New Approach based on Model Driven Engineering for Processing Complex SPARQL Queries on Hive

Mouad Banane[1], Abdessamad Belangour[2]
Ben M'sik Faculty of Science, Hassan 2 University
Casablanca, Morocco

***Abstract*—Semantic web technologies are increasingly used in different domains. The core technology of the Semantic Web is the RDF standard. Today with the growth of RDF data it requires systems capable of handling these large volumes of data and responding to very complex requests at the join level, Several RDF data processing systems have been proposed, but are not dedicated to handling complex SPARQL queries. in this paper we present a new approach based on model driven engineering for processing complex SPARQL queries using one of the big data processing tools named Hive. We evaluate our system using three datasets from LUBM Benchmark. The results of this evaluation show the performance, and the scalability of our approach, also give very interesting results when it is compared with existing works.**

Keywords*—SPARQL; big data; model driven engineering; RDF

I. Introduction

Since its appearance in the early 1990s, the web has profoundly transformed contemporary society. It is now ubiquitous in our lives, whether in the way we communicate, work, play, buy products, and so on. It is now the most used application of the Internet to create, share and use information.

Victim of its success, the web has become a huge reservoir of information that sometimes makes finding information difficult, especially when it comes to finding reliable and relevant information. Faced with this problem, web inventor Tim Berners-Lee came up with the idea of adding "semantics" to web documents [1]. This idea considers the semantic web as an evolution of the web that would allow the available data (content and links) to be more easily usable and interpretable by both human and machine.

In the Semantic Web, information contained in resources such as web pages, databases, services ... will be available and understandable not only for men but also for machines, programs or IT agents. The Semantic Web is designed so that the content of resources on the Web can be made semantically "understandable" and accessible by software. Resources available on the Internet such as documents, images, services or even physical resources that are not on the Internet but their references are available such as physical books, people have an associated semantics. Thanks to this semantics, the organization, the backup, the search for information could be realized, processed automatically by software. The semantics of the content of resources in the Semantic Web must, therefore, be made explicit and available to the machines in a formal and standardized representation. Standardization can help different programs to interoperate or exchange data. The way to represent semantics in the Semantic Web is to use the RDF standard. Today the amount of RDF data continues to grow in a very remarkable way, it is no longer possible to store all linked data sets on a single machine while being able to evolve requests from multiple and varied users. Thus, such volumes of data have raised the need for distributed storage architectures and query processing frameworks. The arrival of Hadoop and especially its MapReduce framework greatly improved the development of massively parallel applications. Nevertheless, the MapReduce API does not allow complex operations making the development of large programs a difficult task for intermediate programmers. To overcome the limitations, higher level languages have been developed as Hive, providing a declarative way of writing programs that are then automatically translated into MapReduce jobs.

As part of model-driven software engineering, model transformation is an increasingly important activity in the development cycle: code generation, maintenance, code optimization, aspect composition, reverse engineering, etc. Thus, model transformation languages represent prime components of a development environment. The basic object of these languages is the model that requires the definition of new operators, among others, construction, navigation, composition, comparison and evaluation.

On the other hand, the explosion of Web data offers a new challenge to manage them. For the management of these large volumes of data we present a new technique of RDF data queries based on the principle of meta-models that allows to transform a given SPARQL queries into a Hive program. To evaluate the SPARQL2Hive approach we use The Lehigh University Benchmark LUBM [2], the results show the efficiency of SPARQL2Hive when the amount of RDF data is very important.

The outline of the paper is as follows: Section 2 exposes some existing related works on this topic. Section 3 describes semantic web technologies RDF and SPARQL, and also Hive. Section 4 presents our main contribution SPARQL2Hive. Section 5 evaluates and analyzes our approach with the Lehigh University Benchmark. Finally, we conclude in Section 6.

