**Loadstar Project – Report**

*Ahmet Melek – Bogazici University*

1. **Introduction & Project Definition**
2. **Compliances and Requirements**
3. **Features**
   1. Table Forming
   2. Data Pulling
   3. Table Querying Tools
   4. Metadata of Tables
   5. Link Previews
4. **Semantic Issues and Discussion**
5. **Appendix**
6. **Introduction & Project Definition**

**Loadstar** is a platform in which users can create **tables** about any kind of concept. [[1]](#footnote-1)

These **tables** could be formed to describe the **concept**, taking notes on a concept, making lists etc.

In the backend, table data is stored in a **Triplestore** database. A triplestore differs from a relational database (such as mySQL) in the sense that it stores **triples** instead of tables with rows. Loadstar visualizes the triple data in tables.

* In a **table**, the **subject** is constant,meaning, all **triples** contain the same subject. Subjects are also optional, if a subject is not provided by the user, it is a blank node.
* The **context** of each and every triple in the table is the table URI: loadstar.com/username/table\_name

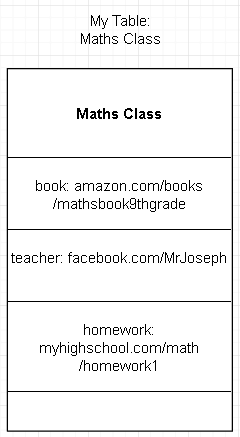


Figure 1: A table could be such as this.

(the subject is defined by user, however, not shown on the table.)

Every row in the table represents a triple.

In this case, the table is named `Maths Class`. In the table, there are three triples:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/book First Triple

o = amazon.com/books/mathsbook9thgrade

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/teacher Second Triple

o = facebook.com/MrJoseph

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/homework Third Triple

o = myhighschool.com/math/homework1

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The subject of the table provides us with the knowledge of ‘Which topic is this table about?’. This is useful when making SPARQL queries, analyses etc.

loadstar.com/ontology is the root path of all ontology elements of Loadstar.

Loadstar also has moderators, which create encyclopedic tables to accumulate knowledge on the platform. With tools that Loadstar provides, data could be pulled automatically from various resources such as DBpedia or Wikidata, in order to form tables in Loadstar.

In the Loadstar platform, there are user profiles.

“Loadstar Platform” has a user profile of its own, located in [www.loadstar.com/users/Loadstar](http://www.loadstar.com/users/Loadstar) .

User profile of Loadstar contains the encyclopedic tables that are created by the moderators.

* When a keyword is searched on Loadstar Platform, tables that are generated by the users will appear based on the popularity and the rating of the tables.(this feature is not implemented yet)

This way Loadstar provides an information serving ecosystem with both manually managed content, created by mods; and natural content created by users.

The mission of Loadstar Platform is to guide people on an endless journey of learning, with an environment that favors diversity of opinions.

The main feature of the platform which is designed to attract users in the initial stage is that, platform is an easy-to-use table forming and table finding tool.

1. **Compliances and Technical Requirements**
2. Project uses Python. RDFlib is used as a base library to work on RDF data.
3. SPARQL technology is utilized via RDFlib’s query module to prepare query tools for users.
4. The APIs used to obtain data are such as:

* Dbpedia to pull table data.
* Wikidata to pull table data.
* linkpreview.net to form link preview boxes.

1. External Programs that are used while developing the project are:

* Protégé to form a table metadata ontology.

1. **Features**

**Careful**: Please change the paths.py file according to your file system firstly. It provides a default path for the database location on your file system. It also provides a default user URI, a default tablename, etc.

**Note:** In all functions, when there is a triple adding or triple deletion operation is performed, the parameter that represents the data (triples) is always in the form of **doubles, instead of triples.** This is due to subjects being constant in a Loadstar table. Subject is represented with another parameter,such as table\_subject\_URI.

