NOSQL Ontology Storage for Indonesian Regional Folk Songs

Mohammad Nazir Arifin[1], Fauzan Prasetyo Eka Putra[2], Riyanarto Sarno[3], Nurul Fajrin Ariyani[4], Leonard Peter Gelu[5]

Department of Informatics Universitas Madura[1,2], Department of Informatics Institut Teknologi Sepuluh Nopember[3,4], Universitas Negeri Timor[5]

nazir@unira.ac.id[1], petergelu0803@gmail.com[5]

ABSTRACT

Big Data has a base of storage that is NOSQL database.All types of data can be stored properly using this database even if the data is unstructured or semi-structured.Using NOSQL database to store ontology can take advantage of Big Data storage. This give benefits in storing Indonesian folk songs storage that usually vary and have sparse / incomplete data. This paper describes an approach to store data of Indonesia folk songs in NOSQL based database. We chose MongoDB database in this paperand present the steps we use in storing, collecting, analyzing and retrieving ontologies on the MongoDB as NOSQL database.

Keywords: Ontology, NOSQ, MongoDB, Indonesian Folk Song, Ontology Storage

1. Introduction.

The large amount of digital data that increases lately makes the management of semantic data should be done better.Presently, technology that supporting the semantic information storage is not mature and database is a better choice of the mass data storage and management [1].According to Kozaki in [2], there are three manners in combining ontologies and Big Data: by analyzing ontologies to databases, providing additional metadata to data using vocabulary defined in ontologies, or converting databases into ontology databases. This paper takes the first topic of how to transform the OWL ontology into the NOSQL database.

Data redundancy and storing of irrelevant are major problems in collecting data from multiple resources [3]. In this case, there is a large amount of data Indonesian folk songs stored over the Internet and most if not all these data are not supplied and managed properly. Managing and collecting data of Indonesian folk song, have some challengessuch as the vary of sources (usually the data is incomplete, contain partial information), and all Indonesian folk songs use their own language that increase the complexity in managing data.

Managing Indonesian folk song in NOSQL database model is not only to get a new way of storing data nor exclusively a new way of querying data, but rather a mixed approach between existing data whilst supporting new ways of storing data and allowing the utilization of the semantics of that data to improve the quality of search results such as the completeness and accuracy of data. The folks song data will be stored andpresented as knowledge that contains relationship, set of rules and concepts for an automated reasoning [4].

The rest of the paper is organized as follows. A semantic web and its stack, NOSQL databaseare presented in Section II. In Section III depicts Ontology storage based on NOSQL database. Section IV gives an audit of relevance and applications of Ontology Database. Conclusion are given in Section V.

Literatur Review

A. *Semantic Web*

A Semantic web is a new design of Web Application constructed by the internet consortium (W3C) for providing important and experimentally right data to the user [3]. It facilitates machine to intelligently match data with related data based on meanings. The Semantic Web can be described as a web of documents linked in such a way so that data becomes readable and understandable to machine in a meaningful way [5]. Semantic web uses XML to define a custom label format and us the RDF to express the data [6]. RDF can only provide some semantics, therefore the OWL (Web Ontology Language) is proposed as the latest W3C recommendation. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans [1].

B. Semantic Web Stack

The Semantic Web Stack architecture is presented in Fig.1. By the inclusion of Semantic Content in the web pages the users are able to convert the current web, which includes structured, unstructured and semi-structured documents, into a web of data [3]. The hierarchy of its layers are explained below:

The bottom layers contain technology that are well known from hypertext web as basis for the semantic web. URI is unique identifier of semantic web resources. XML is a markup language that enables creation of documents composed of structured data. Semantic web gives meaning (semantics) to structured data. XML Namespace provides a way to use other markups from more sources.

The next layer is a layer standardized by the W3C and contains technologies that can be used to create web semantic applications. RDF is framework for creating statements in a form of triples*subject-predicate-object* that enables to represent information about resources in the form of graph. RDF Schema (RDFS) provides basic vocabulary for RDF. By using RDFS it is possible to create hierarchies of classes and properties.

The next layer is Processing and Querying layer and it is known as the central Layer. It is combination of OWL, Rule Inter Change Format (RIF), Taxonomies and Simple Protocol and RDF Query Language (SPARQL) components. OWL extends RDFS by adding more advanced constructs to describe semantics of RDF statements. SPARQL is a RDF query language for RDF-based data (including RDFS and OWL). RIF is a rule interchange format to allow describing relation that cannot be directly described using description logic used in OWL.

