**Semantic Publishing Benchmark**

**With contributions from: Vladimir Alexiev, Venelin Kotsev**

***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.

A fully developed benchmark includes: source code, binary software, data-sets, queries, ontologies and documentation (purpose, choke point descriptions, execution instructions, auditing and disclosure rules). This document describes not just the current status of the benchmark, it can also serve as a user manual for the Semantic Publishing Benchmark.

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, source code, data generator, queries 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.
* Ontologies include core (creative works, company, core concepts, CMS, person, provenance, tagging); domain (news, sport, educational curriculum); and conformance (used for validation);
* Reference datasets include English, Scottish and International football competitions and teams, Formula 1 competitions and teams, People in the UK Parliament, and UK Places from GeoNames.
* *Data Generation:* describes the data generation process, different modelling of relations produced and lists some of the properties that each generated creative work will have.
* *Workloads:* describes the simulated benchmark workload that comprises simultaneous execution of editorial and aggregation query streams:
* Editorial agents simulate the editorial work performed by journalists, editors or automated text annotation engines. This includes insert, and delete operations
* Aggregation agents simulate the retrieval operations performed by journalists, end-users or automated search engines by executing a mix of aggregation queries. These queries include Aggregation, Search, Statistics and Analytical, Drill-down, Faceted search.
* Choke Points: Outlines the various "choke points" that workloads have. Details are also included in LDBC report D4.4.1
* Query definitions - provides a descriptions of each queries part of the workloads
* *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
* Description of Operational Phases: gives information of all the steps that the benchmark executes
* Description of Benchmark Configuration Properties: describes all properties in file: test.properties that drive the benchmark execution
* Description of Benchmark Definitions Properties: describes all properties in file: definitions.properties and their purpose
* *Benchmark Operation:* gives details what needs to be configured in order to run the benchmark through all its operational phases i.e. : Prepare RDF Database, Starting the Benchmark Driver, Generating data, Loading data, 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

[1 Introduction 8](#_Toc389734975)

[1.1 Motivation for the benchmark 8](#_Toc389734976)

[1.2 Relevance to industry 10](#_Toc389734977)

[1.3 General Benchmark Overview 11](#_Toc389734978)

[1.4 Participation of industry and academia 12](#_Toc389734979)

[2 Formal definition 14](#_Toc389734980)

[2.1 Requirements 14](#_Toc389734981)

[2.2 Data 14](#_Toc389734982)

[2.2.1 Data Generation 15](#_Toc389734983)

[2.3 Workloads 18](#_Toc389734984)

[2.3.1 Choke Points 19](#_Toc389734985)

[2.3.2 Basic Workload - Query Definitions 20](#_Toc389734986)

[2.3.3 Advanced Workload - Query Definitions 22](#_Toc389734987)

[2.3.4 Query Templates and Substitution Parameters 27](#_Toc389734988)

[2.4 Validation 27](#_Toc389734989)

[2.5 Performance Metrics 27](#_Toc389734990)

[2.6 Execution Rules 28](#_Toc389734991)

[2.7 Full Disclosure 28](#_Toc389734992)

[2.8 Auditing 29](#_Toc389734993)

[3 Instructions 30](#_Toc389734994)

[3.1 Software Installation 30](#_Toc389734995)

[3.2 Benchmark Configuration 30](#_Toc389734996)

[3.2.1 Description of operational phases 30](#_Toc389734997)

[3.2.2 Description of Benchmark Configuration Properties 31](#_Toc389734998)

[3.2.3 Description of Benchmark Definitions Properties 33](#_Toc389734999)

[3.3 Benchmark Operation 34](#_Toc389735000)

[3.3.1 Prepare RDF Database for tests with SPB 34](#_Toc389735001)

[3.3.2 Starting the Benchmark Driver 34](#_Toc389735002)

[3.3.3 Configure SPARQL Endpoint 34](#_Toc389735003)

[3.3.4 Generate Data 34](#_Toc389735004)

[3.3.5 Load Data 35](#_Toc389735005)

[3.3.6 Generate Query Substitution Parameters 36](#_Toc389735006)

[3.3.7 Run the Benchmark 36](#_Toc389735007)

[3.3.8 Run the Online Replication and Backup Benchmark 37](#_Toc389735008)

[3.3.9 Run Validation of Query Results 37](#_Toc389735009)

[3.3.10 Run OWL2-RL Conformance Test 38](#_Toc389735010)

[3.4 Results Gathering 38](#_Toc389735011)

