LDBC Semantic Publishing Benchmark (SPB) - v0.2 First Public Draft Release

Coordinator : Venelin Kotsev (Ontotext)
With contributions from: Irini Fundulaki (FORTH),
Vladimir Alexiev (Ontotext)

Abstract

The *Semantic Publishing Benchmark* (SPB) is a LDBC benchmark for RDF database engines inspired by the Media/Publishing industry, particularly by the BBC's Dynamic Semantic Publishing approach. As of June 2014 the benchmark has reached the state of draft publication. This document describes the current state of the Semantic Publishing Benchmark software.

The application scenario behind the benchmark considers a media or a publishing organisation that deals with large volume of *streaming content,* namely articles and other "creative works" and "media assets". This content is enriched with *metadata* that describes it and links it to *reference knowledge* – taxonomies and databases that include relevant concepts, entities and factual information. This metadata allows publishers to efficiently retrieve relevant content, according to their various business models.

From a technology standpoint, the benchmark assumes that an RDF database is used to store both the reference knowledge and the metadata. The main interactions with the repository are (i) *updates*, that add new metadata or alter the repository, and (ii) *aggregation queries*, that retrieve content according to various criteria. The engine should handle instantly large number of updates in parallel with massive amount of aggregation queries.

This document describes all features of the SPB : data (reference data-sets, ontologies, data generation), query workloads (descriptions of queries used, choke point descriptions), validation of query results and instructions (how to configure and use the benchmark driver, execution, auditing and disclosure rules)

Executive summary

The *Semantic Publishing Benchmark* (SPB) is a LDBC benchmark for RDF database engines inspired by the Media/Publishing industry, particularly by the BBC's Dynamic Semantic Publishing approach. As of June 2014 the benchmark has reached the state of draft publication.

The application scenario considers a media or a publishing organisation that deals with large volume of *streaming content*, namely articles and other "creative works" and "media assets". This content is enriched with *metadata* that describes it and links it to *reference knowledge* – taxonomies and databases that include relevant concepts, entities and factual information. This metadata allows publishers to efficiently retrieve relevant content, according to their various business models. For instance, some, like the BBC, can use it to maintain rich and interactive web-presence for their content, while others, e.g. news agencies, would be able to provide better defined content feeds, etc.

From a technology standpoint, the benchmark assumes that an RDF database is used to store both the reference knowledge (mostly static) and the metadata (that grows constantly, to stay in synch with the inflow of streaming content). The main interactions with the repository are (i) *updates*, that add new metadata or alter it, and (ii) *aggregation queries*, that retrieve content according to various criteria. The engine should handle instantly large number of updates in parallel with massive amount of aggregation queries. Imagine that each request for a topic page on publisher's website requires several SPARQL queries to retrieve relevant content. Those queries should all be handled reliably within sub-second response time. And their results should reflect new content that came through the pipeline (along with its metadata) just few seconds ago.

The SPB benchmark provides the following business value:

- Media organisations that intend to adopt semantic publishing to foster their business, can use the benchmark as a simple, off-the-shelf means to evaluate suitable RDF database engines for integration into their publishing/journalist pipelines and work flows;
- Vendors of RDF data management software will be able to use the benchmark to find the relevant choke points in their products and provide a research focus for improvement. Vendors will also be able to use the benchmark results to market their products.

The SPB benchmark includes: ontologies, fixed data-sets, the benchmark driver, data generator, query workloads and documentation. The benchmark is flexible enough such that work flows can be tailored for many different use-cases. This allows media organisations to configure the benchmark to suit their own requirements and so quickly and easily run their own internal evaluation of products.

This Introduction section of the document provides the following information:

- Motivation: describes BBC's Dynamic Semantic Publishing platform that spurred industrial interest in semantic
- Relevance to industry: describes other media/publishing organizations that have shown growing interest in semantic technologies
- General Benchmark Overview: describes the basic use cases covered by the benchmark
- Participation of Industry and Academia: describes the major contributions that media companies (in particular the BBC) have made to the benchmark formation

The Formal definition section provides the benchmark specification as follows:

- *Requirements:* basic requirements that must be fulfilled by an RDF repository attempting to implement the benchmark;
- *Data:* describes the ontologies and reference datasets included in the benchmark as well as data generation process
- *Workloads:* describes the simulated benchmark workload that comprises simultaneous execution of editorial and aggregation query streams, choke points that workloads have and query definitions

- *Validation:* gives details about validation of query results and conformance tests that are performed by the benchmark driver
- *Performance Metrics:* describes the metrics used to measure the performance of RDF databases
- *Execution Rules:* describes the rules the must be fulfilled when executing the Semantic Publishing Benchmark in order to have equal start conditions for all benchmarked RDF systems and to be able to publish benchmark results
- *Full Disclosure:* describes the files and parameters that must be documented together with benchmark results
- *Auditing:* the things that an auditor must check before a benchmark can publish its results

Instructions section provides information about the benchmark software such as: installation, configuration, run and results gathering. It can be used for a reference on how to run the benchmark

- *Software installation:* gives details about how to install the benchmark software and descriptions of all important elements of the software distribution
- *Benchmark Configuration:* describes all aspects for configuration of the benchmark software : operational phases, configuration and definition properties
- *Benchmark Operation:* gives details what needs to be configured in order to start the actions that benchmark software can perform e.g. : Prepare RDF Database, Starting the Benchmark Driver, Generating data, Loading data, Warming-up the database, Running the benchmark, Validation of query results, Running conformance to OWL2-RL test etc.
- *Results Gathering:* describes the benchmark results and the log files they are written to

Table of Contents

List of Figures

List of Tables

Abbreviations

Table 1: Abbreviations used in this document

Abbreviation	Meaning
DSP	Dynamic Semantic Publishing
GUID	Globally Unique Identifier
N3	A superset of RDF, provides textual syntax alternative to RDF/XML (N3) W3C (28 March 2011)
N-Quads	Line-based, plain text format for encoding RDF dataset (N-QUADS) W3C (25 February 2014)
N-Triples	Line-based, plain text format for encoding RDF dataset
RDF	Resource Description Format (RDF) W3C (10 February 2004)
RDF/JSON	Textual syntax for RDF (RDF/JSON) W3C (7 November 2013)
RDF/XML	XML Syntax for the RDF (RDF/XML) W3C (10 February 2004)
SPB	Semantic Publishing Benchmark
TriG	Resource Description Format Dataset Language (TRIG) W3C (RDF 1.1 TriG)
TriX	RDF Triples in XML HPL-2004-56
Turtle	Terse RDF Triple Language (TURTLE) W3C (25 February 2014)
URI	Uniform Resource Identifier

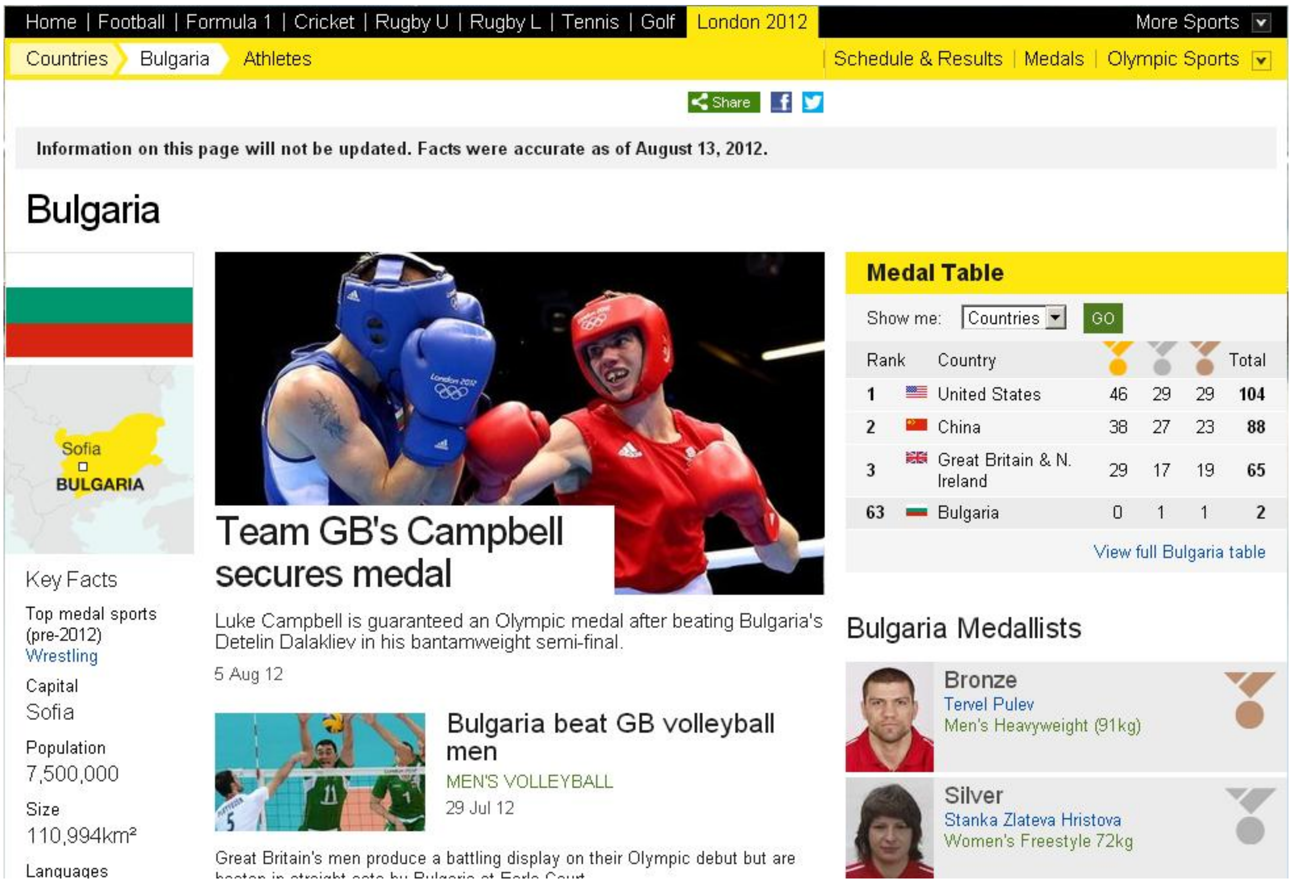
1 Introduction

1.1 Motivation for the benchmark

The Semantic Publishing Benchmark simulates the management and consumption of RDF metadata that describes media assets, or creative works. The scenario involved is a media organization that maintains RDF descriptions of its catalogue of creative works: journalistic assets (articles, photos, videos), papers, books, movies, etc. For this benchmark very useful input is being provided by actual media organizations which make heavy use of RDF (see next section). The benchmark is designed to reflect a scenario where a large number of aggregation agents provide the heavy query workload, while at the same time a steady stream of creative work description management operations are in progress. This benchmark targets RDF database systems, which support at least basic forms of semantic inference.

The inspiration for this benchmark originates in the BBC's development of the "dynamic semantic publishing" (DSP) concept. BBC's deployment of DSP for the World Cup 2010 was one of the first large-scale deployments of semantic technology. This was followed by all of the BBC Sports web site in 2011, culminating in the London Olympics 2012. The deployment of DSP proved to be a success, and it was publicised widely including in-depth technical descriptions. This sparked strong interest across the media and publishing industry worldwide (see next section).

BBC's DSP architecture for the Olympics involved over 10k dynamic aggregations, which are web pages that aggregate journalistic assets about a particular topic: athletes (>10k), country (>200), discipline (400-500), venue, team, etc. An example country aggregation page is shown on Figure 1.


Home | Football | Formula 1 | Cricket | Rugby U | Rugby L | Tennis | Golf
London 2012
More Sports
Countries
Bulgaria
Athletes
Schedule & Results | Medals | Olympic Sports
Share
Sofia
BULGARIA
Show me:
Countries
GO
View full Bulgaria table

Information on this page will not be updated. Facts were accurate as of August 13, 2012.

Bulgaria

Key Facts

Top medal sports (pre-2012)
Wrestling

Capital
Sofia

Population
7,500,000

Size
110,994km²

Languages

Team GB's Campbell secures medal

Luke Campbell is guaranteed an Olympic medal after beating Bulgaria's Detelin Dalakliev in his bantamweight semi-final.

5 Aug 12

Bulgaria beat GB volleyball men

MEN'S VOLLEYBALL

29 Jul 12

Great Britain's men produce a battling display on their Olympic debut but are

Medal Table

Rank	Country	🥇	🥈	🥉	Total
1	United States	46	29	29	104
2	China	38	27	23	88
3	Great Britain & N. Ireland	29	17	19	65
63	Bulgaria	0	1	1	2

Bulgaria Medallists

Bronze
Tervel Pulev
Men's Heavyweight (91kg)

Silver
Stanka Zlateva Hristova
Women's Freestyle 72kg

Figure 1: BBC Olympics 2012 Aggregation Page about Bulgaria

The semantic tagging of journalistic assets is an effort-intensive process. BBC used a semantic annotation pipeline that involves both:

- Automatic concept extraction (SPICE) including advanced probabilistic machine learning models to facilitate disambiguation and increase precision, and dynamic updating of the models based on concepts stored in the semantic repository. A diagram of the process is shown on Figure 2

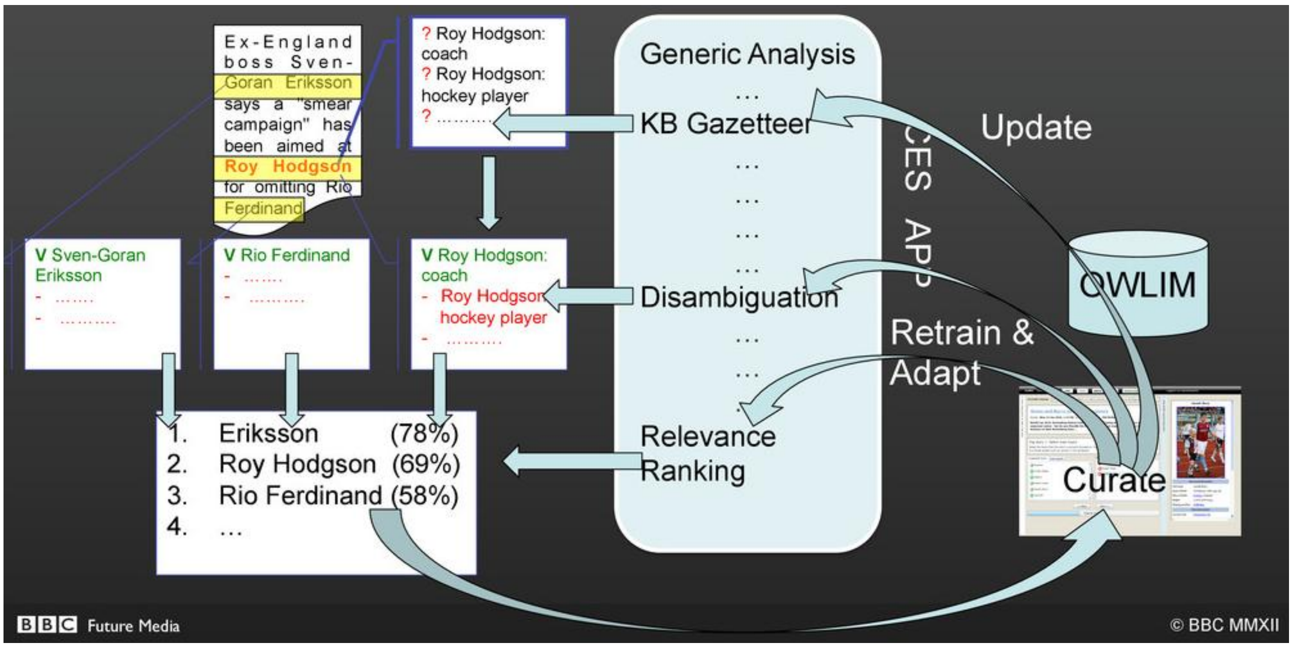
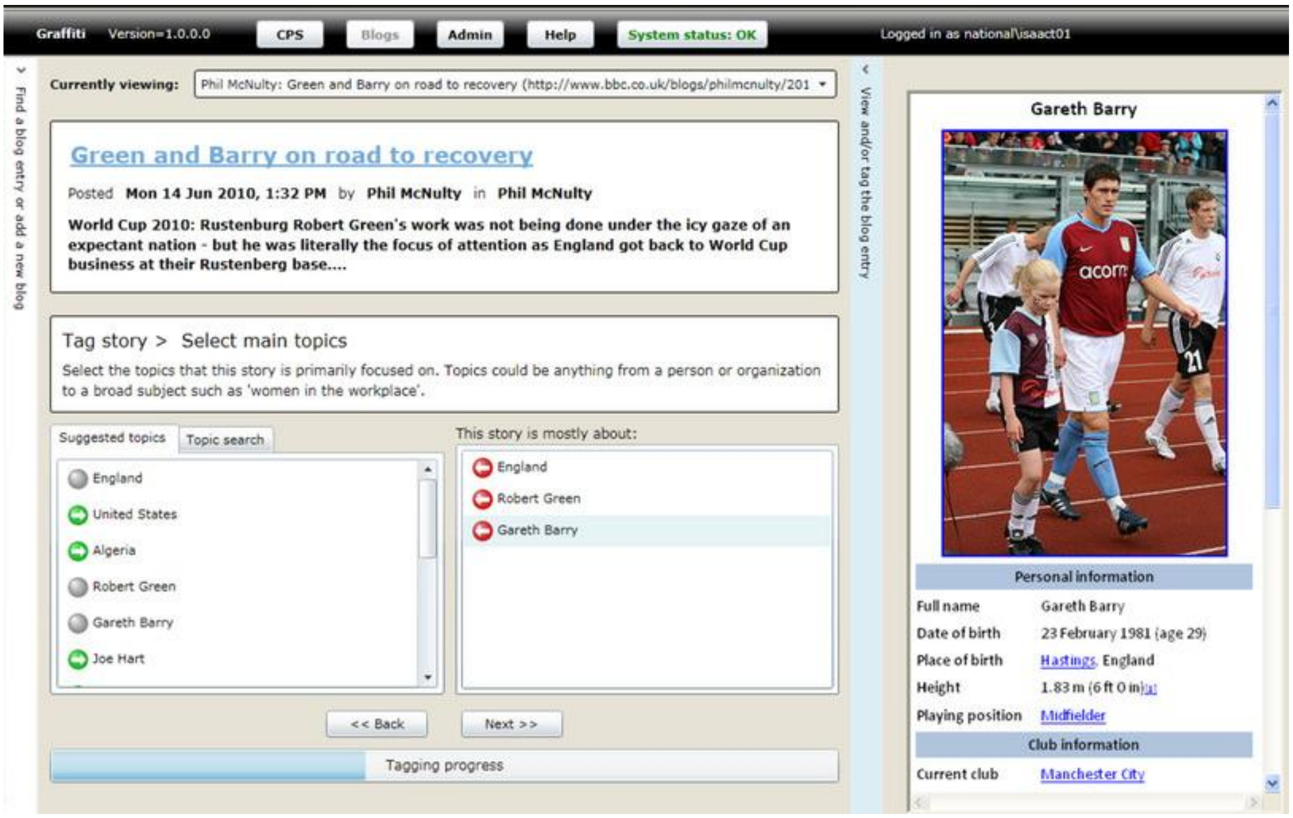


Figure 2: BBC Ontology-aware Text Analytics

- Manual editorial processes (Graffiti) that allow a journalist or editor to adjust semantic tags, as shown on Figure 3.



Graffiti Version=1.0.0.0
CPS
Blogs
Admin
Help
System status: OK
Logged in as national\isaact01
Find a blog entry or add a new blog
View and/or tag the blog entry

Currently viewing: Phil McNulty: Green and Barry on road to recovery (http://www.bbc.co.uk/blogs/philmcnulty/201

Green and Barry on road to recovery

Posted Mon 14 Jun 2010, 1:32 PM by Phil McNulty in Phil McNulty

World Cup 2010: Rustenburg Robert Green's work was not being done under the icy gaze of an expectant nation - but he was literally the focus of attention as England got back to World Cup business at their Rustenberg base....

Tag story > Select main topics

Select the topics that this story is primarily focused on. Topics could be anything from a person or organization to a broad subject such as 'women in the workplace'.

Suggested topics | Topic search

- England
- United States
- Algeria
- Robert Green
- Gareth Barry
- Joe Hart

This story is mostly about:

- England
- Robert Green
- Gareth Barry

<< Back | Next >>

Tagging progress

Gareth Barry

Personal information

Full name	Gareth Barry
Date of birth	23 February 1981 (age 29)
Place of birth	Hastings, England
Height	1.83 m (6 ft 0 in)[1]
Playing position	Midfielder

Club information

Current club	Manchester City

Figure 3: BBC Graffiti: Manual Curation of Semantic Annotation

- Semantic reasoning to infer more tags from the tags created by semantic annotation. E.g. if the knowledge base knows the schedule of a particular sports discipline, then tagging with a particular game occurrence can infer the discipline, countries, potential athletes appearing in the journalistic item.

A lot more technical details about BBC's architecture can be found in [4].

1.2 Relevance to industry

The technique for using semantic technology to annotate, link and consume media assets was originally pioneered at the British Broadcasting Corporation (BBC) in London. Being a publicly funded organisation, the BBC disclosed the technologies they used and have made a concerted effort to raise awareness in the public of their success with this new approach via blogs [1][2], presentations [3] and conference appearances [4].

