SEMANTIC WEB - WINTER 2020
ASSIGNMENT -5

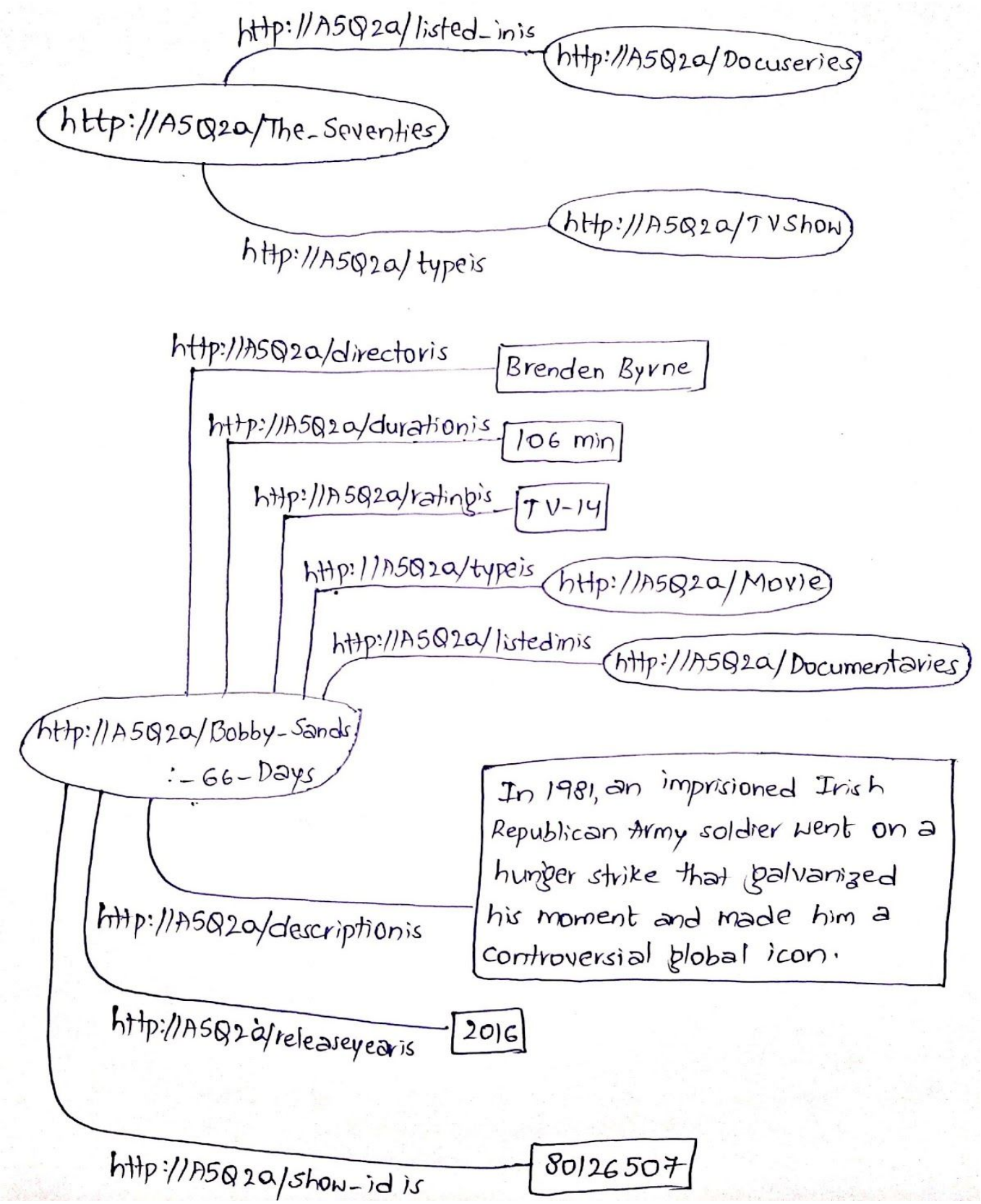
Q2-a)

The selected triples are shown below, some of the objects of triples are IRIs and the others are literals.