[4 References 40](#_Toc389735012)

[5 Annexes 41](#_Toc389735013)

[5.1 Annex. A - Queries Listings 41](#_Toc389735014)

[5.2 Annex. B - Description of Operational Phase : Online Replication And Backup 61](#_Toc389735015)

List of Figures

[Figure 1: BBC Olympics 2012 Aggregation Page about Bulgaria 8](#_Toc389735016)

[Figure 2: BBC Ontology-aware Text Analytics 9](#_Toc389735017)

[Figure 3: BBC Graffitti: Manual Curation of Semantic Annotation 9](#_Toc389735018)

[Figure 4: Data generator, types of produced models in generated data 17](#_Toc389735019)

[Figure 5. Generate Data 35](#_Toc389735020)

[Figure 6. Load Data 35](#_Toc389735021)

[Figure 7. Generate Query Substitution Parameters 36](#_Toc389735022)

[Figure 8. Run The Benchmark 36](#_Toc389735023)

[Figure 9. Run Online Replication And Backup 37](#_Toc389735024)

[Figure 10. Run Query Results Validation 38](#_Toc389735025)

[Figure 11. Run OWL2-RL Conformance Test 38](#_Toc389735026)

List of Tables

[Table 1: Abbreviations used in this document 7](#_Toc389735027)

[Table 2: Required behaviours/functionalities required from a RDF database 14](#_Toc389735028)

[Table 3: Distribution of ‘about’ and ‘mentions’ tags in creative works, analysed in ‘live’ data provided by the BBC 16](#_Toc389735029)