For other media organisations, the application of semantic technology provides several opportunities.

Firstly, as in the BBC use-case, semantic technology can be used to automate many steps in the publication pipeline, especially in the areas of enrichment, linking and aggregation. The result is a better product that provides; a richer end-user experience; that is more dynamic and adaptable; that can adopt and deliver new content almost immediately; and that requires fewer staff (both journalistic and technical) to support.

Secondly, most media organisations have some kind of archive - in some cases stretching back over a hundred years. The digital revolution has enabled this content to be preserved, e.g. by scanning the photo archive and indexing it with the date, caption, owner, etc., however, this small number of attributes does not make the archive easy to use. In fact, it requires a good deal of manual effort using inaccurate keyword search to find anything useful. The advent of semantic technology, supplemented by some automated or semi-automated text analytics process, means that the archive can be ‘semantically annotated’, i.e. concepts can be identified that are relevant to the asset that are described in one or more ontologies. Using semantic annotations, media organisation can exploit their archives to: automate the process of finding relevant content; enrich their existing products; develop new product ranges. In essence, semantic annotation and search enables media organisations to ‘monetise’ their vast archives.

‘Semantic publishing’ is therefore an attractive paradigm for several media sectors, including: news, finance and scientific publications. In fact, the metadata becomes a valuable asset in itself, i.e. when scientific content is annotated to a certain level of detail, the metadata can be mined to find trends, associations, supporting evidence and even proofs - to the point where the actual content may become less important to the person or system conducting the search, the value of the metadata is what enables discovery.

Here follow a few examples of large/international media organisations that are known to be exploring semantic publishing:

- The British Broadcasting Corporation (BBC) is a UK based, publicly funded broadcasting and media organisation tasked. Its main responsibility is to provide impartial public service broadcasting in the United Kingdom, the Channel Islands, and the Isle of Man. It is the largest broadcaster in the world by number of employees, with about 23,000 staff. The BBC originally developed the concept of “dynamic semantic publishing” when creating the World Cup 2010 website. Since then they have rolled out this model to the whole of their Sports product and the news team are in an R&D phase. In order to consolidate the consumption of linked data across all of the BBC products, they are building a dedicated ‘linked data platform’ that supplies both in-house and external linked data from a single in-house service. This platform requires an RDF database capable of handling the creation and management of semantic descriptions (annotations) in real-time, 24 hours per day, for all of their media assets: from news and sport to science and nature - as well as to provide query-answering performance to power all of their distribution channels simultaneously (website, mobile, iPlayer, etc.).
- The UK Press Association (PA) is the national news agency for the UK and Ireland and a leading multi-media content provider across web, mobile, broadcast and print. For the last 145 years PA has been providing fast, accurate feeds of text, data, photos and video. Today the business is increasingly

focused on the delivery of complete products for both digital and print clients. PA have developed a semantic publishing pipeline that is used to improve automation across more than one hundred online products. PA have an enormous archive of audio, video, images and text that with the help of text analytics and semantic technology, is being suitably annotated for consumption.

- Euromoney is an international business-to-business publisher focusing on international finance, macroeconomics, IPOs, bond issuance, M&A deals, banking, capital markets, commodities, foreign exchange, investments, transaction services, and emerging markets. By semantically annotating content, they are able to deliver better products through improved search tools made available to their in-house experts.
- Elsevier is the world's leading provider of science and health information. Elsevier serves more than 30 million scientists, students and health and information professionals worldwide and partners with a global community of 7,000 journal editors, 70,000 editorial board members, 300,000 reviewers and 600,000 authors to help customers advance science and health by providing world-class information and innovative tools that help them make critical decisions, enhance productivity and improve outcomes. Elsevier's 'Smart Content Applications' initiative will offer better discovery through semantic search and navigation; better understanding through analysis and visualization; and the discovery of new knowledge through aggregation and synthesis. The 'Linked Data Repository' is a service platform that utilises a variety of technologies and storage components, where the search function uses a hybrid SPARQL and full-text search (FTS). The requirement of the platform (and the RDF store) is that several billions of RDF triples can be bulk loaded, with incremental updates of several millions. The requirements include: scalability, resilience, update performance, full ACID compliance, reasoning, and the ability to perform deep analysis of the RDF data.
- Wiley is a global publishing company that specializes in academic publishing and markets its products to professionals and consumers, students and instructors in higher education, and researchers and practitioners in scientific, technical, medical, and scholarly fields. The company produces books, journals, and encyclopaedias, in print and electronically, as well as online products and services, training materials, and educational materials for undergraduate, graduate, and continuing education students. Wiley has started a strong foray in semantic technologies, including extensive training of its developers.
- Oxford University Press is a department of the University of Oxford and the largest university press in the world. It has started exploring semantic publishing, and is experimenting with author identification based on named entity recognition, paper content summarization and keyword identification, and other semantic techniques.

For all media organisations that intend to adopt semantic publishing to drive their business, the LDBC Semantic Publishing Benchmark will provide a simple, off-the-shelf means to evaluate suitable RDF databases for integration into their publishing/journalist pipelines and workflows. The benchmark is flexible enough such that workflows can be tailored for many different use-cases. This allows media organisations to configure the benchmark to suit their own requirements and so quickly and easily run their own internal evaluation of products.

Vendors of RDF data management software will be able to use the default benchmark to find the relevant choke points in their products and provide a research focus for improvement. Vendors will also be able to use the benchmark results to market their products.

1.3 General Benchmark Overview

This benchmark simulates the BBC's model where the majority of content is consumed by the public via the Website, iPlayer and mobile devices. However, the model is similar for any high-demand, automated media delivery platform.

Information being consumed is located in an RDF triple store. The core of the publishing system is an RDF Database Engine – which is used to store data about various entities. All entities stored in the database provide information and knowledge about various domains, e.g. politics, sports, persons etc. Entities are additionally being annotated, which adds another layer of relation between them, or just an additional information about them. Annotations can be a simple tag about an entity or a definition of relation between one or group of entities.

People will ask queries about topics or entities stored in the database, create, update, delete annotations about those entities. Queries are returning aggregate or concrete results about topics people are interested in. A constant stream of queries and annotations is sent to the database which is generated by different types of actors in the publishing system e.g. journalists, editors, end users, text analysis engines (semi- or fully-automated). Those actors have been modelled in the benchmark by two types of 'Agents' :

- Aggregation agents - ask queries about some topic or related topics, process result or refine the search based on returned result.
- Editorial agents - generate new annotations about existing entities, update or delete existing ones.

Aggregation and Editorial agents are modelling the interactions that all fore-mentioned actors are having with the publishing system.

The queries vary in complexity. There are simple queries which are asked about a concrete topic and produce a simple result, whereas complex queries produce an aggregate result based on a combination of several properties or constraints. Also there are analytical queries which dynamically alter their criteria based on the returned result and faceted search queries as well.

A required editorial operations rate per second is advisable to be constantly sustained during measurement the aggregation queries rate. Value of the editorial operations rate is to be chosen by the user and is constantly monitored by the benchmark.

A typical description of the benchmark process would be: entities stored in the database will be annotated by journalists or text analysis engines (represented by the editorial agents). Then queried by end-users, journalists or automated search engines (represented by the aggregation agents). Also annotations are edited by editors or journalists after being created (represented by the editorial agents).

LDBC-SPB provides a test driver which is responsible for executing all phases of the benchmark described in this document. Being highly configurable, the test driver is capable of measuring performance on different scales of RDF stores as well as capable of customizing to the limitations that some RDF stores might have.

1.4 Participation of industry and academia

As well as being the inspiration for the LDBC Semantic Publishing Benchmark, the BBC have contributed their data, ontologies and system descriptions, as well as their effort to help develop the benchmark into its current form. Representatives from the BBC have joined almost all of the regular task force conference calls, both TUC face-to-face meetings and have also welcomed LDBC personnel to their premises in London on several occasions.

The Press Association have also been active, having attended the first TUC meeting and commenting on various aspects of the benchmark as it has evolved.

Ontoba are a systems integration company based in London and have provided consultancy, system architect and software development services to a number of UK and non-UK media organisations, including the BBC, the Press Association, Euromoney amongst others.

Nevertheless, the benchmark is very much based upon the BBC use-case, where the BBC have provided:

- Core application ontologies that describe their internal data model, content location, tagging mechanisms, departments and themes
- Domain ontologies for sport, government, people, events
- Datasets for UK football competitions, formula 1 racing and UK government officials

The LDBC conformance ontology supplements the BBC ontologies with more complex reasoning constructs for the dual purpose of:

- Testing for consistency according to OWL2-RL rule-based (RDF-based) semantics
- Testing for correct OWL2-RL inference

This conformance ontology and subsequent data snippets and SPARQL operations were developed between Ontotext, FORTH and the BBC.

2 Formal definition

2.1 Requirements

The following behaviours/functionalities are required from any RDF database engine in order to properly execute the LDBC Semantic Publishing Benchmark (see Table 2).

Table 2: Required behaviours/functionalities required from a RDF database

Feature	**Explanation**
RDF	The database must be capable of storing and processing data in the Resource Description Format (RDF) W3C (10 February 2004)
RDF serialisation	The database under test must be capable of loading RDF data in one of the standard or recommended formats. When this is not possible directly, a separate (manual) step in the benchmark process is necessary to use the appropriate loading tool. Load time is NOT measured as part of this benchmark. The benchmark driver will use Turtle to load ontologies and N-Quads to load the generated creative works data.
RDF named graphs	The database must be capable of storing and isolating separate RDF graphs identified by name (URI), i.e. the database engine must be a ‘quad-store’.
SPARQL	The following SPARQL standards must be supported: SPARQL 1.1 Query (21 March 2013) SPARQL 1.1 Update (21 March 2013) SPARQL 1.1 Protocol (21 March 2013)
RDFS	The semantics of the RDF Vocabulary Description Language 1.0 (RDF Schema) (10 February 2004) must be fully supported in order to return the correct results from queries. These semantics are subsumed by the semantics of OWL2-RL.
OWL	The semantics of the RL profile of Web Ontology Language (OWL2) must be supported in order to pass the conformance test suite.

If the database supports further (non-standard) functionality for processing certain kinds of data or provides custom techniques for accessing this data (via SPARQL) then the test sponsor would be allowed to modify certain queries to take advantage of these features, providing that the results of the query remain unaltered. Databases with those features (e.g. geo-spatial, text indexing) are likely to benefit from such modifications.

2.2 Data

Semantic Publishing Benchmark is distributed with a set of reference data used for data generation and benchmarking. In order to use the benchmark, the database must be initialized with a set of required input data, which is used as a foundation for achieving a realistic use-case scenario. Input data falls into two categories: ***ontologies*** and ***reference datasets***.

Ontologies define the relations between entities in reference data. They can be further divided into three sub-categories:

- **Core ontologies** - describe essential data objects and their properties e.g. creative works. Following is a list of core ontologies: creativework 0.9, company 1.4, coreconcepts 0.6, CMS 1.2, person 0.2, provenance 1.1, tagging 1.0.
- **Domain ontologies** - describe concepts or properties related to a specific domain. Following is a list of domain ontologies:
  - cnews-1.2 - describes the basic concepts that journalists can tag annotations with.
  - sport 2.3 - describes sports, competitions, events.
  - curriculum 4.0 - describes academic entities.
- **Conformance ontologies** - added by the LDBC for enriching existing ontologies and used to test for conformance violations - which is a part of the benchmark - ldbc-conformance.0.3.

More information about ontologies can be found in Appendix. A - Ontologies

Reference Datasets are collections of entities describing various domains e.g. sports, politics, persons, geo-locations, etc. that editorial agents create annotations about and aggregation agents query. They are snapshots of the real datasets provided by the BBC. Additionally DBPedia and Geonames reference datasets have been added for further enriching the annotations with person and geo-location. Table 3 contains descriptions of used reference datasets sorted alphabetically.

Table 3. Reference Datasets

Dataset	**Description**
dbpedia_persondata_en	optional dataset, contains entities describing popular persons world-wide, their names, birth dates and descriptions. Version of the dataset: 3.9, ttl format.
english-football-competitions-1	contains entities describing the English football competitions, e.g. “Premier League”, “League One” etc.
english-football-teams-2	contains entities describing the English football teams, e.g. “Leicester City”, “Macclesfield Town”
formula1-competitions-8	contains entities describing the Formula 1 competitions, e.g. “British Grand Prix”, “German Grand Prix”
formula1-teams-3	contains entities describing the Formula 1 teams, e.g. “Ross County”, “Hamilton”
geonames-EU	optional dataset, contains entities describing places, names and their coordinates (lat, long) for all countries in Europe. Version of the dataset: May 23 21:12:35 CEST 2011.
geonames-GB	contains entities describing places, names and their coordinates (lat, long) for Great Britain. The dataset has been retrieved from the Geonames database. Version of the dataset: May 23 21:12:35 CEST 2011.
international-football-competitions-3	contains entities describing the International football competitions
international-football-teams-2	contains entities describing the International football teams

scottish-football-competitions-1	contains entities describing the Scottish football competitions
scottish-football-teams-2	contains entities describing the Scottish football teams
UK-Parliament-Identifiers-People-7	contains entities describing People in the UK Parliament, e.g. their names, references to external databases like DBpedia

2.2.1 Data Generation

Along with the reference datasets, SPB provides a data generator which is used to produce scalable in size synthetic large data. Following is a description of the Data generator and types of modelled data it generates.

Starting point of the current benchmark is a dataset, consisting of ontologies, reference and generated data – a large number of annotations (or descriptions) of media assets that refer entities found in the reference datasets. All the generated data consists of descriptions of creative work instances, which refer one or several entities from reference datasets.

A creative work can be defined as a meta-data about a real entity (or entities) that exist in reference datasets. A creative work can have various properties like: title, description, creation and modification dates – which all are literal values and other properties like: primaryTopicOf, thumbnail, etc. – referring to other resources. A creative work also has properties: *'about'* and *'mentions'* which hold the references to entities. That way a creative work provides meta-data about one or several entities – facts about them and defines relations between them.

Each creative work resides in its own context. URI identifier used for each creative work and its context differ by a single indirection used in that URI. For creative works that indirection is 'things' and for contexts is 'context'.

Data generator models three types of relations in data :

- **Clustering of data.** The clustering effect is produced by generating *creative works* about a single entity from reference datasets and for a fixed period of time. The number of creative works starts with a high peak at the beginning of the clustering period and follows a smooth decay towards its end. The data generator produces major and minor clusterings with sizes (i.e., number of creative works) of different magnitude. Example of data clustering could be news items that are about events starting with a high number of journalistic assets related to them and following a decay in time as they reach the end of time period, a tendency that mirrors a real world scenario in which a 'fresh' event is popular and its popularity decreases as time goes by.
- **Correlations of entities.** The correlation effect is produced by generating creative works about two or three entities from reference data in a fixed period of time. Each of the entities is tagged by creative works solely at the beginning and end of the correlation period and in the middle of it, both are used as tags for the same creative work. Such an example of data correlation could be that several 'popular' persons (e.g., Stross-Cahn and Barroso) are mentioned together by creative works for a certain period of time.
- **Random tagging of entities.** Random data distributions are created with a bias towards popular entities created when the tagging is performed, that is when entities are assigned to *about* and *mentions* creative work properties. This is achieved by randomly selecting a 5% of all the entities from reference data and marking them as 'popular' when the remaining ones are marked as 'regular'. When creating creative works, 30% percent of the creative works are tagged with randomly selected popular resources and the remaining 70% are linked to the regular ones. Example for random taggings could be every-day events which become less important several days after their start date. Distributions of about and mentions tags analysed from a 'live' dataset provided by the BBC are reproduced in generated data. Table 3 shows the distribution of total about and mentions tags found in creative works and also shows their individual distributions.

Table 4: Distribution of 'about' and 'mentions' tags in creative works, analysed in 'live' data provided by the BBC

Amount	Distribution of about and mentions, %	Distribution of about tags, %	Distribution of mentions tags, %
1	22.33 %	10.06 %	94.77 %
2	32.67 %	23.13 %	3.82 %
3	24.60 %	30.88 %	0.93%
4	11.63 %	22.78 %	0.31 %
5	3.27 %	10.35 %	0.12 %
6	1.52 %	2.80 %	0.05 %
7	1.00 %	0 %	0 %
8	0.74 %	0 %	0 %
9	0.69 %	0 %	0 %
10	0.47 %	0 %	0 %

Illustration of the three types of models that data generator can produce is shown in Figure 4.

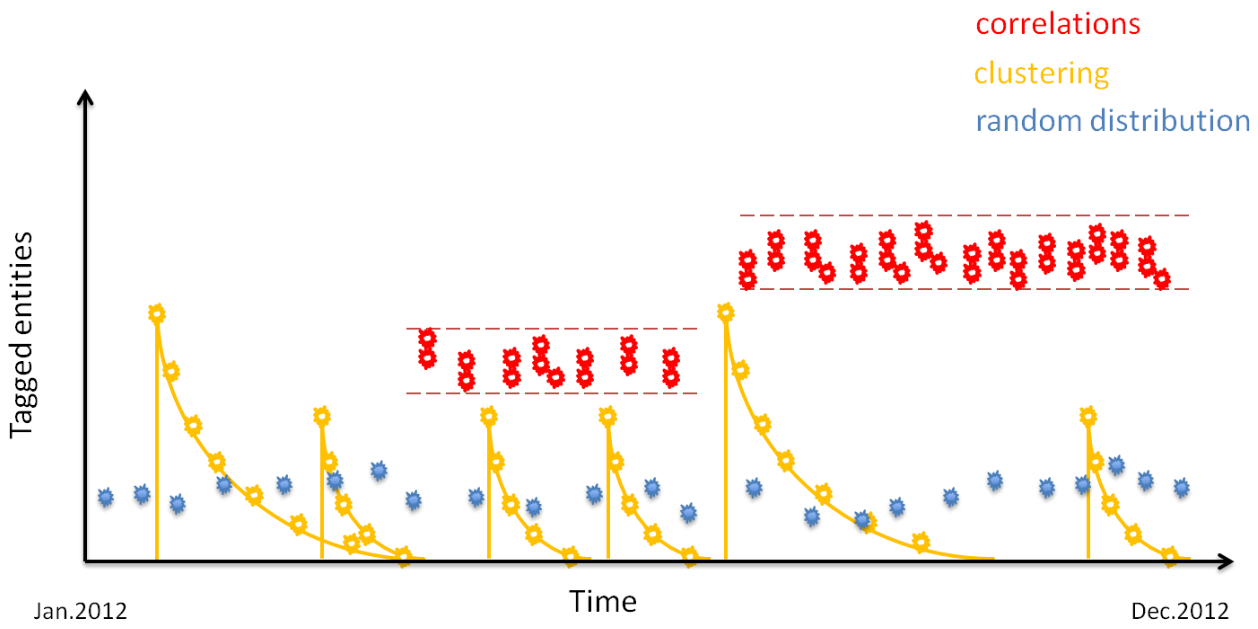


Figure 4: Data generator, types of produced models in generated data

Following is a brief description of the Data generation process :

- First step is to identify all instances from different domain ontologies (also referred to as entities above) that exist in the reference datasets. Identification process consists of execution of several queries that will collect data about stored entities. Once identified, they are used by data generator for tagging when a creative work is created.

- Next step is to use one of the three data models described above and generation of creative works begins following a selected data generation model.

Each creative work will have a set of properties which are described below in Table 5:

Table 5. Creative Work Properties

Property	**Type**	**Description**
title	String	Contains a sentence of randomly selected words from a dictionary file. Maximum number of words is 10 + 2 which is also randomly selected. As a prefix of the title is used the real label of the entity extracted from reference data.
shortTitle	String	Contains a sentence of randomly selected words from a dictionary file. Maximum number of words is 10 and is also randomly chosen.
category	URI	One of pre-defined category types : • Persons • PoliticsPersonsReference • PoliticsPersonsAdditional • SportsTeams • SportsCompetitions
description	String	Contains a sentence of randomly selected words from a dictionary file. Minimal number of words is 8 and maximum 26. A random value in that range is chosen.
about	URI	Contains the URI of an entity that has been tagged by that creative work. Number of *about* properties varies between 1 and 6, distribution of which is shown in Table 4
mentions	URI	Contains the URI of another entity or entities that current creative work also mentions. Number of *mention* properties varies between 1 and 6, distribution of which is shown in Table 4
audience	URI	One of the audience types defined in the creativework ontology : • InternationalAudience • NationalAudience
liveCoverage	Boolean	Defines weather creative work is tagging a live coverage event
primaryFormat	URI	One of the formats defined in the creativeworks ontology : • AudioFormat • VideoFormat • InteractiveFormat • TextualFormat

dateCreated	Date	Date of creation, randomly selected from a seed value.
dateModified	Date	Date of modification, randomly selected and after the moment of creation, also picked randomly within a defined time range.
thumbnail	URI	Contains the URI of a thumbnail image stored in the system. Available types of thumbnail defined in the creativeworks ontology : • CloseUpThumbnail • FixedSize226Thumbnail • FixedSize466Thumbnail • FixedSize66Thumbnail • StandardThumbnail
primaryContentOf	URI	Refers to a URI of a document, assuming that such a document hypothetically exists and is identified by a URI.
altText	String	A text string for enriching thumbnail information

Depending on the type of creative work, certain properties have their pre-defined values. Following Table 6 describes what values of properties are assigned depending on its type.