II. Related Work

Recently, many research efforts have been devoted to developing a new scalable RDF data volume management system, such as Jena-HBase [3]: a distributed RDF triplestore based on HBase [4] the NoSQL database management system

The schema of this system consists of two parts storage and querying, it stores the RDF documents in HBase tables, and for querying this data this system uses the Jena Framework to execute the SPARQL queries. PigSPARQL [5] is an RDF data manipulation system, the principle of PigSPARQL is the translation of SPARQL queries into an executable program of PigLatin [6] language of Apache Pig [7], the translation process starts with the analysis than a compile step of algebra, then optimization of algebra before it is translated into PigLatin, this program will be transformed into a Job MapReduce. The work in [8, 9] presents comparative studies of existing systems based on NoSQL technology and which propose the management of large volumes of RDF data according to the NoSQL models: key/value model, document model, column model, and graph model. From us we have presented a new, more detailed [10] study and brings together about all Distributed RDF Stores based on NoSQL. Galarraga et al. present [11] an evolutionary system that is based on a technique for optimization in the case of large volumes of RDF data.

III. Preliminaries

A. RDF

Developed by the W3C as part of the Semantic Web activities, RDF is not strictly speaking a metadata schema. It constitutes a structured data description model inspired by the logic of first-order predicates and graph theory.

Its genericity and flexibility provide an interoperable framework for describing all types of resources in a networked environment such as the Web. RDF is a model that allows expressing assertions in a very simple model comparable to a simple sentence: [subject] [predicate] [object]. Each assertion forms a triple whose different components are expressed as a URI. The interest of RDF lies in the fact that it is possible to exploit RDF triplets without conversion and whatever the vocabulary used, unlike XML for which it is necessary to convert the data if they do not. Do not use the same scheme. Thus, it does not require the different producers to agree strictly on a metadata structure. The expression "social contract written by john jack rousseau" can be expressed by writing an RDF triple, which can be represented as a subject-predicate-object graph (Fig. 1):

B. SPARQL

To enable the construction of RDF data queries, the W3C has developed the SPARQL standard. It is both a protocol, a query language, and a formalism for the expression of results. SPARQL queries are used to dynamically query data in RDF without downloading all raw data.

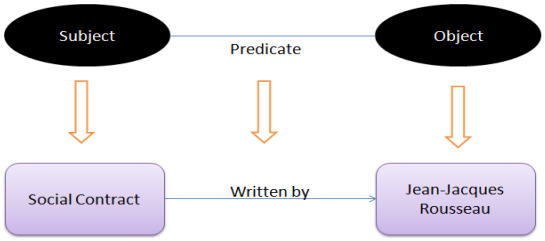

Fig. 1. Example of a RDF Triple.

With RDF and the SPARQL language, it is possible to query the structured information contained in the metadata without having a lower common denominator. As there are many data warehouses structured according to RDF, it is mostly possible to build web or mobile applications with RDF data linked together by URIs. These URIs take mostly the form of URLs, i.e. web addresses. This is the principle of linked data.

C. Model Driven Engineering

In Model Driven Engineering (MDE) [12], a formalism, or modeling language, in which a model is expressed, is described by a meta-model. The meta-model has the particularity of containing all the concepts necessary to create models in a domain, a particular context: the meta-model is at the heart of the MDE. More precisely, the role of a meta-model is to define, as a minimum, the abstract syntax of a formalism, by defining concepts and relations between them [13].

For example, the meta-model of a programming language represents its grammar, the meta-model of an XML file represents its DTD (Document Type Definition).In MDE everything is model, a meta-model is also a model, described according to a certain formalism: the meta-meta-model.

In a MDE context, we call model transformation any program whose inputs and outputs are models. We also speak of "source model" and "target model". Depending on whether a transformation outputs a model or code, it will be referred to as "Model To Model" ("M2M") or "Model To Text" ("M2T"). However, let us nuance this definition, because from a rigorous point of view, in MDE, "everything is model".

