TFT, Tests For Triplestores

Karima Rafes, Julien Nauroy, Cécile Germain

TFT, Tests For Triplestores
Certifying the interoperability of RDF database systems using a continuous delivery workflow

Karima Rafes
INRIA-Saclay / BorderCloud
Orsay, France
Email: karima.rafes@inria.fr

Julien Nauroy
INRIA-Saclay
Orsay, France
Email: julien.nauroy@inria.fr

Cécile Germain
University Paris Sud and CNRS
Orsay, France
Email: cecile.germain@lri.fr

***Abstract*—In March 2013, the W3C recommended SPARQL 1.1 to retrieve and manipulate decentralized RDF data. Real-world usage requires advanced features of Recommendation SPARQL 1.1. As these are not consistently implemented, we propose a software named TFT (Tests For Triplestores) to test the interoperability of the SPARQL endpoint of RDF database systems. To help the developers and end-users of RDF databases, we perform daily tests daily on Jena-Fuseki, Marmotta-KiWistore, 4Store and three other commercial databases. With these tests, we have built a scoring system named SPARQLScore and share our results on the website http://sparqlscore.com.**

***Index Terms*—SPARQL; SPARQL endpoint; interoperability; RDF Database; curation; big data**

I. Pitch

The current W3C recommendation SPARQL 1.1 has been published in its final form in May 2013 [3]. The W3C has defined tests for the compliance of RDF database to this recommendation. Most editors of RDF databases claim to support this latest recommendation but the official implementation report for SPARQL 1.1 [2] shows that none of them pass all the official W3C tests in their entirety. Moreover, software vendors explicitly forbid the disclosure of test compliance results. There exists some reference benchmarks for performance tests, e.g., the Berlin Sparql Benchmark [5], and benchmarks for ontology support, e.g., the University Ontology Benchmark [6] (even though it may be argued that this second type of benchmarks is not representative of real-world applications [7]). Surprisingly enough, it seems that there exists no exhaustive and up-to-date benchmarking facility of the W3C tests that allows the comparison of the RDF databases with respect to their full interoperability. Thus, predicting beforehand the support of a particular SPARQL 1.1 feature in a given RDF database is presently impossible. This is generally damaging to the deployment of the Semantic Web, but has particularly pernicious consequences in scientific research ecosystems.

In the CDS (Center for Data Science) project of Paris-Saclay University, we develop an integrated framework that offers a seamless facility to run and exploit exhaustive testing of RDF databases, in order to help our scientific communities to choose the best solution to share their data. Our panel is really wide: large and well-organized scientific communities such as High Energy Physics (CERN experiences) that have driven computing technology are represented at their best level, as well as small local communities that just discover the need to share beyond short-lived experiments, and many more configurations; the panel includes both hard and soft science.

Our current TFT workflow automatically compiles, deploys and tests every night several hand-picked RDF databases from their sources as well as one SPARQL endpoint offered by a software vendor. It maintains a database of test results accessible from a web interface. In a near future the workflow will be integrated within a platform as a service (PaaS) where the researchers will be able to choose their preferred RDF database. The TFT software will be used to evaluate the conformity of a virtual image hosting an open source RDF database to the latest SPARQL standards, thus providing the scientific end users with critical information for a better informed choice of database based on their needs (performance, support for a particular ontology, etc.), including SPARQL federated queries. The vendors will also be able to propose virtual machines including their own RDF database system, which will then be automatically evaluated using the TFT software before being proposed to researchers.

II. Description

A. Innovation

1) Is interoperability impossible?: The Semantic Web, or Web of Data, aims among other things to share readable information between humans and machines. When this exchange will be possible, new machines will be born to help the humans to use all the information on the Web. The huge amount of information that accumulates on the web is already unusable by humans without using a machine but the majority of the machines are incapable alone of handling this amount of data on the Web.

The machines on the Web become specialized : collector, calculator, semantic parser, databases, etc. The availability of APIs with the WebService technology was the first response to the need to communicate between machines.

