Figures 4, 6, 8, 5, 7,9, 10 and 11 show the results of running execution time of the instantiated queries on the different triple stores. Figure 12 shows how the query (Q1) execution time react wrt different generated dataset size on the selected triple stores.

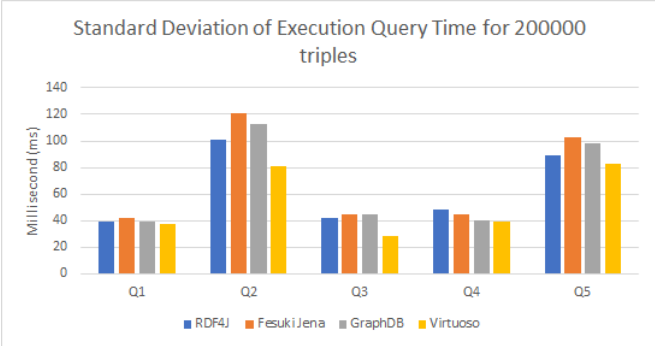

Fig. 5. Standard Deviation σ of Execution Query Time for *200000* Triples

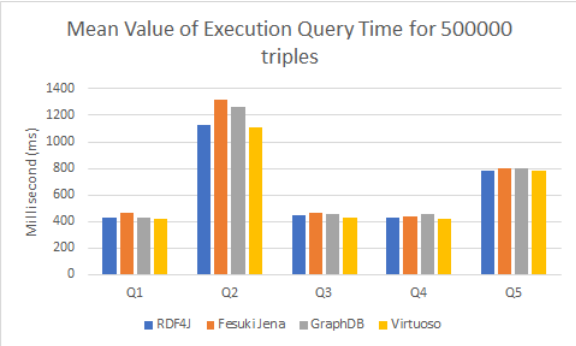

Fig. 6. Mean Value ϕ of Execution Query Time for *500000* triples

6.3. *Real World datasets Experiments: Aviation Safety dataset*

The ontology that we used to evaluate the benchmark is the Aviation Safety ontology. We designed the Aviation Safety Ontology[20] 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 [32]. 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) 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 [32]. The on-

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

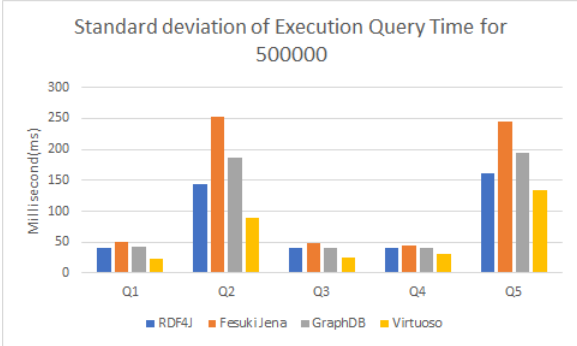

Fig. 7. Standard Deviation σ of Execution Query Time for *500000* Triples

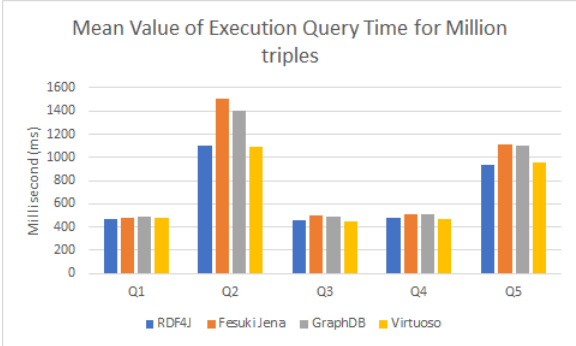

Fig. 8. Mean Value ϕ of Execution Query Time for *million* triples

tology consists of *19421* axioms, *6895* logical axioms, *1725* classes, *129* Object Properties and *2172* individuals (instances). We built Aviation Safety Ontology on top of the Unified Foundational Ontology (UFO)[21] [15, 41]. Figure 13 depicts basic concepts in Aviation Safety Ontology that is represented in UFO. During the ontology development for the requirements election step, we constructed the following competency questions (CQs) that the ontology should answer, the competency questions were derived from an interview of domain users and experts. Also, we map the CQs to the foundational patterns from table 3 to defend that user queries can be generalized to our foundational pattern.