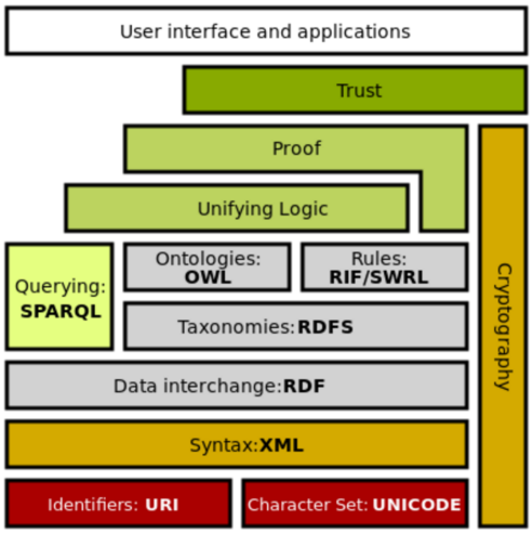


Figure. 1 Semantic Web Stack[3]

The next layer is a Security layer that embeds all the other layers and ensures the source of the semantic web statements as trusted source. This layer uses various cryptographic techniques, including digital signature. Trust can be achieved by verifying authenticity and relying on formal logic during deriving new information.

The last layer (most top layer) is the user interface and application layer. It allows end user to retrieve the authenticated and trusted information for their semantic web application.

C. Ontology Databases

Because of their efficiency in storing and retrieving data, relational database are still used for data storage. Unfortunately, this approach lack of semantics in information, and incompleteness of data [7]. This is due to the loss of most semantic data as well as information during the process of transforming conceptual models to logical models.Although there is currently no proper approach as a guidance in managing and retrieving data and knowledge, there are a number of approaches used to find the relationship between database and ontology[7]. They are as follows:

1. Ontology based databases (OBDB).
2. Database based ontology (DBBO).

In general, DBBO will use an approach to generate ontologies derived from the database by creating a relationship between an ontology with a database schema(e.g. [2]).While on the other side, OBDB in some way is trying to save the example of ontology into the database structure. There were some researches that was done using some tools or APIs that provide methods in storing ontologies into a database structure. Some examples of such tools are Minerva, Sesame, and Jena[7].

D. NOSQL Databases

NOSQL stands for "Not Only SQL" which the database does not use relation, distributed, open-source and can be scaled horizontally[2]. The most fundamental reason of using NOSQL databases in this paper is because they can handle structured and semi-structured data. In [8] and[9], there are four types of NOSQL databases which are key/value, column, and document stores and the last type is graph databases.

Then among the four NOSQL databases, document-oriented databasesare assumed to be the best in ontology learning process for several reasons including flexibility and handling huge amounts of data. In addition, they can store all types of data such as structured, semi-structured and unstructured data that can reduce the complexity of database schemes. This allow represent data from the simple one to complex relation data in a single document properly.

There are so many database that is oriented on document and MongoDB is the most popular one. This database has a very high performance and efficiency. MongoDB uses BSON documents with dynamic schemes to provide simplicity, ease and speed in the integration of application data. MongoDB is particularly suitable for application such CMS, archiving and real-time analysis.

Furthermore, MongoDB can provide dynamic queries with rich document structures. Collections are analogized as tables in a relational database. This collection contains documents that can be nested within a complex hierarchy but still easy to query and index. A document is a set of fields in which a key value pairs. The key is a string where as its associated value may be a basic type, document, or array of value. Each data structure can be managed by MongoDB without the burden even if there are frequent changes to the data.The MongoDB data model can be summarized with following figure.