## Table Forming

Table Creation Steps:

* Create database via create\_db function.(If you did not create it already.)
  + We represent our whole database with a Dataset object when performing operations with Python.
  + This Dataset object aggregates all graphs in our database.
  + In create\_db , we create a Dataset object, to be stored with the database Sleepycat.
  + We perform some checkings in order to not write on an existing database, in case of a database previously existing on the path. We also check for corrupted stores, which typically turn out to be a normal store, but just another type of store, such as SPARQLStore.
  + If there are no problems with the checkings, we proceed and commit our database into the file system of the computer.
* Create a table, with create\_table function.
  + The parameter “double data” is to add triples to the store in the creation process. It is called “double\_data” because a table’s subject is always constant. The type of double data is a list. List consists of **n** number of doubles. Each double consists of a predicate and an object.
  + The subject of the table is to be defined with the “table\_subject\_URI” parameter.
  + The subject, predicates and objects must be of type URI. Otherwise serializations may break.
  + If you have multiple databases for any reason, you can assign the parameter database\_path with an y path you desire. Its default value is defined on paths.py.
* Your table has been created.

To View your table:

* Use print\_graph function. If you did not use the default path, default username, default tablename values, please assign them accordingly, while calling the function.

Some other functions on table CRUD operations :

* open\_db : A function that is used by other functions. Opens an existing database to perform operations.
* table\_add\_tri: To add a list of triples into the table.
* open context: A function that is used by other functions. It firstly opens a database, using open\_db function, then it returns a graph in the database, for which you have assigned values in the parameters.
* / load\_resource: A function used by other functions. Loads a resource on a graph. (ie. A Dbpedia page) The graph is stored in IOMemory and is not persistent. Function returns the graph to be used by other functions.
* / parse\_to\_source\_URI: Used by other functions in order to parse a word such as “dbpedia” into the valid URL such as <http://dbpedia.org/resource>.

## Data Pulling Tools

Autopulling:

* Autopulling is provided by: autopull function.
  + Username, table\_name, database\_path parameters are to find the table to store the pulled data.
  + Source and topic parameters are to find the source page to pull the data from.
  + Function forms a blank graph to be stored in the IOMemory, pulls the data on the graph, then adds every triple in the IOMemory graph to the persistent graph, which is mentioned by username, table\_name and database\_path parameters.
* load\_resource function is used normally by autopull function. It could also be used to preview tables in the other sources.

## Metadata Serving of Tables

A metadata table is a table providing information on tables, such as creation date and creator.

Metadata functions are defined on the file metadataFuncs.py . This is due to the reason that some functions needed to be overwritten, however, since the functions are not defined class-based, they could be overwritten. As a solution, those functions are redefined for metadata table forming, in the file metadataFuncs.py .

First reason they needed to be redefined was that they needed to contain literal values. Those values are not supported by the functions defined on functionsTest.py .

Second reason is thatnormally, a table is formed via the URL configuration:

<http://loadstar.com/username/tablename>

In the metadata functions however, this configuration is designed to be:

<http://loadstar.com/username/tablename/meta>

This difference in the configuration could not be implemented using the same functions on functionsTest.py . Therefore, configuration is redefined for metadata, in metadataFuncs.py .

Creating a Metadata Table:

A metadata table can only be created for an existing table on the database.

* Use autoform\_metadata functions with the relevant table’s creator’s username and name of the relevant table as parameters. Also specify a datetime for the creation date. If this parameter is left blank, datetime.datetime.now() is used to obtain the current time and assign it as the creation date of the table.
* The function is not a dynamic one in the sense that it always puts two rows of data into the metatable. Those rows are the creation date and the creator. If another type of data is desired to be included in the metatable, that data need to be hardcoded into the function.

## Table Query Tools

There is a predefined SPARQL query in the function sparqlCommonObj function. This query is to compare two tables to see if they have any common objects within the triples they have. Function finds the relevant tables with usernames of first and second table, and the names of first and second table.

This functionality is to offer users an easy way to compare two tables, so they do not need to form a query by themselves.

Other queries could be defined to offer further functionality.

## Link Previews

Link previews are to offer users a better experience while viewing the tables.

Idea is, metadata of every object URI in table rows are to be scraped using linkpreview.net API. Then this metadata is to be used to form link previews.