- CQ1: What are the proprieties or qualifications of the safety agents (e.g., the air traffic control agents) (maps P1)

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

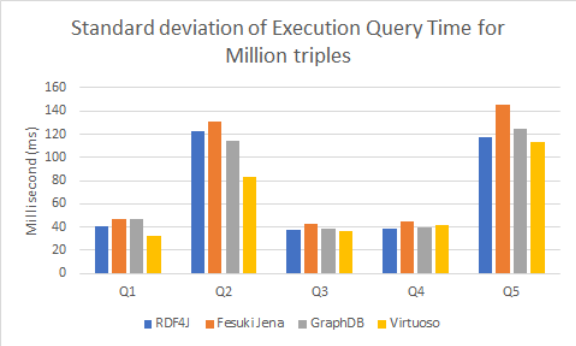

Fig. 9. Standard Deviation σ of Execution Query Time for *million* Triples

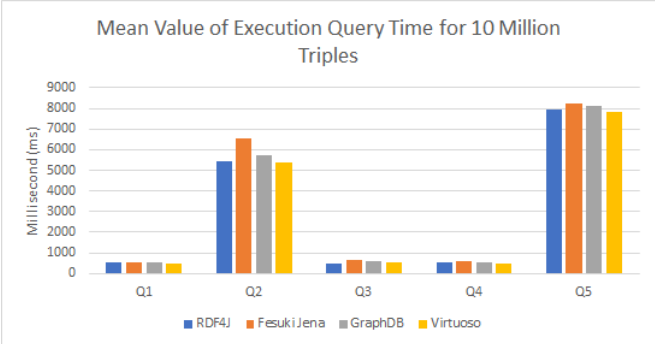

Fig. 10. Mean Value ϕ of Execution Query Time for *10 million* triples

- CQ2: Which people do necessarily participate in a particular event (e.g., specific damage event, a runway incursion)? (maps P2)
- CQ3: Which safety events happen during a particular flight phase (e.g. take-off)?? (maps P4)
- CQ4: Who is responsible for a particular safety operation (e.g., ground handling)? (maps P8)
- CQ5 What are the parts of an Aircraft? (maps P6)

.

For evaluation, we answer the domain user competency questions by using foundational patterns. We ran the following queries against selected triple stores on the instances of user competency questions . The dataset consists of *25000* triples.

- Q1: What are the tropes (properties) that inhere in the air traffic control agent? (instance of P1 and CQ1)

```
SELECT DISTINCT ?trope
```

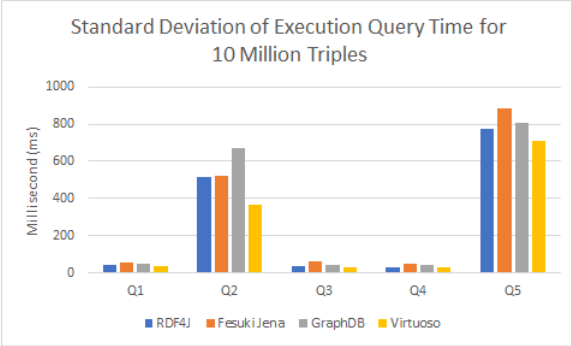

Fig. 11. Standard Deviation σ of Execution Query Time for *10 million* Triples

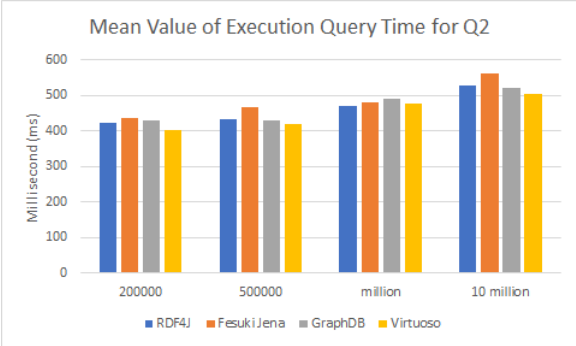

Fig. 12. Mean Value ϕ of Execution Query Time for Q2 on Different Dataset Size

```
FROM NAMED ufo:endurant
WHERE {GRAPH ?g {{ ?trope ufo:
    inheres-in aviation-safety:
    air_traffic_control_agent}
}}
```

