Semantic Publishing Benchmark Task Force Report

Coordinator: Barry Bishop
With contributions from: Venelin Kotsev, Vladimir Alexiev, Atanas Kiryakov

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 September 2013 the benchmark has reached the state of draft publication. This report describes progress made by the Semantic Publishing task-force leading up to the delivery and publication of an LDBC benchmark. Task-force activities progress in parallel to work package tasks, combining the knowledge, research results and knowledge gained into tangible benchmark deliverables.

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 will include: source code, binary software, data-sets, queries, ontologies and documentation (purpose, choke point descriptions, execution instructions, auditing/disclosure/publishing rules). The complete set of objectives for a particular benchmark are likely to change over time as new, relevant problems are uncovered. This report describes not just the current status of the benchmark, but also justifications for its current/final form, for example when feedback from industry, academia and leading experts has been taken into account.

The outstanding tasks required to finish the benchmark are enumerated and a practical plan to see them completed is provided. Finally, this report presents initial results from a "calibration and tuning" experiments, using the current version of the benchmark against OWLIM and Virtuoso – those allow one to observe the impact of different reasoning approach and to get feeling about the complexity profile of the 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 September 2013 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 the reference knowledge, 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 and built binary software, data-set 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 report 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;
- Processes: describes the basic use cases covered by the benchmark;
- Output values: describes the resulting values provided by the benchmark.

The Development section describes the major contributions that media companies (in particular the BBC) have made to the benchmark formation. It also provides discussion on expected changes of scope: the set of objectives for the benchmark may change over time as new relevant problems are uncovered. This section describes possible "variation points" considered by the benchmark's Task Force and justifications for its current form. Variation points include: Ranking, Full-text search (FTS), Geo-spatial queries, Enterprise Features, Reasoning, Drill-down queries.

The Formal definition section provides the benchmark specification as follows:

- *Requirements:* basic requirements that must be fulfilled by a repository attempting to implement the benchmark;

- *Input 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 processing and configuration parameters of the data generator that produces the main business data used by the benchmark (creative works).
- *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).
- *Choke Points:* details are included in LDBC report D4.4.1;
- *Instructions* for parametrising, customising and executing the benchmark:
    - Operational phases: describes the steps that the benchmark executes in order: loadOntologies, loadDatasets, generateCreativeWorks, loadCreativeWorks, warmUp, benchmark, checkConformance, cleanup;
    - Configuration: describes over 15 parameters in test.properties and definitions.properties that drive the benchmark execution;
    - Fine-tuning: describes about 10 parameters that modify the characteristics of the generated data set;
    - Requirements and execution of the benchmark: describes the required platform and instructions for running the benchmark;
    - Results gathering: describes the benchmark results (about 5 numbers) and where log files are written.
- **Disclosure items**: describes the files and parameters that must be documented together with benchmark results;
- **Auditing rules**: the things that an auditor must check before a benchmark can publish its results.

The Current status section describes remaining items: features that were discussed for implementation but are not yet implemented. This includes Full-Text Search and Faceted Search related query loads. It also comments unclear behaviours: describes decisions that do not have a strong justification or have a viable alternative decision. These involve mostly the distribution of various data items, e.g. GeoNames locations related to the creative works. We have published results of the benchmark on two of the leading RDF engines - OWLIM and Virtuoso.

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Motivation for the benchmark