Unfortunately, the definitions of the APIs are written by the developers and there are as many APIs as developers. This heterogeneity makes impossible the implementation of

autonomous agents able to discover and consume the data available on the Web, as it had set a simple API and a unique protocol. Enabling such agents is the aim of SPARQL, by making them capable to discover the data across the Web without downloading the data beforehand. Moreover, it is also a major issue for the Web of Things, where every object becomes a potential Web Agent.

There are many implementations of SPARQL endpoints and the performance of RDF database systems improve each year. However, databases interoperability, ie compliance to SPARQL recommendation, is hard to implement and the differences in the implementation of SPARQL endpoints makes complex two tasks that are critical to widespread adoption of RDF by large and organized scientific communities such as HEP:

- migration between databases and their updgrades;
- the development of agents when experienced manpower is too scarce to adapt agents to different type of endpoints.

Total interoperability might be still a long way to go. Should we wait until it happens? Instead, a medium term strategy can take into account the fact that, in practice, the end-users might not absolutely need all RDF features or could cope with some limitations. However, the tradeoffs are different, thus a first-order requirement is to be able to precisely assess the strengths and flaws of databases wrt interoperability. The tool for evaluating interoperability did not exist. **The TFT software answers this critical need.**

2) The last version is always the better: Interoperability is not enough for scientific communities. The most advanced ones want to use the latest database technology (inferences, velocity, clustering and so on). These innovations are rarely available in the stable versions of databases before several months or even several years. The unstable versions are often available for free download and researchers can install these latest versions very quickly with tools like Git. Moreover, for the small communities, the compromise between compliance to standards and cutting-edge performance is often arbitrated more or less blindly in favor of the latter. The TFT software provides a simple solution to make a better informed decision, creating an incentive for selecting the more interoperable technology within the user requirements.

3) IaaS for the researchers: For a Chief Information Officer (CIO) in the academic world providing services to multiple small and poorly organized scientific communities, the condition of interoperability is not enough to deploy a software in an information system. The CIOs have strong Quality of Service constraints (QoS). And facing the wave of softwares, the CIOs are bypassed by researchers who install theirs tools themselves. The solution of CIO is to offer an IaaS (Infrastructure as a Service, typically a local cloud). With this IaaS, the researcher can create or disappear a virtual machine in a few clicks, and can install her preferred tools without bothering with QoS, security and interoperability. After evaluation of the research results by the peers, the resources deployed on the network may disappear fairly quickly because the corresponding data are still rarely integrated in a long term archiving plan.

These careless methods are doomed. The scientific agencies are enforcing the requirement to linking data to results, with reproducibility as the ultimate goal. Nobody can replace the researchers to save their work and share their findings, with mechanisms such as the Digital Object Identifier (DOI) System [12]. However, our PaaS will help: it will facilitate the transition from small ephemeral silos to permanent repositories within clouds, without sacrificing the agile development that is essential to a significant part of real-world good research.

4) Wrap-up: The TFT software certifies the last version of RDF database systems using a continuous delivery workflow. By providing seamless choice of the last best interoperable databases, our innovation facilitates data sharing within and across scientific communities. Beyond selection, our PaaS contributes to the advent of reproducible science.

B. Detailed features and function

1) Overview: The TFT software facilitates the assessment of RDF database systems providing a SPARQL endpoint. The TFT has 4 parts:

- upload the benchmarks into our RDF database,
- run the tests,
- compute a score for each database systems,
- and share the results in RDF via SPARQL.

In the future, these results can power tools in the cloud that will facilitate the provision of solution of latest generation databases for the researchers and will be maintained by CIOs, by ensuring interoperability of data, and will facilitate their work of preserving data by being able to simply migrate data from one system to another. TFT can be integrated into continuous integration environments of database editors, in order to improve their products and the CIOs can check the RDF database system in their environments.

There are about 40 RDF databases currently on the market as listed in the TripleStore page of Wikipedia. Testing all the available solutions would take several months of work to create the automated workflows. This motivates our implementing a solution to help (the end-users) check RDF database systems on the market. Currently TFT offers a score on interoperability of software and also provide a RDF database of detailed test results.

In 2015, the certified databases will be directly deployable into our instance of PaaS. The software editor will be able install, test and optimize their database themselves.

We intend to reach 90% of the SPARQL 1.1 tests after implementing the federated queries, which should happen before the Semantic Web Challenge conference. In the future, the tests about entailment will be extended in function of the researcher needs. We also intend to support performance measures.