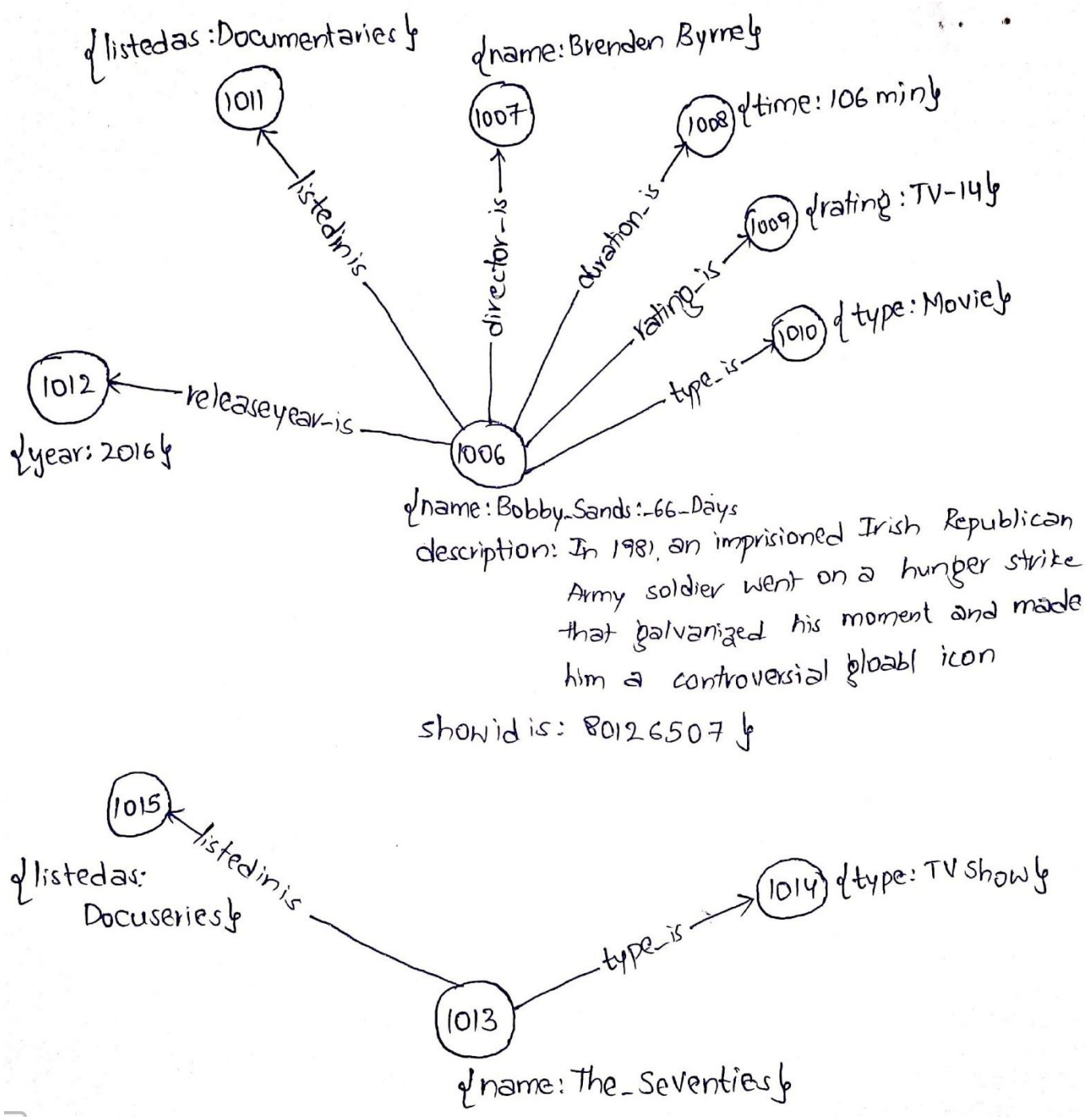
```
@prefix Netflix: <http://A5Q2a/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

Netflix:Bobby_Sands:_66_Days
        Netflix:descriptionis   "In 1981 an imprisoned Irish Republican Army
                                soldier went on a hunger strike that galvanized
                                his movement and made him a controversial global
                                icon."^^xsd:String;
        Netflix:directoris      "Brendan Byrne"^^xsd:String ;
        Netflix:durationis      "106 min" ;
        Netflix:listed_inis     Netflix:Documentaries ;
        Netflix:ratingis        "TV-14"^^xsd:String ;
        Netflix:release_yearis  "2016"^^xsd:Integer ;
        Netflix:show_idis       "80126507"^^xsd:Integer ;
        Netflix:typeis          Netflix:Movie .

Netflix:The_Seventies
        Netflix:listed_inis     Netflix:Docuseries ;
        Netflix:typeis          Netflix:TV Show .
```

- **RDF** graphs have triples of subject, predicate and object whereas **Property graph** has nodes or vertices joined by relationships.
- Some of the objects of the triples are considered to be IRIs and others to be literals, to show the difference between both in the RDF graph.
- In Property graph, the vertices and relationships are associated with attributes which are in the form of key value pairs. Each vertex has a unique id in the property graph using which it is identified.
- In the triples I considered, there are no such properties with which I can associate some attributes. So, the attributes are associated only with vertices.