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

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

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

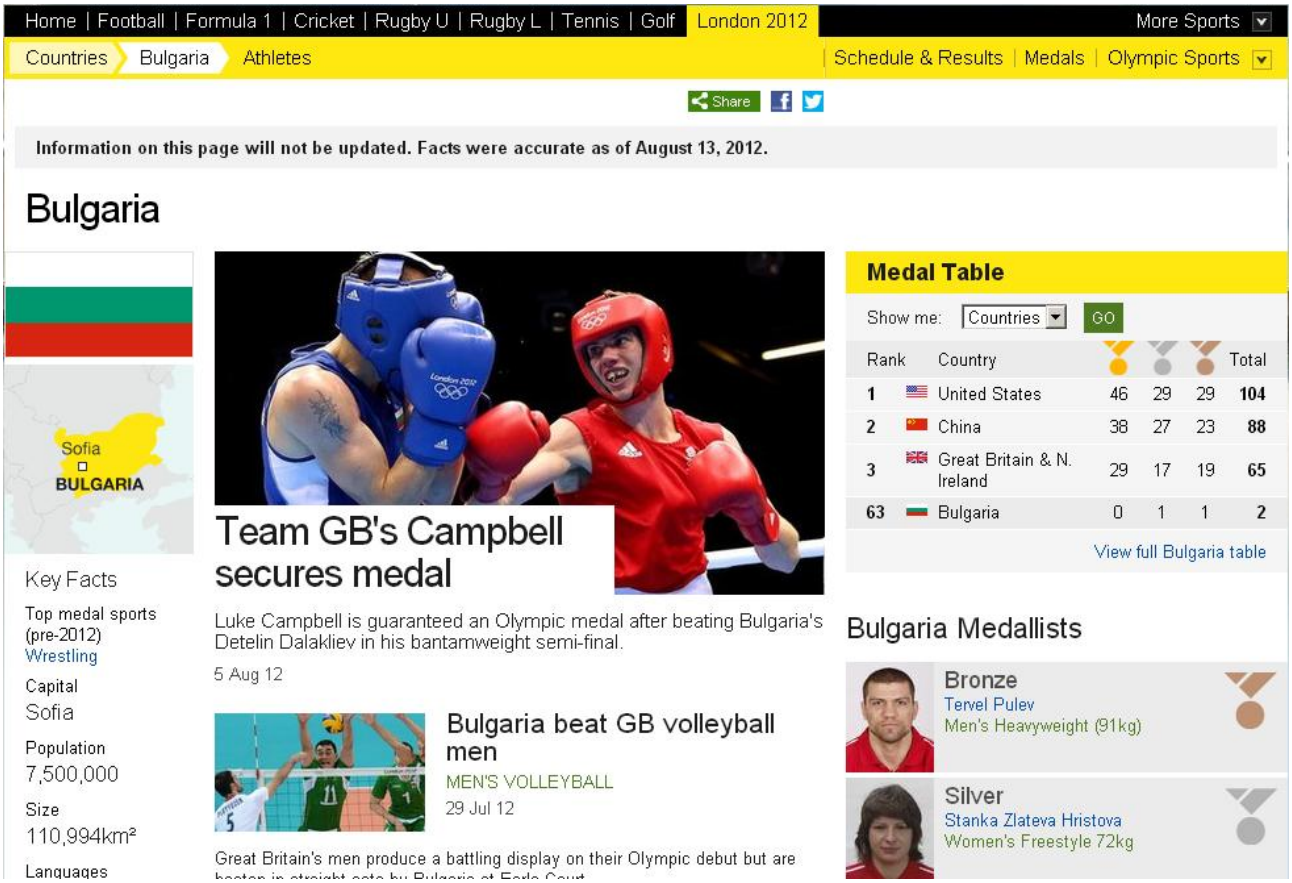

Figure 1: BBC Olympics 2012 Aggregation Page about Bulgaria

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

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

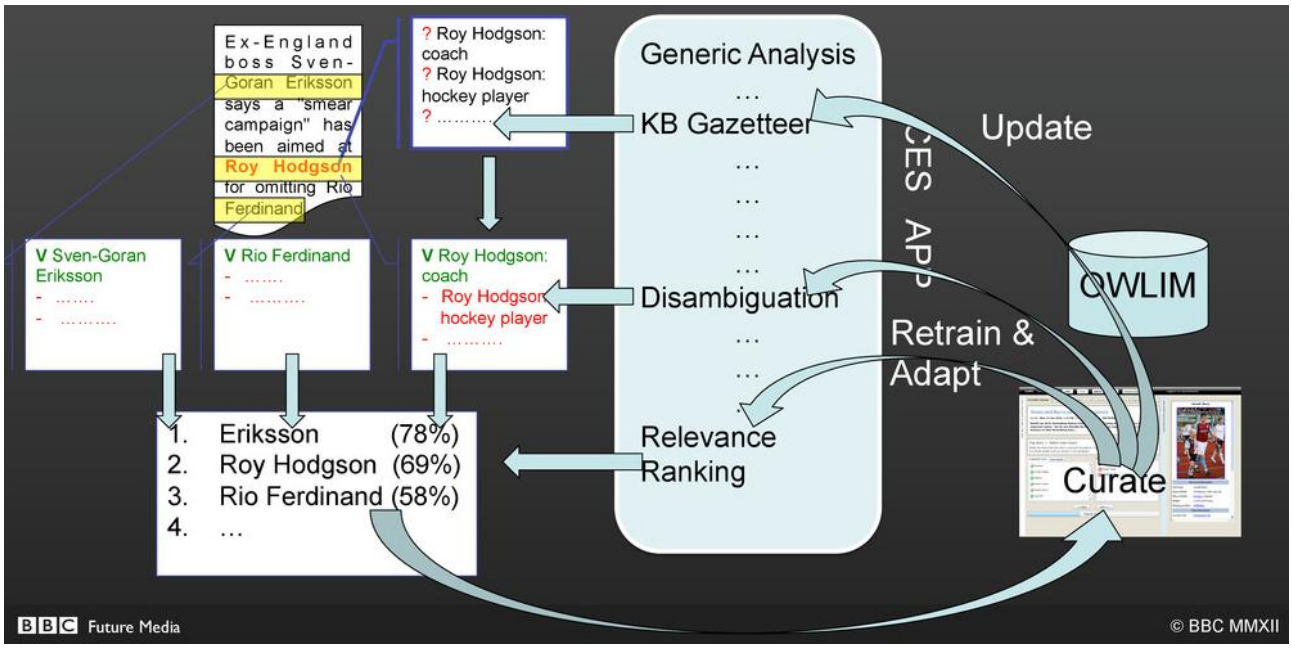

Figure 2: BBC Ontology-aware Text Analytics

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

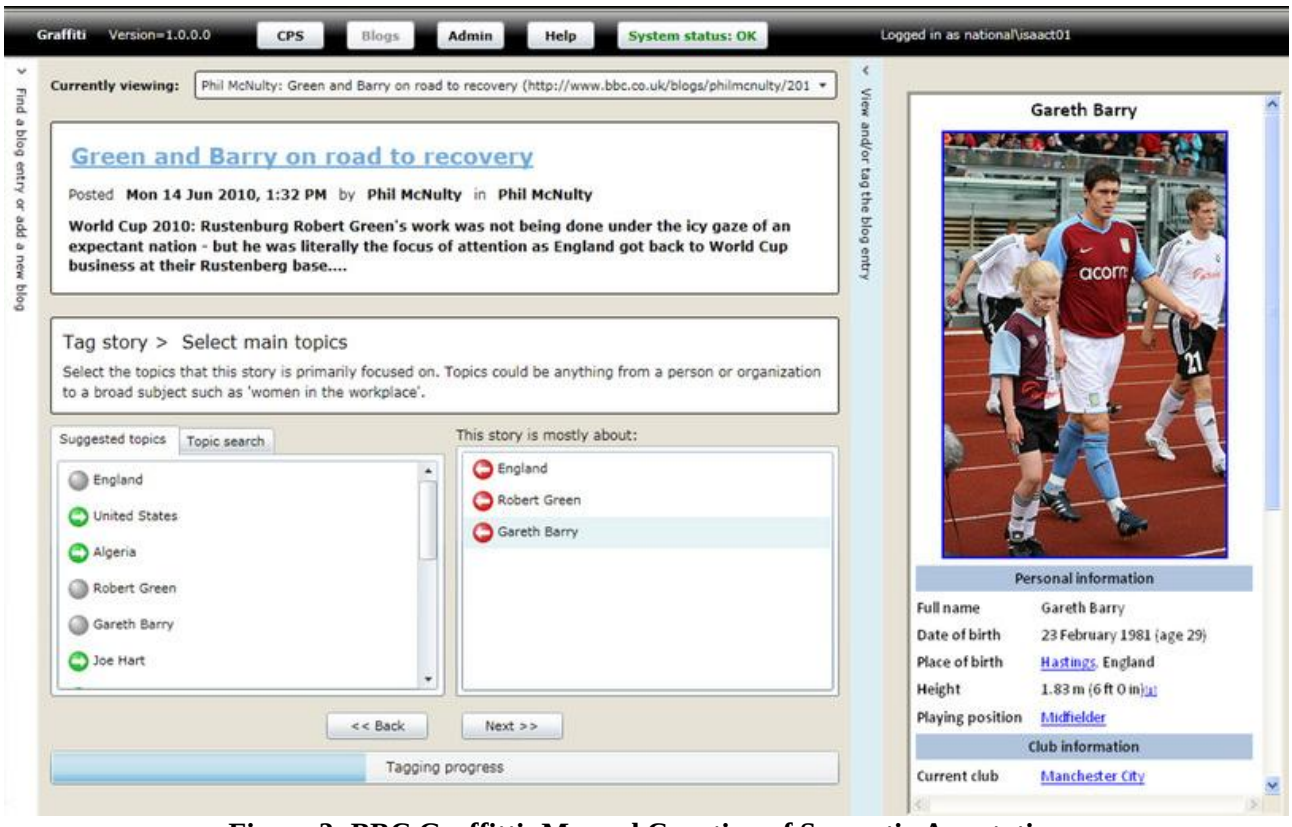

Figure 3: BBC Graffitti: Manual Curation of Semantic Annotation

- 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].