We will reuse these results to propose a workflow for measuring SPARQL endpoints implementations. The workflow will be based on (i) the selection of a SPARQL endpoint to test and (ii) features to validate the endpoint and produce an execution report of the features tested on the endpoint. Regarding step (i),

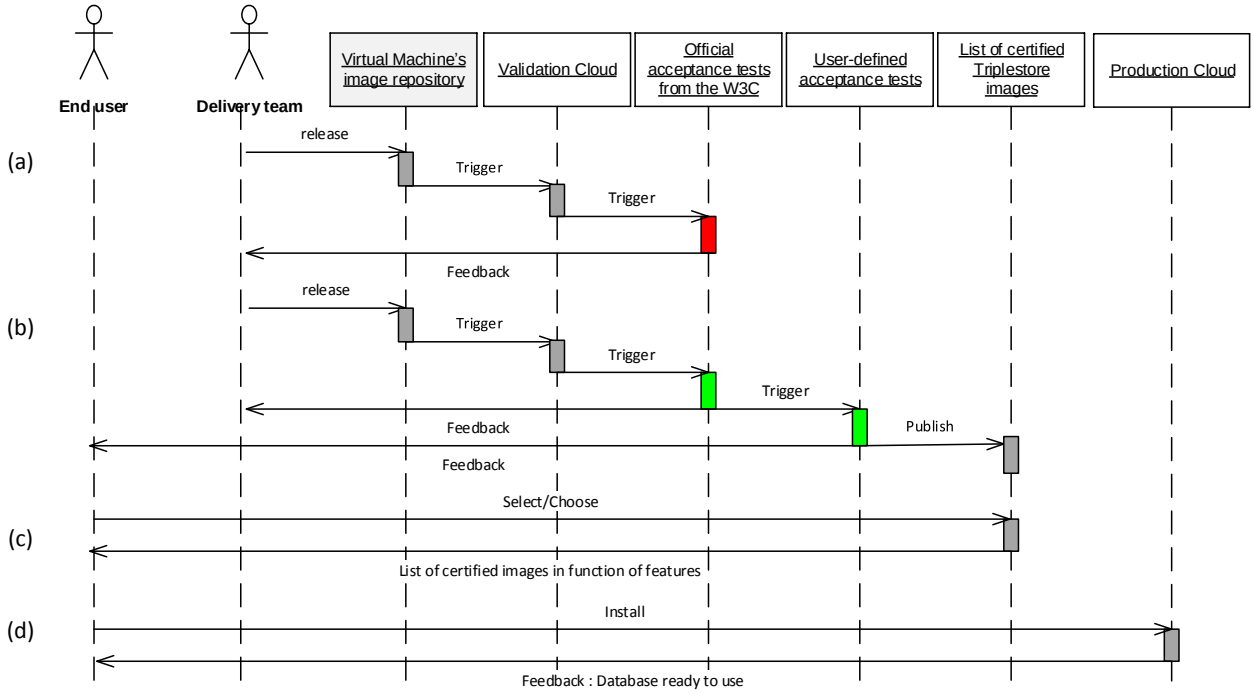


Fig. 1. Certification workflow

our goal is to support both testing a variety of RDF databases, editor-submitted images of virtual machines in our IaaS cloud as well as any user-submitted endpoints available through the Internet. The tests performed at step (ii) include official tests from the W3C,and could include existing benchmarks such as UOBM and user-defined tests based on the support of their ontology. The virtual images will be executed on a validation cloud (also testing performance) and the result of the tests of the chosen SPARQL endpoint will produce a report in EARL via a SPARQL endpoint.

Fig. 1 summarizes the workflow expected in the future. After a new release of a given RDF database, software is made available in the form of a virtual appliance, the image is run on a validation cloud and a set of tests is performed over this new database instance. According to the results of the tests, the validation can either (a) fail and a feedback is provided to the appliances publisher or (b) user-defined tests are run to validate their specific needs. After this second series of tests is run, the appliance is referenced as a W3C-compliant image and feedback is sent back to the users indicating which of their proposed tests pass or fail. This will then allow them to (c) choose a virtual image according to the support of their specific needs and (d) run it on any production cloud available. The comparison of scores or reports obtained from testing different databases will make it possible to choose a database that best fits a specific need, e.g. best supports a given ontology and its inference.