RDF Graph:-


http://A5Q2a/listed_inis
http://A5Q2a/Docuseries
http://A5Q2a/The_Seventies
http://A5Q2a/TVShow
http://A5Q2a/typeis
http://A5Q2a/directoris
Brenden Byrne
http://A5Q2a/durationis
106 min
http://A5Q2a/ratingis
TV-14
http://A5Q2a/typeis
http://A5Q2a/Movie
http://A5Q2a/listedinis
http://A5Q2a/Documentaries
http://A5Q2a/Bobby-Sands:-66-Days
In 1981, an imprisioned Irish Republican Army soldier went on a hunger strike that galvanized his moment and made him a controversial global icon.
http://A5Q2a/descriptionis
http://A5Q2a/releaseyearis
2016
http://A5Q2a/show-id is
80126507

Property Graph:-



****

Q2-b)

Triple stores are the databases which store the statements in the Resource Description framework (RDF), these statements in the RDF are in the form of Subject - Predicate - Object. Querying the triple store means defining a pattern for retrieval of the required triples. Indexing refers to a mechanism whenever we define a pattern for retrieval of triples or query the triple store, how the required triples are indexed and retrieved. Here, I am going to compare the indexing and query optimization in **AllegroGraph** and **Hexastore**.

In **AllegroGraph**, along with the subject(s), predicate(p) and object(o), the other components of the triples are 'graph(g)', 'triple-id(i)' which is unique in triple store and 'triple-attributes'. There may be multiple triples from different repositories with the same triple-id. Index name is permutation of s,p,o,g,i. The order of these letters indicates how the triples are sorted and stored. For example, if the order is '*spogi',* first the triples are sorted based on subject, if the subjects are same, then the triples are sorted on predicate, if the predicates are same then they are sorted on object, if the objects are same, they are sorted on graph, if the graphs are also same they they will be sorted based on the index which is obviously unique. So, always the index is positioned last in permutation. The seven default indices created by triple stores are '*spogi, posgi, ospgi, gspoi, gposi, gospi and i*'. The indices can be added and deleted easily. When more indices are added, it takes more time to identify the triple making indices less efficient. The efficiency of indices are given by *Oscore,* the optimal value of oscore is 1.0 and higher value indicates less efficient. AllegroGraph performs auto optimization of indices whenever the value of oscore is not as threshold.

★ How about Query Performance in **AllegroGraph**?

The query is optimal when the index starts with *sp* or *ps* as it directly leads to triples with matching subject and predicate, so always the query is optimal when the indexing starts with either *sp* or *ps*. But, if the active index does not start as required, then the query searches for an active index which either starts with *s* or *p*. For example, the active index is '*sopgi*' , the query selects first based on the subject but the further component is object and not predicate. So, the query takes all the objects and checks for the required predicate and returns the required results. But, always the number of triples checked are much more than the number of triples returned.

Hexastore tries to overcome the vertical partitioning indexing scheme where 'n' is the number of two column tables with subject and object are the columns. 'N' is the number of unique properties in the data, but this suits only for property bound queries. Property bound queries may not only be the necessity always. So, Hexastore
came up with a new indexing scheme with six different combinations of subject, predicate and object. Each of these six index structures centered around one RDF element and decides about the prioritization of other two elements. For example, consider a subject headed index, for every value of the subject, all the properties associated to are stored in the form of

list and every item of that list (every property) will be pointing to another list which contains the list of objects associated to the subject with that particular property. This scheme is followed for the indexing scheme as '*spo*' , the structure of the data storage changes with the index scheme followed providing equal importance to all types of queries.

Pros of **HexaStore** :-

- Efficient handling of multi valued resources.
- As only valid values are stored, this can avoid NULL values.
- HexaStore accesses only those items which are required by the query, hence reducing I/O cost.

Cons of **HexaStore** :-

- The main drawback of HexaStore is its five-fold storage space in its worst-case.
- Hard to handle updates and deletions in HexaStore.