1.2 Relevance to industry

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

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

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

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

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

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

- The British Broadcasting Corporation (BBC) 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 publishing benchmark will provide a simple, off-the-shelf means to evaluate suitable RDF databases for integration into their publishing/journalist pipelines and workflows. The benchmark is flexible enough such that workflows can be tailored for many different use-cases. This allows media organisations to configure the benchmark to suit their own requirements and so quickly and easily run their own internal evaluation of products.

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

1.3 Processes

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, 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 so called 'drill-down' or analytical queries which dynamically alter their criteria based on the returned result.

A required editorial operations rate per second is advisable to be constantly sustained during measurement the aggregation queries rate. Editorial operations rate should be calculated by using formula:

$$editorialOperationsRate = log10(dataset\ size) + 1$$

Following table (Table 1) shows the minimal editorial operations rate which is to be constantly sustained for various dataset sizes.

Table 1 : Example of editorial operations rate for various dataset sizes

Dataset size	Editorial operations rate per second
1M	7
10M	8
50M	8.7
100M	9
500M	9.6
1B	10

A typical description of the 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).

1.4 Result Metrics

Result metrics produced by the benchmark describe how fast an RDF database can execute queries (by simultaneously running aggregation agents) while at the same time requiring simultaneous editorial operations to be performed (by running editorial agents) having a predefined amount of data stored in it and utilizing a predefined set of ontologies.

The benchmark is self-contained and requires only the input "dataset size" from which the benchmark generates all extra necessary data.

The result of the benchmark describes the query (or operation) execution rate per second for each of the agent types. There are two outputs: the 'update rate' and the 'query rate' i.e. editorial operations and aggregation operations.

More detailed information is displayed for each executed query by the aggregation agents and each executed operation by the editorial agents, e.g.

- total number of executed queries (operations) - each aggregation and editorial agent will report back the status of execution for queries from the aggregation or editorial query mix
- number of failed queries (operations) - each agent will report back the failure to execute a certain query

2 Development

2.1 Participation of industry and academia

As well as being the inspiration for the LDBC 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.

2.2 Changes of scope

There are many possible features of a publishing benchmark that could be implemented and would have direct relevance to large media/publishing organisations. However, for simplicity and focus, only the most important features were prioritised. The remaining features that were left out are described in the following table (Table 2).

Table 2 : Features of a publishing benchmark

Feature	Explanation
Ranking	Being graph-oriented, RDF databases often reveal some ability to query the structure of a graph, e.g. by accessing the number of statements made about each resource and using this to compute an 'importance' factor. When these values are made available at query time then they provide an additional option for ordering query results. This kind of feature is very useful for 'sieving' a large number of query results to bring the interesting values earlier in the result processing. However, despite its usefulness, the addition of a ranking element to the benchmark rapidly became overly complex and contrived. It was considered a distraction from the main thrust of the use-case.
Full-text search (FTS)	Some RDF database products offer full-text search, hence an aspect of this could be interesting for a media/publisher that naturally has a large amount of text to store and process. One benchmarking challenge for such a feature is that the indexing methods (e.g. stemming algorithms used) as well as the ranking method, while typically sharing common traits is non-standard, hence not only would the performance of such system tests vary, but also the content of their answers. A genuine FTS benchmark further should besides performance also measure precision and recall against some ground truth. Despite this being a meaningful activity, it was again considered distracting from the benchmark scenario. However, some small element remains with the inclusion of one or more 'aggregation' queries that use regular expressions to search text attributes of creative works. Any RDF database that provides a more efficient mechanism to search text is able to re-write these queries to make use of any feature they offer.
Geo-spatial	Despite a standardisation effort for processing geo-spatial data encoded in RDF, this is far from complete or well-supported. However, there does exist a W3C ontology for encoding geo-spatial data using the WGS84 ontology and this was utilised in some of the aggregation queries using geo-spatial constraints that are easy to capture in the SPARQL query language, i.e. to search for creative works tagged with a geo-location within a box defined by the latitude and longitude of its corners. The inclusion of this style of query permits them to be re-written for those RDF databases that have more efficient methods to apply geo-spatial constraints to SPARQL queries.
Enterprise Features	All enterprises are concerned with the resilience and robustness of the data storage platform. If the database is clustered, it should be able to continue functioning in the event of the loss of one of its nodes. In this state, it should be able to maintain a degraded, but acceptable level of performance, even when recovery/synchronisation processes are running. Day-to-day administration activities should also not unduly affect system performance, e.g. executing an online backup. Unfortunately, the scope of enterprise environments and range of administration activities would justify a suite of benchmarks by itself. It was considered a diversion from the main goal of the publishing benchmark and was discontinued.

Feature	Explanation
Reasoning	The ontologies and use-case provided by the BBC require fairly simple reasoning that does not extend past RDFS and a few OWL primitives. Since both of these standards have reached 'recommendation' status, the task force considered it appropriate to extend the BBC use-case in order to verify the commonly used inference primitives, i.e. all of the RDF Schema Vocabulary rules, as well as majority of the OWL2-RL profile (both inference rules and integrity constraints). In order to achieve this, a further ontology was added to the input data collection, which makes use of further OWL language features. Also, a separate suite of test cases was created that test the more complex inferences and verifies that the constraints hold and prevent any inconsistencies from being entered into the database. This separate suite of tests is run once as a set of 'checkbox' items. Decision for a separate suite was influenced by the fact that a heavy inferencing during the benchmark would have a heavy impact on performance results.
Drill-down queries	The BBC use-case does not currently have any drill-down or faceted search functionality. However, this is something that they intend to make available in the near future. The task force decided that such functionality presents an interesting set of problems for query-optimizers, so several queries were added to the aggregation workload that simulate user drill-down into creative works.

3 Formal definition

3.1 Requirements

3.1.1 Base level requirements

The following behaviours/functionalities are required from any RDF database engine in order to properly execute the LDBC publishing benchmark (see Table 3).