Abbreviations

Table : Abbreviations used in this document

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

1. Introduction
   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.

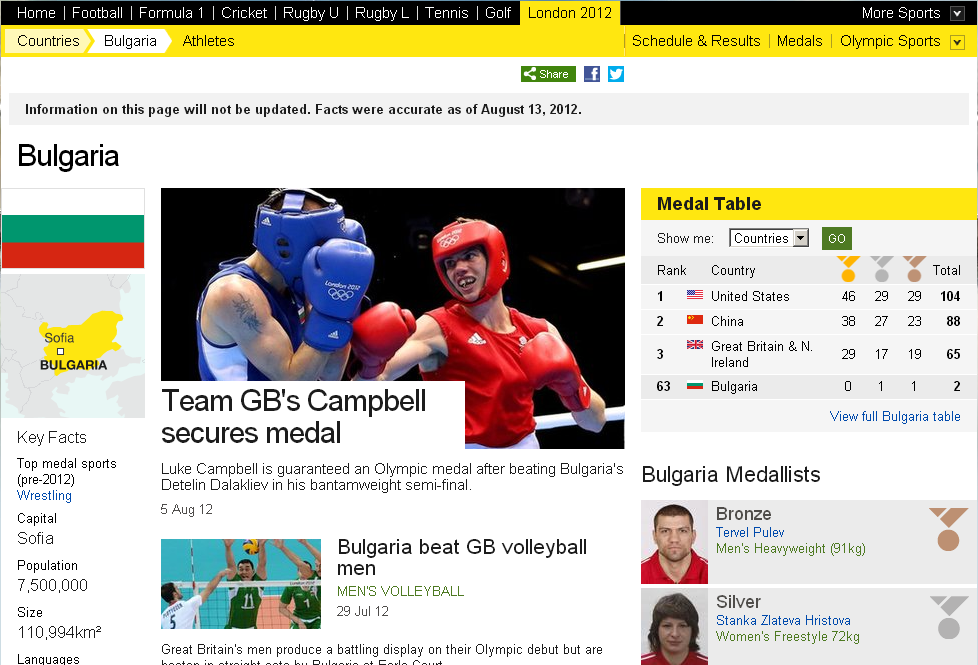


Figure : BBC Olympics 2012 Aggregation Page about Bulgaria

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

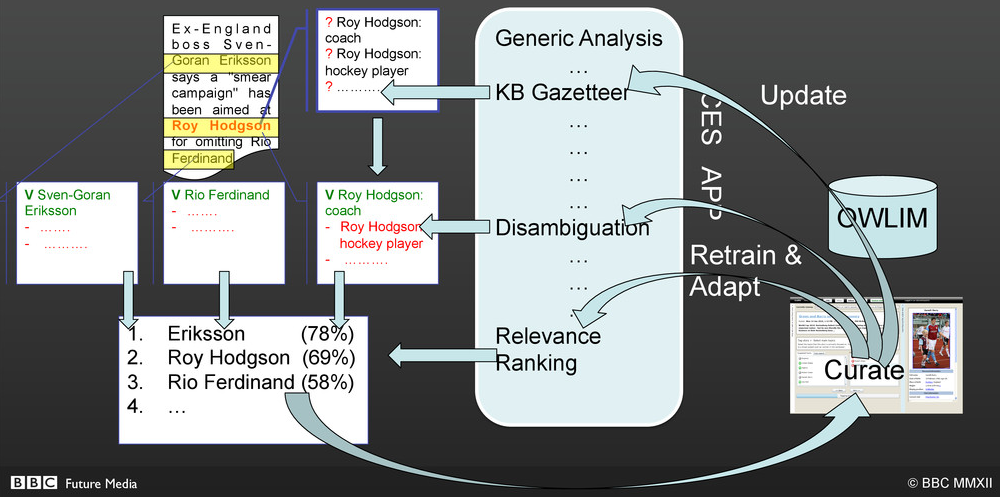


Figure : BBC Ontology-aware Text Analytics

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

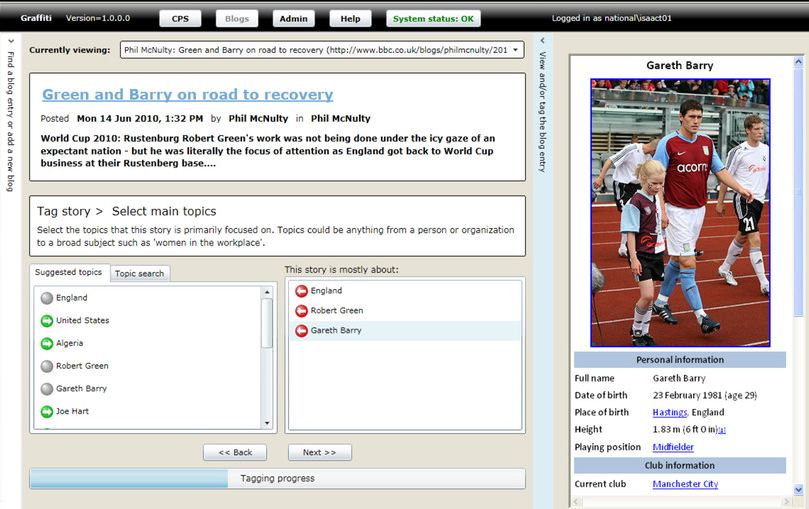


Figure : BBC Graffitti: Manual Curation of Semantic Annotation

* 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. 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) are 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 encyclopedias, 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. 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).

* 1. 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

Full details of the ontologies and datasets are given in deliverable D2.2.2 Data Generator

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.

1. Formal definition
   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 : 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.

* 1. 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.

*Details can be found in Deliverable D2.2.2, 2.1 Semantic Publishing: Ontologies [5]*

* **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.

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. Following is a list of used reference datasets:

* 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”
* scottish-football-competitions-1: contains entities describing the Scottish football competitions
* scottish-football-teams-2: contains entities describing the Scottish football teams
* international-football-competitions-3: contains entities describing the International football competitions
* international-football-teams-2: contains entities describing the International 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
* 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.
* 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.
* 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.

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 described 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 : 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.