Table 6. Creative Work types and specific properties

Creative Work Type	Description	Properties
NewsItem	Journalistic asset about some news item. It could be reoccurring or a every-day news event.	• audience : restricted to National audience • liveCoverage : set to false • primaryFormat : set to TextualFormat and InteractiveFormat
BlogPost	Creative work about a blog post, a tweet or other post.	• audience : restricted to International audience • liveCoverage : set to false • primaryFormat : set to TextualFormat and randomly adding another primaryFormat of InteractiveFormat
Programme	Creative work about a Programme, e.g. a television broadcast or a radio programme	• audience : restricted to International audience • liveCoverage : set to true • primaryFormat : set randomly to either AudioFormat or VideoFormat

During generation of creative works, following distribution by type is produced :

- 45% - BlogPost,
- 35% - NewsItem
- 20% - Programme

Generated data is saved in files with a configurable serialization file format. All available serialization formats are supported, but as not all serialization formats are context aware only those supporting RDF contexts are suitable for the benchmark i.e. *N-Quads* and *TriG*.

Although there are no strict scale sizes defined for generated data, following Table 7 provides information on several characteristics of produced synthetic data. Last column shows statistics about entities from optional datasets (if included in the data generation). Even if no optional datasets have been added, results shown in other columns are similar.

Table 7. Characteristics of generated data at different scales

Scale size, triples	**Actual triples saved**	**Size on disk, GB**	**Creative Works**	**Entities Used (from optional datasets)**
5M	5 000 068	0.94	228 892	542 021
50M	50 000 061	9.29	2 288 516	642 021
200M	200 000 067	37.3	9 154 691	702 227
1B	1 000 000 083	188	45 776 626	772 124

2.3 **Workloads**

The workload in the Semantic Publishing Benchmark is defined by simultaneous execution of *editorial* and *aggregation* agents, simulating a constant load generated by end-users, journalists, editors or automated engines. Further it stresses important functionalities that RDF stores must cope with.

Editorial agents simulate the editorial work performed by journalists, editors or automated text annotation engines by executing following operations:

- **Insert operations**: generate new creative work descriptions (content metadata) following the models and distribution rules defined in Data generation section. Each creative work is added to the database in a single transaction by execution of an insert SPARQL query.
- **Update operations**: update an existing creative work. Update operation consists of two actions, executed in one transaction, following the BBC's use-case for update of a creative work. First action is to delete the context where a creative work description resides along with all its content. Second is to insert the same creative work (using its current ID) with all its properties – current and updated ones.

- **Delete operations**: delete an existing creative work. Delete operation will erase the context where a creative work resides along with all of its content.

Each editorial agent will execute a mix of editorial operations in a constant loop, until the benchmark run has finished. Editorial operations executed by an agent are chosen pseudo-randomly following the distribution: 80% INSERT operations, 10% UPDATE operations, 10% DELETE operations.

Important note to add here is on how IDs of new creative works and their contexts are created. Instead of using a randomly generated Globally Unique Identifier (GUID) which would be otherwise a reasonable choice, the IDs of creative works are created by incrementally increasing a number starting from 1. That way when simulating all editorial operations, there is no need to explicitly have a knowledge of all creative work IDs stored in the database, but retrieving the greatest ID is enough, and that retrieval is performed just once – before benchmark phase begins. That approach saves time for retrieval of creative work IDs and is keeping the complexity of IDs creation low.

Aggregation agents simulate the retrieval operations performed by journalists, end-users or automated search engines by executing a mix of aggregation queries described in Table 8.

Table 8. Aggregation query types

Query Type	**Description**
Aggregation	Queries that take longer to execute as their purpose is to accumulate a result of creative works that match certain criteria, e.g. creative works about some topic , or creative works modified within some time range
Search	Those queries execute faster and are searching for creative works that match a concrete fact
Statistical	Queries that produce statistics about existing creative works, their distribution, etc. For example: most popular creative work types, most popular topics creative works are about or mention, shows the greatest number of mention tags that creative work has
Full-text	Perform text search in the literal properties of creative works
Geo-spatial	Aggregate result based on geo-spatial region
Analytical	Performs analysis on stored data, e.g. find those creative works that are most similar by calculating a score about entities being tagged by them or finding the two most popular entities tagged by creative works for a period of time
Drill-down	Dynamically re-configure their criteria, based on the result produced during previous execution. E.g. retrieve all creative works modified in August 2012. From result pick a creative work and further enhance modification time criteria by adding a time interval of few hours around its modification time, etc.
Faceted search	Queries that retrieve creative works by applying multiple filters incrementally

Each aggregation agent will execute a mix of those query types described above in a constant loop, until the benchmark run finishes. Each agent executes a query and waits for response (or a time-out), after receiving the response next query is executed (queries executed by agents are not of the same type). Query order of execution is pseudo-randomly chosen following an even distribution for each query defined in the benchmark's configuration.

2.3.1 Choke Points

A benchmark can be characterized as valuable if its workload stresses important technical functionalities that database engines must manage. A definition of "choke points" could be the technical challenges that each database needs to overcome in order to satisfy the need for fast and reliable service. Queries in SPB are addressing a variety of choke points, each testing the certain capabilities of the RDF database engine :

- **CP1 : JOIN ORDERING** : tests the ability to consider cardinality constraints and decide which type of join to be used
- **CP2 : AGGREGATION** : tests the ability to evaluate sub-selects first
- **CP3 : OPTIONALs AND NESTED OPTIONALs** : tests the ability of the query optimizer to produce a plan where the execution of the triple patterns inside optional section is the last to be performed since they do not affect the size of the intermediate results
- **CP4 : REASONING** : tests the ability whether systems can handle efficiently RDFS and OWL constructs
- **CP5 : PARALLEL EXECUTION OF UNIONs** : tests the ability of optimizer to produce query plans where unions are executed in parallel. This is helpful if the involved sub-queries produce a large number of results
- **CP6 : OPTIONALs WITH FILTERs** : tests the ability to execute as early as possible such expressions in order to eliminate possible large number of intermediate results
- **CP7 : ORDERING** : tests the ability of the optimizer to choose query plan that facilitates the ordering of results
- **CP8 : GEO-SPATIAL PREDICATES** : tests the ability to handle queries for geo-spatial data, mentioning entities within a geospatial range
- **CP9 : FULL-TEXT Search** : tests the ability of the systems to utilize custom optimizations e.g. indexing when dealing with text search
- **CP10 : DUPLICATE ELIMINATION** : tests the ability to identify duplicate entries during the creation of intermediate results
- **CP11 : COMPLEX FILTER CONDITIONS** : tests the ability of query optimizer to split complex filter conditions into conjunction of conditions and execute them in parallel as soon as possible

Two types of workloads are offered by the Semantic Publishing Benchmark - *basic* and *advanced*.

Basic workload consists of 9 queries of types : search, aggregate, geo-spatial, full-text search, time-range queries. *Advanced* workload consists of 25 queries which in addition to basic workload adds : analytical, drill-down and faceted search queries.

Following are descriptions of queries from basic and advanced query mixes:

2.3.2 Basic Workload - Query Definitions

Identifier	**query1.txt**
Description	Retrieve the 10 most recent creative works, that are about or mention different topics. For each creative work a graph is returned that comprises of the work's *title,*

	shortTitle, *dateCreated, dateModified, description, primaryFormat* and *primaryContentOf* properties; if the creative work has a thumbnail, then return its *thumbnailAltText* and *thumbnailType*. Also retrieve existing properties of the topics that the creative work is about or mentions: *aboutLabel, aboutShortLabel, aboutPreferred-Label, mentionsLabel, mentionsShortLabel* and *mentionsPreferredLabel*.
Type	Aggregation
Choke Points	CP1, CP2, CP3, CP4, CP5

Identifier	**query2.txt**
Description	Retrieve details about a given resource that is an instance of class *CreativeWork* and its subclasses, namely its *title*, *dateCreated, dateModified* the topic the resource is about, its *primaryContentOf* and the *webDocumentType* thereof.
Type	Search
Choke Points	CP1, CP3, CP4

Identifier	**query3.txt**
Description	Retrieve a list of creative works that are instances of classes *BlogPost* or *NewsItem*, that are about a specific topic, the value of *primaryFormat* is one of *TextualFormat, InteractiveFormat* or *PictureGalleryFormat*. If the creative work has an audience then the creative work should be obtained using this value. Given the retrieved list of creative works, return for each resource its related nodes, and then order the result in descending order of the value of their *creationDate* property.
Type	Aggregation
Choke Points	CP3, CP5, CP6, CP10, CP11

Identifier	**query4.txt**
Description	Return a list of all creative works with all their properties, that are *about* a given topic, have a given *primaryFormat*, and are instances of a certain subclass of class *CreativeWork*. The results should be ordered by property *creationDate* and only N results should be returned.
Type	Search
Choke Points	CP1, CP7

Identifier	**query5.txt**
Description	Return the list of most popular topics that creative works of a given type are about, have a given audience, and they have been created in a specific time range (*dateModified* is between a start and an end date). For each retrieved topic get its properties (if they exist) *cannonicalName* and *preferredLabel*; and if any of the labels exist, return it as a result along with the count of topics with this label. The result

	should be returned on descending order of the count of labels.
Type	Aggregation
Choke Points	CP1, CP2, CP3, CP7, CP11

Identifier	**query6.txt**
Description	Retrieve all instances of class *CreativeWork* that mention a geo-location that is within the boundaries of a given rectangle area. Along with each retrieved creative work retrieve property *geonamesId* and its lat (latitude) and long (longitude) values. Limit the result to 100 creative works (geo-spatial query).
Type	Geo-spatial
Choke Points	CP7, CP8, CP10, CP11

Identifier	**query7.txt**
Description	Retrieve properties *dateModif, title, category, liveCoverage, audience* for all creative works that are of a given type. The value of property *dateModif* of the retrieved creative works should be within a certain time range. Return 100 results ordered in ascending order by their *dateModif*.
Type	Aggregation
Choke Points	CP1, CP7, CP11

Identifier	**query8.txt**
Description	Retrieve the graphs of resources that are instances of class *CreativeWork* considering also its subclasses that comprise the resources' *type, title, description, dateCreated, dateModified, category*, the topic they are about, their *primaryContentOf* and the latter's *webDocumentType*. Each of the returned resources should contain in its title or description a given string. (full text search query)
Type	Full-text search
Choke Points	CP3, CP4, CP9, CP11

Identifier	**query9.txt**
Description	Retrieve 10 similar creative works, by calculating their similarity score. The creative works should be ordered by the computed score and in descending order of the value of property *dateModified*.
Type	Analytical
Choke Points	CP2, CP7, CP10

2.3.3 Advanced Workload - Query Definitions

Identifier	**query1.txt**
Description	Retrieve the 10 most recent creative works, that are about or mention different topics. For each creative work a graph is returned that comprises of the work's *title, shortTitle, dateCreated, dateModified, description, primaryFormat* and *primaryContentOf* properties; if the creative work has a thumbnail, then return its *thumbnailAltText* and *thumbnailType*. Also retrieve existing properties of the topics that the creative work is about or mentions: *aboutLabel, aboutShortLabel, aboutPreferred-Label, mentionsLabel, mentionsShortLabel* and *mentionsPreferredLabel*. (same as query1.txt from basic query mix)
Type	Aggregation
Choke Points	CP1, CP2, CP3, CP4, CP5

Identifier	**query2.txt**
Description	Retrieve details about a given resource that is an instance of class *CreativeWork* and its subclasses, namely its *title*, *dateCreated*, *dateModified* the topic the resource is about, its *primaryContentOf* and the *webDocumentType* thereof. (same as query2.txt from basic query mix)
Type	Search
Choke Points	CP1, CP3, CP4

Identifier	**query3.txt**
Description	Retrieve creative works that have been modified in a time range of one hour. This query returns (a) creative works ordered by each minute of the hour they have been modified in a certain time range specified in an filter constraint and (b) their number
Type	Aggregation
Choke Points	CP1, CP7, CP11

Identifier	**query4.txt**
Description	Retrieve the most popular types of creative works that have been modified for a period of time (2/3 of one year). This select query retrieves all instances of only subclasses of class *CreativeWork*, the latter is filtered out by using a filter expression. Additional constraint is added on their modification time. Results are grouped by creative works' type, ordered by their number for each type and a slice of the first 10 is selected
Type	Aggregation

Choke Points	CP1, CP4, CP7, CP11

Identifier	**query5.txt**
Description	Retrieve the N most popular topics that creative works are *about*, selecting from two categories. This select query returns the first N results grouped by topic, ordered in descending order using the number of uses of each topic. Result is further limited by selecting only two of categories of creative works
Type	Aggregation
Choke Points	CP1, CP7, CP11

Identifier	**query6.txt**
Description	Retrieve the N most popular topic types that creative works are *about*. Adds a filter on *coverage* and *audience* properties. This query returns the first ten results grouped by topic type, ordered in descending order using the number of uses of each topic type.
Type	Aggregation
Choke Points	CP1, CP4, CP7, CP11

Identifier	**query7.txt**
Description	Retrieve the N most popular topics that creative works *mention*. This query returns the first ten results grouped by topic, ordered in descending order by the number of tagging of each topic. Result is further limited by adding a condition on primary contents for each creative work.
Type	Aggregation
Choke Points	CP1, CP2, CP7

Identifier	**query8.txt**
Description	Retrieve the N most popular topics creative works that have been modified in a time range of one hour are about. This query returns results, grouped by topic, ordered in descending order by the number of appearances of each topic. A filter expression checks whether the modification of a creative work is within the specified time range, and whether the audience of the creative work has the specified value.
Type	Aggregation
Choke Points	CP1, CP7, CP11

Identifier	**query9.txt**
Description	Retrieve the largest number of mentioned topics in creative works. This query uses a sub-select to count the creative works that are *mentioning* 'things'. Results are grouped

	by creative work, and the max aggregate expression is calculated over the group of retrieved 'mentions' properties for a creative work.
Type	Analytical
Choke Points	CP2

Identifier	**query10.txt**
Description	Retrieve a list of N creative works that are *mentioning* the maximum number of topics, their number, creation date and topics. This query uses *query9* (as a sub-query) to retrieve the greatest number of mentioned topics in creative works and a second sub-query which retrieves all creative works that mention as many topics as the creative work that has the greatest number of mentioned topics. This match is achieved through a filter constraint.
Type	Aggregation
Choke Points	CP1, CP2, CP7

Identifier	**query11.txt**
Description	Retrieve similar creative works regarding the ’things’ they are *about* or *mention*. This query searches for similar creative works, by counting the set of intersecting about and mentions values, to which a metric is applied and scores the similarity, e.g. similarity score equals (number of identical about tags) * 2 + (number of mentioning about tags) * 1 + (number of identical mentions tags) * 0.5
Type	Analytical
Choke Points	CP2, CP5, CP7, CP10

Identifier	**query12.txt**
Description	This query searches for similar creative works, by counting the set of intersecting *about* and *mentions* values, to which a metric is applied and scores the similarity. The similarity score equals (number of identical about tags) * 2 + (number of mentioning about tags) * 1 + (number of identical mentions tags) * 0.5. This query is similar to *query11* and takes advantage of the *rdfs:subPropertyOf* subsumption hierarchy.
Type	Analytical
Choke Points	CP2, CP4, CP7

Identifier	**query13.txt**
Description	Retrieve a list of N creative works, the ’things’ they are *about* and *mention*, their categories, the modification date. Filter results by two categories. This is a star-shaped query that involves five triple patterns (four joins).
Type	Aggregation

Choke Points	CP1, CP4, CP7, CP11

Identifier	**query14.txt**
Description	Retrieve a list of N creative works, the ’things’ they are *about* and *mention*, their *categories*, the *modification* date, *thumbnail*, and *primary format*. Filter results by *audience type*, *document type* and *primary format*. This is a star-shaped query that involves nine triple patterns (eight joins).
Type	Aggregation
Choke Points	CP1, CP3, CP4, CP7, CP11

Identifier	**query15.txt**
Description	Retrieve a list of N creative works, the ’things’ they are *about* and *mention*, their *categories*, the *modification date*, *thumbnail* and *primary format*. Additional constraint is added by a filter that selects creative works containing a word which is not so commonly used. This is a star-shaped query (similar to query14) that involves nine triple patterns (eight joins). The filter expression involves both conjunction and disjunction of constraints.
Type	Aggregation, Full-text search
Choke Points	CP1, CP3, CP7, CP9, CP10

Identifier	**query16.txt**
Description	Retrieve a list of N creative works, their *title*, the ’things’ they are *about* and *mention*, their *categories* and *titles* containing certain word. This select query returns the first distinct 500 creative works, ordered by their tag, that have an optional national audience. The filter expression selects those creative works whose title contains the word “policy”. This is a large query that involves reasoning for the rdfs:subClassOf and rdfs:subPropertyOf subsumption hierarchies. This is a star-shaped query that involves five triple patterns.
Type	Aggregation, Full-text search
Choke Points	CP1, CP4, CP6, CP7, CP9, CP10

Identifier	**query17.txt**
Description	A drill-down query which retrieves a list of Creative Works found within a defined geo-range. Result is analyzed and a random element is selected whose geo-coordinates are again used to further narrow-down the range. Query is re-executed, produces a smaller result. Process is repeated several times, until no results or iterations limit has been reached.
Type	Geo-spatial, Drill-down

Choke Points	CP7, CP8, CP10, CP11

Identifier	**query18.txt**
Description	A drill-down query which retrieves a list of Creative Works that have been modified within a defined date-time range. Result is analyzed and another random element is selected. Its modification time is further used again to narrow-down the date-time range. Query is re-executed, produces a smaller result. Process is repeated several times, until no results or iterations limit has been reached.
Type	Aggregation, Drill-down
Choke Points	CP1, CP4, CP7

Identifier	**query19.txt**
Description	Retrieve most popular topics that creative works are *about* in a fixed date-time range, ordered by most-recent modification date. A time-range query.
Type	Aggregation
Choke Points	CP1, CP4, CP7, CP11

Identifier	**query20.txt**
Description	Construct creative works, which contain a certain word in their *title* or *description*. A full-text search query.
Type	Aggregation, Full-text search
Choke Points	CP1, CP3, CP4, CP9, CP11

Identifier	**query21.txt**
Description	Retrieve various properties of creative works in several iterations by adding up more filter conditions to previous ones. A faceted search query. Executing incrementally following steps: - FTS search on a random word in titles and adding a category type constraint - Group by year and month of creation - Group by tag (tagged entity being mentioned or tagged about) - Group by primary format - Finally selecting title and date of creation by adding a constraint on: specific date (day, month, year)
Type	Faceted search
Choke Points	CP1, CP4, CP7, CP9, CP11

Identifier	query22.txt
Description	Retrieve various properties of creative works in several iterations by adding up more filter conditions to previous ones. A faceted search query similar to query21, differs in final iteration, where constraint is added on a specific month of the year
Type	Faceted search
Choke Points	CP1, CP4, CP7, CP9, CP11

Identifier	query23.txt
Description	Retrieve various properties of creative works in several iterations by adding up more filter conditions to previous ones. A faceted search query similar to query21, differs in final iteration, where days and count of tags are returned grouped by tag
Type	Faceted search
Choke Points	CP1, CP4, CP7, CP9, CP11

Identifier	query24.txt
Description	Retrieve a time-line of relatedness between two entities in database. Query explores correlations of entities tagged together in creative works i.e. creates a history of interactions. Additional constraint of time interval of 6 months limits the result. An analytical query.
Type	Aggregation, Analytical
Choke Points	CP1, CP10, CP11

Identifier	query25.txt
Description	Query retrieves the 10 most popular entities related to a selected one. Having selected one entity, retrieves an ordered list of other entities that have participated in a correlation with that entity for maximum number of days : looks for co-occurrences and takes into account how long co-occurrences last. An analytical query.
Type	Aggregation, Analytical
Choke Points	CP1, CP2, CP4, CP7, CP10

Details on choke points classification can be found in D4.4.1 Analysis and classification of choke points, [6]

2.3.4 Query Templates and Substitution Parameters

Queries in SPB are using templates for their parameterization thus allowing to use parameter substitution when executed. SPB driver generates and initializes the substitution parameters prior to execution of workloads. The use of substitution parameters ensures that different runs of the benchmark will employ the same values and as a consequence will produce comparable results. The benchmark driver allows the user to configure the number of the substitution parameters per query which are produced in a random manner and saved along with generated data.

During the benchmark run, the driver goes through an initialisation phase where all parameters are loaded in memory and each query receives a corresponding set of substitution parameters. Once a query executes the last set of parameters, next execution will start to iterate from the beginning of parameters list.

2.4 Validation

Validation of query results is an important part of a benchmark, because it provides information to the benchmark user about correctness of query results and about inference capabilities of the database engine.

The SPB driver performs two types of validation :

- Validation of results - all queries that are part of the Basic and Advanced Workloads are validated for correctness of returned results. For that purpose a pre-generated validation dataset of one million triples is loaded into an empty RDF database and each of the queries with pre-generated set of substitution parameters are executed. Expected results are matched against returned result values and report is produced per-query and per-workload.
- Conformance Check - tests inference capabilities of the RDF database for conformance to the OWL2-RL rule-set. Again this validation test is performed on an empty database (OWL2-RL rules would take a larger amount of time to execute if validation test is performed on a larger dataset) where required validation data are loaded and checked. A report is produced with information about the results of conformance validation check.