Table 3: 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 the following features are likely to benefit from such modifications.

3.2 Input data

In order to run 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 used can be further divided into 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.

Full 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, that editorial agents create annotations about and aggregation agents query. They are snapshots of the real datasets provided by the BBC. Additionally a Geonames reference dataset has been added for further enriching the annotations with geo-locations data and allows to performing geo-spatial queries. 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.

3.3 Data generation

Starting point of the current benchmark is a dataset, consisting of ontologies, reference datasets (covered in section: Input Data) 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 the 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, modification date – 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 refer to the entities from reference datasets. That way a creative work provides meta-data about one or several entities – facts about them and relations.

Main purpose of the data generator is to reproduce the distribution of about and mentions tags which was analysed and taken from a 'live' dataset provided by the BBC. Following Table 4 shows the distribution of total about and mentions tags found in creative works, also shows their individual distributions.

Table 4 : Distribution of 'about' and 'mentions' in creative works, analysed from '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 %

First step of data generation process is to identify all instances from different domain ontologies (also referred to as entities above) that exist in the reference datasets. For more details on the query used, refer to *D 2.2.2 - 2.3 Semantic Publishing: Data Generator [5]*.

Once identified, they are used for tagging when a creative work is generated. Next step is to create a bias towards popular entities when tagged. This is achieved by randomly selecting an amount of 5% of all instances (or entities) to be 'popular' instances, and the rest 95% to be 'regular'. Another allocation is used which applies the bias towards popular entities by using popular instances for 30% of generated creative works and 'regular' instances for the rest 70%.

Additional properties are added to each creative work, e.g.:

- randomly generated and sized sentences used for the title, shortTitle and description properties;
- randomly generated date-time which is within a range of one year from current date;
- type of creative works is distributed as follows: 45% of creative works will have a type: BlogPost, 35% - NewsItem, 20% - Programme;
- 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.

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 it is ‘things’ and for its context - ‘context’.

All generated creative works are stored in files with a file format of choice. 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 will generate creative works each residing in its own context.

3.4 Workloads

The workload in the Semantic Publishing Benchmark is created by the simultaneous execution of the editorial and aggregation agents. Simulating a constant load generated by end-users, journalists, editors, automated engines, etc.

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 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 creative works. First action is to delete the context where a creative work description resides along with all its content. Second action is to insert the same creative work (using its current ID) with all 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 long 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 just retrieve the greatest ID stored, 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 data
- Statistics 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
- Analytical queries: so called ‘drill-down’ queries, which would dynamically re-configure their criteria, based on the result produced in their 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.

Each aggregation agent will execute a mix of the queries described above in a constant loop, until the benchmark run has finished. Query order for execution is pseudo-randomly chosen following a distribution per query (defined in the benchmark's configuration) which limits the execution of the 'heavy' aggregation queries as they would take longer to execute and favours execution of 'faster to execute' ones - e.g. search queries or statistics queries, also analytics queries fall into that group.

Pseudo random allocation of query execution is controlled by an allocations module built-in the benchmark which after being initialized with corresponding to agent type's distribution values would produce the next allocation value used for selecting the next query or operation for execution. Allocations module is using a built-in random utility which initialized with a seed value guarantees the pseudo randomness of its results. Thus during each execution of the benchmark same pseudo-random sequence of results will be produced.

The result of workload execution is constantly updated and shown as a benchmark result, updated once per second. Each update of the benchmark result is adequately showing the 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 query encountered by each of the agent is registered and shown too.

The number of editorial and aggregation agents that will run can be configured before to running the benchmark.

For further details on queries descriptions see D2.2.2. - 2.2 Semantic Publishing: Workloads [5]

3.5 Choke points

By using the term "choke points" we mean the technical challenges that each RDF store needs to overcome in order to satisfy the need for a fast and reliable service using real-world data and real-world queries.

Following is a description of choke points that can be identified in each of the aggregation queries executed during the benchmark's run:

Identifier	**query1.txt** - Retrieve creative works *about* thing **t** (or such that *mention* thing t)
Description	• Join ordering based on cardinality evaluation of functional properties: cwork:dateModified, cwork:dateCreated. • OPTIONAL and nested OPTIONAL clauses which are treated by the query optimizer as nested sub-queries (i.e. optimized separately and added to the main query plan)
Optimisation	Optimizer should use an efficient cost evaluation method for choosing the optimal join tree (among all join trees that the query has) Optimizer should be able to decide whether to put the OPTIONAL triples on top of the join tree and delay their execution until the last possible moment

Identifier	**query2.txt** - Retrieve creative works that are *about* or *mention* things that have specific properties
Description	• Join ordering based on cardinality evaluation of functional properties: cwork:dateModified, cwork:dateCreated • OPTIONAL clauses which are treated by the query optimizer as nested sub-queries (i.e. optimized separately and added to the main query plan)

	• FILTER constraint
Optimisation	Optimizer should use an efficient cost evaluation method for choosing the optimal join tree (among all join trees that the query has) Optimizer should be able to decide whether to put the OPTIONAL triples on top of the join tree and delay their execution until the last possible moment or consider also the FILTER condition on variables from the OPTIONAL clause – in which case putting the OPTIONAL triples on top of the join tree will not be optimal (pushing it down as much as possible will reduce the amount of intermediate results)

Identifier	**query3.txt** - Retrieve creative works that have been modified in a time range of one hour
Description	• full scan query • FILTER constraint on time interval defining start end end periods • GROUP BY, ORDER BY
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time Optimizer should be able to split the FILTER conditions in conjunction of conditions and push them down the join tree as much as possible, which will limit the amount of intermediate results.