2) The benchmarks:

a) Upload the tests: For the moment, there are two collections of tests: the SPARQL 1.1 test suite comprising 453 tests and the GO3.0 test suite comprising 6 tests. The file config.ini defines the collection of tests and can be extended when necessary.

The SPARQL 1.1 test suite is the one used in the official implementation report for SPARQL 1.1 [1].

The GO3 test suite has been defined by the Grid Observatory 3.0 project. This project uses an OWL ontology of the European Grid Initiative middleware to convert the raw and heterogenous traces into linked data. At this early stage of our development, we use this project as a simple test case of real world requirements. The objective is to reproduce a past experiment [8] using an RDF database and SPARQL queries when data integration and a priori knowledge was previously required to process the raw files hosting the required datasets.

Each collection of tests has a separate folder in the project TFT-tests on GitHub [16]. Every folder contains a file named manifest-all.ttl containing pointers to other test files. This mimics the behavior of the W3C test suite. The sample of code Fig. 2 is an example extracted from the GO3 test suite; the file is named manifest-all.ttl

In the manifest.ttl file (Fig. 3), the suite of tests and the description of each test are defined. For example, test_1 describes a query comprising the following elements:

- Its name is “Substract date”
- The query is in the file q01.rq
- The input is in pbs.ttl

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix mf:     <http://www.w3.org/2001/sw/DataAccess/tests/test-manifest#> .
@prefix qt:     <http://www.w3.org/2001/sw/DataAccess/tests/test-query#> .

<> a mf:Manifest ;
        rdfs:label "Grid Observatory tests" ;
        mf:include (
                <ERT-ART/manifest.ttl>
        ).

<http://grid-observatory.org/> rdfs:label "Grid Observatory 3.0" ;
        mf:conformanceRequirement (
                <ERT-ART/manifest.ttl>
        ).
```

Fig. 2. File GO3/manifest-all.ttl in the project TFT-tests [16].

```
<>  rdf:type mf:Manifest ;
    rdfs:comment "Query tests to calculate ERT-ART" ;
    mf:entries
    (
       :test_1
       :test_10
) .

:test_1 rdf:type mf:QueryEvaluationTest ;
    mf:name    "Substract date" ;
    dawgt:approval dawgt:Approved;
    mf:action
         [ qt:query  <q01.rq> ;
           qt:data   <pbs.ttl> ] ;
    mf:result  <q01.srx> .
```

Fig. 3. Sample of file GO3/ERT-ART/manifest.ttl in the project TFT-tests [16]. You can see the list of tests and the definition of first test.

```
:test_10  rdf:type mf:QueryEvaluationTest ; #Type of test
          mf:name    "Query to calculate ERT-ART" ;
          dawgt:approval dawgt:Approved;
          mf:feature sd:BasicFederatedQuery ; #Type of tool to run the test
          mf:action
                [ qt:query  <q10.rq> ;

                  qt:serviceData [
                                   qt:endpoint <http://example1.org/sparql> ;
                                   qt:data     <pbs.ttl>
                                 ];
                  qt:serviceData [
                                   qt:endpoint <http://example2.org/sparql> ;
                                   qt:data     <bdii.ttl>
                                 ]
                ] ;
          mf:result  <q10.srx> .
```

Fig. 4. Sample of file GO3/ERT-ART/manifest.ttl in the project TFT-tests [16]. You can see an example of test for a federated query.

```
SELECT ...
WHERE {
        SERVICE <http://example2.org/sparql> { ... }
        SERVICE <http://example1.org/sparql> { ... }
}
```

Fig. 5. Sample of file GO3/ERT-ART/q10.rq in the project TFT-tests [16] where the endpoints are the same as in the definition of test Fig. 4

- The expected result is in q01.srx

Fig. 4 shows an example of a test with a federated query. This test needs to have two remote endpoints to execute the query. The file pbs.tll contains the input to be loaded into the first remote endpoint and the file bdii.ttl contains the input to be loaded into the second remote endpoint. The URI of the endpoints, http://example1.org/sparql and http://example2.org/sparql, are the same URI used in the query, see Fig. 5.