2.5 Performance Metrics

Performance metrics produced by the SPB benchmark describe how fast an RDF database can execute queries (by simultaneously running aggregation agents) while at the same time running simultaneous editorial operations (by simultaneously running editorial agents) and operating over an amount of generated data in the RDF database. Data generator of the SPB is capable of producing data in different scales, thus allowing to have a various performance results for those scales.

Semantic Publishing Benchmark outputs two types performance metrics :

- Minimum, Maximum and Average execution times for each individual query and editorial operation during the whole benchmark run
- Average execution rate per second for all queries and editorial operations

Performance metrics are produced per-second during the benchmark run and saved to log files. Further all of the information related to query execution and query results is saved to log files with different level of details.

2.6 Execution Rules

Following is a list of execution rules, that must be kept while running the benchmark. They will provide an equal starting conditions for database systems under test and a steady ground for comparing results.

- run SPB on a single database node or a cluster - single instances of RDF databases or cluster configurations can be benchmarked
- it is up to the user to decide what types of indexes to use during the benchmark
- warm-up time should not exceed half of the benchmark time
- sustain editorial operations rate above a configured threshold level - a required editorial operations rate per second is to be constantly sustained during the benchmark. Value of editorial operations rate is to be defined by the benchmark user and reported in the benchmark results.
- benchmark time at sustained editorial operations rate - minimum time for running the benchmark should be at least 30 minutes after the moment of reaching editorial operations threshold level described above.
- time needed to reach editorial operations threshold level should not exceed the benchmark time. Once the editorial operations rate threshold has been reached, the SPB driver will constantly monitor if that level is sustained and will report in case of a drop below the configured threshold value
- sustaining editorial operations rate above a threshold level is not mandatory. If choosing to discard editorial operations rate threshold check (i.e. if choosing to discard previous three bullet points) - that should be stated explicitly.
- query substitution parameters are to be generated prior to running the benchmark
- run on a 'cold' database - right before running the benchmark, RDF Database System should be started and no data should be stored in it, except SPB's ontologies, reference knowledge datasets and generated synthetic data.
- serialization format for generated data should be context aware, e.g. N-Quads

2.7 Full Disclosure

This section provides guidelines for user of the benchmark, regarding items that need to be disclosed when modifications have been made. Not all RDF databases will support the full requirements of the Semantic Publishing Benchmark, in which case a test sponsor could modify some of the benchmark's definitions parameters. Such modifications might be: changes to query templates, disabling execution of queries, or altering behaviour of the data-generator. Any modification should be coordinated with the LDBC before implementing.

The benchmark driver has been designed to be highly configurable and many of its features have been made flexible for tuning. Items that need to be disclosed, if modified, are:

- properties in definitions.properties file. All properties can be modified to alter the behaviour of data-generator or the distribution of query / operations executions. Each modification of any property in that file needs to be disclosed;
- aggregation, editorial and conformance queries are saved to template files allowing the user to view or modify each one. e.g. a reason for modification could be to alter a query template (as long as end result produced by it is the same) to enable a non-standard feature provided by certain database engine. Modifications to query templates are acceptable only if modified version produces equal result to the original query. Each modification of query template different than corresponding one in '*aggregation_standard'* folders in basic and advanced query mixes must be disclosed;
- any modifications to the source code of the benchmark;

- any modifications to third-party library components used by the benchmark, e.g. updating a library component to a newer version;
- any modification to ontologies, reference datasets and the dictionary file (WordsDictionary.txt)

Additional items that need to be disclosed:

- The total dataset size
- Number of aggregation and editorial agents configured to execute queries and operations
- Benchmark driver's time to load generated synthetic data
- A report of query validation results should be included
- Benchmarks' run time (value of parameter *benchmarkRunPeriodSeconds*)
- Benchmarks' configuration files : test.properties and definition.properties used during the benchmark run
- Benchmarks' log files as well as generated query substitution parameter files
- The hardware used to run the benchmark including : CPU, Chipset, Physical memory, Hard disks - specifications and configuration, network adapters, etc.
- Operating System (for the benchmark test driver and for the database under test)
- Total cost of hardware listed at full price

2.8 **Auditing**

Execution of the LDBC Semantic Publishing Benchmark in sponsored test conditions will be audited in a manner consistent with all of the LDBC benchmarks. These auditing rules will be defined by the LDBC.

Over and above the general rules, the following key points need to be verified by the auditor:

- Input data size set and checked: the input data size is set in the configuration file passed to the test driver at start-up. After completion of the benchmark, it should be verified that at least this much data is present in the database - this can be achieved by executing a query similar to: SELECT (COUNT(*) as ?c) WHERE { GRAPH ?g { ?s ?p ?o} };
- Execution rules: auditor should confirm that execution rules are kept during the benchmark run
- Hardware: visually confirm the hardware specification and where possible login to the test machine(s) and execute appropriate utility software to interrogate the hardware;
- Collection of output values: the final output from the test driver contains the actual benchmarked performance values.

3 Instructions

3.1 Software Installation

The SPB benchmark driver is distributed as a file: semantic_publishing_benchmark -*.jar (suffix stands for the different workload versions - basic and advanced) accompanied by all necessary configuration files, ontologies and reference data. Sources are available for download on GitHub :

- https://github.com/ldbc/ldbc_spb_bm - *the benchmark driver*
- https://github.com/ldbc/ldbc_spb_optional_datasets - *optional reference datasets, can be added for enriching generated data, especially when generating large-scale synthetic data*

Installation of the benchmark driver requires building it from sources and moving all contents of the distribution folder to a selected destination (using the benchmark software from distribution folder is also acceptable).

Building the benchmark software requires Apache Ant and Java Development Kit 1.6 or later.

Following is a list of items that are the result of the benchmark build process :

- *semantic_publishing_benchmark.jar* - the benchmark driver
- *data/* – a folder containing all required data (ontologies, reference datasets and query templates, etc.) to run the benchmark
- *test.properties* – a configuration file with parameters for configuring the benchmark driver. Some properties need to be modified according to the database under test, others have default values ready for use
- *definitions.properties* – a definitions file containing various allocation parameters which define the behaviour of SPB driver, contents of this file are not to be modified by the benchmark user
- *readme.txt* – a text file with information about configuring and running the benchmark

3.2 Benchmark Configuration

3.2.1 Description of Operational Phases

The SPB Benchmark process consists of execution of several operational phases started in a sequence one after another. Each of the operational phases corresponds to a boolean property in *test.proeprties* file where they can be enabled or disabled.

Following is a list of the operational phases which are executed in the order listed below :

- ***initialization*** - default operational phase, always started first or when necessary, retrieves information about stored reference and generated data in the database
- ***loadOntologies*** **–** populates the RDF database with required ontologies. A required phase, can be done manually by uploading all .ttl files located at : /data/ontologies/*
- ***loadDatasets*** **–** populates the RDF database with required reference data. A required phase, can be done manually by uploading all .ttl files located at : /data/datasets
- ***generateCreativeWorks*** **–** generates the synthetic data used for benchmarking. Data is saved to files of pre-defined size (see *'generatedTriplesPerFile'*) and total number of triples (see *'datasetSize'*). Requires operational phases : *loadOntologies*, *loadDatasets*

- ***loadCreativeWorks*** **–** load generated synthetic data from previous phase into the RDF database. Some databases may not be able to support this phase, in that case generated data can be loaded manually. Requirement is to use context aware serialization format (e.g. N-Quads). Requires operational phases : *loadOntologies, loadDatasets, generateCreativeWorks.*

- ***generateQuerySubstitutionParameters*** **-** enables generation of query substitution parameters used during the benchmark phase. For each query a substitution parameters file is saved into the *'creativeWorksPath'* location. If no such files exist, then randomly picked parameter values will be used. Requires operational phases : *loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks*

- ***validateQueryResults*** **-** validates for correctness query results for editorial and aggregate operations against a validation dataset. To be run independently of the *runBenchmark* phase on an empty database. Requires operational phases : *loadOntologies, loadDatasets*

- ***warmUp*** **–** runs the aggregation agents for *'warmupPeriodSeconds'* seconds, results are not collected. Requires operational phases : *loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*

- ***runBenchmark*** **–** runs the benchmark for *'benchmarkRunPeriodSeconds'* seconds, results are collected. Editorial and aggregation agents are run simultaneously. Requires operational phases : *loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*

- ***runBenchmarkOnlineReplicationAndBackup*** **-** benchmark measures the performance under currently ongoing backup process. Verifies that certain conditions are met such as milestone points at which backup has been started. Requires additional implementation of provided shell script files (/data/enterprise/scripts) for each database with database-specific commands. Requires operational phases : *loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*. Also making a full backup is required prior to starting that operational phase. For a description of the steps performed in that operational phase see: *Annex.B. Description of Operational Phase : Online Replication And Backup*

- ***checkConformance*** **–** runs tests for conformance to the *OWL2-RL* rules. Requires operational phase : *loadOntologies*. To be run independently of the *runBenchmark* phase on an empty database (using OWL2-RL rule-set). Requires operational phase : *loadOntologies*

- ***cleanup*** **–** optional phase, can be used to clear all data from the database after benchmark run has completed. To be used with caution as it will erase all data in the database

3.2.2 Description of Benchmark Configuration Properties

Before running the benchmark some configuration properties need to be edited in configuration file : *test.properties*. Following Table 9 provides a description of all configuration parameters:

Table 9. Benchmark Configuration Properties

Property	**Description**
adjustRefDatasetsSizes	optional, if additional datasets have been added (files with extension '.adjustablettl'), then for each, a new .ttl file is created with adjusted size depending on the value of *datasetSize* parameter, *default value is true*
aggregationAgents	number of aggregation agents that will execute a mix of queries simultaneously, *requires updating*
allowSizeAdjustmentsOnDataModels	allows data generator to dynamically adjust the amount of

	correlations, clusterings and randomly generated models keeping a ratio of 1/3 for each in generated data model. This property overrides definitions.properties' parameters : *majorEvents, minorEvents, correlationsAmount*. *Default value is true*
benchmarkByQueryRuns	sets the number of queries which the benchmark phase will execute and stop. If value is greater than zero then parameter *benchmarkRunPeriodSeconds* is ignored. e.g. if set to 100, benchmark will stop after 100 queries have been executed
benchmarkRunPeriodSeconds	benchmark period in seconds, *requires updating*
creativeWorkNextId	sets the next ID of Creative Works. When running the benchmark driver to generate synthetic data in separate processes, in order to guarantee that all generated creative works will not overlap their IDs, add an increment in value ~ 2.6M for each 50M generated triples
creativeWorksInfo	name of file containing information about generated synthetic dataset (saved in *creativeWorksPath*), *default: dataset.info*
creativeWorksPath	path to location where generated synthetic data will be saved, *default: ./data/generated*
dataGeneratorWorkers	number of worker threads used by the data generator to produce synthetic data, *requires updating*
datasetSize	defines the size of generated synthetic data (triples) produced by the data-generator
definitionsPath	path to definitions.properties configuration file, *default: ./definitions.properties*
editorialAgents	number of editorial agents that will execute a mix of update operations simultaneously, *requires updating*
endpointUpdateURL	URL of SPARQL endpoint used for update operations, *requires updating*
endpointURL	URL of SPARQL endpoint provided by the RDF database, *requires updating*
generateCreativeWorksFormat	serialization format for generated synthetic data. Available options are : TriG, TriX, N-Triples, N-Quads, N3, RDF/XML, RDF/JSON, Turtle. Use exact names. Required are context aware serialization formats such as: N-Quads and TriG
generatedTriplesPerFile	number of triples per file. Generated synthetic data is saved to files with number of *generatedTriplesPerFile* each
generatorRandomSeed	use it to set the random seed for the data generator (default value is 0). Can be useful in cases when several benchmark drivers are started in separate processes to generate synthetic data faster. Should be used with *creativeWorkNextId* parameter for each new process
ontologiesPath	path to ontologies from reference knowledge data, *default: ./data/ontologies*

queriesPath	path to query templates, *default: ./data/sparql*
querySubstitutionParameters	number substitution parameters set that will be generated for each query, *default value is 100000*
queryTimeoutSeconds	defines the timeout for query execution, *default value is 300s*
referenceDatasetsPath	path to datasets from reference knowledge, *default: ./data/datasets*
systemQueryTimeoutSeconds	system queries timeout, default value is 1h (system queries are executed during the initialization operational phase and may take longer to complete depending on the size of stored synthetic data in the database)
minUpdateRateThresholdOps	defines the minimum rate of editorial operations per second which should be reached during the first N% (see *updateRateThresholdReachTimePercent)* of benchmark run time and should be kept during the rest of the benchmark. If set to zero, update rate threshold is ignored. e.g. if required update rate is set to 6.3 update operations per second, then benchmark will consider that value during its benchmark run phase and will report invalid results if that rate drops below 6.3 editorial operations per second
minUpdateRateThresholdReachTimePercent	defines the time window during which property value *minUpdateRateThresholdOps* should be reached. Default value is 0.1 (10%). e.g. if set to 0.1 then the update rate defined in *updateRateThresholdOps* should be reached during those first 10% of the benchmark run time
maxUpdateRateThresholdOps	defines the maximum rate of editorial operations per second. If set to zero that threshold is ignored
validationPath	location of validation data related to the validation operational phase, *default value can be used*
warmupPeriodSeconds	warmup period in seconds, *requires updating*
interruptSignalLocation	defines the location of the interrupt signal (a file) which is used to interrupt current driver's run when such interrupt signal has been set by another driver
enableEditorialOpeartionsValidation	enables validation of editorial operations (insert/delete) during benchmark run. Validation is performed on each *editorialOpsValidationInterval* operation, *default : true*
editorialOpsValidationInterval	sets the validation interval for editorial operations, *default : 100*
enableCompressionOnGeneratedData	enables gzip compression on generated data, *default: false*
benchmarkByQueryMixRuns	sets the count of query mixes that will be executed by the benchmark. If value is zero, then execution of query mixes will not be controlled by this parameter, *default:0*

3.2.3 Description of Benchmark Definitions Properties

Additional definition values have been stored in a property file called : *definitions.properties*. Values in that file are not to be modified by the benchmark user. Following Table 10 provides a description of each property :

Table 10. Benchmark Definitions Properties

Property	Description
aboutAndMentionsAllocation	defines the allocation amount of *about* or *mentions* in aggregation criteria
aboutsAllocations	defines the allocation amount of *about* tags in a creative work during generation of data
aggregationOperationsAllocation	defines the allocation distribution of aggregation queries starting from query1, query2, query3...etc.
correlationDuration	defines the duration of a correlation between two entities based on total data generation period
correlationEntityLifespan	defines the lifespan of each entity that participates in a correlation based on total generation period
correlationsAmount	defines the number of correlations between entities that will be generated in the dataset
correlationsMagnitude	defines maximum amount of Creative Works that will be generated for a particular correlation in a single day
creativeWorkTypesAllocation	defines the allocation amount of Creative Work types during data-generation
dataGenerationPeriodYears	defines the period (in years) in generated data, starting from *seedYear*
editorialOperationsAllocation	defines the allocation distribution of INSERT, UPDATE, DELETE queries that each editorial agent will execute
entityPopularity	defines popularity ratio of an entity in the reference datasets
exponentialDecayRate	defines the exponential decay rate. Used values to be in range 0.01 (for gentle slope) to 1 (for steeper slope)
exponentialDecayThresholdPercent	defines the threshold in percents of exponential decay, below that threshold values will be ignored. Threshold is defined as the ratio of : *currentExponentialDecayValue* / *exponentialDecayUpperLimitOfCWs*
exponentialDecayUpperLimitOfCWs	defines the maximum number of Creative Works that an entity can be tagged about. The exponential decay function will start from that defined value
majorEvents	defines the maximum number of *major* events that could happen during data generation period. Each major event will be tagged by a number of Creative Works which will decay exponentially in time
maxLat	defines maximum latitude, related to geo-spatial data
maxLong	defines maximum longitude, related to geospatial data
mentionsAllocations	defines the allocation amount of *mention* tags in a creative work

mileStoneQueryPosition	defines the position in terms of percent in the benchmark run time, at which a milestone query is executed, related to Online and Replication Benchmark operational phase
minLat	defines minimum latitude, related to geo-spatial data
minLong	defines minimum longitude, related to geo-spatial data
minorEvents	defines the maximum number of *minor* events that could happen during data generation period. Each minor event will be tagged by a number of Creative Works which will decay exponentially in time
seedYear	defines a seed year that will be used for generating the Creative Works date properties
usePopularEntities	defines allocation amount of popular entities to be used for tagging in Creative Works or in aggregation queries
queryPools	Defines a pool of queries, where each pool contains unique set of queries. During query execution, each query from the pool gets executed just once until all queries have been executed. The query pool is defined by a set of curly braces {}. If empty value is used, then query pool is not created and all queries are executed according to distributions defined in parameter 'aggregationOperationsAllocation'

3.3 **Benchmark Operation**

3.3.1 Prepare RDF Database for tests with SPB

Prior to using the benchmark, following setup for the RDF Database is required :

- RDFS rule-set - RDFS is the minimum rule-set requirement for the RDF database repository
- Enable context indexing (if available)
- Enable geo-spatial indexing (if available)
- Enable text-indexing (if available)

3.3.2 Starting the Benchmark Driver

For starting each action, i.e. Generate data, Run the benchmark, Validate query results, Check OWL2-RL conformance, etc. one sets up appropriate configuration options (in test.properties file; for each phase a new .properties file can be created) and executes following command line in console :

java -jar semantic_publishing_benchmark.jar <test.properties>

Note :

- *A fair amount of memory may be required for java maximum heap size, e.g. -Xmx8G*
- *LDBC Semantic Publishing Benchmark requires Java Runtime Environment 1.6 (JRE) or higher.*
- *Each of the operational phases can be executed independently of the next one, i.e. one may decide to execute the first phase (loadOntologies) only and stop. Then continue with next phase from the*

sequence (loadDatasets) and stop, etc. Once all previous phases have been completed next one can be executed without continuing further.

3.3.3 Configure SPARQL Endpoint

Update following configuration properties :

- endpointURL
- endpointUpdateURL

3.3.4 Generate Data

To generate synthetic data, that will be used by SPB driver for benchmarking, following configuration properties need to be set/enabled in test.properties file :

- dataGeneratorWorkers=N *(data generator thread can be set equal to the number of CPU cores)*
- adjustRefDatasetsSizes=true
- allowSizeAdjustmentsOnDataModels=true
- loadReferenceDatasets=true
- loadOntologies=true
- generateCreativeWorks=true
- datasetSize=M *(set the total number of triples which Data Generator will produce)*
- generatedTriplesPerFile=K *(select number of generated triples per file, generated data is saved to files)*
- generateCreativeWorksFormat=n-quads
- generateQuerySubstitutionParameters=true

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets.*

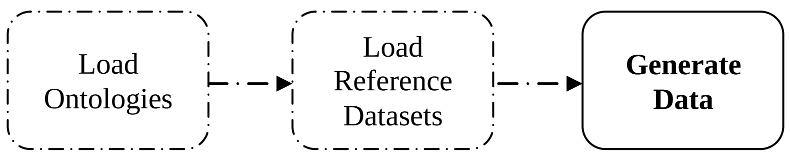


Figure 5. Generate Data

Note : it is not required to load ontologies and reference datasets for each data generation. Once ontologies and reference data have been loaded (loadOntologies, loadDatasets need to be executed just once), data generation can be started without running previous two phases.

3.3.5 Load Data

SPB driver can load generated synthetic data into the RDF Database by uploading it to the SPARQL end-point :

- loadCreativeWorks=true

However, generated data can be loaded manually to the RDF database by uploading generated files in *creativeWorksPath.*

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets, generateCreativeWorks*

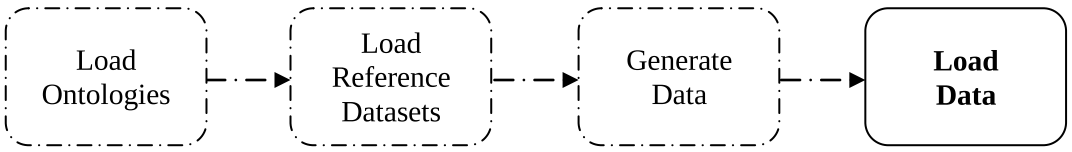


Figure 6. Load Data

Note : it is not required to generate data for each load. Once data has been generated (loadOntologies, loadDatasets, generateCreativeWorks need to be executed just once), loading of data can be started without running previous three phases.

3.3.6 Generate Query Substitution Parameters

This action is intended to start together with data generation, but can also be executed independently.

- generateQuerySubstitutionParameters=true

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets, generateCreativeWorks, loadCreativeWorks (i.e. generated data should be loaded in RDF database)*

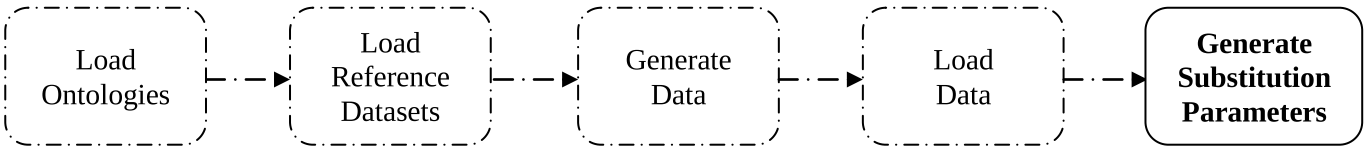


Figure 7. Generate Query Substitution Parameters

Note : it is not required to generate and load data for each generation of substitution parameters. Once data has been loaded (loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks need to be executed just once), generation of query substitution parameters can be started without running previous four phases.