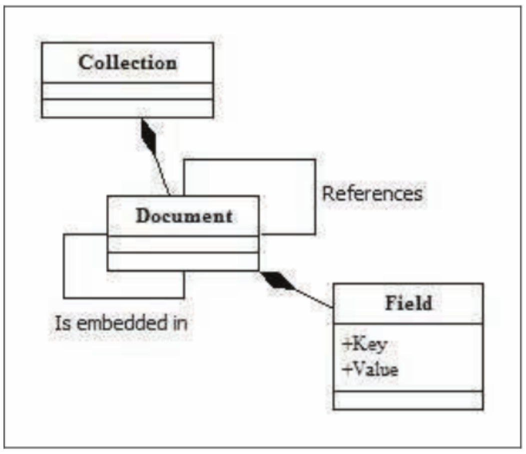


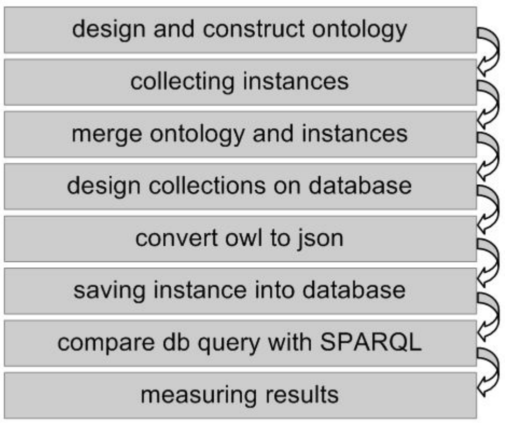
Figure. 2 MongoDB Data Model [2]

2. NOSQL ONTOLOGY DATA STORAGE FOR INDONESIAN FOLK SONG

The proposed model is explained in the Fig. 3. It consists of the following eight blocks/functions: namely, 1. Design and construct ontology, 2. Collectiong instances of Indonesian folk song, 3. Merge ontology and collected instances, 4. Design database collection on NOSQL database, 5. Convert OWL to JSON, 6. Saving structure and instances into MongoDB, 7. Compare database query with SPARQL and 8. Measuring results of research. The different functions of each block are explained in the following sub-sections.

A. Design and construct Ontology

This paper only coversthe ontology is created with Protege, an ontology editor and saved in OWL format. We create one class named IndonesiaFolkSong with several object properties and data properties. IndonesianFolkSong class is subclass of Musical Item class from Music Ontology. We considered some aspect that must be included in an instance of IndonesiaFolkSong, they are the creator/author of folksong, the language that is used by the song, the origin of song based on regional name, the audio file on internet, the title of song and the lyrics of song if any. Object properties that are needed in this workare: 1. creator, 2. file, 3. lyrics, 4. origin, 5. singer, and 6. wikipedia. Data properties that we use are: languange, title from Dublin Core, name from Foaf, regional name, songurl and text from music ontology.



design and construct ontology
collecting instances
merge ontology and instances
design collections on database
convert owl to json
saving instance into database
compare db query with SPARQL
measuring results

Figure. 3 Proposed Model

Object property creator will link IndonesiaFolkSong class (as domain) with ontology from foaf (friend of a friend) “Person” (as range). Based on foaf, thePerson class has firstName, lastName and name attributes.By using resources such as Dublin Core, Geonames, Music Ontology and Foaf, Indonesian Folk Song ontology become more linked with existing resource or ontologies.

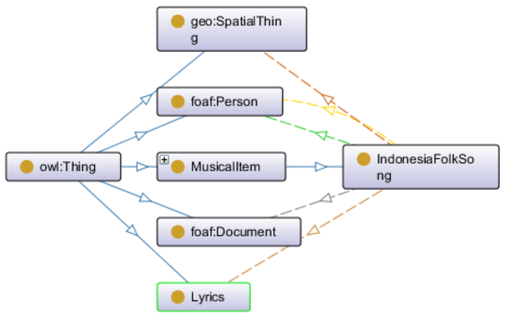


Figure. 4 Classes on Indonesian Folk Song Ontology

B. *Collecting instances and Merge to Ontology*

One of the challenge in creating Indonesian folk song is collecting data (Indonesian folk song). Indonesia is a country which has many cultures and languages. In this paper we will only use 100 data of Indonesia folk song with vary in language, origin, author and sources. Several items use content from Youtube and wikipedia page. Songs’ lyrics usually get from web log content.

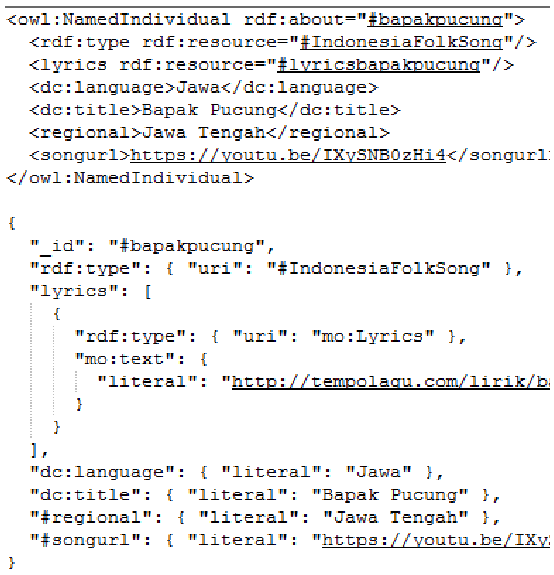
Mergin/inserting instances to the already created ontology is notoriously process because each data must be inserted one by one.We use three type of instances, there are: person for the creator of folk song, lyric, and song. Each instance of song will link their creator (if any) to the person instance and also link their lyrics to their lyrics instance. The reason for this is Person and Lyrics are different class of IndonesiaFolkSong and also to prevent redundancy of data since some songs may refers the same Person for their creator.