IV. System Architecture

In this section, we present a meta-modeling of our approach. The Metamodel describes our approach independently of the source and target models used. Then, we illustrate the two meta-models of Hive and SPARQL with the case of the transformation of a query of the SPARQL language into a HiveQL program.

The aim of the SPARQL2Hive approach is to transform a given model, expressed in a formalism: SPARQL standard of the Semantic Web into another model expressed in another formalism: Hive tool for the management of Big Data. Fig. 2 illustrates the different stages of operation of our approach. The source model contains a set of elements to transform. We randomly assign a transformation possibility for each element of the source model.

We evaluate, via an objective function, the quality of the proposed transformation. Finally, the last step is to refine the solution or proposed solutions and iterate the different steps until converging towards an acceptable solution (target model of good quality).

We use the principle of model-driven engineering to realize this transformation of SPARQL to Hive, the help is to first realize the two metamodels SPARQL and Hive then in the second stage we propose a transformation between its two meta-models using transformation languages like ATL [14,15].

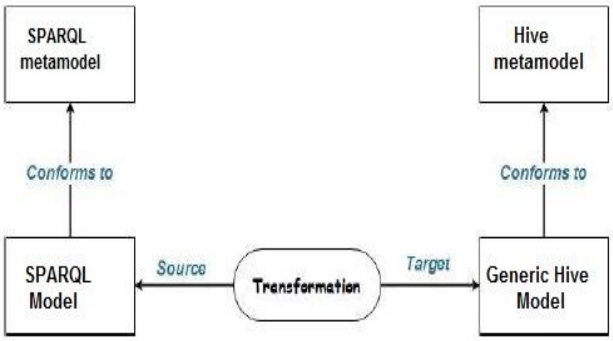

Fig. 2. System Architecture.

A. SPARQL Metamodel

The structure of a SPARQL query is very similar to that used in the SQL language, so a SPARQL query can be a SELECT, ASK, CONSTRUCT query. A SELECT query, of the interrogative type, is used to extract from the RDF graph a subgraph corresponding to a set of resources that satisfy the conditions defined in a WHERE clause. But a CONSTRUCT request, of constructive type, generates a new graph which completes the interrogated graph. In addition, a SPARQL query can have other purposes than providing a set of matches to the variables specified in the SELECT. Indeed, in the SPARQL language, it is possible to ask if a request has at least one solution. To do this, the SELECT is replaced by an ASK.

The SELECT clause contains as SQL: SELECT, FROM and WHERE, There are also other elements in the SPARQL language that make it possible to specify prefixes (PREFIX), conditions (FILTER), disjunctions (UNION), filters on the production of results (LIMIT and OFFSET). Fig. 3 presents the SPARQL meta-model. Fig. 4 presents the Hive meta-model.

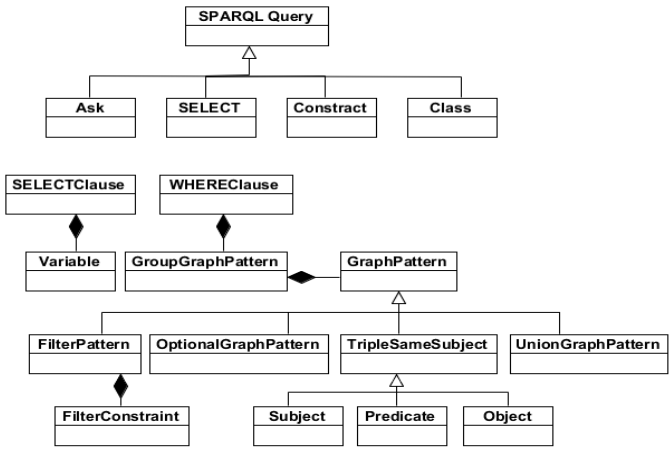

Fig. 3. SPARQL Meta-Model.