3.3.7 Start the Warm-up

To start the Warm-up of the database, following properties need to be configured :

- aggregationAgents
- editorialAgents
- warmupPeriodSeconds
- benchmarkRunPeriodSeconds
- updateRateThresholdOps *(if set to zero, update rate threshold limit will be disabled, but results will not meet the benchmark execution rules)*
- updateRateThresholdReachTimePercent
- warmUp=true
- runBenchmark=false

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*

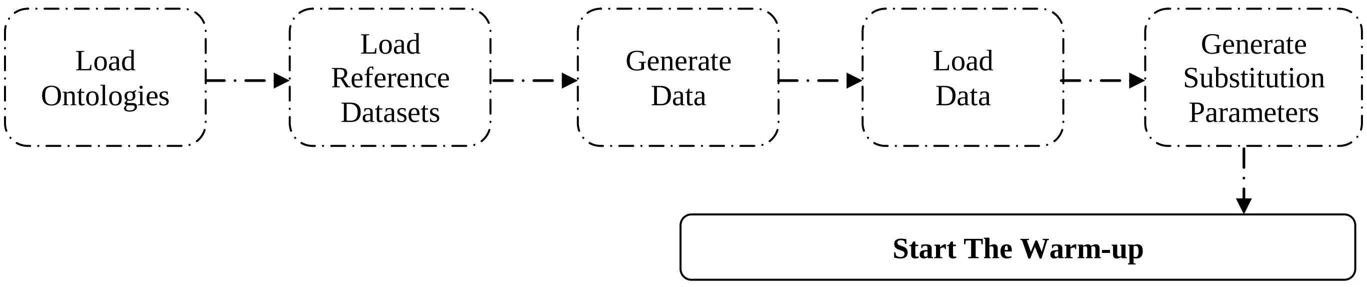


Figure 8. Start The Warm-up

Note : If generating and loading the data into the RDF database have been completed (as well as generating the query substitution parameters) earlier, then all configuration options related to generating and loading the data can be disabled when running benchmark operational phases.

3.3.8 Start the Benchmark Run

To start the benchmark run, following properties need to be configured :

- aggregationAgents
- editorialAgents
- warmupPeriodSeconds
- benchmarkRunPeriodSeconds

- updateRateThresholdOps *(if set to zero, update rate threshold limit will be disabled, but results will not meet the benchmark execution rules)*
- updateRateThresholdReachTimePercent
- warmUp=false
- runBenchmark=true

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*

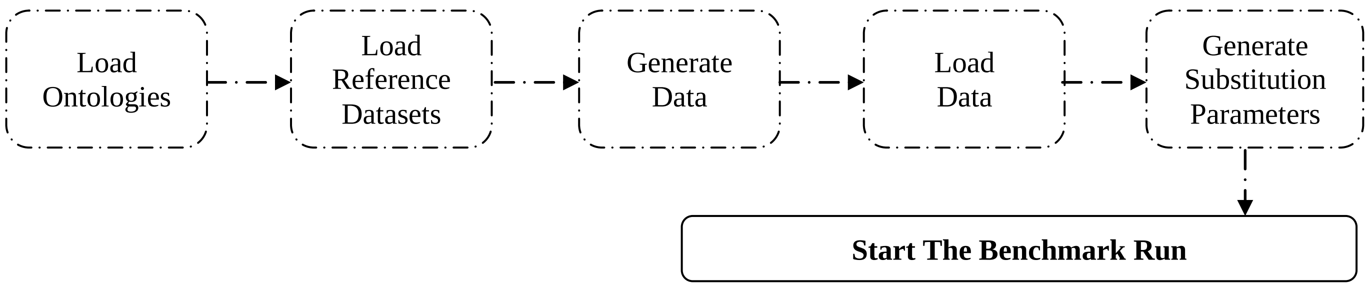


Figure 9. Start The Benchmark Run

Note : If generating and loading the data into the RDF database have been completed (as well as generating the query substitution parameters) earlier, then all configuration options related to generating and loading the data can be disabled when running benchmark operational phases.

3.3.9 Start Online Replication and Backup Benchmark

Prior to running, each RDF database vendor should implement all provided scripts (/data/enterprise/scripts) with database specific commands, also a full backup prior to running the benchmark for later restore point is required. See Annex.B for description of steps executed during the Online Replication and Backup action

- runBenchmark=false
- runBenchmarkOnlineReplicationAndBackup=true
- benchmarkByQueryRuns=N (benchmark will run until N number of queries have been executed, property *benchmarkRunPeriodSeconds* is ignored)

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets, generateCreativeWorks, loadCreativeWorks, generateQuerySubstitutionParameters*

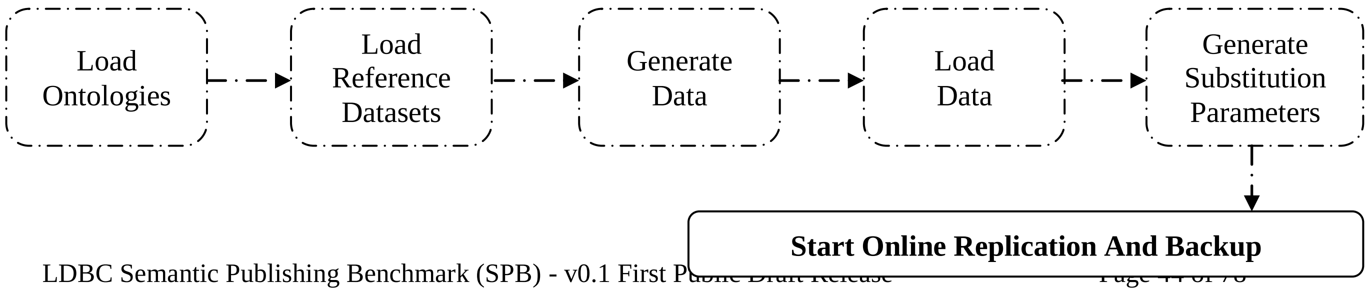

Load Ontologies
Load Reference Datasets
Generate Data
Load Data
Generate Substitution Parameters
Start Online Replication And Backup

Figure 10. Start Online Replication And Backup

3.3.10 Run Validation of Query Results

Starting that action requires an empty RDF Database with RDFS rule-set. It should be run independently of benchmark phases and no generation of data is required. Can be run before benchmark phases in order to have validated RDF database's capabilities to return correct query results.

- loadOntologies=true
- loadDatasets=true
- validateQueryResults=true

Required operational phases that should be enabled : *loadOntologies, loadReferenceDatasets*

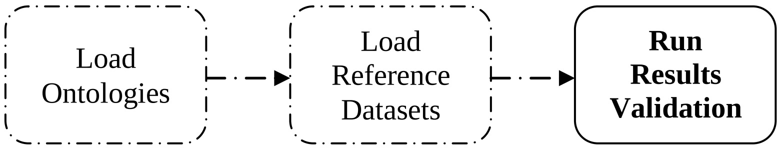


Figure 11. Run Query Results Validation

3.3.11 Run OWL2-RL Conformance Test

Starting that action requires an empty RDF Database and configured for OWL2-RL rule-set.

- loadOntologies=true
- checkConformance=true

Required operational phases that should be enabled : *loadOntologies*

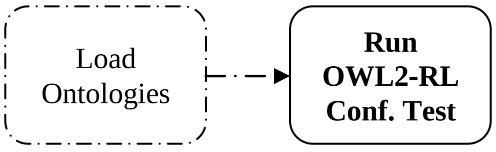


Figure 12. Run OWL2-RL Conformance Test

3.4 Results Gathering

Benchmark results are produced during the benchmark run phase and consist of constantly updated and saved status to a set of log files. Each status update is giving information about : current state of each type of agents, how many queries or operations have been executed, what is the current operations or query execution rate. Failures to execute an operation or queries (timeouts) are logged too.

Results are also displayed on the console during the whole operation of the benchmark driver. As mentioned earlier, logs of those results as well as more detailed data for the benchmark run are saved to three different types of log files, limited in size of 250 Mb each. Once that limit has been reached, a new file is created :

- ***semantic_publishing_benchmark_queries_brief.log*** **-** stores a brief log of each executed query or operation, returned number of results, execution time
- ***semantic_publishing_benchmark_queries_detailed.log*** **-** stores a detailed log for each query or operation - contents and results
- ***semantic_publishing_benchmark_results.log*** **-** stores a summary of results of the benchmark, updated each second during the whole benchmark run - queries and operations execution rate, details about execution times for each of the queries

Logging details can be controlled by a configuration file: *log4j.xml* saved in the distributed benchmark driver (semantic_publishing_benchmark.jar). After modifying *log4j.xml*, benchmark driver must be updated with contents of the new xml file.

Summary of the benchmark result includes:

- total seconds of benchmark run time, or total executed aggregation queries
- total number of editorial and aggregation agents
- average number of editorial and aggregation operations and queries per second
- average, min and max time of execution per operation and query

Results that should be gathered after the benchmark run has completed *(semantic_publishing_benchmark_results.log)* are :

- editorial agents number
- aggregate agents number
- time (*benchmarkRunPeriodSeconds)* or executed queries (in case *benchmarkByQueryRuns* has been used)
- average operations per second and their total amount
- average queries per second and their total amount
- optional, average, min and max execution times for each editorial operation
- optional, average, min and max execution times for each query

Benchmark driver will report at the end of its run if benchmark results are valid in case benchmark execution rules have been fulfilled.

4 References

[1] http://www.bbc.co.uk/blogs/bbcinternet/2010/07/bbc_world_cup_2010_dynamic_sem.html

[2] http://www.bbc.co.uk/blogs/bbcinternet/2012/04/sports_dynamic_semantic.html

[3] http://www.slideshare.net/JemRayfield/dsp-bbcjem-rayfieldsemtech2011

[4]https://speakerdeck.com/jemrayfield/bbc-dynamic-semantic-publishing-sport%7Colympics-semtechbiz-uk

[5] http://www.ldbc.eu:8090/download/attachments/1671227/D2.2.2-final.pdf

[6] http://www.ldbc.eu:8090/download/attachments/1671227/LDBC_2_2_1_final_v2.pdf

5 Appendices

5.1 Appendix. A - Ontologies

CreativeWork 0.9 : this ontology defines the classes and properties for creative works. Figure 12 shows the classes and properties for the creative works ontology. A creative work (also called a journalistic asset) is something created by the publisher’s editorial team. It is not a representation of the item itself (which could be text, a photo, a video, an audio recording, etc), rather the metadata that describes it and its location (in an appropriate content management system). A creative work has a title, shortTitle, exactly one description, modification and creation date (properties cwork:description, cwork:dateModified and cwork:dateCreated respectively). Property cwork:liveCoverage indicates that the creative work is the live coverage of an event. It has zero or more audiences (property cwork:audience), instances of class cwork:Audience, a single format (property cwork:primaryFormat), instance of class cwork:Format. Creative works can be tagged (property tag) by anything (instance of class core:Thing), and is associated with exactly one category (property cwork:category) that can be any URI. Properties cwork:about and cwork:mentions are subproperties of property cwork:tag. Class cwork:Audience describes the kinds of audience for the story presented by a creative work; instances of this class are cwork:NationalAudience, cwork:InternationalAudience. Class cwork:Format collects all different kinds of formats of a creative work. These are PictureGalleryFormat, AudioFormat, InteractiveFormat, VideoFormat, TextualFormat.

A creative work has exactly one thumbnail (property cwork:thumbnail). Thumbnails have at most one type (property cwork:thumbnailType), instance of class cwork:ThumbnailType: namely StandardThumbnail, CloseUpThumbnail, FixedSize66Thumbnail, FixedSize228Thumbnail and FixedSize466Thumbnail and a text description (property cwork:altText).

There are different types of creative works: news article (class cwork:NewsArticle), a programme (class cwork:Programme) and a blog post (class cwork:BlogPost); these types are represented using the rdfs:subClassOf RDFS property and are subclasses of cwork:CreativeWork class.

Company 1.4 : This ontology describes the relationship between the web documents produced by a content management system (class bbc:WebDocument), BBC products (class bbc:Product) and creative works (class cwork:CreativeWork). A BBC product can be a blog (bbc:Blogs), education (bbc:Education), news (bbc:News), music (bbc:Music) or sport (bbc:Sport), all instances of bbc:Product. A web document (instance of class bbc:WebDocument) has an associated product (property bbc:product). These documents have exactly one primary topic (property core:primaryTopic) that can be anything (instance of core:Thing). Such documents are presented in at most one platform such as bbc:Mobile, bbc:HighWeb, instances of class bbc:Platform. A creative work can be the primary content (bbc:primaryContentOf) of at least one and at most two web documents. Finally, a BBC web document has an associated product, the former being the primary content of a creative work. Figure 13 presents the classes and properties of the company ontology.

Core Concepts 0.6 : defines the main classes used in BBC datasets such as core:Person , core:Place, core:Event, core:Organization and core:Theme. These are all subclasses of the class core:Thing. core:Thing is defined as equivalent of class (using the owl:sameAs property) owl:Thing, that is the “class of all individuals in the OWL world”. An instance of class core:Thing has short, preferred labels (properties core:shortLabel, core:preferredLabel), disambiguation hint (property core : disambiguationHint:). Finally, each instance of class core:Thing has an associated URI slug (property core:slug) that is the fragment of a URI that uniquely identifies a resource within a domain. For instance, in the case of Wikipedia the URI slug for the entry Stoat: http://en.wikipedia.org/wiki/Stoat is “Stoat”. core:primaryTopicOf property is the inverse of core:primaryTopic. An instance of core:Thing can be the primary topic of more than one web documents. Properties bbc:twitter, bbc:facebook and bbc:officialHomepage are subproperties (modelled using the RDFS built-in rdfs:subPropertyOf relationship) of core:primaryTopicOf that are used to indicate the different kinds of web documents that have a “thing” as primary content. The main classes and properties of the core concepts ontology are shown in Figure 14.

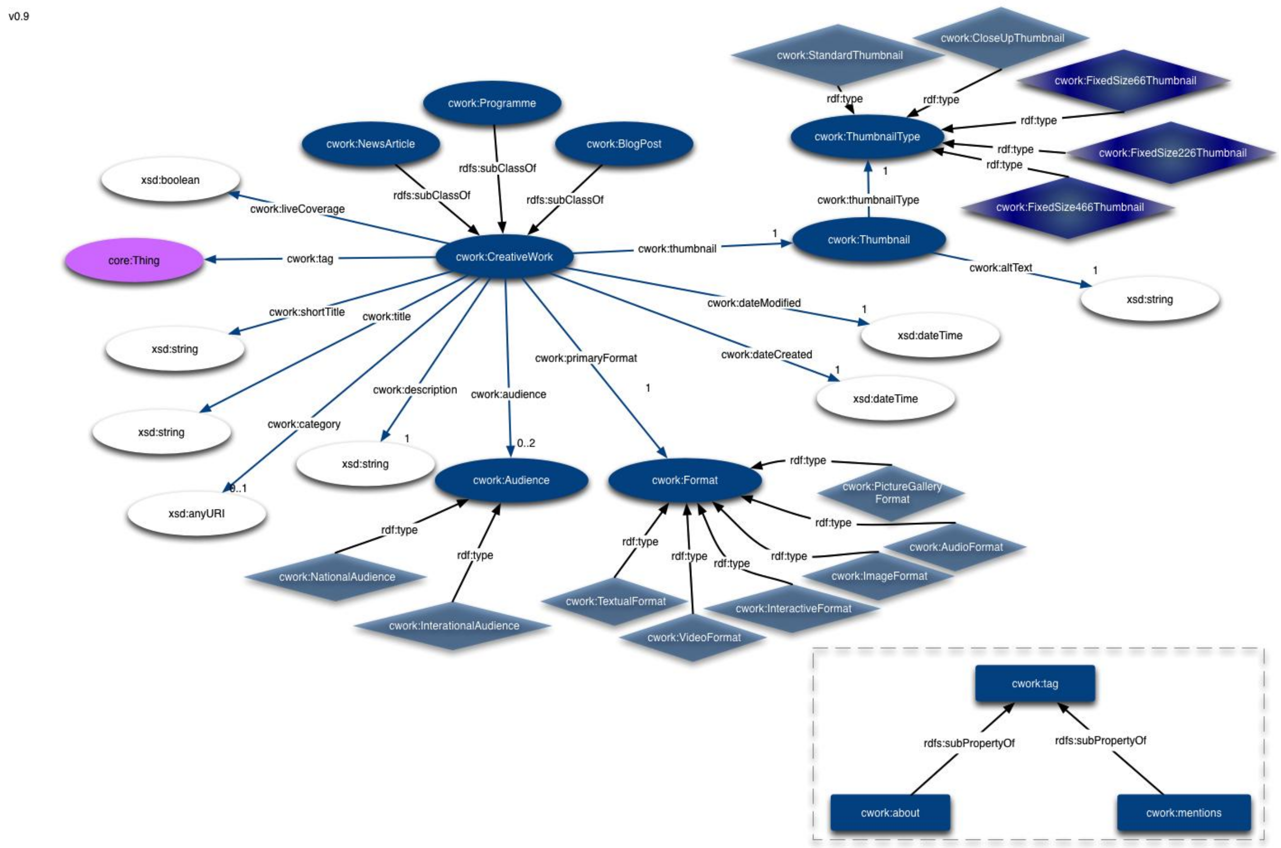


Figure 13. Creative Work Ontology 0.9

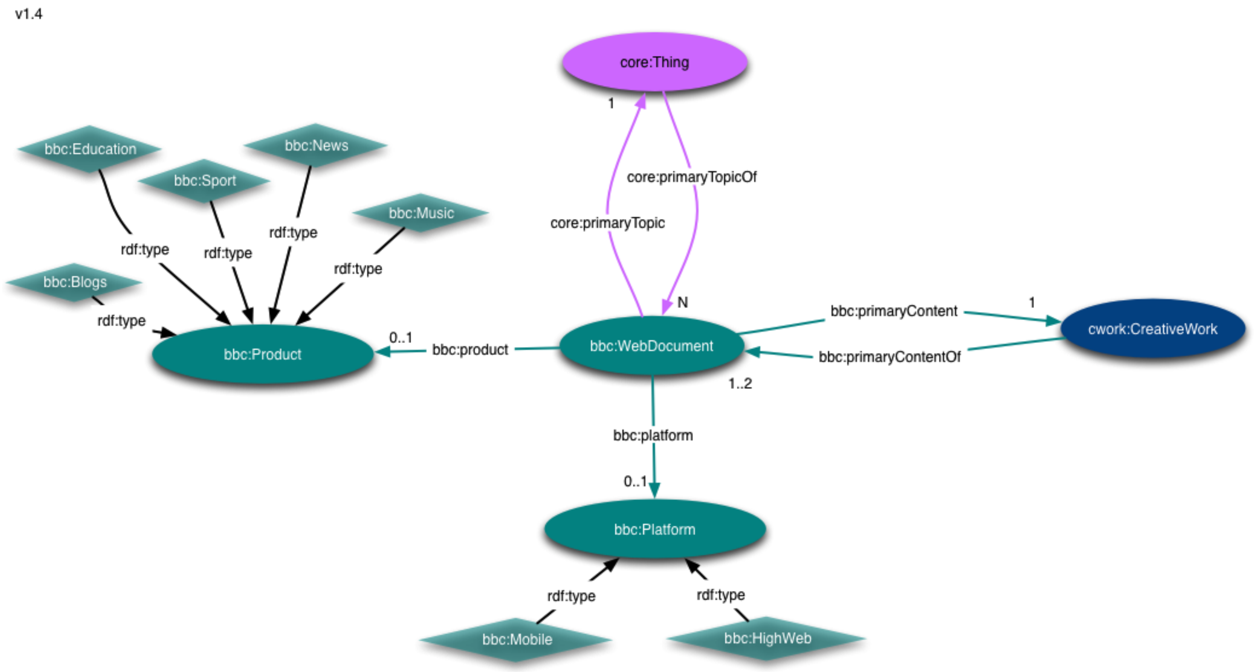


Figure 14. Company Ontology 1.4

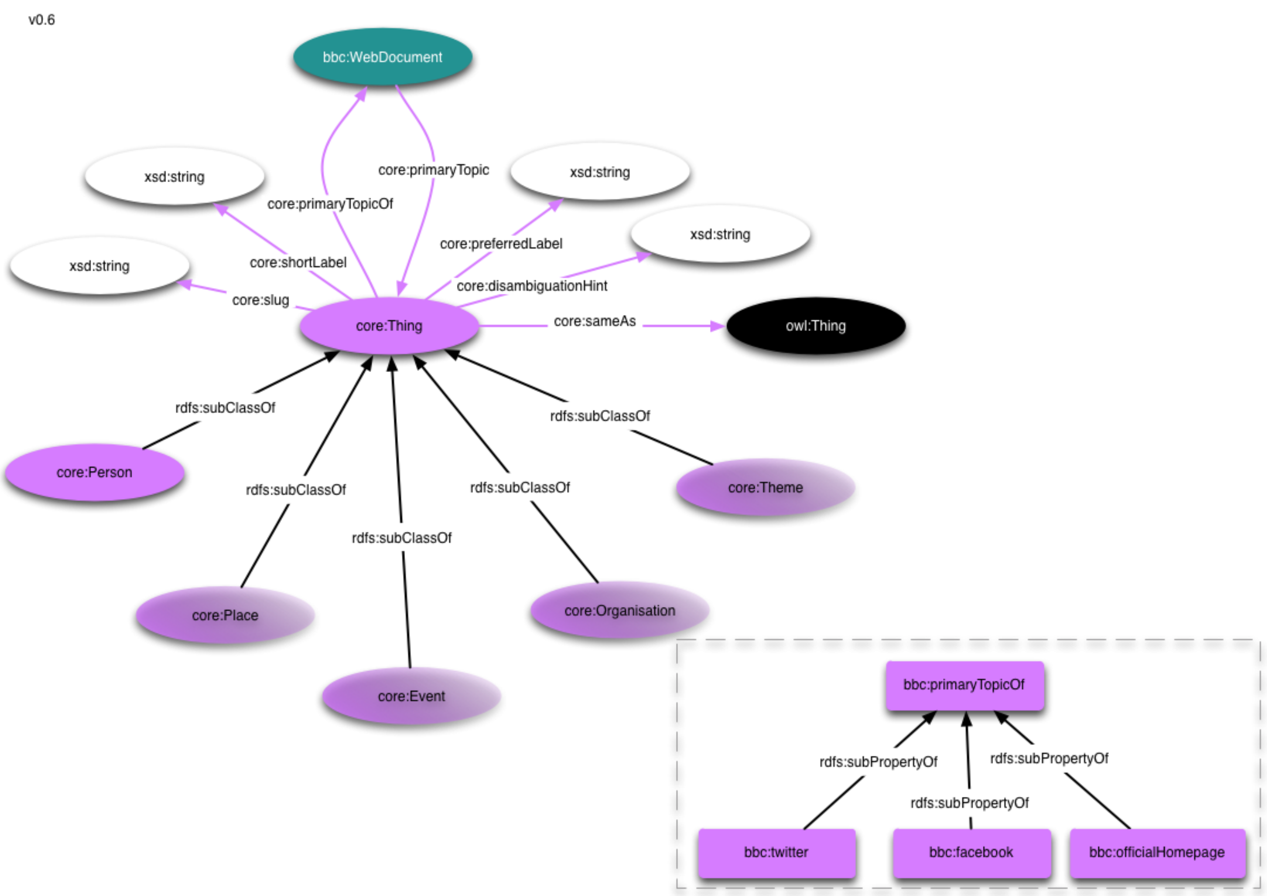


Figure 15. Core Concepts Ontology 0.6

CMS 1.2 : the ontology is used for interpreting locators into various specialized content management systems. A creative work and a BBC “thing” can have multiple such locators, that can be sport-stats (class cms:Sports), music bootstrap (class cms:MusicBootstrap), iscript (class cms:iScript) and content api (class cms:ContentApi). The CMS classes are all subclasses of cms:Locator. The CMS ontology used in BBC is shown in Figure 15.