- Q2: what are the Participants of a Damage event? (instance of P2 and and CQ2)

```
SELECT DISTINCT ?Participants
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {aviation-safety
    :Damage_manifestation ufo:has
    -participant ?Participants} }
```

- Q3: What are the parts of a specific Flight? (instance of P4 and and CQ3)

```
SELECT DISTINCT ?part
```

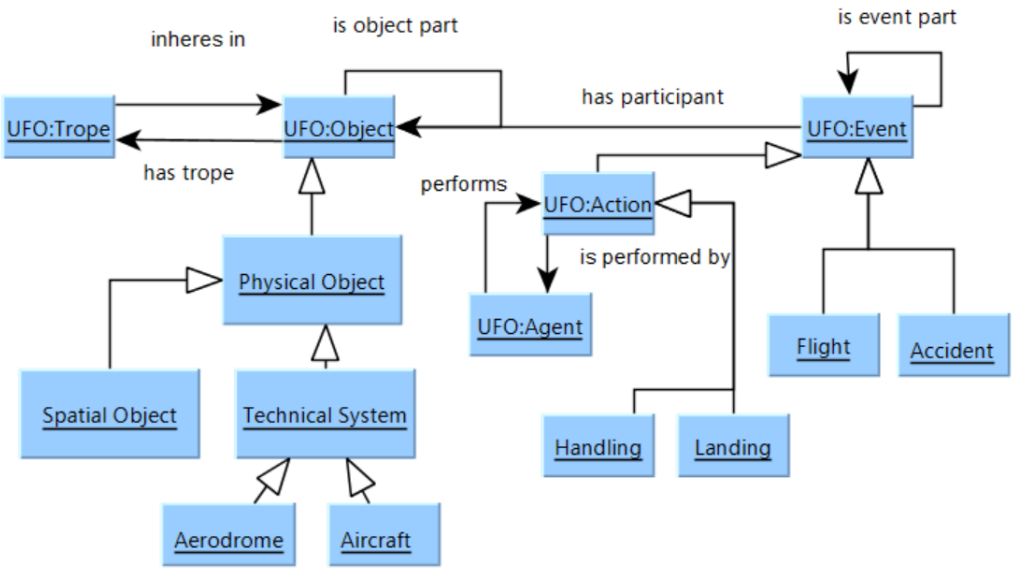

Fig. 13. Aviation Safety Ontology

```
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {?part ufo:is-
    part-of aviation-safety:Fligt
    -i} }
```

– Q4: Who performs Ground handling operation-i? (instance of P8 and CQ4)

```
SELECT DISTINCT ?agent
FROM NAMED ufo:endurant
WHERE {GRAPH ?g {?agent ufo:
    performs aviation-safety:
    Ground_handling_operation-i}
    }
```

– Q5: Select everyone that performs actions in aviation domain and filter all participating relation? (instance of P8,P2 and CQ2,CQ4)

```
SELECT ?term
FROM NAMED ufo:perdurant
 WHERE {GRAPH ?g {
 ?action ufo:is-performed-by ?
     agent
FILTER (
   NOT EXISTS {
     ?action ufo:has-participant ?
         agent
        }
  )
    }}
```

– Q6: What are the parts of aircraft-i and in which event this object participates? (instance of P'6 and CQ5)

```
SELECT DISTINCT ?part ?event
FROM NAMED ufo:endurant
WHERE {GRAPH ?g {?part ufo:is-
    object-part-of aviation-
    safety:aircraft-i.
 aviation-safety:Aircraft-i ufo:
     is-participant-of ?event
} }
```

The result are presented in the figures 14 and 15.