API is RESTful, meaning, it is possible to communicate with it via HTTP requests and responses. A get request such as this:

http://api.linkpreview.net/?key=””””mykeyhere”””&q=http://ahmetmelek.com

could be sent in order to receive a response such as:

{

"title":"Ahmet Melek",

"description":"I am interested in Machine Learning and Brain-Computer Interfaces. Check below for more.",

"image":"",

"url":"http:\/\/ahmetmelek.com\/"

}

This data in JSON then could be used in frontend to form link previews.

**4. Semantic Issues and Discussion**

**4.1 Table Names and Subjects – Contexts in Data Model**

Tables are hosted in URIs that are formed with usernames and tablenames:

loadstar.com/username/tablename

In Dbpedia, the name of the table is also the subject of each and every triple in the table. The table is hosted in dbpedia.org/resource/SubjectName. While continuing to read below, please keep in mind that both in Dbpedia and Loadstar, subjects are constant in a table. (There is no information obtained about the backend of Dbpedia, however, at least data is presented with constant subject tables.) Every triple in a table has the same subject. This constant subject will be mentioned as “**the subject”** below in order to make the situation easier to understand.

Since Dbpedia is an encyclopedic data source, this constitutes no trouble. The table itself, and **the subject** may safely be the same **thing**, and be represented by the same URI.

However, in the case of Loadstar platform, there are Tables that are formed by users instead of moderators. This brings the issue of “How to mention about a table?” . A user may not agree with a table and may want to comment about the table, referencing its link. Aim of user is to comment on the table itself. Not to comment on the **concept** which table describes.

Given that a person wants to make a comment on a Dbpedia table, it is not possible for the user to distinguish between the table and the concept that the table describes. Let’s say the table is about Mathematics. When the user makes a comment using URI of the table[[2]](#footnote-2), it is not possible to tell if the user is commenting on the table or if the user is expressing an opinion on the concept “Mathematics”.

That is why Loadstar utilizes contexts. In the platform, tables are represented by the context URIs and **the subject** is represented by a separate URI. A pattern for triples (actually quads) that exist in Loadstar databases would be:

< subject predicate object context>.

Subject is optional and may be blank. It is to tell what concept this table is about.

Predicates and objects are to form rows that are somehow relevant to the subject.These rows are also related to each other in the sense that they all have the same subject and context.

Sometimes the relevance between subject and rows could be obvious. However there could be times that relevance is not obvious. A user could form a table about their maths class, and may put “dbpedia.org/resource/Mathematics” as table subject. A maths class is not the same thing as the concept Mathematics, however they are somehow relevant in table creator’s mind. The concept Mathematics has a conception on table creator’s mind, and that conception is his/her maths class.

Asking the question of:

“Which URI do you consider to put as your table subject, on your table that is named ‘Maths Class’? ”

could be thought as the same of asking this question:

“What is the **entity (subject)** that comes to your mind first when you think about your table ‘Maths Class’? ”

and that **entity** should exist in Dbpedia or another RDF data API.

If subject is left as blank, (meaning the subject of the table is a blank node) the relation of one row with another row is that they share the same context. (same table name) However, no further inferences can be made on the relation, since there is no entity (subject) defined to tell us about the relation of one row with another.

The context represents the table (and the location of it on the Web). Context URI is derived as: loadstar.com/username/tablename

* 1. **Fixing the Number of Subjects in a Table to One – Concepts and Conceptions**

In every table, the number of subjects is fixed to one, as mentioned in the previous chapter. There is a reasoning on this. That reasoning and advanteges-disadvanteges of the situation will be discussed in this chapter, after talking on **“concepts”** and **“conceptions”** on Loadstar.

Loadstar is founded on the idea of representing different ideas on a topic semantically. These ideas may be called as **conceptions**, and the topic which the ideas are upon, may be called **concept**. Concepts are objective, and conceptions are subjective. An example could be the Dbpedia page on Mathematics. It is a **concept**. It represents information that could be identified as objective. However a table formed in Loadstar by users is a subjective set of statements. That is why they are called **conceptions.** In the below box, this issue is further explained.

The philosophy behind Loadstar’s **concept** and **conception** denotation can be explained as:

In a commucation between two or more individuals; there may exist a particular opinion or a set of opinions that are shared and agreed among all of those individuals. If this is the case, individuals will name those opinions as being objective truths.