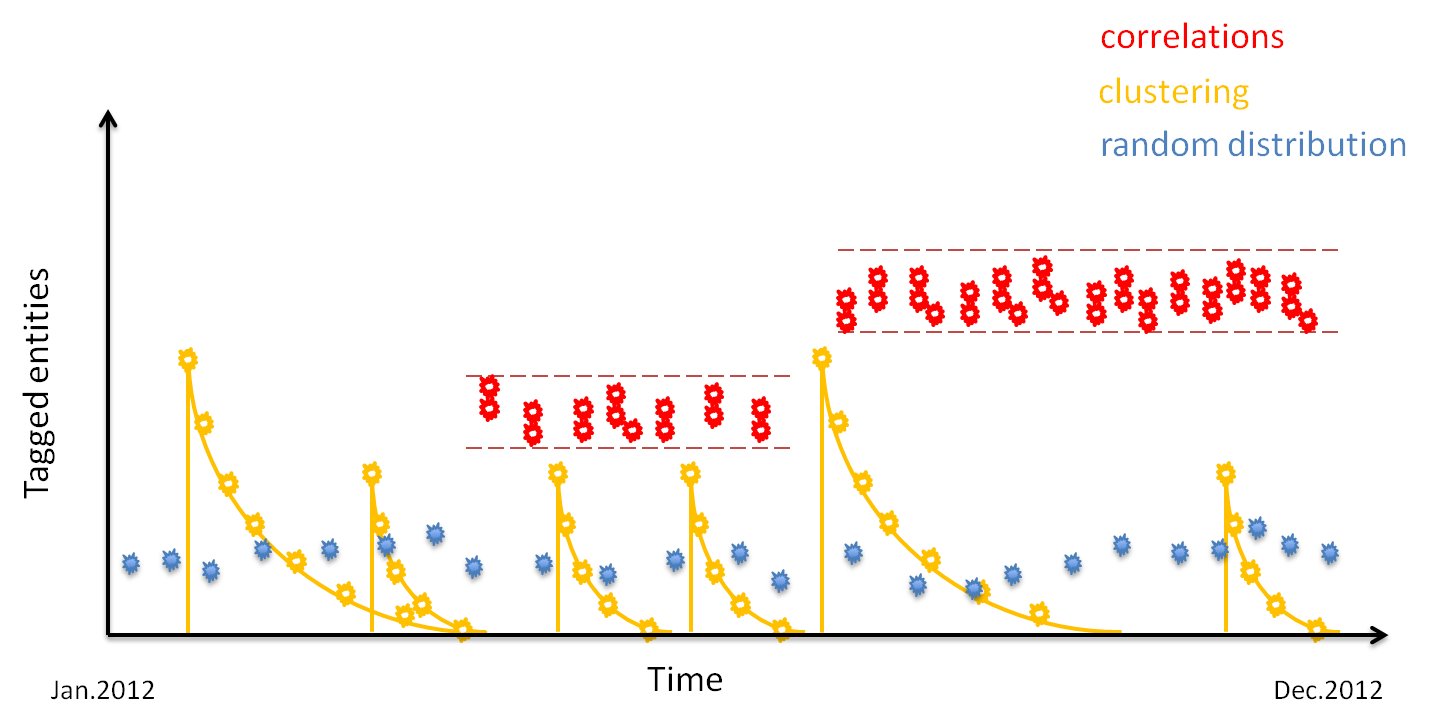


Figure 4: Data generator, types of produced models in generated data

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 reference dataset 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 tree data models described above and generation of creative works begins following a selected data generation model.

Each creative work will have varying amount properties like depending on its type like :

* randomly generated and sized sentences used for the title, shortTitle and description properties using a words dictionary
* randomly generated date-time which is within a range of one based on a configuration value
* Different types of creative works are distributed as follows: 45% of creative works will have a type: BlogPost, 35% - NewsItem, 20% - Programme. Types of creative works define a different set of properties for them
* audience type : chosen depending on the type of creative work, e.g. for BlogPost the audience chosen is: InternationalAudience, for NewsItem – audience chosen is: NationalAudience;
* liveCoverage property: a boolean property whose value is chosen based on the type of the creative work;
* primaryFormat property: depending on creative work type - TextualFormat, InteractiveFormat, VideoFormat, AudioFormat;
* thumbnail property: using a randomly generated URI, assuming that thumbnails will be identified by a Uniform Resource Identifier;
* altText property: a randomly generated text string, used in case a thumbnail cannot be resolved;
* primaryContentOf property: referring to a randomly generated URI of a document, assuming that such a document hypothetically exists and is identified by a URI.

Generated data is saved in files with a configured serialisation file format. Available serialization formats are: TriG, TriX, N-Triples, N-Quads, N3, RDF/XML, RDF/JSON, Turtle. It should be noted that not all of these serialization formats do have support for contexts so not all of the serialization formats are suitable for benchmark purposes.

* 1. 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:

* Aggregation queries : 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 queries : those queries execute faster and are searching for creative works that match a concrete fact
* Statistical queries : 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 : queries that perform text search in the literal properties of creative works
* Geo-spatial : queries that aggregate result based on geo-spatial region
* Analytical queries : queries that perform 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 queries : 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 : 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. Query order of execution is pseudo-randomly chosen following a distribution per query defined in the benchmark’s configuration.

Pseudo random allocation of query execution is controlled by an allocations module in the benchmark which would select the next query or operation to run.