Identifier	**query4.txt** - Retrieve the most popular *types* of creative works, skip the base type
Description	• full scan query • FILTER constraint excluding entities of base class from result set • GROUP BY, ORDER BY
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time Optimizer should be able to push the FILTER condition down the join tree as much as possible and apply it as soon as variables in it have been bound Optimizer should not consider the GROUP BY and ORDER BY as important clauses in cases where all results are counted (COUNT(*))

Identifier	**query5.txt** - Retrieve the N most popular topics that creative works are *about* **query6.txt** - Retrieve the N most popular topic types that creative works are *about* **query7.txt** - Retrieve the N most popular topics that creative works *mention*
Description	• full scan query • GROUP BY, ORDER BY
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time

	Optimizer should not consider the GROUP BY and ORDER BY as important clauses in cases where all results are counted (COUNT(*))

Identifier	**query8.txt** - Retrieve the N most popular topics creative works that have been modified in a time range of one hour are about.
Description	• full scan query • Join ordering based on cardinality evaluation of functional property : cwork:dateModified • FILTER constraint on time interval defining start and end periods • GROUP BY, ORDER BY
Optimisation	Optimizer should use an efficient cost evaluation method for choosing the optimal join tree (among all join trees that the query has) Optimizer should be able to split the FILTER conditions into conjunction of conditions and push them down the join tree as much as possible, which will limit the amount of intermediate results Optimizer should not consider the GROUP BY and ORDER BY as important clauses in cases where all results are counted (COUNT(*))

Identifier	**query9.txt** - Retrieve the largest number of mentioned topics in creative works
Description	• full scan query • aggregation query and sub-query • COUNT, MAX, GROUP BY
Optimisation	Optimizer should not consider the GROUP BY as important clause in cases where all results are counted (COUNT(*))

Identifier	**query10.txt** - Retrieve a list of N creative works that are mentioning the maximum number of topics and their number
Description	• two aggregate sub-queries • FILTER constraint • COUNT, MAX, GROUP BY
Optimisation	Optimizer should identify the possibility of asynchronous execution of the aggregate sub-queries. Optimizer should be able to identify the aggregate (COUNT, MAX) sub-query and use the right type of join operation (intersection). Optimizer should push the FILTER condition down the join tree as much as possible and apply it as soon as variables in it have been bound.

Identifier	**query11.txt** - Retrieve similar creative works regarding the 'things' they are about or mention. Calculates a score for a particular Creative Work, about most similar articles. **query12.txt** – A simplified version of query11 – all optimisations except for UNIONS are true for it.
Description	• three aggregation star-shaped sub-queries, one select sub-query • Join ordering based on cardinality evaluation of functional property : cwork:about, cwork:mentions • COUNT • DISTINCT • UNION
Optimisation	Optimizer should identify the possibility of asynchronous execution of the aggregate sub-queries. Optimizer should consider cardinality of star-shaped sub-queries for choosing the optimal join ordering. Optimizer should identify the possibility to run the UNIONs in term and the DISTINCT in parallel. Optimizer should consider the selectivity of the DISTINCT for choosing the right execution plan. The distinct's state should be shared between threads or should be merged after the top order sort.

Identifier	**query13.txt** – Retrieve a list of N creative works, the 'things' they are about and mention, their categories, the modification date.
Description	• star-shaped query • DISTINCT • ORDER BY, LIMIT • FILTER
Optimisation	Optimizer should consider correctly estimated cardinalities of all triple patterns in order to select the optimal join ordering for the execution plan. Depending on estimated cardinalities, to select the right type of join operator. Optimizer should push the FILTER condition down the join tree as much as possible and apply it as soon as variables in it have been bound.

Identifier	**query14.txt** - Retrieve a list of N creative works, the 'things' they are about and mention, their categories, the modification date, their thumbnail, and primary format.

	query15.txt - Similar to query14, differs in FILTER constraints **query16.txt** - Similar to query14, differs in simplified FILTER constraint, but introduces an additional OPTIONAL clause **query17.txt** - Similar to query16 **query18.txt** - Similar to query17, FILTER condition is placed inside the OPTIONAL clause
Description	• star-shaped query • DISTINCT • ORDER BY, LIMIT • OPTIONAL • FILTER, multiple conditions
Optimisation	Optimizer should consider correctly estimated cardinalities of all triple patterns in order to select the optimal join ordering for the execution plan. Depending on estimated cardinalities, to select the right type of join operator. Optimizer should split the FILTER condition in conjunction of conditions and push them as deep as possible in the join tree, thus starting their execution as soon as possible. Disjunctions in filter condition should be transformed into UNIONs. Equality between a variable and a constant should be handled by replacing every match of the variable with the constant. (Query rewriting) Optimizer should be able to decide whether to put the OPTIONAL triples on top of the join tree and delay their execution until the last possible moment. (query16, query17). Also should consider the FILTER condition on variables from the OPTIONAL clause – in which case putting the OPTIONAL triples on top of the join tree will not be optimal (pushing it down as much as possible will reduce the amount of intermediate results (query18)).

Identifier	**query19.txt** - Retrieve a list of N creative works, their thumbnail and the thumbnail's alternative text. **query20.txt** - similar to query19, differs from it by a more complex FILTER condition.
Description	• star-shaped query • OPTIONAL • FILTER • owl:sameAs • owl:propertyChainAxiom • owl:ObjectProperty
Optimisation	Optimizer should be able to decide whether to put the OPTIONAL triples on top of the join tree and delay their execution until the last possible moment. Also should consider the FILTER condition on variables from the OPTIONAL clause – in which case putting the OPTIONAL triples on top of the join tree will not be optimal

	(pushing it down as much as possible will reduce the amount of intermediate results). Reasoning support for owl:propertyChainAxiom is required (query19). Reasoning support for owl:sameAs is required (query20).

Identifier	**query21.txt** - Retrieve a list of N sport disciplines
Description	• path query
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time. Optimizer should consider correctly estimated cardinalities not only for building the optimal logical execution plan, but also for choosing the appropriate physical operators (join). Traversal should be started with the more selective part of the triple pattern.

Identifier	**query22.txt** - Retrieve a list of N creative works, the document they are primary content of, and its platform
Description	• owl:inverseOf • OPTIONAL • FILTER
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time. Optimizer should be able to decide whether to put the OPTIONAL triples on top of the join tree and delay their execution until the last possible moment. Also should consider the FILTER condition on variables from the OPTIONAL clause – in which case putting the OPTIONAL triples on top of the join tree will not be optimal (pushing it down as much as possible will reduce the amount of intermediate results). Equality between a variable and a constant should be handled by replacing every match of the variable with the constant. (Query rewriting)s Reasoning support for owl:inverseOf is required.

Identifier	**query23.txt** - Retrieve an ordered list of web documents and their subdocuments.
Description	• owl:AsymmetricProperty • FILTER
Optimisation	Optimizer should be able to decide to use appropriate indexes for achieving optimal execution time. Consistency support of owl:AsymmetricProperty is required as the query should return empty result.

Identifier	**query24.txt** - Retrieve creative works within a certain range defined by geo-coordinates. Retrieves a list of all creative works that are mentioning entities within a geo-range. A drill-down query, starts with an initial range and narrows down with selected results.
Description	• geo-spatial query
Optimisation	Optimizer could recognize the existence of pair lat and long and try to ignore its 'data independence' assumption. The query gives an opportunity for each RDF engine to use its custom implementation of geo-spatial functionality. Requires building a specialized geo-spatial index and use of custom functions.