REFERENCES FOR Q2(B):-

1) https://franz.com/agraph/support/documentation/current/triple-index.html
2) Weiss, Cathrin & Karras, Panagiotis & Bernstein, Abraham. (2008). Hexastore. Proceedings of the VLDB Endowment. 1. 1008-1019. 10.14778/1453856.1453965.
3) https://cs.uwaterloo.ca/~gweddell/cs848/presentations/MSabriCS848Fall2013.pdf

****

Q2-c)

As discussed in the previous question, triple store databases are the databases used to store RDF statements whose important components are subject, predicate and object. Property graph stores are the databases which are used to store highly processed and connected data. Use cases of such highly connected data include social networking and restaurant recommendation. For example, consider Facebook as a use case, If my account is considered as a vertex all my friends will be connected to my vertex and their friends will be connected to that specific vertex. Here, I have discussed the important aspects of the **HexaStore** triple store and **Amazon Neptune** property graph store along with comparison among them using pros and cons.

Hexastore tries to overcome the vertical partitioning indexing scheme where 'n' is the number of two column tables with subject and object are the columns. 'N' is the number of unique properties in the data, but this suits only for property bound queries. Property bound queries may not only be the necessity always. So, Hexastore
came up with a new indexing scheme with six different combinations of subject, predicate and object. Each of these six index structures centered around one RDF element and decides about the prioritization of other two elements. For example, consider a subject headed index, for every value of the subject, all the properties associated to are stored in the form of list and every item of that list (every property) will be pointing to another list which contains the list of objects associated to the subject with that particular property. This scheme is followed for the indexing scheme as '*spo*' , the structure of the data storage changes with the index scheme followed providing equal importance to all types of queries.

Pros of **HexaStore** :-

- Efficient handling of multi valued resources.
- As only valid values are stored, this can avoid NULL values.
- HexaStore accesses only those items which are required by the query, hence reducing I/O cost.

Cons of **HexaStore** :-

- The main drawback of HexaStore is its five-fold storage space in its worst-case.
- Hard to handle updates and deletions in HexaStore.

Amazon Neptune is a fast, reliable, fully managed database for building and handling highly connected graphs, this is a graph database engine which is highly optimized for storing billions of graph components and retrieving the information within milliseconds of latency. For applications like social networking, if we would try to use relational databases, there has to be multiple databases with many foreign keys. Some other popular use cases of Amazon Neptune include Fraud Detection, Recommendation

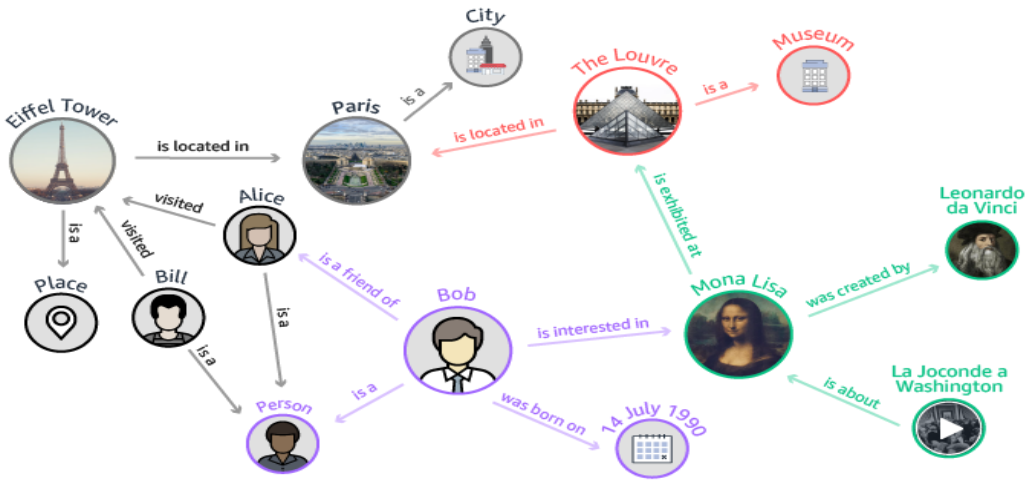
Engines, Knowledge graphs, Life sciences, Network/IT operations. Amazon Neptune has only two data types, vertices and edges which have properties in the form of key value pairs.

Pros of **Amazon Neptune**:-

- Supports query languages like Gremlin and SPARQL.
-

Cons of **Amazon Neptune** :-

- Graph visualization is not native, we have to use some third party resources for graph visualization.
- Not an open source and payment is just like other amazon services. Pay per use method.



REFERENCES FOR Q2(C):-

1) Weiss, Cathrin & Karras, Panagiotis & Bernstein, Abraham. (2008). Hexastore. Proceedings of the VLDB Endowment. 1. 1008-1019. 10.14778/1453856.1453965.
2) https://aws.amazon.com/neptune/
3) https://medium.com/faun/aws-neptune-overview-amazons-graph-database-service-9e3a0f688e26
4) Image Source: https://aws.amazon.com/neptune/

****