```
[SERVICE]
endpoint["http://example.org/sparql"] = "http://onevm-194.lal.in2p3.fr/sparql/"
endpoint["http://example1.org/sparql"] = "http://onevm-60.lal.in2p3.fr/sparql/"
endpoint["http://example2.org/sparql"] = "http://onevm-194.lal.in2p3.fr/sparql/"
```

Fig. 6. File config.ini in the software TFT [9] where the remote endpoints are defined.

```
git clone --recursive https://github.com/BorderCloud/TFT.git
cd TFT

./tft-testsuite -a -t fuseki \
  -q http://example.com:3030/tests/query \
  -u http://example.com:3030/tests/update

./tft \
  -t fuseki \
  -q http://example.com:3030/tests/query \
  -u http://example.com:3030/tests/update \
  -tt fuseki -tq http://127.0.0.1/ds/query -tu http://127.0.0.1/ds/update \
  -o ./junit \
  -r ${BUILD_URL} \
  --softwareName=Fuseki \
  --softwareDescribeTag=v${VERSIONFUSEKI}  \
  --softwareDescribe="${BUILD_TAG}#${FILEFUSEKI}"
```

Fig. 7. This script downloads TFT, uploads, passes and saves the tests and the results in a RDF database.

During the tests, TFT will replace the URIs of the remote endpoints with the URI contained in the file config.ini (Fig. 6).

The test suites needs 3 remote endpoints but in practice, there is no limit on the maximum number of endpoints that can be defined. In our case, the remote endpoints are hosted into the StratusLab [10] IaaS cloud and the virtual images of the RDF databases are available in its marketplace [15]. The remote endpoints currently need to be instantiated before running TFT but we plan on instantiating these databases on the fly in a near future using images available in StratusLab marketplace.

Before running tests, TFT with the script tft-testsuite uploads all the file turtle in a RDF database. With another script, TFT reads each test and saves its result in the same RDF database via SPARQL queries. The SparqlScore website uses this RDF database to share the results of TFT.

b) Pass the tests: We use Jenkins, a continuous integration server, because the database systems are often Open Source and in a Git repository.

Fig. 7 shows an example of a script execution by our continuous integration server. The script named tft follows the same workflow for each test.

- Delete all data from the main test database and the remote test database(s)
- Load initial data to define the initial state in the main database and the remote database(s)
- Run the tests in the main database; in case of federated queries, the main database is responsible for contacting the remote ones (this is the normal behavior of federated queries).
- Monitor the response to the test and/or control the final state obtained in the databases.

After the compliance tests have been run, the script tft saves and shares the results.

c) Computing a score: The current tests database currently comprises a total of 459 tests. You can see the details in the Table. I.

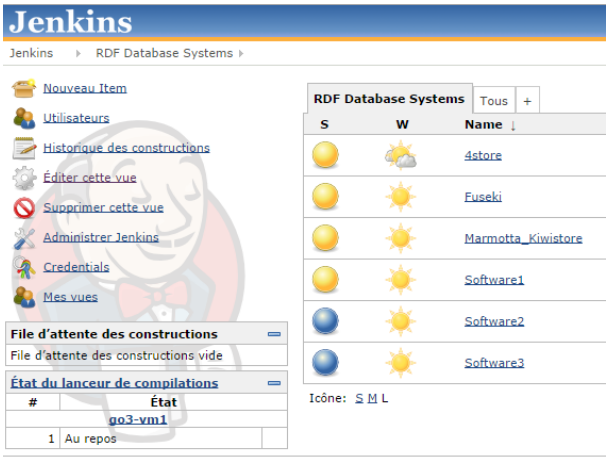
TABLE I
DETAILS ABOUT THE TYPES OF TESTS

Title of test suites	Number of tests
Grid Observatory 3.0	6
SPARQL 1.1 Entailment Regimes	13
SPARQL 1.1 Federation Extensions	10
SPARQL 1.1 Query Language	268
SPARQL 1.1 Query Results CSV and TSV Formats	6
SPARQL 1.1 Query Results JSON Format	4
SPARQL 1.1 Update	80
Total :	459

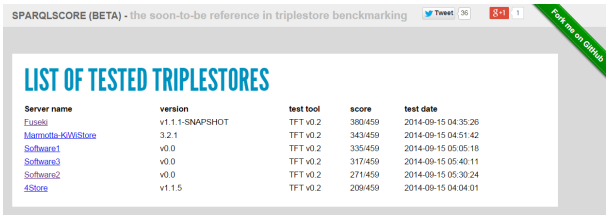
Choosing a particular weight for each test would be highly debatable. We calculate a simple global score for each RDF database system: one point is given for each passed test. The script tft-score calculates this score and share these scores with the results of tests.

Two test suites were removed from the global test suite, namely "SPARQL 1.1 Service Description" and "SPARQL 1.1 Protocol", because they had no test to run. We would develop the tests of "SPARQL 1.1 Protocol" when the classics tests will be insufficient to compare the database together.

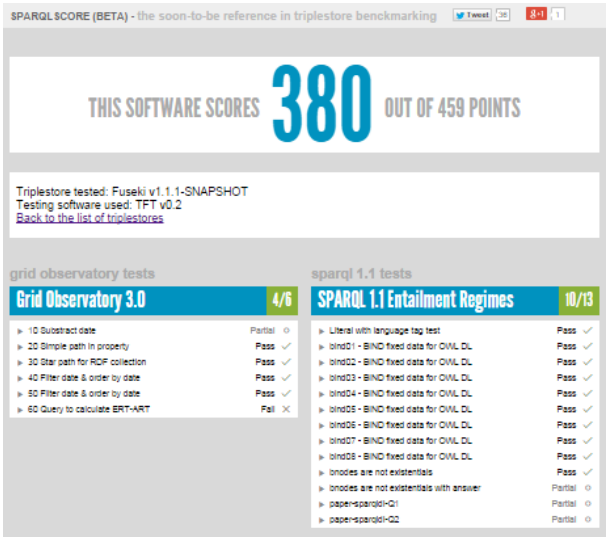
The tests of "SPARQL 1.1 Service Description" will imple-



(a) We use Jenkins, a continuous integration server, that runs the tests.