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*. |
| **Choke Points** | CP1, CP2, CP3, CP4 |

|  |  |
| --- | --- |
| **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. |
| **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. |
| **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. |
| **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. |
| **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). |
| **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*. |
| **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) |
| **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*. |
| **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) |
| **Choke Points** | CP1, CP2, CP3, CP4 |

|  |  |
| --- | --- |
| **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) |
| **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 |
| **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 |
| **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 |
| **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. |
| **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. |
| **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. |
| **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. |
| **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. |
| **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 |
| **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. |
| **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). |
| **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). |
| **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. |
| **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. |
| **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. |
| **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. |
| **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. |
| **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. |
| **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) |
| **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 |
| **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 |
| **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. |
| **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. |
| **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 substitution for them when executed. SPB driver generates and initializes the substitution parameters prior to execution of the 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 one to specify the amount of the substitution parameter sets per query which are produced in a random manner and saved along with generated data.

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

* 1. Validation

Validation of query results is 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.
  1. 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.

* 1. 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 configuration can be benchmarked
* warm-up time should not exceed benchmark time
* sustain editorial operations rate above a configured threshold - a required editorial operations rate per second is to be constantly sustained during measurement the aggregation queries rate. 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 45 minutes after reaching the editorial operations threshold 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
* query substitution parameters are to be generated prior to running the benchmark
* run on empty database - each run of the benchmark should be performed on an empty database
* run on a 'cold' database - right before running the benchmark, RDF Database System should be started and no data should be stored in it
* serialization format for generated data should be context aware, e.g. N-Quads
  1. 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 templates needs to 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
* 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
  1. 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, the first draft of which will likely be in deliverable D6.6.3 Auditor Training M24.

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.

1. Instructions
   1. Software Installation

The Publishing 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\_semanticpub\_bm - *the benchmark driver*
* https://github.com/ldbc/ldbc\_semanticpub\_bm\_additional\_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
  1. Benchmark Configuration
     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
  + 1. Description of Benchmark Configuration Properties

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

* ***ontologiesPath* *-*** path to ontologies from reference knowledge data, *default: ./data/ontologies*
* ***referenceDatasetsPath* *-*** path to datasets from reference knowledge*, default: ./data/datasets*
* ***creativeWorksPath -*** path where generated synthetic data will be saved*, default: ./data/generated*
* ***queriesPath* -** path to query templates, *default: ./data/sparql*
* ***definitionsPath* *-*** path to definitions.properties configuration file*, default: ./definitions.properties*
* ***endpointURL* *-*** URL of SPARQL endpoint provided by the RDF database*, requires updating*
* ***endpointUpdateURL -*** URL of SPARQL endpoint used for update operations*, requires updating*
* ***datasetSize -*** defines the size of generated synthetic data (triples) produced by the data-generator
* ***generatedTriplesPerFile -*** number of triples per file. Generated synthetic data is saved to files with number of *generatedTriplesPerFile* each.
* ***adjustRefDatasetsSizes -*** optional, if additional datasets have been added (files with extension '.adjustablettl'), then for each of them, a new .ttl file is created with adjusted size depending on the value of *datasetSize* parameter, *default value is true*. Additional datasets can be taken from project ldbc\_semanticpub\_bm\_additional\_datasets (see section 3.1 Software Installation)
* ***allowSizeAdjustmentsOnDataModels -*** allows data generator to dynamically adjust the amount of correlations, clusterrings and randomly generated models keeping a ratio of 1/3 for each in generated data model*, default value is true*
* ***queryTimeoutSeconds -***  defines the timeout for query execution, *default value is 300 s*
* ***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 generated synthetic data in the database)
* ***validationPath -*** location of validation data related to the validation operational phase, *default value can be used*
* ***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, TriX, TriG
* ***warmupPeriodSeconds -*** warmup period in seconds, *requires updating*
* ***benchmarkRunPeriodSeconds -*** benchmark period in seconds, *requires updating*
* ***aggregationAgents -*** number of aggregation agents that will execute a mix of queries simultaneously, *requires updating*
* ***editorialAgents -*** number of editorial agents that will execute a mix of update operations simultaneously, *requires updating*
* ***dataGeneratorWorkers -*** number of worker threads used by the data generator to produce synthetic data, *requires updating*
* ***generatorRandomSeed -*** use it to set the random seed for the data generator (default value is 0). e.g. in cases when several benchmark drivers are started in separate processes in order to generate synthetic data faster - to be used with *creativeWorkNextId* parameter for each new process
* ***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*
* ***querySubstitutionParameters -*** number substitution parameters set that will be generated for each query, *default value is 100000*
* ***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
* ***updateRateThresholdOps -*** defines the update 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 run in order to have a valid result. 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
* ***updateRateThresholdReachTimePercent -*** defines the time window during which value set to property *updateRateThresholdOps* 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, if not reached, the result is not considered to be valid
* ***enableLogs -*** enables or disables logging of benchmark results and details, default: true
  + 1. 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 is a description of each property :