Imagine two people arguing each other on the color of a dress. Individual A is claiming the dress to be the color of white and yellow, and individual B is claiming the dress to be the color of blue and black. It might be seen as those individuals are having an argument, however, they also have an agreement. They both agree that there is a physical entity there, and that entity is used as a wearable by humans. That is why those individuals have an argument on the color of a ‘dress’ at first place. They firstly agree upon a set of ideas, which can be summed up as “There is a dress existing.”, and then, they have disagreements on another set of opinions, which can be summed up as “The color of the dress is…”

In the above case, there is an objectivity that is substantial to those individuals. This objectivity can be called as **a recognized** **concept** by that group of individuals. That **concept** exists for the group, and all individuals in the group agree upon the basic characteristics of that **concept**. The **concept** is **objective**. In the above example, that **concept** is the word “dress”. The word holds certain assumptions in itself and individuals agree on those assumptions.

The set of opinions which are based upon **concepts** and not agreed among different individuals may be named as **conceptions** on that particular **concept**. In the above example, conceptions are the ideas on the color of the dress and they are based upon the concept dress. (Check Appendix 2 for a second example)

After emphasizing this mindset, it is possible to clarify the reason Loadstar implements such logic :

Platform’s aim is to provide people with easily consumable information on concepts. Principle value here is the easy consumption. We are living in an era where all products and services are produced in an easily consumable manner, to attract customers. It is impossible for an educational or information based platform to survive otherwise. Education should be upgraded into forms that it is compatible with the digital age we are living in.

The first semantic characteristic is to keep pace with the expressionism that digital age brings with it, such as every person in Twitter being able to comment individually on Donald Trump’s tweets. **When the identities of the tables are seperated from the topic that the table is about, it is possible to make expressions on the tables. Loadstar provides this functionality with context-subject seperation.**

The second semantic characteristic is to keep pace with the fast-consuming environment. The market for information is getting more and more competitive, and a platform where information-consuming is not fast is only to survive with the cost of having appeal for a small community. In order for platforms to attract consumers, information must be easily consumable. **When the tables are about only one topic, it will be much more easier for consumers to understand and to internalize the information.**

* 1. **Other Semantics Characteristics**

Loadstar has certain semantic characteristics that it implements with taking inspiration from other semantically formed platforms such as Dbpedia. Loadstar also makes a number of modifications on those characteristics, to string along with its purpose.These semantic characteristics can be listed as:

* + Serving tables with Context URIs
  + Serving tables which have only one Subject
  + Serving the information components with their own URIs
  + Serving the components with predicate relations
  + Providing automatic tools to pull RDF data from other data sources
  + Providing easy SPARQL querying

**5. Appendix**

# Ontology

Here is the ontology to form Metatables:

@prefix : <http://www.loadstar.com/ontology#> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix xml: <http://www.w3.org/XML/1998/namespace> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@base <http://www.loadstar.com/ontology> .

<http://www.loadstar.com/ontology> rdf:type owl:Ontology .

**Object Properties**

### http://loadstar.com/ontology#created\_by

<http://loadstar.com/ontology#created\_by> rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf owl:topObjectProperty ;

rdfs:domain <http://loadstar.com/ontology#Table> ;

rdfs:range <http://loadstar.com/ontology#Agent> .

**Data Properties**

### http://loadstar.com/ontology#creation\_time

<http://loadstar.com/ontology#creation\_time> rdf:type owl:DatatypeProperty ;

rdfs:subPropertyOf owl:topDataProperty ;

rdfs:domain <http://loadstar.com/ontology#Table> ;

rdfs:range xsd:dateTime .

**Classes**

### http://loadstar.com/ontology#Agent

<http://loadstar.com/ontology#Agent> rdf:type owl:Class .

### http://loadstar.com/ontology#Table

<http://loadstar.com/ontology#Table> rdf:type owl:Class .

**Individuals**

### http://loadstar.com/ontology#Loadstar

<http://loadstar.com/ontology#Loadstar> rdf:type owl:NamedIndividual ,

<http://loadstar.com/ontology#Agent>.

1. This concept is inspired by Dbpedia. [↑](#footnote-ref-1)
2. Dbpedia.org/resource/Mathematics [↑](#footnote-ref-2)