Identifier	**query25.txt** - Retrieve creative works that have been modified within a randomly selected date-time range. A drill-down query, starts with a time range of a year, then narrows down to a month, day, hour, etc.
Description	• time-range query
Optimisation	Optimizer should consider correctly estimated cardinalities not only for building the optimal logical execution plan, but also for choosing the appropriate physical operators (join) The query gives an opportunity for each RDF engine to utilize a specialized index for faster look-ups of date/time object values.

Identifier	**query26.txt** - Retrieve creative works and their properties, which contain a a certain word in their title or description, using a regex expression.
Description	• full-text search query
Optimisation	Optimizer should consider correctly estimated cardinalities not only for building the optimal logical execution plan, but also for choosing the appropriate physical operators (join) The query gives an opportunity for each RDF engine to utilize a specialized index for ull-text search

For further details on choke points classification see D4.4.1 Analysis and classification of choke points, [6]

3.6 Instructions

The Publishing benchmark driver is distributed as a single file: semantic_publishing_benchmark.jar. All necessary configuration and definition files, ontologies and reference data comes packed in that jar file. Additionally sources are available for download.

Deployment of the benchmark driver consist of saving the distribution jar file to a folder and extracting the following items from it:

- *test.properties* – a configuration file containing all required parameters for running the benchmark. Properties file needs to be modified according to the current setup of the database being tested.
- *definitions.properties* – a definitions file containing various allocation parameters which can be changed to do additional fine-tuning to the benchmark's data-generator or operation / query execution behaviour of editorial and aggregation agents
- *data/* – a folder containing all required ontologies, reference datasets and query templates
- *readme.txt* – a text file with information about configuring and running the benchmark (same information will be provided here)

There exist some operational considerations that have been described in the following list.

3.6.1 Description of operational phases

- *loadOntologies* – loads ontologies (in folder 'data/ontologies') into database, a required step;
- *loadDatasets* – loads the reference datasets (in folder 'data/datasets') into database, a required step;
- *generateCreativeWorks* – using ontologies and reference data from previous two phases, generates creative works and saves them in files. Generated files need to be loaded into database manually (or automatically). Note: in order to execute current phase, ontologies and reference data from previous two phases must be stored in the database;
- *loadCreativeWorks* – automatically loads generated creative works files into database (currently tested for N-Quads);
- *warmUp* – series of Aggregation queries are executed for a configurable period of time;
- *benchmark* – all aggregation and editorial agents are started and kept running for a configurable period of time (see parameter benchmarkRunPeriodSeconds);
- *checkConformance* – executes conformance queries (in folder 'data/sparql/conformance'). That phase can be executed independetly of previous ones (with exception of loadOntologies which needs to be executed always first);
- *cleanup* – optional phase, can be used to clear all data from database after benchmark run has finished.

3.6.2 Configuration

Before starting the benchmark some configuration parameters need to be changed by editing file: *test.properties*. Following is a description of all configuration parameters:

- *ontologiesPath* : path to ontologies, provided with the distribution jar file, e.g. "./data/ontologies"
- *referenceDatasetsPath* : path to reference datasets, provided with the distribution jar file, e.g. "./data/datasets"
- *creativeWorksPath* : path to folder where generated creative works data will be saved, e.g. "./data/generated". Folder is created automatically.

- *queriesPath* : path to all query templates, executed during the benchmark run, e.g. "./data/sparql"
- *definitionsPath* : path to the definitions.properties file
- *endpointURL* : URL of the endpoint used for the benchmark, e.g.

  "http://localhost:8080/openrdf-sesame/repositories/ldbc_pub"
- endpointUpdateURL : URL of the endpoint for execution of update queries, e.g.

  "http://localhost:8080/openrdf-sesame/repositories/ldbc_pub/statements"
- *datasetSize* : define the size of generated data (triples) which will be produced by the data-generator of the benchmark
- *generatedTriplesPerFile* : define a limit on the number of triples per file (generated data is saved to files - path is defined by parameter -creativeWorksPath)
- *queryTimeoutSeconds* : define the timeout for queries execution
- *verbose* : if set to true, a more detailed status of the benchmark's state is output to console
- *generateCreativeWorksFormat* : define the serialization format used by the data-generator. Available options for generated file formats are : TriG, TriX, N-Triples, N-Quads, N3, RDF/XML, RDF/JSON, Turtle. Use the exact name as shown in list.
- *warmupPeriodSeconds* : define the warmup period (seconds) in which aggregation agents will execute queries without reporting to the benchmark result
- *benchmarkRunPeriodSeconds* : define the benchmark's run period (seconds) during which editorial and aggregation agents will run simultaneously with benchmark result data being recorded and shown
- *aggregationAgents* : define the number of aggregation agents which will concurrently execute a mix of aggregation queries (query mix can be tuned by changing parameter aggregationOperationsAllocation in definitions.properties file)
- *editorialAgents* : define the number of editorial agents which will concurrently execute a mix of editorial queries (query mix can be tuned by changing parameter editorialOperationsAllocation in definitions.properties file)

Following parameters are used to define and configure benchmark's execution phases. Parameters are ordered, starting from initial phase and following their logical execution order.

- *loadOntologies* : populates the database with required ontologies (it is possible to upload ontology files manually, if the phase has not been enabled)
- *loadReferenceDatasets* : populates the database with required reference datasets (it is possible to upload reference datasets files manually, if the phase has not been enabled)
- *generateCreativeWorks* : in order to run that phase, ontologies and reference data from previous two phases must be stored in the database. Data-generator produces required for the benchmark creative works data and saves it to -creativeWorksPath
- *loadCreativeWorks* : uploads generated creative works data into database. This phase is optional, and uploading can be done manually (tested for N-Quad files)
- *warmUp* : runs aggregation agents simultaneously, no benchmarking is performed
- *runBenchmark* : runs the benchmark - all aggregation and editorial agents are started and simultaneously execute query mixes of the aggregation queries and editorial operations. Benchmark results are recorded.
- *checkConformance* : starts a set of conformance tests by executing a set of queries which verify the capabilities of the RDF database engine. Note : before running that phase, running loadOntologies is

required. The checkConformance phase may not be part of the benchmarking process and can be run independently.