C. *Design collections in database*

The following steps are related to query in database instead of using SPARQL. As NOSQL database is not using table or general scheme (MongoDB using collection instead) so we have to design the pattern of ontology that will be stored in database. We use three collections in our database, there are: namespace, person and song. Lyrics collection as reflection of lyrics instance is not created because we embed the lyric directly into the song documents as we believe each song must be have one lyrics.

D. *Convert OWL to JSON*

All instances with their relationship in OWL will be converted to Javascript Object Notation (JSON) in order to be saved into MongoDB database. Figure 5 shows the change form of OWL instance into JSON.



```
<owl:NamedIndividual rdf:about="#bapakpucung">
  <rdf:type rdf:resource="#IndonesiaFolkSong"/>
  <lyrics rdf:resource="#lyricsbapakpucung"/>
  <dc:language>Jawa</dc:language>
  <dc:title>Bapak Pucung</dc:title>
  <regional>Jawa Tengah</regional>
  <songurl>https://youtu.be/IXySNB0zHi4</songurl
</owl:NamedIndividual>

{
  "_id": "#bapakpucung",
  "rdf:type": { "uri": "#IndonesiaFolkSong" },
  "lyrics": [
    {
      "rdf:type": { "uri": "mo:Lyrics" },
      "mo:text": {
        "literal": "http://tempolagu.com/lirik/b
      }
    }
  ],
  "dc:language": { "literal": "Jawa" },
  "dc:title": { "literal": "Bapak Pucung" },
  "#regional": { "literal": "Jawa Tengah" },
  "#songurl": { "literal": "https://youtu.be/IXy
}
```

Figure. 5 Convertion RDF to JSON

E. *Saving instance into database*

We create a web based tool in PHP to

automate the process of parsing and extracting instances and their relations from OWL (XML) into JSON byusing EasyRDF library. The tool will save each instance in JSON form into MongoDB database. In this paper, we use three collection in MongoDB named "ns" to store namespace / prefixes, "song" to store all folksong instances and the last is "person" to store all author instances.

There are additional prefixes that we use in OWL file and then used in query process is as follows:

TABLE I. PREFIXES

Prefix	Url
foaf	http://xmlns.com/foaf/0.1/
dc	http://purl.org/dc/elements/1.1/
mo	http://purl.org/ontology/mo/
ifs	http://www.semanticweb.org/indonesia-folk-song#

F. Compare Query Result

After saving instances into database, matching and compare of query result between SPARQL query and MongoDB database query is conducted. The important part is all query result (SPARQL and MongoDB query) must satisfy the required data. In this paper, query execution time is not explicitly measured as we only introduce prototype model in using NoSQL database for ontology. Apache Jena Fuseki 2.6.0 is installed and used as SPARQL query endpoint and MongoDB query will be executed directly by Windows using shell commands.

3. Discussion.

Experiments of evalution were carried out on computerwith Protege 5.1.0. To create database collections, MongoDB 3.4.2 is installed. The data set to evaluate our approach consists with 6 classes, 5 object properties, 7 data properties and 100 individuals. The main reason to select this data set is to evaluate our approach, the dataset should contain unstructured or semi-structured data and all candidate to be stored in the ontology and in database as well.

Here use of queries to extract instance of folksong in NOSQL based database. These queries cover some aspects to show how data can be extracted such as using regular expression, get single and multiple result, and query using relation from other collection. The query is as follows:

Example 1 Find song title "ampar-ampar pisang"

db.song.findOne({ "dc:title": { "literal": "Ampar-Ampar Pisang" } });

Result:

Instance ID
"_id": "ifs:amparamparpisang"

Above is a simple query, which can be used to extract an instance of song that has literal title "Ampar-Ampar Pisang". By storing instance into MongoDB as ontology storage, we practically can get complete semantic relationship on the resulted data as example in Figure 5. If there are some instances missing one or more of property, we still can show the result because of NOSQL storage that can store semi or unstructured data in same document.