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 16 and 17, the left set of bars is with UFO index, while the right bars are without UFO index. These figures indicate that using UFO-indexing approach makes the

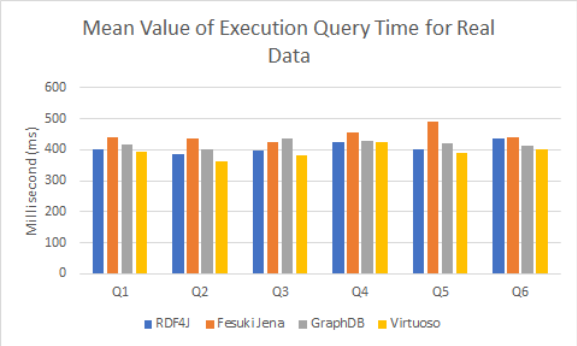

Fig. 14. Mean Value ϕ of Execution Query Time for *Real Data*

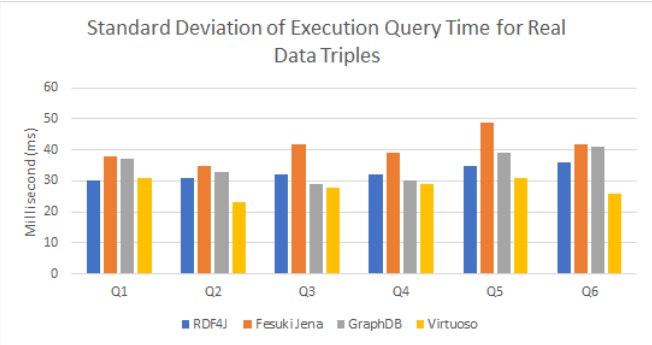

Fig. 15. Standard Deviation σ of Execution Query Time for *Real Data*

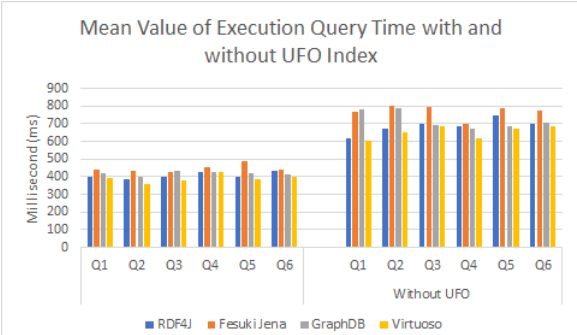

Fig. 16. Mean Value ϕ of Execution Query Time with and without UFO Index

search process easier and faster, as we demonstrated in [3].