B. Hive Meta-model

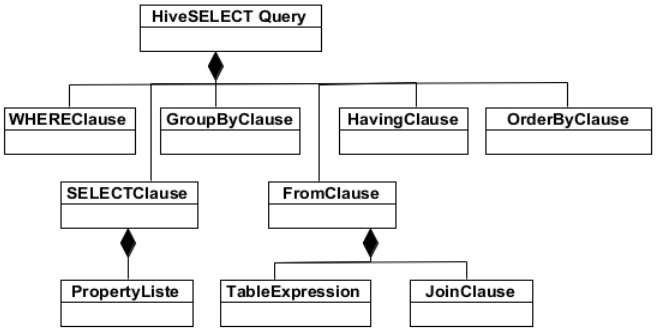

Fig. 4. Hive Meta-Model.

C. Transformation

After the creation of the OMG MDA standard, many tools based on this approach have emerged, such as Atlas Transformation Language (ATL) [15] for model transformation. This language is close to the standard QVT (Query, View, Transformation), proposed by the OMG. This resemblance is historic since ATL is the first attempt to implement QVT [15]. ATL is now one of the most mature model transformation languages, so naturally we chose this language. Fig. 5 shows our transformation engine.

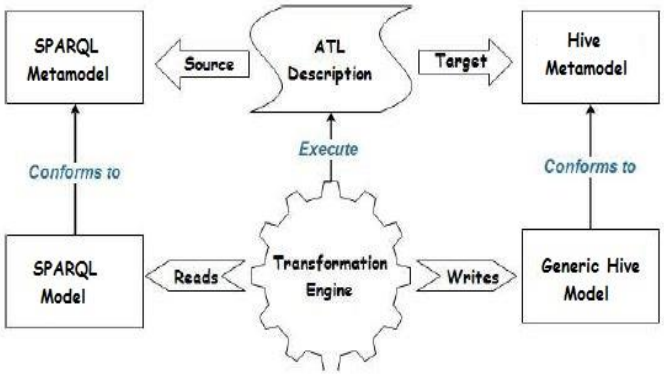

Fig. 5. Transformation Engine.

An ATL program consists of rules that specify how the elements of the target model should be created based on the elements in the source model. These rules are always established according to the following schema:

```
rule Nom_de_la_regle
    from
       i: Nom_MM_Source!Nom_Méta-Classe_Source

    to
        o: Nom_MM_Cible!Méta-Classe_Cible
              Attribut1 <- i.AttributA,
              Attribut2 <- i.AttributB+i.AttributA,
```

- Rule, from and to are the language instructions.

- *i* (resp., o) is the name of the variable that in the body of the rule represents an instance of the source (or target) meta-class in the source (or target) model.
- *Attribute1,2* (respectively A, B) are the attributes of the target meta-class (or source) of the target (or source) meta-model.

The exclamation point is used to specify which meta-model belongs to a meta-class. A translation into everyday language would give:

The Rule *rule* creates for each instance *i* identified in the source model an instance o of the target meta-class in the target model, giving Attribut1 the value of *AttributA* and Attribut2 the value of the sum *AttributA* plus *AttributB*.

ATL makes it possible to factorize the rules with the use of helper, which one can assimilate to functions.

The execution of this transformation gives the result obtained in the following figure (Fig. 6).