SELECT ?song
WHERE { ?song dc:title "Ampar-Ampar Pisang" }

When using SPARQL query above, we can get the same id result but we missed the complete semantic relationship since in Fuseki the result only show link of instance.

Example 2 Find all song from Bali

db.song.find({ "ifs:regional": { "literal": "Bali" } });

Result:

Instance ID
"_id": "ifs:janger"
"_id":"ifs:meyongmeyong"
"_id":"ifs:goakmaling"
"_id":"ifs:dondapdape"
"_id":"ifs:dadongdauh"
"_id":"ifs:jurupencar"

“_id”:”ifs:putriceningayu”
“_id”:”ifs:bibirangda”

We try to find more than one instance of songs that has regional / from Bali. We can expect that we still get complete semantic relationship from the results as each row of result is complete instance of song. From the query above we getseven instance of song from Bali and each instance has their own semantic properties (title, languange, url, etc).

SELECT ?song
WHERE { ?song ifs:regional "Bali" }

SPARQL query above has the same result with the query but same with first example, the result only shows the instance id and missed the complete semantic relationship.

Example 3 Find all song that has URL from Youtube

db.song.find({ “ifs:songurl”: { “literal”: /youtube/i } });

Query above facilitates regular expression in MongoDB to find instances that has url on youtube.com. As we collect folksong data mostly from youtube so the result almost get all of song instance in database.

SELECT ?song
WHERE {
?x ifs:songurl ?song FILTER regex(?song, "youtube", "i")
}

SPARQL also supports query using regular expression so the query above can capture same result as MongoDB query does.

Example 4 Find all song that has creator “Yusuf Alamudi”

db.song.find({ “ifs:creator”: { “uri”: “ifs:yusufalamudi” } });

Result:

Instance ID
“_id”: “ifs:alosiripolodua”

This query will try to link song instance with their author since an instance of song only contains the URI of author. Author instances is in the same OWL file and saved in author collection. As MongoDB is non relational database so we can’t use join and use an iterator instead.

SELECT ?song
WHERE { ?song ifs:creator ifs:yusufalamudi }

SPARQL query above is as simple as MongoDB query and the result is the same.

4. Conclusion.

In this paper, we have tried to offer an alternative approach that uses NOSQL databases to store ontology data as an effort to improve the process of searching and data or knowledge management. With this approach we were able to optimize the query and data storage of Indonesian folk songs more efficiently. We developed our approach using a prototype as well as tool for evaluating the tests against the proposed approach. We have compared the results with conventional approaches that perform queries and knowledge using OWL / RDF files.

There are still many points of view for future work in this field. The following are some possibilities: 1. Validate the approach with larger data samples. Further experiments should validate this approach with large amounts of data from existing Indonesian folk songs. 2. Change SPARQL to MongoDB or vice versa. By using a data query in accordance with the desired language it will be a better step forward.

We assume that our approach has been able to achieve its objectives as a system and can provide a framework to optimize storage and query mechanisms from ontology or knowledge data. Where practicable, we strongly recommend using our approach to optimize data management and knowledge not only for local folk songs in Indonesia.

Bibliographies

[1] A. Fatima, C. Luca, and G. Wilson,

"New Framework for Semantic Search Engine," in 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 446–451.

[2] A. S. Hettiarachchi, J. Goonatillake, G. Wikramanayake, and A. Walisadeera, "Hybrid Approach for Optimal Management and Querying of Data and Knowledge," in International Conference on Advances in ICT for Emerging Regions (ICTer), 2016, pp. 176–185.

[3] H. Abbes, S. Boukettaya, and F. Gargouri, "Learning ontology from Big Data through MongoDB database," Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA, vol. 2016–July, 2016.

[4] H. A. Hartanto, R. Sarno, M. Nurul, and F. Ariyani, "Linked Warning Criterion on Ontology-Based Key Performance Indicators," pp. 211–216, 2016.

[5] M. S. P. Babu and S. Govathoti, "A Semantic model for building integrated Ontology databases," in 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 595–598.

[6] S. Wang and X. Zhang, "A high-efficiency ontology storage and query method based on relational database," 2011 Int. Conf. Electr. Control Eng. ICECE 2011 - Proc., no. 102102210409, pp. 4253–4256, 2011.

[7] Z. Zhou and Y. Xing, "A study on ontology storage based on relational database," Conf. Anthol. IEEE, pp. 1–5, 2013.