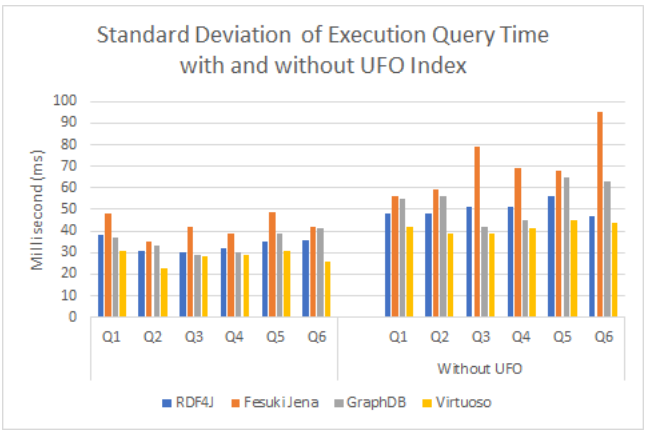

Fig. 17. Standard Deviation σ of Execution Query with and without UFO Index

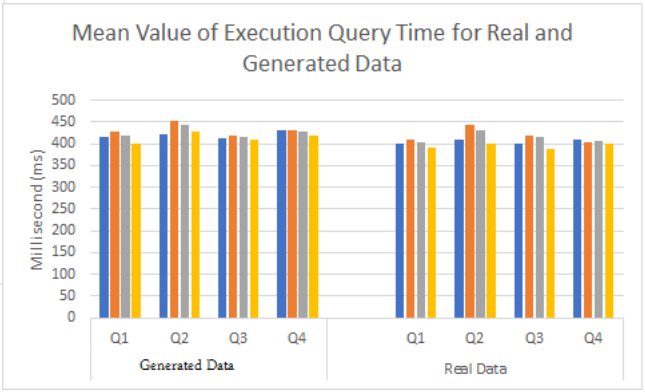

Fig. 18. Mean Value ϕ of Execution Query Time on Generated vs Real Data

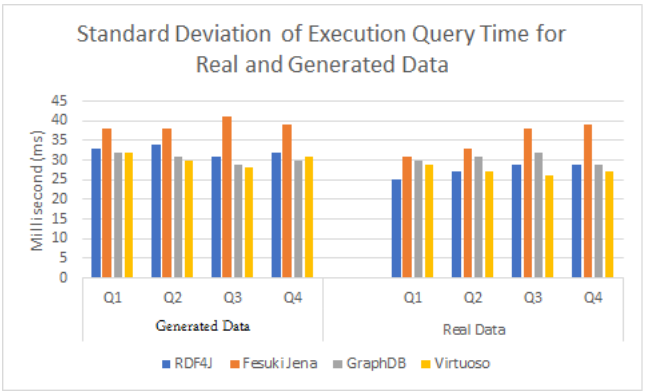

Fig. 19. Standard Deviation σ of Execution Query on n Generated vs Real Data

6.4. 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 dataset based on a unified foundational ontology, i.e., we can run these fondational patterns for different UFO based datasets.

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

```
SELECT DISTINCT ?object ?event
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {?event ufo:has-
    participant ?object} }
```

- Q2: What are all *properties* in each dataset and in which entity inhere?

```
SELECT DISTINCT ?trope ?entity
FROM NAMED ufo:endurant
WHERE {GRAPH ?g {?entity ufo:has
    -trope ?trope} }
```

- Q3: What are all *actions* that happened in both datasets with their parts?

```
SELECT DISTINCT ?event ?subEvent
FROM NAMED ufo:perdurant
WHERE {GRAPH ?g {?event ufo:has-
    part ?subEvent} }
```

- Q4: Who are the *participants* in all event in each dataset and what are the properties of every *participant*?

```
SELECT DISTINCT ?object ?event ?
    trope FROM NAMED
ufo:perdurant
WHERE {{GRAPH ?g {?event ufo:has
    -participant ?object} }
UNION{
{GRAPH ?g {?object ufo:has-trope
     ?trope} }
}}
```

Figures 18 and 19 present the mean values and standard deviations of the execution queries time on both datasets.

7. 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 datasets 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 dataset sizes and Virtuoso is better than RDF4J and GraphDB. However, RDF4J (Sesame) is better than GraphDB, taking into the consideration that in many cases, RDF4J is almost equal to GraphDB performance. Moreover, 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 dataset plays an important role in a triple store performance.

Regarding the number of results, all of the selected triple stores return the same number of results. But, the result set size plays an important role on triple stores performance. Also, figure 12 shows that the dataset size affect the performance of triple stores.

Moreover, the results of the real dataset 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 datasets as we demonstrated in section 6.4. However, from the results, we note that the performance of triple stores w.r.t. real word data was somehow better than generated data even they have the same size, that could be because of the result set size.

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 16 and 17. Also, the results indicate that the performance of Fuseki Jena-TDB is the lowest and Virtuoso is the best.

8. 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 datasets 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 will 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 datasets.

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] 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.*.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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$_8$.

[18] 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.

[19] 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$_8$.

[20] Guizzardi, G. & Wagner, G. (2010c). *Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages* (pp. 175–196). doi:10.1007/978-90-481-8847-5$_8$.

[21] 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,).

[22] 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*.

[23] 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$_2$.

[24] 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.

[25] 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*.

[26] 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.

[27] 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.

[28] 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.

[29] 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/.

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

[31] 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/.

[32] 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.

[33] 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.

[34] Kremen, P. & Necaský, M. (2019). Improving discoverability of open government data with rich metadata descriptions using semantic government vocabulary. *J. Web Semant.*, **55**, 1–20. doi:10.1016/j.websem.2018.12.009.

[35] 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.

[36] Ledvinka, M. & Kremen, P. (2015). JOPA: Accessing Ontologies in an Object-oriented Way. In *ICEIS*.

[37] Lefever, E. (2016). A hybrid approach to domain-independent taxonomy learning. *Applied Ontology*, **11**(3), 255–278.

[38] 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.

[39] 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.

[40] Murray, P.R., Rosenthal, K.S., Kobayashi, G.S. & Pfaller, M.A. (2002). *Medical Microbiology* (4th ed.). St. Louis: Mosby.