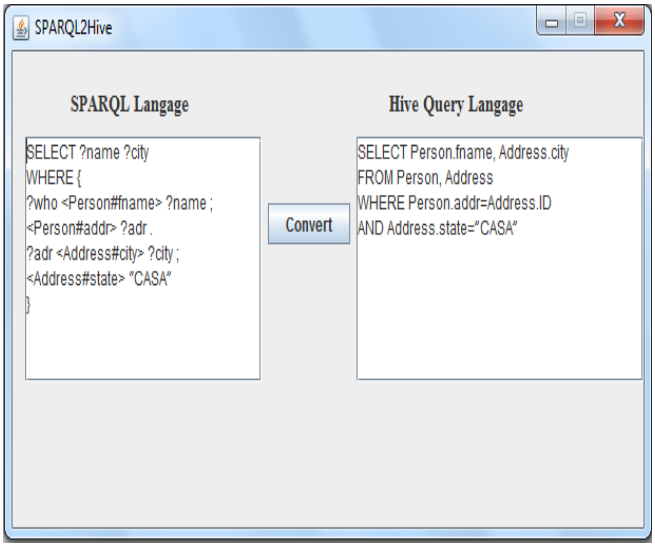

Fig. 6. Conversion Example of SPARQL Query to HiveQL.

V. Evaluation

To evaluate our approach, we performed a validation using the LUBM Benchmark [16], we execute LUBM queries on three datasets of different sizes to better analyze our SPARQL2Hive system. We present, in the first subsection, the context of our experiments. Then we will analyze the results obtained. Finally, we will evaluate the impact of the size of the sample database on the quality of model transformation.

SPARQL2Hive is implemented on the Hadoop 3.xy version and the Hive 3.1.0 version on a machine with a 2.3 GHz Intel Xeon processor; this machine can store up to 4 TB of hard disk storage and RAM storage of 16 GB. LUBM1, LUBM2 and LUBM5 these three datasets used in this experiment, they have the following triplets' number: 138 million triples, 275M and 689M and the sizes of these three datasets are: 11.4 GB, 22, 77 GB and 56, 8 GB. The results obtained for the loading time of these three games to give are presented in Table I:

TABLE I. Loading Time

Dataset	LUBM1	LUBM2	LUBM5
Loading Time(ms)	1,26	3,05	7,9

TABLE II. System Runtime for LUBM Queries (MS)

Queries	LUBM1	LUBM2	LUBM5
Q1	481	537	752
Q2	429	516	641
Q3	535	583	633
Q4	509	621	627
Q5	743	797	851
Q6	657	720	773
Q7	678	736	794
Q8	179	216	201
Q9	129	130	142
Q10	181	237	252
Q11	121	135	150
Q12	83	103	126
Q13	376	405	451
Q14	325	361	404

Table II illustrates the results of running the 14 LUBM queries on the three instances of this Benchmark.

We compare our SPARQL to Hive system with Jena by always using the three datasets LUBM1, LUBM2, LUBM5, generally on the majority of the queries; SPARQL2hive is more powerful than Jena at the runtime of LUBM Benchmark queries. Figures 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16 show the results of this comparison for all LUBM queries.

The results obtained after this experiment show the SPARQL2Hive efficiency when the RDF data volume is very large, SPARQL2Hive does not take a lot of time to load the data. Because it performs a simple translation of a given SPARQL query into a program Hive [17], Query Language. But with the Jena Framework, the operation becomes a little complicated because the request goes through a set of steps, which takes a lot of time, especially for loading data, preparing data for recovery, more than Jena uses a lot of resources such as RAM.

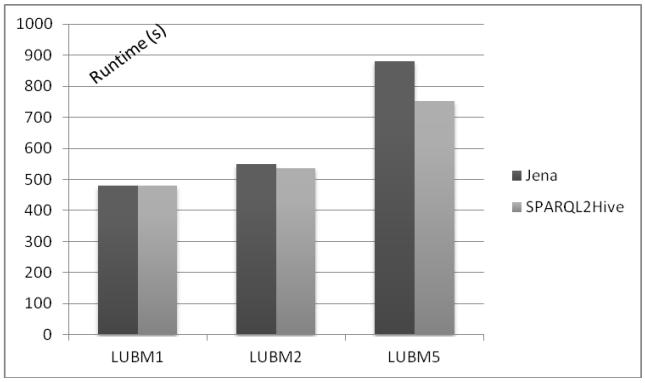

Fig. 7. LUBM Q1 Runtime.