3.6.3 Fine-tuning

The data generator of the publishing benchmark can be further tuned on data-generation and execution distributions of aggregation query / editorial operations, so that a different data distributions and workload to be achieved. Tuning can be done by changing parameter values in definition.properties file. Following is a description of those parameters:

- *aboutsAllocations* : define number of about tags used when generating creative works
- *mentionsAllocations* : define number of mentions tags used when generating creative works
- *entityPopularity* : define amount of entities to be considered as popular among all entities found in the reference dataset
- *usePopularEntities* : define the amount of tags that will use popular entities during data-generation and aggregation (set bias towards popular entities)
- *creativeWorkTypesAllocation* : define the allocation of different types of creative works (BlogPost, NewsItem, Programme)
- *aboutAndMentionsAllocation* : define ratio of about / mentions tags to use in aggregation queries as an aggregation criteria
- *editorialOperationsAllocation* : define distribution of execution frequency for each of the editorial operation from the editorial query mix (queries in folder /data/sparql/editorial)
- *aggregationOperationsAllocation* : define distribution of execution frequency for each of the aggregation queries from the aggregation query mix (queries in folder /data/sparql/aggregation)

3.6.4 Requirements to execute the benchmark

Publishing benchmark requires for its execution Java Runtime Environment 1.6 or higher.

3.6.5 Execution of the benchmark

Publishing benchmark can be started by executing the following command in console:

> java -jar semantic_publishing_benchmark.jar <path_to_test.properties>

3.6.6 Results gathering

Results produced by the benchmark (diagnostic and benchmark results) are shown on the console output during the whole operation of the benchmark driver. Additionally, the same results of the benchmark are saved to three different types of log files, limited in size of 25 Mb each (and saving to a next-version of the file after exceeding the 25 Mb limit):

- *semantic_publishing_benchmark_queries_brief.log* : stores a brief log of each executed query or operation;
- *semantic_publishing_benchmark_queries_detailed.log* : stores a detailed log for each query or operation - contents and results;

- *semantic_publishing_benchmark_results.log* : stores the summary results of the benchmark, saved each second during the run. Summary results shown on the console output are also saved to that file.

Summary of the benchmark result includes:

- total seconds of benchmark run time
- total number of editorial and aggregation agents
- total number of editorial and aggregation operations and queries
- average number of editorial and aggregation operations and queries for the total benchmark run

3.7 Disclosure items

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 behavior 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) used by the data-generator.

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)
- The hardware used to run the benchmark including : CPU, Chipset, Physical memory, Hard disks - specifications and configuration, network adapters, etc.
- Operating System used (for the benchmark test driver and for the database under test)
- Total cost of hardware listed at full price

3.8 Auditing rules

Execution of the 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} };
- Hardware: visually confirm the hardware specification and where possible login to the test machine(s) and execute appropriate utility software to interrogate the hardware;
- Collection of output values: the final output from the test driver contains the actual benchmarked performance values.

3.9 Publication rules

At the moment of writing of this document publication rules are one of the remaining items.

4 Status of the benchmark

4.1 Remaining items

Remaining items covered here were initially planned to be implemented for the Semantic Publishing Benchmark, which have not been implemented at the moment of writing of the current document:

- Validation of query results;
- A set of queries for testing database capabilities when executing: Faceted Search;
- Publication rules and publication strategy.

4.2 Unclear behaviours and/or risks

This section describes the aspects of the benchmark that could raise doubts in its effectiveness. When designing the benchmark, certain assumptions have been made as a starting point for the development of the benchmark. Following is a list of unclear behaviours or risks that might occur:

- distribution of popular entities and in particular the correlations that in practice are likely to occur, but which have not been modeled in the data generator. Choosing which entity to be 'popular' and which - 'not popular' (or regular) was based on the assumption of a random distribution of popular entities among all. Tagging of all entities (as a part of the data-generation process) then is further biased towards the popular ones (by selecting randomly among all 'marked' as popular), which might not always represent the real-life use-case (where data is not properly distributed and clustering of popular entities could take to extreme levels). Properly distributed data provides optimization opportunities for database query engines, but if it is randomly distributed - that optimization can't be applied;
- distribution ratios of queries / operations for each of the agents types : aggregation and editorial agents execute a mix of queries / operations with a pre-defined distribution for each query / operation. The assumption was made that queries that will take longer to execute (i.e. harder to process by the database engine) should be executed less frequently than the faster-to-execute queries. Although that distribution can be modified for both types of agents, it is up to the database vendor to decide, if results produced by the benchmark are relevant;
- the ratio between editorial and aggregation agents number : in the real-life BBC use case, this ration has not been studied yet, but an assumption was made for a constant rate of update operations (see section 2.3 Processes - editorialOperationsRate = log10(dataset size) + 1), which binds the number of editorial agents indirectly to the dataset size
- the distribution of geonames locations : selection of geonames locations for each generated creative work was decided to be random. Geonames locations thus are randomly distributed among all generated creative works. A certain bias towards more popular places could be used, as is in the 'real-life' case, e.g. distribution of geonames locations around the area of big cities or locations of greater importance should be higher than the rest of all other places;
- random generation of text strings - even though all 'synthetic' text strings and sentences were generated using a random selection of words from a dictionary, still that approach could not be the most proper for some vendors with more specific requirements, e.g. the strings length are randomly selected for a fixed min or max interval, or frequency of certain word inside generated sentences is based on the distribution provided by the random algorithm used;
- the use of context aware serialisation RDF format for generated data - it might be the case that some database vendors will not to be able to easily meet all requirements of the semantic publishing benchmark. For example - generated data must be saved in a context aware RDF serialization format. There is no guarantee that all the databases will be capable of importing generated data saved in such serialization formats e.g. N-Quad, TriG, TriX. Nevertheless, the data-generator can be configured to

produce data in all supported serialization formats and vendors could make an effort to ‘adapt’ the generated data to their specific requirements for importing it.

4.3 Future work

Future work addresses several items encountered during the course of development and use of the semantic publishing benchmark such as:

- additional tuning of queries from the aggregate query-mix is needed – currently the output results produced by the benchmark in terms of values could look discouraging from a user's point of view (e.g. a rather low result value for aggregate operations could be misleading for an adequate estimation of RDF engine's capabilities);
- validation of results – validation of results is an important feature of the benchmark. Further work is required for setting a “ground truth” for validation of query results of the queries;
- data generation – finding the balance between generated metadata (creative works) and the amount of reference data (tagged by the creative works). This could be important for users, because it could provide a more realistic results of RDF engine's capabilities. Also distribution of entities and the correlations that are likely to occur between them needs to be further enhanced and modelled;
- implementation of additional query types, e.g. faceted search queries.