[41] Mylopoulos, J. (1992). Conceptual modelling and Telos. *Conceptual Modeling, Databases, and Case An integrated view of information systems development.*, 49–68.

[42] 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.

[43] 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$_4$.

[44] Saleem, M., Szárnyas, G., Conrads, F., Bukhari, S.A.C., Mehmood, Q. & Ngonga Ngomo, A.-C. (2019). How Representative Is a SPARQL Benchmark? An Analysis of RDF Triplestore Benchmarks. In *The World Wide Web Conference*. WWW '19 (pp. 1623–1633–). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3308558.3313556.

[45] Schmidt, M., Schallhorn, T., Lausen, G. & Pinkel, C. (2009). SP2Bench: A SPARQL performance benchmark (pp. 222–233). doi:10.1109/ICDE.2009.28.

[46] 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.

[47] Vargas-Vera, M. & Motta, E. (2004). AQUA - Ontology-based question answering system. *Micai 2004: Advances in Artificial Intelligence*, **2972**, 468–477.

[48] Vila, H. Lluis; Reichgelt (1996). The token reification approach to temporal reasoning.

[49] von Malottki (2008). Architecture for OWL Binding (JAOB),.

[50] 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.

[51] Wilson, E. (1991). Active vibration analysis of thin-walled beams. PhD thesis, University of Virginia.

[52] 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).

[53] (2012). OWL 2 Web Ontology Language Document Overview. Technical report December, W3C Consoritum. http://www.w3.org/TR/owl2-overview/.
[54] (2017). Shapes constraint language (SHACL). Technical report, W3C. https://www.w3.org/TR/shacl/.

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 5

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

	Q1	**Q2**	**Q3**	**Q4**	**Q5**
RDF4J	422ms	632ms	428 ms	417 ms	711 ms
JenaTDB	438ms	761ms	479 ms	475 ms	775 ms
GrapgDB	430ms	744ms	464 ms	462 ms	754 ms
Virtouso	402ms	939ms	409 ms	410 ms	689 ms

Table 6

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

	Q1	**Q2**	**Q3**	**Q4**	**Q5**
RDF4J	39ms	101ms	42 ms	48 ms	89 ms
JenaTDB	42ms	121ms	45 ms	45 ms	103 ms
GrapgDB	39ms	113ms	45 ms	40 ms	98 ms
Virtouso	37ms	81ms	29 ms	39 ms	83 ms

Table 7

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

	Q1	**Q2**	**Q3**	**Q4**	**Q5**
RDF4J	433ms	1131ms	445 ms	425 ms	778 ms
JenaTDB	467ms	1321ms	467 ms	435 ms	801 ms
GrapgDB	429ms	470ms	457 ms	454 ms	803 ms
Virtouso	420ms	1110ms	428 ms	420 ms	781 ms

Table 8

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

	Q1	**Q2**	**Q3**	**Q4**	**Q5**
RDF4J	41ms	143ms	40 ms	40 ms	162 ms
JenaTDB	51ms	252ms	49 ms	45 ms	245 ms
GrapgDB	42ms	187ms	41 ms	41 ms	195 ms
Virtouso	24ms	89ms	25 ms	30 ms	134 ms

Table 9

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

	Q1	**Q2**	**Q3**	**Q4**	**Q4**
RDF4J	469ms	1101ms	457 ms	478 ms	935 ms
JenaTDB	481ms	1501ms	501 ms	509 ms	1115 ms
GrapgDB	490ms	1407ms	494 ms	510 ms	1105 ms
Virtouso	476ms	1095ms	453 ms	463 ms	954 ms

Table 10

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

	Q1	**Q2**	**Q3**	**Q4**	**Q4**
RDF4J	41ms	121ms	38 ms	39 ms	117 ms
JenaTDB	47ms	131ms	43 ms	45 ms	145 ms
GrapgDB	47ms	114ms	39 ms	40 ms	125 ms
Virtouso	32ms	83ms	37 ms	42 ms	113 ms