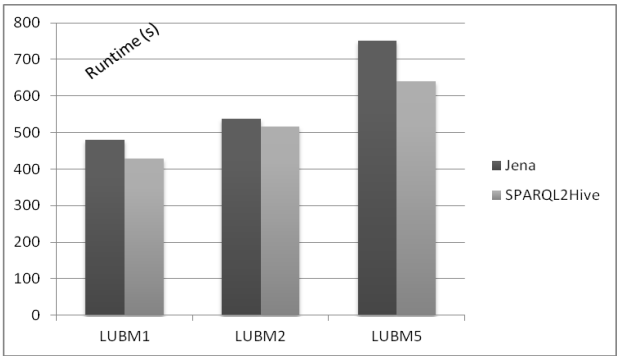

Fig. 8. LUBM Q2 Runtime.

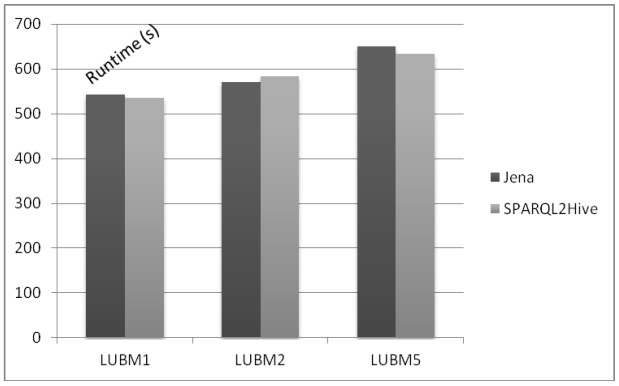

Fig. 9. LUBM Q3 Runtime.

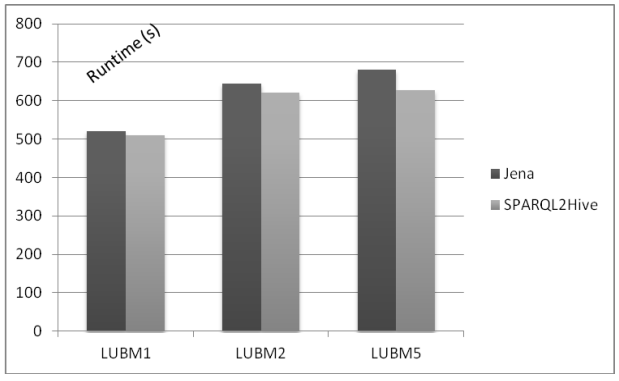

Fig. 10. LUBM Q4 Runtime.

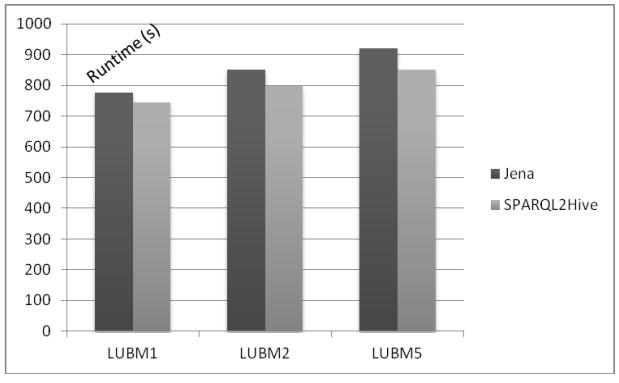

Fig. 11. LUBM Q5 Runtime.

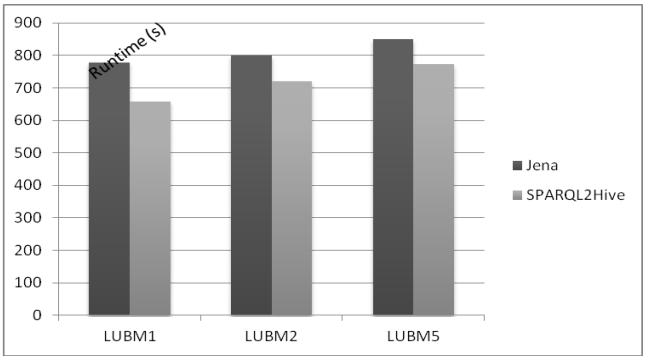

Fig. 12. LUBM Q6 Runtime.