(b) We use SparqlScore.com to share the score of RDF database systems.



(c) SparqlScore prints the result of each test.

Fig. 8. Jenkins and SparqlScore uses the results of tests.

mented in another software with a different goal, namely to test a database with its final data. The aim to test the Service Description will be to alert a researcher that the description of his data is insufficient.

3) Share the results: After the compliance tests have been run, we share three results for: * the developers, the editors of database system, * the machines, the other software in an intranet to propose the last best and stable databases to researchers, * and the humains, ie the end-users of RDF databases worldwide.

a) With the editors: TFT creates a report in JUnit format compliant with the Jenkins software (Fig. 8,a). Jenkins can check the last push in the Git repository about a software and can give a feedback to developers in real-time. If a software editor integrates TFT in its Jenkins server, he will also be able to reject automatically the last delivery if a test show a regression of interoperability. An example of tests is available in the project TFT-tests on GitHub [16] and the developers can see the tests of the end-users and can reproduce the same tests. They can also add their own tests easily.

b) With the machine: After the compliance tests have been run, the script tft generates two results: a report in JUnit format for our own Jenkins server and a report in RDF/EARL (Evaluation and Report Language) format [14].
The report in EARL format is saved in an RDF database exposing a SPARQL endpoint. Thus another machine can check easily the compliance of RDF database systems. So, we can integrate new software almost in real-time following the last deliveries of developers in our future platform PaaS and check the compatibility. The continuous integration platform can alert if there is a regression in the software and the machine can detect the improvements, propose the last best stable databases and migrate automatically the database in the best last stable solutions for the researchers.

c) With the end-users: The SparqlScore.com website, Fig. 8, illustrates a machine that can reuse the test results and associated scores. Since the amount of data to fetch and process can be quite high and in order to increase the users experience of the website as well as to relieve our database, we use the Smarty [11] library to cache the results of the SPARQL queries to build the report in HTML5.
With this website, the end-users can see the real interoperability of database and in the future, others indicators.

C. Design choices

We integrate the linked data technologies where the input are the tests in the turtle format with the ontology defined by the SPARQL 1.1 WG; the output is a RDF database that is feeded by a SPARQL Update query after each test.
This design offer the possibility to write quickly a new test and everybody can propose a new test or fork the tests via a project in the GitHub's Service [16].

D. Lessons learned

The TFT software currently runs 80% of the SPARQL 1.1 tests from the W3C. The last tests to implement need

specific configuration during the installation of the database such as allowing federated queries or need additional and database-specific parameters, such as specific HTTP headers for choosing Entailment Regimes.

Already with 80% of tests, we saw the gap between the different databases. We pushed the results in a website: http://sparqlscore.com. Very soon after the launch of the website a first database vendor contacted us to include their software in our tests, providing a specifically set-up SPARQL endpoint. The integration went smoothlessy. We hope that the sparqlscore.com website will encourage other editors to test their software and improve their compatibility to the standards.