* ***aboutsAllocations -*** defines the allocation amount of *about* tags in a Creative Work
* ***mentionsAllocations -*** defines the allocation amount of *mention* tags in a Creative Work
* ***entityPopularity -*** defines popularity ratio of an entity in the reference datasets
* ***usePopularEntities -*** defines allocation amount of popular entities to be used for tagging in Creative Works or in aggregation queries
* ***creativeWorkTypesAllocation -*** defines the allocation amount of Creative Work types during data-generation
* ***aboutAndMentionsAllocation -*** defines the allocation amount of *about* or *mentions* in aggregation criteria
* ***editorialOperationsAllocation -*** defines the allocation distribution of INSERT, UPDATE, DELETE queries that each editorial agent will execute
* ***aggregationOperationsAllocation -*** defines the allocation distribution of aggregation queries starting from query1, query2, query3...etc.
* ***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
* ***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*
* ***majorEventsPerYear -*** defines the maximum number of *major* events that could happen in one year. Each major event will be tagged by a number of Creative Works which will decay exponentially in time
* ***minorEventsPerYear -*** defines the maximum number of *minor* events that could happen in one year. 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
* ***dataGenerationPeriodYears -*** defines the period (in years) in generated data, starting from *seedYear*
* ***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
* ***correlationDuration -*** defines the duration of 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
* ***minLat -*** defines minimum latitude, related to geo-spatial data
* ***maxLat -*** defines maximum latitude, related to geo-spatial data
* ***minLong -*** defines minimum longitude, related to geo-spatial data
* ***maxLong -*** defines maximum longitude, related to geospatial data
* ***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
  1. Benchmark Operation
     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 requiremed for the RDF database repository
* Enable context indexing (if available)
* Enable geo-spatial indexing (if available)
* Enable text-indexing (if available)
  + 1. Starting the Benchmark Driver

For starting each of the operational phases, 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>*

*Notes :*

* *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.*
  + 1. Configure SPARQL Endpoint

Update following configuration properties :

* endpointURL
* endpointUpdateURL
  + 1. 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.*

Load

Reference

Datasets

Load

Ontologies

**Generate**

**Data**

Figure . Generate Data

* + 1. 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*

Load

Reference

Datasets

Load

Ontologies

Generate

Data

**Load**

**Data**

Figure . Load Data

* + 1. Generate Query Substitution Parameters

This operational phase is intended to start together with data generation, but can also be started independently.

* generateQuerySubstitutionParameters=true

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

Load

Reference

Datasets

Load

Ontologies

Generate

Data

Load

Data

**Generate**

**Substitution**

**Parameters**

Figure . Generate Query Substitution Parameters

* + 1. Run the Benchmark

To run the benchmark operational phase, 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=true

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

Load

Reference

Datasets

Load

Ontologies

Generate

Data

Load

Data

Generate

Substitution

Parameters

**Run The Benchmark**

Figure . Run The Benchmark

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.

* + 1. Run the 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 for the Online Replication and Backup phase

* 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

Reference

Datasets

Load

Ontologies

Generate

Data

Load

Data

Generate

Substitution

Parameters

**Run Online Replication And Backup Benchmark**

Figure . Run Online Replication And Backup

* + 1. Run Validation of Query Results

Running that operational phase 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*

Load

Reference

Datasets

Load

Ontologies

**Run**

**Results**

**Validation**

Figure . Run Query Results Validation

* + 1. Run OWL2-RL Conformance Test

Running that operational phase requires an empty RDF Database and configured for OWL2-RL rule-set.

* loadOntologies=true
* checkConformance=true

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

Load

Ontologies

**Run**

**OWL2-RL**

**Conf. Test**

Figure . Run OWL2-RL Conformance Test

* 1. 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 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

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.

1. 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](https://speakerdeck.com/jemrayfield/bbc-dynamic-semantic-publishing-sport|olympics-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](http://www.ldbc.eu:8090/download/attachments/1671227/LDBC_2_2_1_final_v2.pdf?version=1&modificationDate=1380549454000)

1. Annexes
   1. Annex. A - 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 . }  }  OPTIONAL { ?creativeWork bbc:primaryContentOf ?webDocument .  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 . }  }  OPTIONAL { ?creativeWork bbc:primaryContentOf ?webDocument .  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 |

* 1. Annex. B - 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)