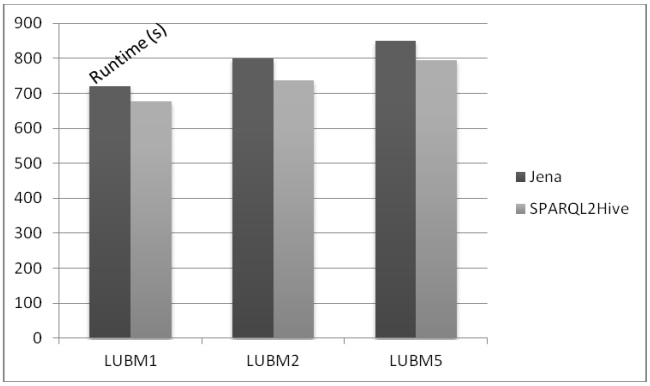

Fig. 13. LUBM Q7 Runtime.

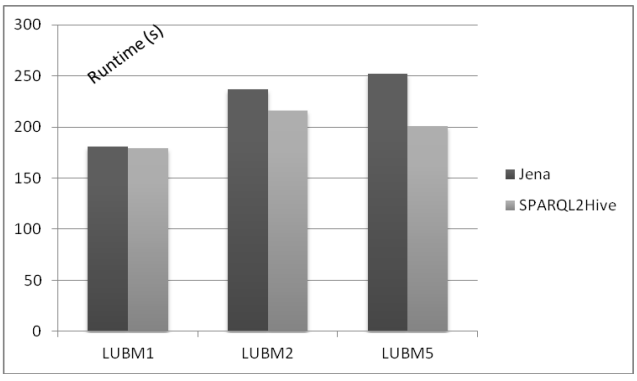

Fig. 14. LUBM Q8 Runtime.

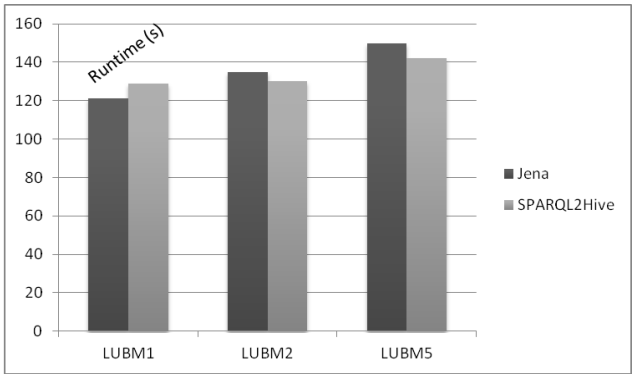

Fig. 15. LUBM Q9 Runtime.

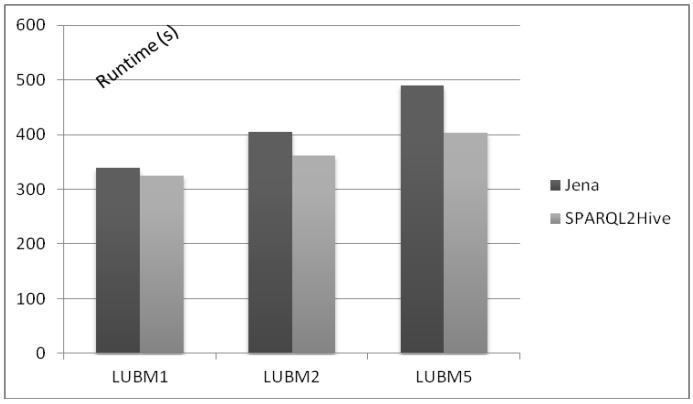

Fig. 16. LUBM Q14 Runtime.