To add more databases in the benchmark, we have to build a specific script that compiles and deploys it from the latest sources (preferably). The deployment can be complex (e.g. security rules) and sometimes the protocol is not exactly the same as for other databases (update queries or queries with a specific entailment regime). The current prototype was developed in 2 months and another month was required to write the scripts to compile and install the five RDF databases we selected. As discussed before, the sixth RDF database was provided as an endpoint by the software vendor.

E. Web access

The TFT software is under a Creative Commons Attribution-ShareAlike 4.0 International License. The aim of this license is to share the same software to tests and compare objectively the databases on the market. TFT and TFT-tests (the collections of tests) are available via their repositories [9][16].

The SparqlScore software is also available via its repository [16] and everybody can read the last results with our continuous integration plateform on the website http://sparqlscore.com/ (Fig. 8).

We can not share officially the endpoint of our RDF database because our server is not designed to meet hundreds of users. Maybe in the future, we share officially. We can give the url upon request.

APPENDIX

In the Table. II, you can see that we meet the requirements of the Semantic Web challenge 2014.

ACKNOWLEDGMENTS

This work has been partially funded by the FUI project TIMCO and by France Grilles.

REFERENCES

[1] SPARQL1.1: Test case structure. http://www.w3.org/2009/sparql/docs/tests/, 2012.

[2] Official implementation report for sparql 1.1. http://www.w3.org/2009/sparql/implementations/, March 2013.

[3] Recommendations of the w3c : Sparql 1.1 (protocol and rdf query language). http://www.w3.org/TR/sparql11-overview/, March 2013.

[4] Andy Seaborne, W3C RDF Data Access Working Group. Vocabulary for DAWG test cases. http://www.w3.org/2001/sw/DataAccess/tests/test-dawg, 2001.

[5] Christian Bizer and Andreas Schultz. The Berlin SPARQL benchmark. *Int. Jal. On Semantic Web and Information Systems*, 4(2):1–24, 2009.

[6] L. Ma et al. Towards a Complete OWL Ontology Benchmark. In *The Semantic Web: Research and Applications*, volume 4011 of *LNCS*, pages 125–139. 2006.

[7] T. Weithner et al. Whats wrong with owl benchmarks. In *2nd Int. Workshop on Scalable Semantic Web Knowledge Base Systems*, pages 101–114, 2006.

[8] Cecile Germain-Renaud, Alain Cady, Philippe Gauron, Michel Jouvin, Charles Loomis, Janusz Martyniak, Julien Nauroy, Guillaume Philippon, and Michèle Sebag. The grid observatory. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 114–123. IEEE, 2011.

[9] Karima Rafes, Inria. Repository Git of software TFT. https://github.com/BorderCloud/TFT, 2014.

[10] Charles Loomis, Mohammed Airaj, Marc-Elian Bégin, Evangelos Floros, Stuart Kenny, David O'Callaghan, et al. Stratuslab cloud distribution. *European Research Activities in Cloud Computing*, 2012.

[11] New Digital Group, Inc. What is Smarty? http://www.smarty.net/docs/en/what.is.smarty.tpl, 2014.

[12] Norman Paskin. Digital object identifier (doi) system. *Encyclopedia of library and information sciences*, 3:1586–1592, 2008.

[13] Semantic Web Challenge 2014. Challenge Criteria of Semantic Web Challenge. http://challenge.semanticweb.org/2014/criteria.html, 2014.

[14] Shadi Abou-Zahra, W3C/WAI. Evaluation and Report Language (EARL) 1.0 Schema. http://www.w3.org/TR/EARL10-Schema/, 2011.

[15] Stratuslab cloud distribution. Marketplace of images for the Stratuslab cloud. https://marketplace.stratuslab.eu/marketplace/metadata, 2014.

[16] The W3C SPARQL Working Group and The grid observatory. Repository Git of test suite SPARQL1.1 and of Grid Observatory. https://github.com/BorderCloud/TFT-tests, 2014.

TABLE II
CHALLENGE CRITERIA [13]

General criteria	**0ur applicant**
Demonstrate a clear commercial potential	The data produce objectively by the software be able to feed the PaaS plateform and will help the end users to choose objectively their interoperable databases.