Table 11

Mean Value ϕ of Execution Query Time for *10 Million* triples

	Q1	**Q2**	**Q3**	**Q4**	**Q4**
RDF4J	527ms	5431ms	506 ms	540 ms	7965 ms
JenaTDB	561ms	6551ms	629 ms	605 ms	8243 ms
GrapgDB	520ms	5722ms	594 ms	531 ms	8123 ms
Virtouso	505ms	5368ms	531 ms	477 ms	7814 ms

Table 12

Standard Deviation σ of Execution Query Time for *10 Million* triples

	Q1	**Q2**	**Q3**	**Q4**	**Q4**
RDF4J	43ms	513ms	39 ms	31 ms	773 ms
JenaTDB	54ms	521ms	62 ms	46 ms	886 ms
GrapgDB	46ms	679ms	40 ms	43 ms	809 ms
Virtouso	37ms	368ms	30 ms	29 ms	713 ms

Table 13

Mean Value ϕ of Execution Query Time for Real Data

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
RDF4J	401ms	386ms	397 ms	424 ms	402 ms	437 ms
FusekiJena	440ms	435ms	425 ms	455 ms	490 ms	442 ms
GraphDB	417ms	401ms	437 ms	429 ms	421 ms	412 ms
Virtouso	395ms	361ms	382 ms	425 ms	389 ms	402 ms

Table 14

Standard Deviation σ of Execution Query Time on Real Data

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
RDF4J	30ms	31ms	32 ms	32 ms	35 ms	36 ms
FusekiJena	38ms	35ms	42 ms	39 ms	49 ms	42 ms
GraphDB	37ms	33ms	29 ms	30 ms	39 ms	41 ms
Virtouso	31 ms	21ms	28 ms	29 ms	31 ms	26 ms

Table 15

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

	Q1 (With UFO / Without UFO)	**Q2 with /without UFO**	**Q3 with/ without UFO**	**Q4 with /without UFO**	**Q5 (With UFO / Without UFO)**	**Q6 (With UFO / Without UFO)**
RDF4J	401/616 ms	386/672 ms	397/698 ms	424/685 ms	402/745 ms	437/697 ms
FusekiJena	440/767ms	435/801ms	425/792 ms	453/698 ms	490/785 ms	440/775 ms
GraphDB	412/782 ms	401/685 ms	437/692 ms	428/671 ms	422/685 ms	412/705 ms
Virtuoso	395/602 ms	361/655 ms	383/688ms	425/621 ms	389/675 ms	401/685 ms

Table 16

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

	Q1 (With UFO / Without UFO)	**Q2 with /without UFO**	**Q3 with/ without UFO**	**Q4 with /without UFO**	**Q5 (With UFO / Without UFO)**	**Q6 (With UFO / Without UFO)**
RDF4J	29/48 ms	31/48 ms	32/51 ms	37/51 ms	35/56 ms	36/47 ms
FusekiJena	40/56ms	35/59ms	42/79 ms	45/69 ms	49/68 ms	42/95 ms
GraphDB	41/56 ms	33/55 ms	29/42 ms	29/45 ms	39/65 ms	39/63 ms
Virtuoso	34/52 ms	21/39 ms	28/39ms	32/45 ms	28/45 ms	26/45 ms

Table 17

Mean Value ϕ of Execution Query Time for both datasets

	Q1 (generated / real)	**Q2 (generated / real)**	**Q3 (generated / real)**	**Q4 (generated / real)**
RDF4J	416/399 ms	421/408 ms	412/399 ms	430/410 ms
FusekiJena	427/409ms	452/422ms	420/416 ms	430/402 ms
GraphDB	419/402ms	441/430 ms	417/416 ms	429/407 ms
Virtuoso	401/392ms	428/400 ms	409/389 ms	420/401 ms

Table 18

Standard Deviation σ of Execution Query Time for both datasets

	Q1 (generated / real)	**Q2 (generated / real)**	**Q3 (generated / real)**	**Q4 (generated / real)**
RDF4J	33 /25 ms	34/27 ms	31/23 ms	32/26 ms
FusekiJena	38/31ms	38/29ms	41/38 ms	39/39 ms
GraphDB	32/30 ms	31/31ms	29/32 ms	30/29 ms
Virtuoso	32/29 ms	30/27ms	28/26 ms	31/27 ms