Person 0.2 : describes information related to persons, instances of class person:Person, considered to be a subclass of class bbc:Thing. A person can have a role, a first and a last name (properties person:role, person:firstName, person:lastName). The person ontology is shown in Figure 16.

Provenance 1.1 : specifies the main concepts and properties used to describe versioning and change log information for the BBC datasets. The main class of the ontology is provenance:Graph that carries information about different versions of a dataset. The information that carries a provenance graph are the owner and provider of the dataset (properties provenance:owner, provenance:provider) that can be any web resource. The provision date, the reason a dataset changed and its version, a canonical location and a previous hash version (properties provenance:provided, provenance:changeReason, provenance:version, provenance:canonicalLocation, provenance:previousVersionHash). The ontology is shown in Figure 17.

Tagging 1.0: this ontology is used for connecting creative works with concepts from domain ontologies. The main concept is the tagging:TagConcept which is a subclass of bbc:Thing. A tag concept is associated with a set of tags (instances of class tagging:TagSet) and a locator of a content management system (instance of class cms:Locator). The ontology is shown in Figure 18.

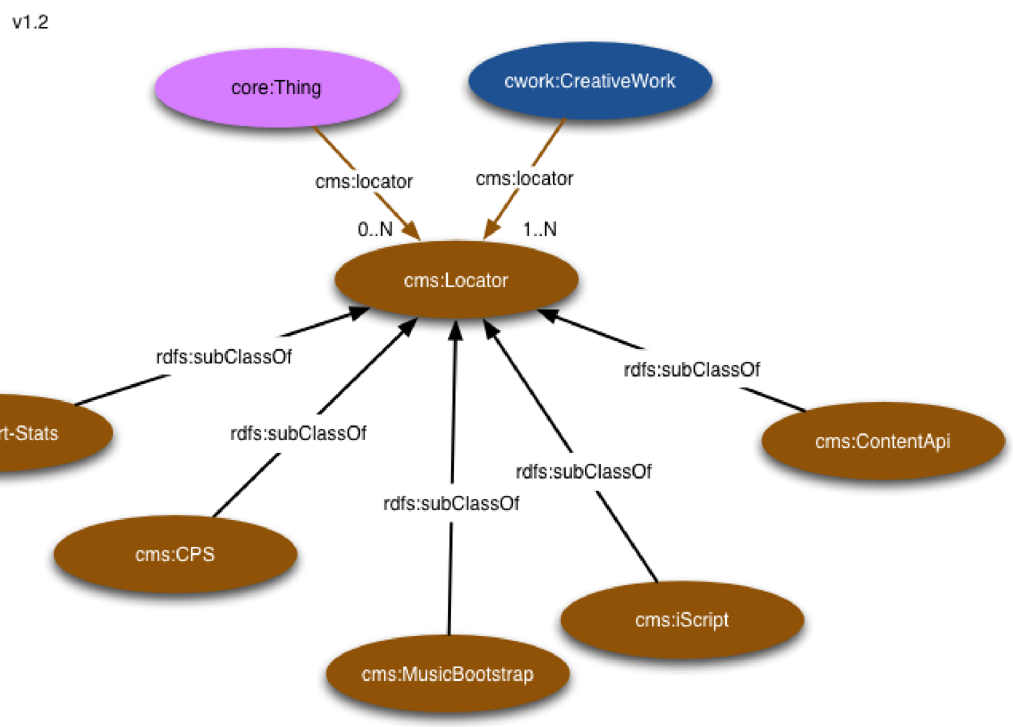


Figure 16. CMS Ontology 1.2

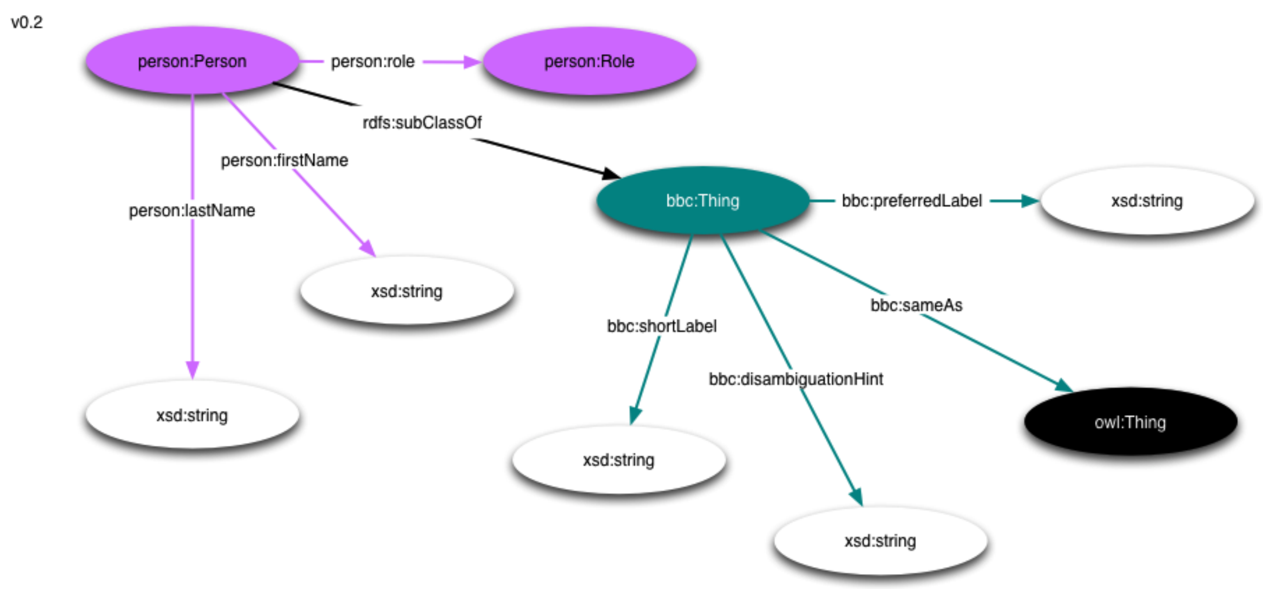


Figure 17. Person Ontology 0.2

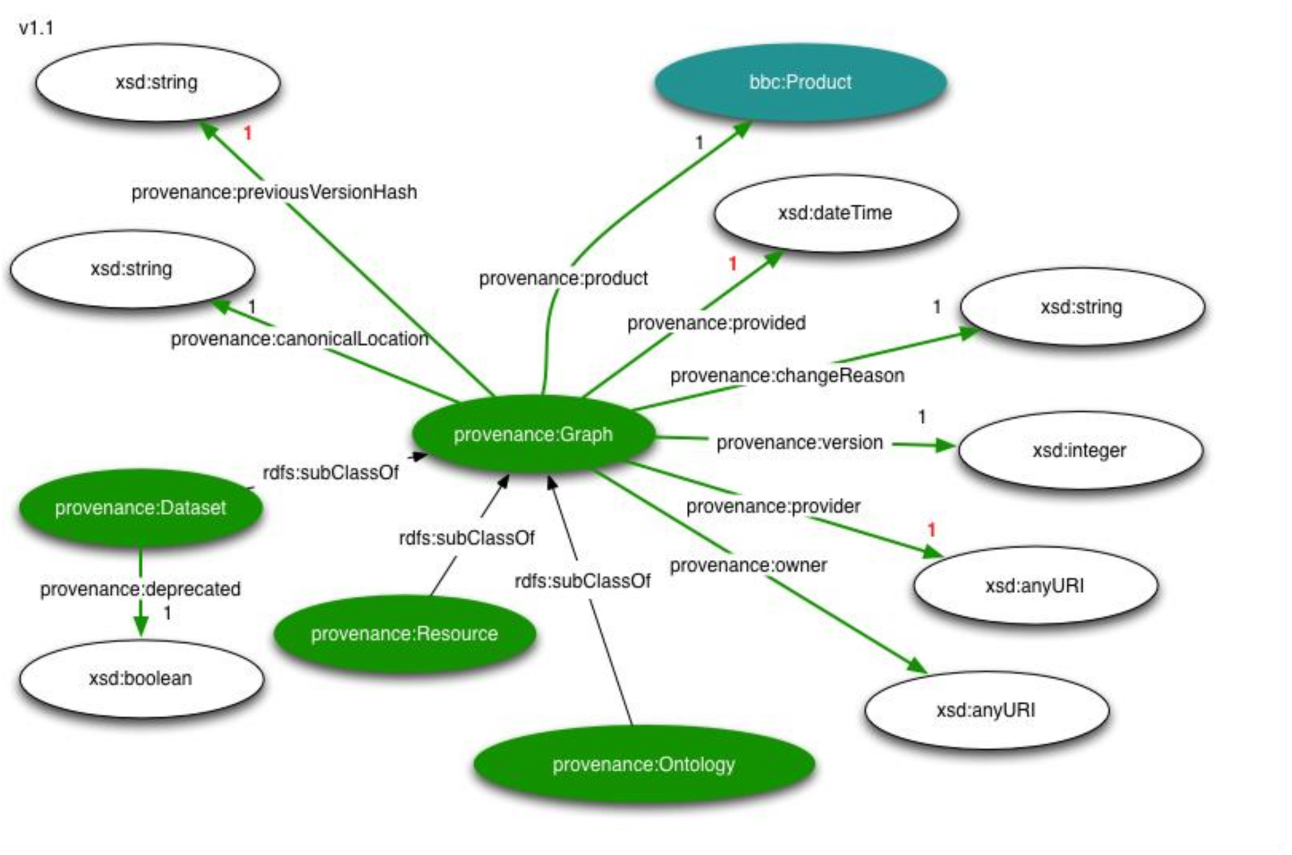


Figure 18. Provenance Ontology 1.1

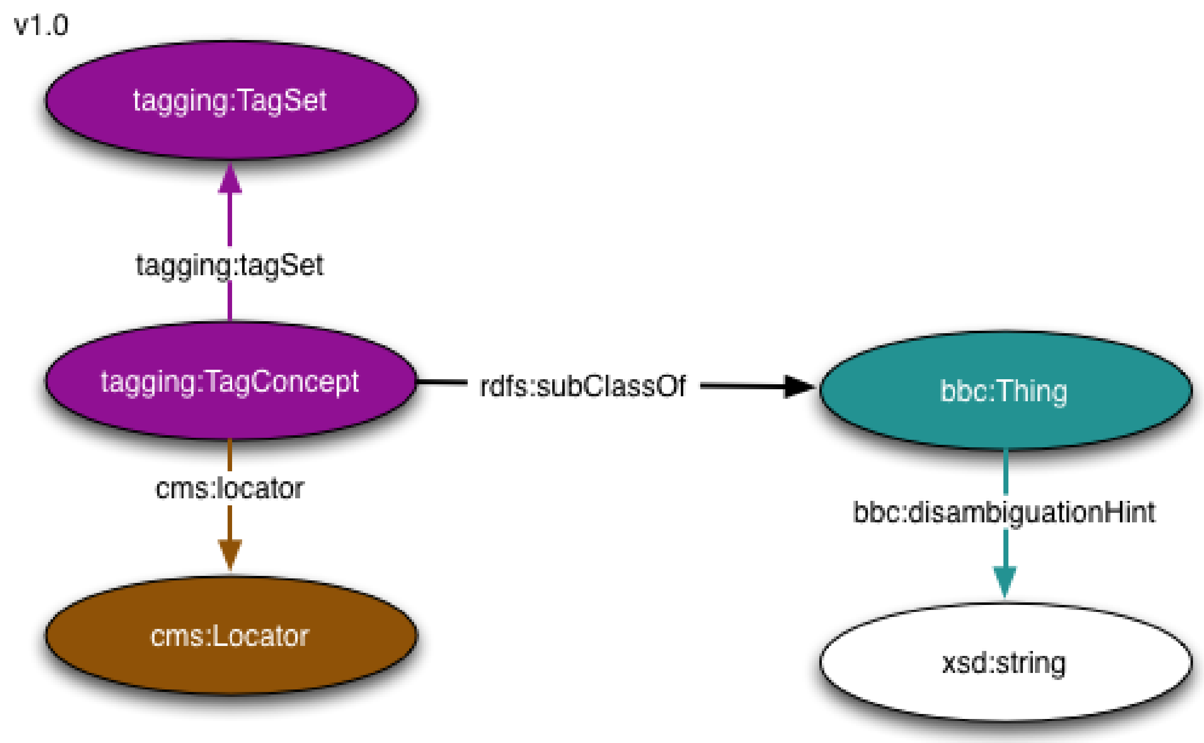


Figure 19. Tagging Ontology 1.0

In addition to the aforementioned general ontologies, concepts from domain ontologies are used as tagging concepts: the sports ontology contains concepts for describing sports, competitions and sporting events, the curriculum ontology describes entities in academia and finally the news ontology describes the basic concepts that a creative work can be tagged with.

Figure 19 presents an overview of the ontologies that comprise the BBC schema. The main classes of each of the ontologies are shown, in a colour-coded fashion to indicate the ontology where they come from.

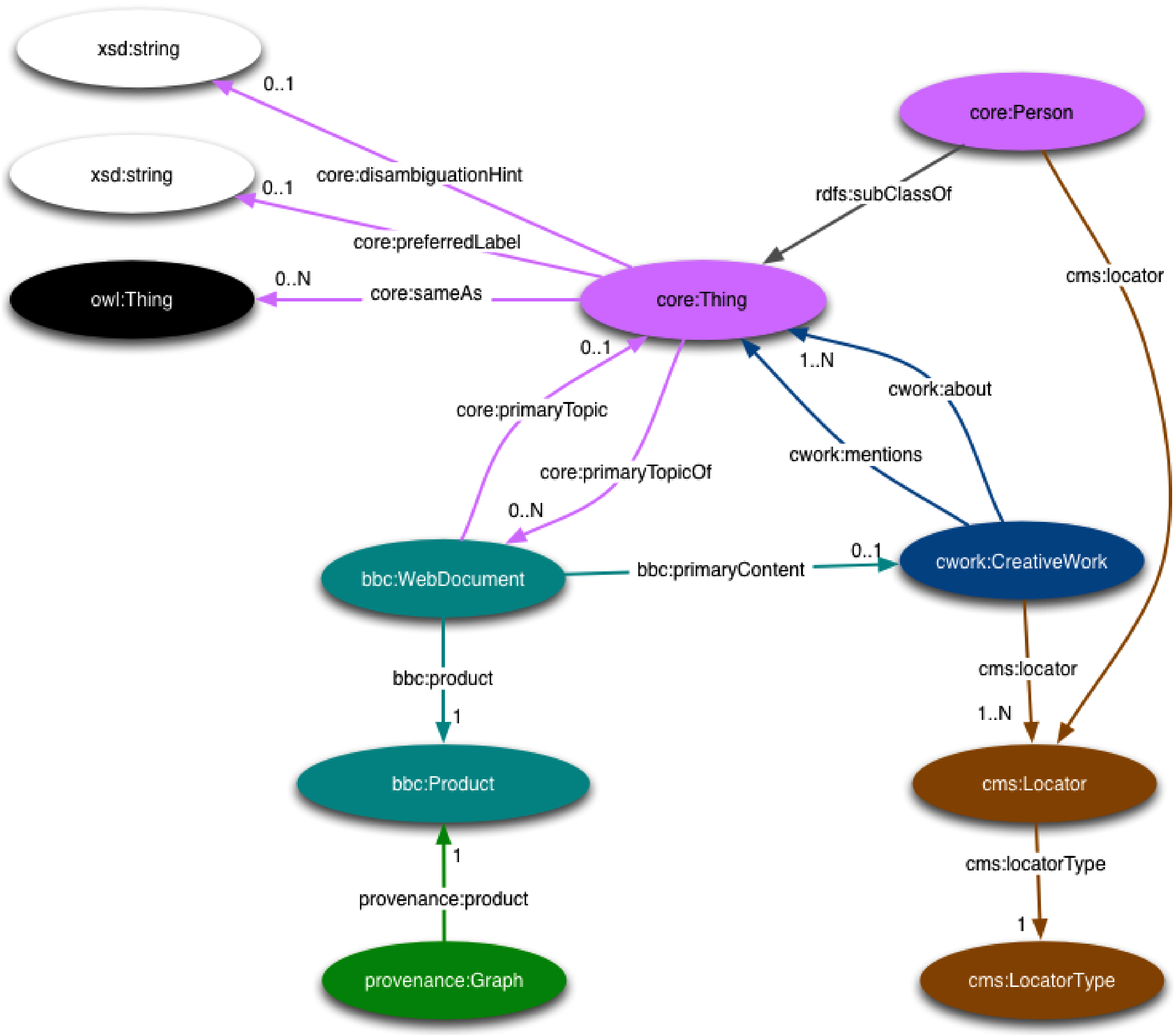


Figure 20. Overview of Ontologies 0.2

5.2 Appendix. B - Queries Listings

Following are listings of all queries from both Basic and Advanced query-mixes, extracted from query logs with initialized values for their template parameters.