Demonstrate a large existing user base; or functionality that is useful and of societal value	Clearly the solution can help the end users of databases to really share their data in the Linked (Open) Data and it will help the editors of database to improve their softwares. The software is part of the CDS open data platform. CDS federates 25 laboratories and involves more than 200 researchers on the major French scientific campus University Paris-Saclay
Open Track criteria	**Our applicant**
1. The application has to be an end-user application, i.e. an application that provides a practical value to general Web users or, if this is not the case, at least to domain experts. It should show-case functionalities that the use of semantic web technologies can bring to an application.	The website Sparqlscore.com gives an end-user application and an example to reuse the data of TFT software.
2.1 The information sources used should be under diverse ownership or control;	It is possible to add tests of diverse ownership beyond the tests of W3C working group.
2.2 The information sources used should be heterogeneous (syntactically, structurally, and semantically); and	The tests in input are in the the format turtle with the DAWG's ontology [4] and the result are accessible via a SPARQL endpoint with the EARL's ontology [14].
2.3 The information sources used should contain substantial quantities of real world data (i.e. not toy examples).	We use the real tests of the SPARQL 1.1 Working Group [1].
3.1 The meaning of data has to play a central role. Meaning must be represented using Semantic Web technologies;	TFT uses the Semantic Web technologies (Turtle, SPARQL, RDF databases) and tests the interoperability to help to build the Semantic Web.
3.2 Data must be manipulated/processed in interesting ways to derive useful information;	We produce an useful information to compare the interoperability of RDF database.
3.3 This semantic information processing has to play a central role in achieving things that alternative technologies cannot do as well, or at all;	Our software shares our data with a SPARQL endpoint between our systems. So, we eat our own hot dog and demonstrate the validity to Linked Data technology.
Additional Desirable Features	**Our applicant**
The application provides an attractive and functional Web interface (for human users)	The website Sparqlscore.com gives an end-user application
The application should be scalable (in terms of the amount of data used and in terms of distributed components working together).	The system can be scalable because we can test quickly a new SPARQL endpoint (if possible hosted by the editor).
Ideally, the application should use all data that is currently published on the Semantic Web.	not applicable but we help to build the real interoperability on the Semantic Web.
Rigorous evaluations have taken place that demonstrate the benefits of semantic technologies, or validate the results obtained.	Our tests are reproductible, tranparent and so rigorous.
Novelty, in applying semantic technology to a domain or task that have not been considered before	Using the Linked Data to share the results of a integration continuous plateform is new and the schema EARL [14] is still a draft recommendation to W3C.
Functionality is different from or goes beyond pure information retrieval	We produce new linked data.
Contextual information is used for ratings or rankings	not applicable
Multimedia documents are used in some way	not applicable
There is a use of dynamic data (e.g. workflows), perhaps in combination with static information	The workflow of our continuous integration platform consumes and produces each day dynamic data about the latest version of RDF database systems.
The results should be as accurate as possible (e.g. use a ranking of results according to context)	The website Sparqlscore.com gives an end-user application where you can see the details of the outcome of each test.
There is support for multiple languages and accessibility on a range of devices	The tests' results are accessible via an SPARQL endpoint and so any device can reuse these results.