VI. Conclusion and Future work

The explosion of RDF data offers a new challenge for researchers to manage these large volumes of RDF data, searches are oriented towards Big Data storage systems likes HBase, and for management we find query systems like Hive. In this work, we presented SPARQL2Hive a system based on the principle of meta-models for transforming a SPARQL query into a HiveQL program. in our future work we are going to work on RDF data management in real time, we combine the solution to present in this work with a streaming system like Spark streaming and Storm.

References

[1] Schmidt, M. (2018). Fondations of SPARQL query optimization. PhD Thesis, Albert-Ludwigs-Universität Freiburg (Germany).

[2] Chebotko, A., Lu, S., Jamil, H. M. et Fotouhi, F. (2006). Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns. Technical Report TR-DB-052006-CLJF.

[3] V. Khadilkar, M. Kantarcioglu, B. Thuraisingham, et P. Castagna, « Jena-HBase: A Distributed, Scalable and Efficient RDF Triple Store », p. 4

[4] Azqueta-Alzúaz, Ainhoa, et al. "Massive data load on distributed database systems over HBase." Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE Press, 2017.

[5] Schätzle, Alexander, Martin Przyjaciel-Zablocki, and Georg Lausen. "PigSPARQL: Mapping SPARQL to pig latin." Proceedings of the International Workshop on Semantic Web Information Management. ACM.

[6] Fuad, Ammar, Alva Erwin, and Heru Purnomo Ipung. "Processing performance on Apache Pig, Apache Hive and MySQL cluster." Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014. IEEE.

[7] I. Kurtev, « State of the Art of QVT: A Model Transformation Language Standard », in Applications of Graph Transformations with Industrial Relevance, vol. 5088, A. Schürr, M. Nagl, et A. Zündorf, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, p. 377-393.

[8] Banane, M., Belangour, A., & Labriji, E. H. (2018). RDF Data Management Systems Based on NoSQL Databases : A Comparative Study, 58(2), 98–102

[9] Banane M., Belangour A., El Houssine L. (2019) Storing RDF Data into Big Data NoSQL Databases. In: Mizera-Pietraszko J., Pichappan P., Mohamed L. (eds) Lecture Notes in Real-Time Intelligent Systems. RTIS 2017. Advances in Intelligent Systems and Computing, vol 756. Springer, Cham.

[10] Banane M., Belangour A. (2019) A Survey on RDF Data Store Based on NoSQL Systems for the Semantic Web Applications. In: Ezziyyani M. (eds) Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). AI2SD 2018. Advances in Intelligent Systems and Computing, vol 915. Springer, Cham

[11] Galárraga L., Hose K., Schenkel R.: Partout: A distributed engine for efficient RDF processing. Proceedings of the companion publication of the 23rd international confer- ence on World Wide Web companion, International World Wide Web Conferences Steering Committee, 2014, p. 267÷268

[12] Da Silva, Alberto Rodrigues. "Model-driven engineering: A survey supported by the unified conceptual model." Computer Languages, Systems & Structures 43 (2015): 139-155.

[13] Boussaïd, Ilhem, Patrick Siarry, and Mohamed Ahmed-Nacer. "A survey on search-based model-driven engineering." Automated Software Engineering 24.2 (2017): 233-294.

[14] ATLAS group, LINA & INRIA: ATL: Atlas Transformation Language, User Manual, Nantes, January 2005,

[15] F. Jouault et I. Kurtev, « Transforming Models with ATL », in Satellite Events at the MoDELS 2005 Conference, vol. 3844, J.-M. Bruel, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 128-138.

[16] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. J. Web Sem., 3(2-3):158–182, 2005

[17] Du, Dayong. Apache Hive Essentials. Packt Publishing Ltd, 2015.