Basic Query-mix

Identifier	**query1.txt**

Listing

```
CONSTRUCT {
  ?creativeWork a cwork:CreativeWork ;
    a ?type ;
    cwork:title ?title ;
    cwork:shortTitle ?shortTitle ;
    cwork:about ?about ;
    cwork:mentions ?mentions ;
    cwork:dateCreated ?created ;
    cwork:dateModified ?modified ;
    cwork:description ?description ;
    cwork:primaryFormat ?primaryFormat ;
    bbc:primaryContentOf ?webDocument .
  ?webDocument bbc:webDocumentType ?webDocType .
  ?about rdfs:label ?aboutLabel ;
    bbc:shortLabel ?aboutShortLabel ;
    bbc:preferredLabel ?aboutPreferredLabel .
  ?mentions rdfs:label ?mentionsLabel ;
    bbc:shortLabel ?mentionsShortLabel ;
    bbc:preferredLabel ?mentionsPreferredLabel .
  ?creativeWork cwork:thumbnail ?thumbnail .
  ?thumbnail a cwork:Thumbnail ;
    cwork:altText ?thumbnailAltText ;
    cwork:thumbnailType ?thumbnailType .
}
WHERE {
  {
    SELECT ?creativeWork
      WHERE {
              ?creativeWork cwork:about <http://www.wikidata.org/wiki/Q263166> .
              ?creativeWork a cwork:CreativeWork ;
              cwork:dateModified ?modified .
      }
      ORDER BY DESC(?modified)
      LIMIT 10
  }
  ?creativeWork a cwork:CreativeWork ;
                a ?type ;
                cwork:title ?title ;
```

```
          cwork:dateModified ?modified .
  OPTIONAL { ?creativeWork cwork:shortTitle ?shortTitle . }
  OPTIONAL { ?creativeWork cwork:description ?description . }
  OPTIONAL { ?creativeWork cwork:about ?about .
        OPTIONAL { ?about rdfs:label ?aboutLabel . }
        OPTIONAL { ?about bbc:shortLabel ?aboutShortLabel . }
        OPTIONAL { ?about bbc:preferredLabel ?aboutPreferredLabel . }
       }
  OPTIONAL {
        ?creativeWork cwork:mentions ?mentions .
        OPTIONAL { ?mentions rdfs:label ?mentionsLabel . }
        OPTIONAL { ?mentions bbc:shortLabel ?mentionsShortLabel . }
        OPTIONAL { ?mentions bbc:preferredLabel ?mentionsPreferredLabel . }
       }
  OPTIONAL { ?creativeWork cwork:dateCreated ?created . }
  OPTIONAL { ?creativeWork cwork:primaryFormat ?primaryFormat . }
  OPTIONAL {
       {
        ?webDocument bbc:primaryContent ?creativeWork .
        OPTIONAL { ?webDocument bbc:webDocumentType ?webDocType . }
       }
       UNION
       {
        ?webDocument ^bbc:primaryContentOf ?creativeWork .
        OPTIONAL { ?webDocument bbc:webDocumentType ?webDocType . }
       }
  }
  OPTIONAL { ?creativeWork cwork:thumbnail ?thumbnail .
        OPTIONAL { ?thumbnail cwork:altText ?thumbnailAltText . }
        OPTIONAL { ?thumbnail cwork:thumbnailType ?thumbnailType . }
  }
}
```

Identifier	**query2.txt**

Listing	CONSTRUCT {
	  ?cWork a cwork:CreativeWork ;
	    a ?type ;
	    cwork:title ?title ;
	    cwork:dateCreated ?dateCreated ;
	    cwork:dateModified ?dateModified ;
	    cwork:about ?about ;
	    bbc:primaryContentOf ?pco .

```
  ?pco bbc:webDocumentType ?webDocType .
}
WHERE {
  ?cWork a cwork:CreativeWork ;
    a ?type ;
    cwork:title ?title .
  ?type rdfs:subClassOf cwork:CreativeWork .
  OPTIONAL { ?cWork cwork:dateCreated ?dateCreated . }
  OPTIONAL { ?cWork cwork:dateModified ?dateModified . }
  OPTIONAL { ?cWork cwork:about ?about .}
  OPTIONAL {
    ?cWork bbc:primaryContentOf ?pco .
    ?pco bbc:webDocumentType ?webDocType .
  }
  FILTER (?cWork = <http://www.bbc.co.uk/things/86916#id>)
}
```

Identifier	**query3.txt**

Listing

```
DESCRIBE ?creativework
WHERE {
  {
    SELECT DISTINCT ?creativework
    {
      ?creativework cwork:dateCreated ?created .
      ?creativework cwork:about <http://rdf.freebase.com/ns/m.060q82> .


      {
        { ?creativework cwork:primaryFormat cwork:TextualFormat . }
        UNION
        { ?creativework cwork:primaryFormat cwork:InteractiveFormat . }
        UNION
        { ?creativework cwork:primaryFormat cwork:PictureGalleryFormat . }
      }
          # formats
      {
        { ?creativework a cwork:NewsItem . }
        UNION
        { ?creativework a cwork:BlogPost . }
      }
      OPTIONAL { ?creativework cwork:audience ?audience } .
      FILTER (!BOUND(?audience) || ?audience = cwork:InternationalAudience) .
    }
```

	}
	}
	ORDER BY DESC(xsd:dateTime(str(?created)))
	LIMIT 16

Identifier	**query4.txt**

Listing	DESCRIBE ?creativework
	WHERE {
	  ?creativework cwork:about <http://rdf.freebase.com/ns/m.03f77> ;
	    cwork:dateCreated ?created ;
	    cwork:primaryFormat cwork:TextualFormat ;
	    a cwork:BlogPost .
	}
	ORDER BY DESC(xsd:dateTime(str(?created)))
	LIMIT 12

Identifier	**query5.txt**

Listing	SELECT (COALESCE(?preferredLabel, ?cannonicalName, "none") as ?label) ?cnt
	{
	  {
	    SELECT ?topic ((COUNT(*)) AS ?cnt)
	    WHERE
	    {
	      ?creativeWork a cwork:BlogPost ;
	        cwork:about ?topic ;
	        cwork:dateModified ?dt ;
	        cwork:audience cwork:InternationalAudience .
	      FILTER (?dt > "2011-09-15T10:12:09Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dt < "2011-09-15T11:12:09Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	    }
	    GROUP BY ?topic
	  }
	  OPTIONAL {
	    ?topic domain:canonnicalName ?cannonicalName .
	  }
	  OPTIONAL {
	    ?topic bbc:preferredLabel ?preferredLabel .
	  }
	}
	ORDER BY DESC(?cnt)

Identifier	**query6.txt**

Listing	SELECT DISTINCT ?cwork ?geonamesId ?lat ?long ?dateModified {
	
	?cwork a cwork:CreativeWork .
	?cwork cwork:mentions ?geonamesId .
	?cwork cwork:dateModified ?dateModified .
	?geonamesId geo:lat ?lat .
	?geonamesId geo:long ?long .
	FILTER(CONTAINS(STR(?geonamesId), "geonames")) .
	
	BIND(51.87930219264732 AS ?referenceLat) .
	BIND(-1.4231757610303737 AS ?referenceLong) .
	BIND(0.2232188735361287 AS ?deviation) .
	
	FILTER(
	(xsd:double(?lat) >= (?referenceLat - ?deviation)) &&
	(xsd:double(?lat) <= (?referenceLat + ?deviation)) &&
	(xsd:double(?long) >= (?referenceLong - ?deviation)) &&
	(xsd:double(?long) <= (?referenceLong + ?deviation))) .
	}
	
	LIMIT 100

Identifier	**query7.txt**

Listing	SELECT ?cwork ?dateModif ?title ?category ?liveCoverage ?audience {
	?cwork a cwork:BlogPost .
	?cwork cwork:dateModified ?dateModif .
	?cwork cwork:title ?title .
	?cwork cwork:category ?category .
	?cwork cwork:liveCoverage ?liveCoverage .
	?cwork cwork:audience ?audience .
	FILTER(?dateModif >= "2011-09-01T00:00:00.000"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateModif < "2011-09-30T23:59:59.999"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	}
	
	LIMIT 100

Identifier	**query8.txt**
Listing	CONSTRUCT { ?cWork a cwork:CreativeWork ; a ?type ;

```
    cwork:title ?title ;
    cwork:description ?description ;
    cwork:dateCreated ?dateCreated ;
    cwork:dateModified ?dateModified ;
    cwork:about ?about ;
    cwork:category ?category ;
    bbc:primaryContentOf ?pco .
  ?pco bbc:webDocumentType ?webDocType .
}
WHERE {
  ?cWork a cwork:CreativeWork ;
    a ?type ;
    cwork:title ?title ;
    cwork:description ?description .
  ?type rdfs:subClassOf cwork:CreativeWork .
  OPTIONAL { ?cWork cwork:dateCreated ?dateCreated . }
  OPTIONAL { ?cWork cwork:dateModified ?dateModified . }
  OPTIONAL { ?cWork cwork:about ?about .}
  OPTIONAL { ?cWork cwork:category ?category . }
  OPTIONAL {
    ?cWork bbc:primaryContentOf ?pco .
    ?pco bbc:webDocumentType ?webDocType .
  }
  FILTER (CONTAINS(?title, "started") || CONTAINS(?description, "evident")) .
}
LIMIT 1000
```

Identifier	**query9.txt**

Listing

```
SELECT ?other ?dt ((?cnt_2 * 2 + ?cnt_1 + ?cnt_0_5 * 5e-1) AS ?score)
WHERE {
  {
    SELECT (COUNT(*) AS ?cnt_2)
    WHERE {
      ?other cwork:about ?oa .
      <http://www.bbc.co.uk/things/70111#id> cwork:about ?oa .
    }
  } .
  {
     SELECT (COUNT(*) AS ?cnt_1)
     WHERE {
       ?other cwork:mentions ?oa .
       <http://www.bbc.co.uk/things/70111#id> cwork:about ?oa .
```

	}
	  } .
	  {
	    SELECT (COUNT(*) AS ?cnt_0_5)
	    WHERE {
	      ?other cwork:mentions ?om .
	      <http://www.bbc.co.uk/things/70111#id> cwork:mentions ?om .
	    }
	  } .
	  {
	    SELECT DISTINCT ?other ?dt
	    WHERE {
	      <http://www.bbc.co.uk/things/70111#id> cwork:tag ?tag .
	      ?other cwork:tag ?tag .
	      ?other cwork:dateModified ?dt .
	    }
	  }
	}
	ORDER BY DESC(?score) DESC(?dt)
	LIMIT 10

Advanced Query-mix

Identifier	**query1.txt**

Listing	CONSTRUCT {
	  ?creativeWork a cwork:CreativeWork ;
	    a ?type ;
	    cwork:title ?title ;
	    cwork:shortTitle ?shortTitle ;
	    cwork:about ?about ;
	    cwork:mentions ?mentions ;
	    cwork:dateCreated ?created ;
	    cwork:dateModified ?modified ;
	    cwork:description ?description ;
	    cwork:primaryFormat ?primaryFormat ;
	    bbc:primaryContentOf ?webDocument .
	  ?webDocument bbc:webDocumentType ?webDocType .
	  ?about rdfs:label ?aboutLabel ;
	    bbc:shortLabel ?aboutShortLabel ;

```
   bbc:preferredLabel ?aboutPreferredLabel .
  ?mentions rdfs:label ?mentionsLabel ;
   bbc:shortLabel ?mentionsShortLabel ;
   bbc:preferredLabel ?mentionsPreferredLabel .
  ?creativeWork cwork:thumbnail ?thumbnail .
  ?thumbnail a cwork:Thumbnail ;
   cwork:altText ?thumbnailAltText ;
   cwork:thumbnailType ?thumbnailType .
}
WHERE {
  {
   SELECT ?creativeWork
     WHERE {
             ?creativeWork cwork:about <http://www.wikidata.org/wiki/Q263166> .
             ?creativeWork a cwork:CreativeWork ;
             cwork:dateModified ?modified .
     }
     ORDER BY DESC(?modified)
     LIMIT 10
  }
  ?creativeWork a cwork:CreativeWork ;
              a ?type ;
              cwork:title ?title ;
              cwork:dateModified ?modified .
  OPTIONAL { ?creativeWork cwork:shortTitle ?shortTitle . }
  OPTIONAL { ?creativeWork cwork:description ?description . }
  OPTIONAL { ?creativeWork cwork:about ?about .
           OPTIONAL { ?about rdfs:label ?aboutLabel . }
           OPTIONAL { ?about bbc:shortLabel ?aboutShortLabel . }
           OPTIONAL { ?about bbc:preferredLabel ?aboutPreferredLabel . }
          }
  OPTIONAL {
           ?creativeWork cwork:mentions ?mentions .
           OPTIONAL { ?mentions rdfs:label ?mentionsLabel . }
           OPTIONAL { ?mentions bbc:shortLabel ?mentionsShortLabel . }
           OPTIONAL { ?mentions bbc:preferredLabel ?mentionsPreferredLabel . }
          }
  OPTIONAL { ?creativeWork cwork:dateCreated ?created . }
  OPTIONAL { ?creativeWork cwork:primaryFormat ?primaryFormat . }
  OPTIONAL {
          {
            ?webDocument bbc:primaryContent ?creativeWork .
            OPTIONAL { ?webDocument bbc:webDocumentType ?webDocType . }
```

```
        }
        UNION
        {
         ?webDocument ^bbc:primaryContentOf ?creativeWork .
         OPTIONAL { ?webDocument bbc:webDocumentType ?webDocType . }
        }
  }
  OPTIONAL { ?creativeWork cwork:thumbnail ?thumbnail .
        OPTIONAL { ?thumbnail cwork:altText ?thumbnailAltText . }
        OPTIONAL { ?thumbnail cwork:thumbnailType ?thumbnailType . }
  }
}
```

Identifier	**query2.txt**

Listing

```
CONSTRUCT {
  ?cWork a cwork:CreativeWork ;
    a ?type ;
    cwork:title ?title ;
    cwork:dateCreated ?dateCreated ;
    cwork:dateModified ?dateModified ;
    cwork:about ?about ;
    bbc:primaryContentOf ?pco .
  ?pco bbc:webDocumentType ?webDocType .
}
WHERE {
  ?cWork a cwork:CreativeWork ;
    a ?type ;
    cwork:title ?title .
  ?type rdfs:subClassOf cwork:CreativeWork .
  OPTIONAL { ?cWork cwork:dateCreated ?dateCreated . }
  OPTIONAL { ?cWork cwork:dateModified ?dateModified . }
  OPTIONAL { ?cWork cwork:about ?about .}
  OPTIONAL {
    ?cWork bbc:primaryContentOf ?pco .
    ?pco bbc:webDocumentType ?webDocType .
  }
  FILTER (?cWork = <http://www.bbc.co.uk/things/63607#id>)
}
```

Identifier	**query3.txt**

Listing

```
SELECT ?minute ((COUNT(*)) as ?count)
```

	WHERE {
	 ?r cwork:dateModified ?dateTime .
	 FILTER (?dateTime > "2010-04-20T06:20:20Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateTime < "2010-04-20T07:20:20Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	 }
	 GROUP BY (MINUTES(?dateTime) as ?minute)
	 ORDER BY ?minute

Identifier	**query4.txt**

Listing	SELECT ?type ((COUNT(*)) as ?count)
	 WHERE {
	 ?creativeWork a ?type .
	 ?creativeWork a cwork:CreativeWork .
	 ?creativeWork cwork:dateModified ?dateModified.
	 FILTER(?type != cwork:CreativeWork) .
	 FILTER(?dateModified > "2010-08-22T02:56:17Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateModified < "2011-04-22T02:56:17Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	 }
	 GROUP BY ?type
	 ORDER BY DESC(?count)
	 LIMIT 10

Identifier	**query5.txt**

Listing	SELECT ?about ((COUNT(*)) AS ?count)
	 WHERE {
	 ?creativeWork cwork:about ?about .
	 ?creativeWork cwork:category ?category .
	 ?about a sport:RecurringCompetition .
	 FILTER((?category = <http://www.bbc.co.uk/category/SportsCompetitions>) || (?category = <http://www.bbc.co.uk/category/SportsTeams>)) .
	 }
	 GROUP BY ?about
	 ORDER BY DESC(?count)
	 LIMIT 1000

Identifier	**query6.txt**
Listing	SELECT ?aboutType ((COUNT (*)) as ?count) ?coverage ?audience WHERE { ?creativeWork cwork:about ?about . ?about a ?aboutType . ?creativeWork cwork:liveCoverage ?coverage . ?creativeWork cwork:audience ?audience .

	FILTER ((?coverage = "false"^^<http://www.w3.org/2001/XMLSchema#boolean>) && (?audience = cwork:InternationalAudience)) .
	}
	GROUP BY ?aboutType ?coverage ?audience
	ORDER BY DESC(?count)
	LIMIT 10

Identifier	**query7.txt**

Listing	SELECT ?mentions ((COUNT(*)) as ?count)
	WHERE {
	  ?creativeWork cwork:mentions ?mentions .
	  {
	    SELECT ?creativeWork (count(*) as ?pcCount) {
	      ?creativeWork bbc:primaryContentOf ?pc .
	    }
	    GROUP BY (?creativeWork)
	  }
	  FILTER(?pcCount > 1)
	}
	GROUP BY ?mentions
	ORDER BY DESC(?count)
	LIMIT 10

Identifier	**query8.txt**

Listing	SELECT ?topic ((COUNT(*)) AS ?count)
	WHERE {
	  ?creativeWork a cwork:BlogPost ;
	    cwork:about ?topic ;
	    cwork:dateModified ?dt ;
	    cwork:audience cwork:InternationalAudience .
	  FILTER (?dt > "2010-12-21T02:33:56Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dt < "2010-12-24T02:33:56Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	}
	GROUP BY ?topic
	ORDER BY DESC(?count)

Identifier	**query9.txt**

Listing	SELECT ((MAX(?cnt)) AS ?mxc)
	WHERE {
	  {
	    SELECT ?cw ((COUNT(*)) AS ?cnt)

```
    WHERE {
      ?cw cwork:mentions ?thing
    }
    GROUP BY ?cw
  }
}
```

Identifier	query10.txt

Listing

```
SELECT ?creativeWork2 ?mentionsCount ?dateCreated ?thing2
WHERE {
  {
    SELECT ((MAX(?cnt)) AS ?maxcnt)
    WHERE {
      {
        SELECT ?creativeWork ((count(*)) AS ?cnt)
        WHERE {
          ?creativeWork cwork:mentions ?thing
        }
        GROUP BY ?creativeWork
      }
    }
  } .
  {
    SELECT ?creativeWork2 ((COUNT(*)) AS ?mentionsCount)
    WHERE {
      ?creativeWork2 cwork:mentions ?thing2 .
      ?creativeWork2 cwork:dateCreated ?dateCreated .
    }
    GROUP BY ?creativeWork2
  } .
  FILTER (?mentionsCount = ?maxcnt)
  ?creativeWork2 cwork:mentions ?thing2 .
  ?creativeWork2 cwork:dateCreated ?dateCreated .
}
ORDER BY DESC(?dateCreated)
LIMIT 10
```

Identifier	query11.txt

Listing

```
SELECT ?other ?dt ((?cnt_2 * 2 + ?cnt_1 + ?cnt_0_5 * 5e-1) AS ?score)
WHERE {
  {
```

```
  SELECT ((COUNT(*)) AS ?cnt_2)
  WHERE {
    ?other cwork:about ?oa .
    <http://www.bbc.co.uk/things/98747#id> cwork:about ?oa .
  }
} .
{
  SELECT ((COUNT(*)) AS ?cnt_1)
  WHERE {
    ?other cwork:mentions ?oa .
    <http://www.bbc.co.uk/things/98747#id> cwork:about ?oa .
  }
} .
{
  SELECT ((COUNT(*)) AS ?cnt_0_5)
  WHERE {
    ?other cwork:mentions ?om .
    <http://www.bbc.co.uk/things/98747#id> cwork:mentions ?om .
  }
} .
{
  SELECT DISTINCT ?other
  WHERE {
    {
      {
        <http://www.bbc.co.uk/things/98747#id> cwork:about ?topic
      }
      UNION
      {
        <http://www.bbc.co.uk/things/98747#id> cwork:mentions ?topic
      }
    } .
    {
      {
        ?other cwork:about ?topic
      }
      UNION
      {
        ?other cwork:mentions ?topic
      }
    } .
  }
  } .
  ?other cwork:dateModified ?dt .
}
ORDER BY DESC(?score) DESC(?dt)
LIMIT 10
```

Identifier	**query12.txt**

Listing

```
SELECT ?other ?dt ((?cnt_2 * 2 + ?cnt_1 + ?cnt_0_5 * 5e-1) AS ?score)
WHERE {
  {
    SELECT ((COUNT(*)) AS ?cnt_2)
    WHERE {
      ?other cwork:about ?oa .
      <http://www.bbc.co.uk/things/49609#id> cwork:about ?oa .
    }
  } .
  {
     SELECT ((COUNT(*)) AS ?cnt_1)
     WHERE {
       ?other cwork:mentions ?oa .
       <http://www.bbc.co.uk/things/49609#id> cwork:about ?oa .
     }
  } .
  {
     SELECT ((COUNT(*)) AS ?cnt_0_5)
     WHERE {
       ?other cwork:mentions ?om .
       <http://www.bbc.co.uk/things/49609#id> cwork:mentions ?om .
     }
  } .
  {
    SELECT DISTINCT ?other ?dt
    WHERE {
      <http://www.bbc.co.uk/things/49609#id> cwork:tag ?tag .
      ?other cwork:tag ?tag .
      ?other cwork:dateModified ?dt .
    }
  }
}
ORDER BY DESC(?score) DESC(?dt)
LIMIT 10
```

Identifier	**query13.txt**

Listing	SELECT DISTINCT ?thing ?about ?mentions ?category ?dateModified
	WHERE
	{
	  ?thing rdf:type cwork:CreativeWork .
	  ?thing cwork:about ?about .
	  ?thing cwork:mentions ?mentions .
	  ?thing cwork:category ?category .
	  ?thing cwork:dateModified ?dateModified .
	  FILTER ((?category = <http://www.bbc.co.uk/category/PoliticsPersons>) || (?category = <http://www.bbc.co.uk/category/PoliticsPersonsReference>))
	}
	ORDER BY ?dateModified
	LIMIT 100

Identifier	**query14.txt**

Listing	SELECT ?thing ?about ?mentions ?category ?dateModified ?thumbnail ?primaryFormat
	WHERE
	{
	  ?thing rdf:type cwork:CreativeWork .
	  ?thing cwork:tag ?tag .
	  ?thing cwork:category ?category .
	  ?thing cwork:dateModified ?dateModified .
	  ?thing cwork:thumbnail ?thumbnail .
	  ?thing cwork:audience ?audience .
	  ?thing cwork:primaryFormat ?primaryFormat .
	  ?thing bbc:primaryContentOf ?primaryContent .
	  ?primaryContent bbc:webDocumentType ?webdoc .
	  OPTIONAL {
	    ?thing cwork:mentions ?mentions .
	    ?thing cwork:about ?about .
	  }
	  OPTIONAL {
	    ?thing cwork:audience cwork:InternationalAudience .
	  }
	  FILTER ((?audience = cwork:InternationalAudience) && (?webdoc = bbc:Mobile) && ((?primaryFormat = cwork:TextualFormat) || (?primaryFormat = cwork:PictureGalleryFormat)))
	}
	ORDER BY DESC(?dateModified)
	LIMIT 200

Identifier	query15.txt

Listing	SELECT DISTINCT ?thing ?about ?mentions ?entityType ?category ?title
	WHERE
	{
	  ?thing rdf:type cwork:CreativeWork .
	  ?thing rdf:type ?class .
	  ?class rdfs:subClassOf cwork:CreativeWork .
	  ?thing cwork:about ?about .
	  ?thing cwork:mentions ?mentions .
	  ?mentions rdf:type ?entityType .
	  ?about rdf:type ?entityType .
	  ?thing cwork:category ?category .
	  ?thing cwork:title ?title .
	  FILTER (CONTAINS (?title, "policy")) .
	  OPTIONAL {
	    ?thing cwork:audience cwork:InternationalAudience .
	  }
	}
	ORDER BY ?about
	LIMIT 100

Identifier	query16.txt

Listing	SELECT DISTINCT ?thing ?tag ?category ?title
	WHERE
	{
	  ?thing rdf:type cwork:CreativeWork .
	  ?thing rdf:type ?class .
	  ?class rdfs:subClassOf cwork:CreativeWork .
	  ?thing cwork:tag ?tag .
	  ?thing ?a ?o .
	  ?a rdfs:subPropertyOf cwork:tag .
	  ?thing cwork:category ?category .
	  ?thing cwork:title ?title .
	  FILTER (CONTAINS (?title, "policy") && (?tag = ?o)) .
	  OPTIONAL {
	    ?thing cwork:audience ?audience .
	    FILTER (?audience = cwork:InternationalAudience) .
	  }
	}
	ORDER BY ?tag
	LIMIT 100

Identifier	query17.txt

Listing	SELECT DISTINCT ?cwork ?geonamesId ?lat ?long ?dateModified {
	
	?cwork a cwork:CreativeWork .
	?cwork cwork:mentions ?geonamesId .
	?cwork cwork:dateModified ?dateModified .
	?geonamesId geo:lat ?lat .
	?geonamesId geo:long ?long .
	FILTER(CONTAINS(STR(?geonamesId), "geonames")) .
	