4.4 Implementation plan for addressing remaining items

Implementation plan for the remaining items consists of design, implementation and testing of samples for: Faceted Search queries, design and implementation of validation functionality for query results. Those tasks have been assigned to Ontotext as it takes the main part of the publishing benchmark development process.

4.5 Publication strategy

At the moment of writing of this document publication rules are one of the remaining items.

4.6 Results

The work on fine-tuning and calibration of the Semantic Publishing Benchmark started in October with an effort for benchmarking few of the most popular RDF database engines. Overall, this effort appears to take longer than expected. All the results below should be considered preliminary and are provided for the sake of giving initial feeling about the complexity and the characteristics of the benchmark. Those are not meant to indicate real world performance or to serve for comparison between different engines.

Although we experiment with several engines, there is substantial progress and usable results only for two of those: Virtuoso 7.0 Opensource and OWLIM-SE 5.4. The results presented below come from benchmarking efforts at Ontotext. There is a parallel effort for benchmarking taking place at Forth – those will be published when they become available.

Both databases have been benchmarked on the same hardware configuration shown in Table 5:

Table 5: Hardware configuration used for benchmark results

CPU	2 x Intel(R) Xeon(R) CPU E5-2687W 0 @ 3.10GHz CPU
RAM	256GB

CPU	2 x Intel(R) Xeon(R) CPU E5-2687W 0 @ 3.10GHz CPU
Storage	SSD Drives, RAID-0
OS	SunOS Solaris 5.11

In Table 6 are shown benchmark configuration properties:

Table 6: Benchmark configuration properties

Warm up time	60 s
Benchmark time	300 s
Editorial agents	2
Aggregation agents	14

Results that follow have been measured by execution of a reduced query mix which contains 7 queries – a subset of the full query mix. A reduced query mix has been used to speed up the process of initial calibration of the benchmark.

Most of the experimentation took place with a dataset of 50 million explicit statements (all together, ontologies, reference knowledge and metadata), that includes metadata for 2.4 million creative works (18.7 statements per asset on average). The following table (Table 7) presents the overall results for different datasets sizes, both for OWLIM and Virtuoso.

Table 7: Benchmark results for different dataset sizes

Dataset size	**OWLIM**		**Virtuoso**	
	Editorial ops.	Aggregation ops.	Editorial ops.	Aggregation ops.
10 M	9.1	68.8	142.7	*(? 2.9)*
50 M	8.1	52.9	140.7	17.8
100 M	5.8	39.2	3.55	*(? 0.5)*

With the increase of dataset size, the rates of editorial operations drops for OWLIM, which could be explained by the fact that OWLIM is does a forward-chaining and materialisation, i.e. OWLIM performance reasoning during load or update; with the 50 million statements dataset, after materialization OWLIM deals with 82 million statements. The decline in the aggregation operations can be explained with the fact that some of the queries require "full scan". Overall, the decline in the performance looks reasonable – it is the range of 40% between the 10M dataset and the 100M one, both for editorial and aggregation operations.

For Virtuoso the rate of editorial operations is relatively constant and not dependent on the data set size, which can be explained with the backward-chaining inferencing employed by that engine, that performs reasoning at query time.

As one can observe, within these tests we had troubles to get very consistent results out of Virtuoso. Performance results vary in some ranges, which needs further investigation. This is something that should be straightforward to fix with some help from OpenLink. It should also be noted, that the results for OWLIM are measure when its geo-spatial index is used, while we were not able to craft a query that uses the special-purpose geo-spatial indices of Virtuoso. Again, this should be straightforward and should allow Virtuoso to demonstrate better results on aggregation queries.

The performances of OWLIM for each of the aggregation queries is shown in the next table (Table 8), using average execution time per query. Query execution performances for Virtuoso will not be analysed because of the non-deterministic performance results shown by Virtuoso so far.

Table 8: Query performances, OWLIM

Dataset Size	Average Execution time, ms						
	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
10M	670	7	59	39	44	17	565
50M	700	7	252	101	57	38	601
100M	793	6	628	180	144	22	677

Following is an analysis of the of the query performances for OWLIM shown in Table 8.

Q1 – equivalent to query1 in choke points analysis (Section 3.5). Increasing execution times are expected with the increase of the dataset sizes, as this query contains an aggregate sub-select. Query contains multiple optionals and nested optionals, which are treated as nested sub-queries and optimized separately.

Q2 – equivalent to query2 in choke points analysis (Section 3.5). Execution times are relatively constant because the query optimizer employs various indexes effectively and the query is about an exact entity in the dataset.

Q3 – a query which is not in the initial query mix, containing several UINON sections, OPTIONAL and a FILTER constraint. Result is ORDER BY a date-time value. Motivation to add this query was to explore behaviour of the RDF engine with a query containing unions in combination with optionals and filter constraints. Query execution times rapidly increases with larger dataset sizes which shows that UNION sections present a choke point for the RDF database.

Q4 – a query which is not in the initial query mix. Motivation to add this query is to have a simple query containing few triple patterns and which orders the results by some criteria (currently date of creation). Query execution times increase with dataset sizes, which could be explained with the fact that larger dataset sizes require more time for a full scan and for ordering results later.

Q5 – equivalent to query8 in choke points analysis (Section 3.5). A full scan query with ordering based on a date-time FILTER constraint. Again execution time increases with dataset size.

Q6 – equivalent to query24 in choke points analysis (Section 3.5). It is a geo-spatial query. The query gives opportunity to RDF databases for utilizing their geo-spatial optimisations. Results are produced with geo-spatial optimisations enabled for the three dataset sizes. As expected, execution times should not vary too much as geo-spatial optimisations were utilized.

Q7 – a shorter version of Q1, with all nested optionals removed. Motivation to add this version of the query is to compare its execution performance to the nested OPTIONALS version of Q1. Again increase of execution time is observed with the increase of dataset size. And decrease in query execution time is present compared with Q1 for the same dataset. What is interesting to note is that nested OPTIONALs do not add too much “weight” in the query performance compared to Q1.

4.7 Conclusion

In this status report we have reviewed the progress of the Semantic Publishing Benchmark.

First section describes the motivation for the benchmark and relevance to the industry.

Further we have added detailed information about some of the key features of the benchmark like: input data, data generation, workloads, a description of technical challenges that queries would present a RDF engine. Also we have presented additional information on configuration, execution and fine tuning of the benchmark and data generation.

In the last section we have outlined remaining items that need to be implemented, unclear behaviours encountered during the course of development, and future work that would address those issues and further evolve the benchmark. We have published results of the benchmark on two of the leading RDF engines - OWLIM and Virtuoso.

References

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

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

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

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

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

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