	BIND(51.83950716912813 AS ?referenceLat) .
	BIND(-0.22870856488321856 AS ?referenceLong) .
	BIND(0.20338699677432331 AS ?deviation) .
	
	FILTER(
	(xsd:double(?lat) >= (?referenceLat - ?deviation)) &&
	(xsd:double(?lat) <= (?referenceLat + ?deviation)) &&
	(xsd:double(?long) >= (?referenceLong - ?deviation)) &&
	(xsd:double(?long) <= (?referenceLong + ?deviation))) .
	}
	LIMIT 1000

Identifier	query18.txt

Listing	SELECT ?cwork ?dateModif ?title ?category ?liveCoverage ?audience {
	?cwork a cwork:BlogPost .
	?cwork cwork:dateModified ?dateModif .
	?cwork cwork:title ?title .
	?cwork cwork:category ?category .
	?cwork cwork:liveCoverage ?liveCoverage .
	?cwork cwork:audience ?audience .
	FILTER(?dateModif >= "2010-05-01T00:00:00.000"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateModif < "2010-05-31T23:59:59.999"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
	}
	LIMIT 100

Identifier	query19.txt
Listing	SELECT ?topic ((COUNT(*)) as ?topicsCount) (MAX(?dateModif) AS ?maxDate) { ?cwork a cwork:BlogPost . ?cwork cwork:audience cwork:InternationalAudience . ?cwork cwork:about ?topic . ?cwork cwork:dateModified ?dateModif .

	`FILTER(?dateModif >= "2010-03-01T00:00:00.000"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateModif < "2010-03-31T23:59:59.999"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .`
	`}`
	`GROUP BY ?topic`
	`ORDER BY DESC(?maxDate) DESC(?topicsCount)`
	`LIMIT 100`

Identifier	**query20.txt**

Listing	`CONSTRUCT {`
	`  ?cWork a cwork:CreativeWork ;`
	`    a ?type ;`
	`    cwork:title ?title ;`
	`    cwork:description ?description ;`
	`    cwork:dateCreated ?dateCreated ;`
	`    cwork:dateModified ?dateModified ;`
	`    cwork:about ?about ;`
	`    cwork:category ?category ;`
	`    bbc:primaryContentOf ?pco .`
	`  ?pco bbc:webDocumentType ?webDocType .`
	`}`
	`WHERE {`
	`  ?cWork a cwork:CreativeWork ;`
	`    a ?type ;`
	`    cwork:title ?title ;`
	`    cwork:description ?description .`
	`  ?type rdfs:subClassOf cwork:CreativeWork .`
	`  OPTIONAL { ?cWork cwork:dateCreated ?dateCreated . }`
	`  OPTIONAL { ?cWork cwork:dateModified ?dateModified . }`
	`  OPTIONAL { ?cWork cwork:about ?about .}`
	`  OPTIONAL { ?cWork cwork:category ?category . }`
	`  OPTIONAL {`
	`    ?cWork bbc:primaryContentOf ?pco .`
	`    ?pco bbc:webDocumentType ?webDocType .`
	`  }`
	`  FILTER (CONTAINS(?title, "on") || CONTAINS(?description, "automatically")) .`
	`}`
	`LIMIT 1000`

Identifier	**query21.txt**

Listing	`SELECT ?title ?description ?category ?tag ?audience ?liveCoverage ?primaryFormat ?year ?month {`
	`        ?creativework a cwork:CreativeWork .`
	`        ?creativework cwork:title ?title .`

```
        ?creativework cwork:description ?description .
        ?creativework cwork:category ?category .
        ?creativework cwork:tag ?tag .
        ?creativework cwork:audience ?audience .
        ?creativework cwork:liveCoverage ?liveCoverage .
        ?creativework cwork:primaryFormat ?primaryFormat .
        ?creativework cwork:dateCreated ?dateCreated .
        BIND (day(?dateCreated) as ?day) .
        BIND (year(?dateCreated) as ?year) .
        BIND (month(?dateCreated) as ?month) .
        FILTER (CONTAINS(?title, "broadcasters") || CONTAINS(?description, "begins")) .
        FILTER (?category = <http://www.bbc.co.uk/category/SportsTeams>) .


}
ORDER BY ?year ?month
LIMIT 500
```

Identifier	**query22.txt**
Listing	

```
SELECT ?year ?month ?tag ((COUNT(*)) as ?count) {
        ?creativework a cwork:CreativeWork .
        ?creativework cwork:title ?title .
        ?creativework cwork:description ?description .
        ?creativework cwork:category ?category .
        ?creativework cwork:tag ?tag .
        ?creativework cwork:audience ?audience .
        ?creativework cwork:liveCoverage ?liveCoverage .
        ?creativework cwork:primaryFormat ?primaryFormat .
        ?creativework cwork:dateCreated ?dateCreated .
        BIND (day(?dateCreated) as ?day) .
        BIND (year(?dateCreated) as ?year) .
        BIND (month(?dateCreated) as ?month) .
        FILTER (CONTAINS(?title, "us") || CONTAINS(?description, "browser")) .
        FILTER (?category = <http://www.bbc.co.uk/category/SportsTeams>) .
}
GROUP BY ?year ?month ?tag
ORDER BY ?year ?month ?count
LIMIT 500
```

Identifier	**query23.txt**
Listing	

```
SELECT ?year ?month ((COUNT(*)) AS ?count) {
        ?creativework a cwork:CreativeWork .
        ?creativework cwork:title ?title .
        ?creativework cwork:description ?description .
        ?creativework cwork:category ?category .
        ?creativework cwork:tag ?tag .
        ?creativework cwork:audience ?audience .
        ?creativework cwork:liveCoverage ?liveCoverage .
        ?creativework cwork:primaryFormat ?primaryFormat .
        ?creativework cwork:dateCreated ?dateCreated .
        BIND (day(?dateCreated) as ?day) .
        BIND (year(?dateCreated) as ?year) .
        BIND (month(?dateCreated) as ?month) .
        FILTER (CONTAINS(?title, "enjoyment") || CONTAINS(?description, "retain")) .
        FILTER (?category = <http://www.bbc.co.uk/category/PoliticsPersons>) .
}
GROUP BY ?year ?month
ORDER BY ?year ?month
LIMIT 500
```

Identifier	**query24.txt**
Listing	

```
SELECT DISTINCT ?year ?month ?day ((COUNT(*)) AS ?cwsPerDay) {
  ?cw a cwork:CreativeWork .
  ?cw cwork:about <http://news.bbc.co.uk/democracylive/hi/representatives/profiles/25491.stm> .
  ?cw cwork:about <http://rdf.freebase.com/ns/m.027c3m> .
  ?cw cwork:dateCreated ?dateCreated .
  BIND (day(?dateCreated) AS ?day) .
  BIND (month(?dateCreated) AS ?month) .
  BIND (year(?dateCreated) AS ?year) .
  FILTER (?dateCreated >= "2010-12-28T05:44:58Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> && ?dateCreated < "2011-09-28T05:44:58Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>) .
}
GROUP BY ?year ?month ?day
ORDER BY ?year ?month ?day
```

Identifier	**query25.txt**
Listing	

```
SELECT ?who ?interactionDays {
  {
    SELECT ?who ((COUNT(*)) AS ?interactionDays) {
      {
        SELECT DISTINCT ?year ?month ?day ?who {
          ?cw a cwork:CreativeWork .
          ?cw cwork:about <http://news.bbc.co.uk/democracylive/hi/representatives/profiles/25491.stm> .
          ?cw cwork:about ?who .
```

	?cw cwork:dateCreated ?dateCreated .
	BIND (day(?dateCreated) AS ?day) .
	BIND (month(?dateCreated) AS ?month) .
	BIND (year(?dateCreated) AS ?year) .
	FILTER (?who != <http://news.bbc.co.uk/democracylive/hi/representatives/profiles/25491.stm>) .
	}
	GROUP BY ?year ?month ?day ?who
	}
	}
	GROUP BY ?who
	}
	}
	ORDER BY DESC(?interactionDays) ?who
	LIMIT 10

5.3 Appendix. C - Description of Operational Phase : Online Replication And Backup

Following are description of steps performed in Online Replication And Backup Operational Phase of SPB. Additional requirement for each RDF database is to implement all of the provided scripts (/data/enterprise/scripts) with database specific commands for completing those steps.

1. System start + bulk load + full backup (manual step, **full_backup_start.sh**)
2. Starting a warmup run
3. Starting a Benchmark run, fixed number of queries will be executed - timing is started
4. Adding a milestone point at ½ of executed queries + (**incremental_backup_start.sh**). Location of the milestone is configurable.
5. Completing the benchmark run
6. System shutdown (**system_shutdown.sh**)
7. System restart (**system_start.sh**) + verification query1 (ok - if milestone exists) - end of timing
8. System shutdown (**system_shutdown.sh**)
9. Starting backup restore (**full_backup_restore_start.sh**)
10. System restart (**system_start.sh**) + verification query (ok - if milestone point does not exists)

5.4 Appendix. D - Sample of Benchmark Results

Following are samples benchmark results taken from the three types of log files saved during the benchmark run :

- **semantic_publishing_benchmark_queries_brief.log** - provides a brief log of query execution order. Brief log contains information about : query name, query id (id is formed as sequential increment of a query execution counter - one per query), execution time (ms), returned results (triples). e.g. :

```
...
11:06:32.861 :        [query7.txt, id:199] Query executed, execution time : 406 ms, results : 100
11:06:32.869 :        [query8.txt, id:198] Query executed, execution time : 103 ms, results : 1000
11:06:32.876 :        [query1.txt, id:224] Query executed, execution time : 13 ms, results : 0
11:06:32.917 :        [query8.txt, id:199] Query executed, execution time : 75 ms, results : 1000
11:06:32.919 :        [query4.txt, id:209] Query executed, execution time : 47 ms, results : 122
11:06:32.931 :        [insert.txt, id:0] Query executed, execution time : 1953 ms
11:06:32.981 :        [query8.txt, id:200] Query executed, execution time : 99 ms, results : 1000
11:06:33.000 :        [query7.txt, id:200] Query executed, execution time : 335 ms, results : 100
...
```

- **semantic_publishing_benchmark_queries_detailed.log** - provides a detailed information of each executed operation. Detailed log contains the query body and result from its execution. e.g. :

```
>> 11:16:01.066 [AggregationAgent:Thread-14] :
*** Query [query3.txt, id:2124], execution time : 337 ms, results : 448
PREFIX bbcevent:<http://www.bbc.co.uk/ontologies/event/>
...
PREFIX fb:<http://rdf.freebase.com/ns/>

#  Query name : query3
#  Query Description :
#  Describes all creative works about a topic with certain fixed properties and order them by creation date

DESCRIBE ?creativework
WHERE {
  {
    SELECT DISTINCT ?creativework
    {
      ?creativework cwork:dateCreated ?created .
      ?creativework cwork:about <http://dbpedia.org/resource/Rachel_Reeves> .
      {
        { ?creativework cwork:primaryFormat cwork:TextualFormat . }
        UNION
        { ?creativework cwork:primaryFormat cwork:InteractiveFormat . }
        UNION
        { ?creativework cwork:primaryFormat cwork:PictureGalleryFormat . }
      }
        # formats
```

```
   {
     { ?creativework a cwork:NewsItem . }
     UNION
     { ?creativework a cwork:BlogPost . }
   }
   OPTIONAL { ?creativework cwork:audience ?audience } .
   FILTER (!BOUND(?audience) || ?audience = cwork:NationalAudience) .
  }
 }
}
ORDER BY DESC( xsd:dateTime(str(?created)) )
LIMIT 16


---------------------------------------------
*** Result for query [query3.txt, id:2124] :
Length : 43621
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
	xmlns:bbcevent="http://www.bbc.co.uk/ontologies/event/"
	xmlns:geo-pos="http://www.w3.org/2003/01/geo/wgs84_pos#"
	xmlns:bbc="http://www.bbc.co.uk/ontologies/bbc/"
	xmlns:time="http://www.w3.org/2006/time#"
	xmlns:event="http://purl.org/NET/c4dm/event.owl#"
	xmlns:music-ont="http://purl.org/ontology/mo/"
	xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
	xmlns:foaf="http://xmlns.com/foaf/0.1/"
	xmlns:provenance="http://www.bbc.co.uk/ontologies/provenance/"
	xmlns:owl="http://www.w3.org/2002/07/owl#"
	xmlns:cms="http://www.bbc.co.uk/ontologies/cms/"
	xmlns:news="http://www.bbc.co.uk/ontologies/news/"
	xmlns:cnews="http://www.bbc.co.uk/ontologies/news/cnews/"
	xmlns:cconcepts="http://www.bbc.co.uk/ontologies/coreconcepts/"
	xmlns:dbp-prop="http://dbpedia.org/property/"
	xmlns:geonames="http://sws.geonames.org/"
	xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
	xmlns:domain="http://www.bbc.co.uk/ontologies/domain/"
	xmlns:dbpedia="http://dbpedia.org/resource/"
	xmlns:geo-ont="http://www.geonames.org/ontology#"
	xmlns:bbc-pont="http://purl.org/ontology/po/"
	xmlns:tagging="http://www.bbc.co.uk/ontologies/tagging/"
	xmlns:sport="http://www.bbc.co.uk/ontologies/sport/"
	xmlns:skosCore="http://www.w3.org/2004/02/skos/core#"
	xmlns:dbp-ont="http://dbpedia.org/ontology/"
	xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
	xmlns:curric="http://www.bbc.co.uk/ontologies/curriculum/"
	xmlns:cwork="http://www.bbc.co.uk/ontologies/creativework/"
```

```
            xmlns:fb="http://rdf.freebase.com/ns/"
            xmlns:sesame="http://www.openrdf.org/schema/sesame#"
            xmlns:fn="http://www.w3.org/2005/xpath-functions#">

<rdf:Description rdf:about="http://www.bbc.co.uk/things/64991#id">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
            <rdf:type rdf:resource="http://www.bbc.co.uk/ontologies/creativework/CreativeWork"/>
            <rdf:type rdf:resource="http://www.bbc.co.uk/ontologies/creativework/NewsItem"/>
            <rdf:type rdf:resource="http://www.bbc.co.uk/ontologies/creativework/Thumbnail"/>
            <bbc:primaryContentOf rdf:resource="http://www.bbc.co.uk/things/972600547#id"/>
            <bbc:primaryContentOf rdf:resource="http://www.bbc.co.uk/things/1813497345#id"/>
            <bbc:primaryContentOf rdf:resource="http://www.bbc.co.uk/things/1161261914#id"/>
            <cwork:about rdf:resource="http://www.bbc.co.uk/things/g5b3a44d9-bc98-41b3-8c68-d3692a0bc8b6#id"/>
            <cwork:about rdf:resource="http://www.guardian.co.uk/politics/person/8855/"/>
            <cwork:about rdf:resource="http://www.guardian.co.uk/politics/person/9245/"/>
            <cwork:about rdf:resource="http://dbpedia.org/resource/Rachel_Reeves"/>
            <cwork:tag rdf:resource="http://www.bbc.co.uk/things/g5b3a44d9-bc98-41b3-8c68-d3692a0bc8b6#id"/>
            <cwork:tag rdf:resource="http://www.guardian.co.uk/politics/person/8855/"/>
            <cwork:tag rdf:resource="http://www.guardian.co.uk/politics/person/9245/"/>
            <cwork:tag rdf:resource="http://dbpedia.org/resource/Rachel_Reeves"/>
            <cwork:tag rdf:resource="http://sws.geonames.org/7701617/"/>
            <cwork:altText>thumbnail atlText for CW http://www.bbc.co.uk/context/64991#id</cwork:altText>
            <cwork:audience rdf:resource="http://www.bbc.co.uk/ontologies/creativework/NationalAudience"/>
            <cwork:category rdf:resource="http://www.bbc.co.uk/category/PoliticsPersonsReference"/>
            <cwork:dateCreated                    rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2011-04-
02T05:58:40.867Z</cwork:dateCreated>
            <cwork:dateModified                   rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2012-04-
01T21:06:23.660Z</cwork:dateModified>
            <cwork:description> 23 fixtures movement town since large wrestling cases contribution charge into lead competition official
competition enactments.</cwork:description>
            <cwork:liveCoverage rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</cwork:liveCoverage>
            <cwork:mentions rdf:resource="http://sws.geonames.org/7701617/"/>
            <cwork:primaryFormat rdf:resource="http://www.bbc.co.uk/ontologies/creativework/TextualFormat"/>
            <cwork:primaryFormat rdf:resource="http://www.bbc.co.uk/ontologies/creativework/InteractiveFormat"/>
            <cwork:shortTitle> venue power ambition europe amounts make affords whole union global.</cwork:shortTitle>
            <cwork:thumbnail rdf:resource="http://www.bbc.co.uk/thumbnail/2140388969"/>
            <cwork:title>David Davis domestic information help official our ceased machinery reinforced mediums decided.</cwork:title>
</rdf:Description>
...
```

- **semantic_publishing_benchmark_results.log** - benchmark results are saved to that file. Benchmark results are updated each second and appended to it. Result contains two sections : Editorial and Aggregate. Each section describes number of agents that execute operations/queries, number of each executed operation/query as well as statistics for each - min, max, average execution times. Each section summarizes results at the end with total number of executed operations/queries and average operations/queries per second, number of timed-out operations/queries. e.g. :

```
...
Seconds run : 597
```

```
Editorial:
          2 agents

          4058  inserts (avg : 205     ms, min : 59      ms, max : 3889    ms)
          492   updates (avg : 474     ms, min : 186     ms, max : 5663    ms)
          507   deletes (avg : 225     ms, min : 60      ms, max : 2330    ms)

          5057 operations (4058 CW Inserts (0 timed-out), 492 CW Updates (0 timed-out), 507 CW Deletions (0 timed-out))
          8.4707 average operations per second

Aggregation:
          16 agents

          2181  Q1   queries (avg : 947     ms, min : 6       ms, max : 8381    ms, 0 timed-out)
          2013  Q2   queries (avg : 10      ms, min : 3       ms, max : 1812    ms, 0 timed-out)
          2074  Q3   queries (avg : 605     ms, min : 163     ms, max : 21738   ms, 0 timed-out)
          2059  Q4   queries (avg : 157     ms, min : 27      ms, max : 3536    ms, 0 timed-out)
          2021  Q5   queries (avg : 126     ms, min : 7       ms, max : 2552    ms, 0 timed-out)
          2011  Q6   queries (avg : 71      ms, min : 13      ms, max : 3322    ms, 0 timed-out)
          2007  Q7   queries (avg : 554     ms, min : 148     ms, max : 3905    ms, 0 timed-out)
          2119  Q8   queries (avg : 134     ms, min : 43      ms, max : 2152    ms, 0 timed-out)
          2072  Q9   queries (avg : 1811    ms, min : 88      ms, max : 12311   ms, 0 timed-out)

          18557 total retrieval queries (0 timed-out)
          34.3648 average queries per second
...
```