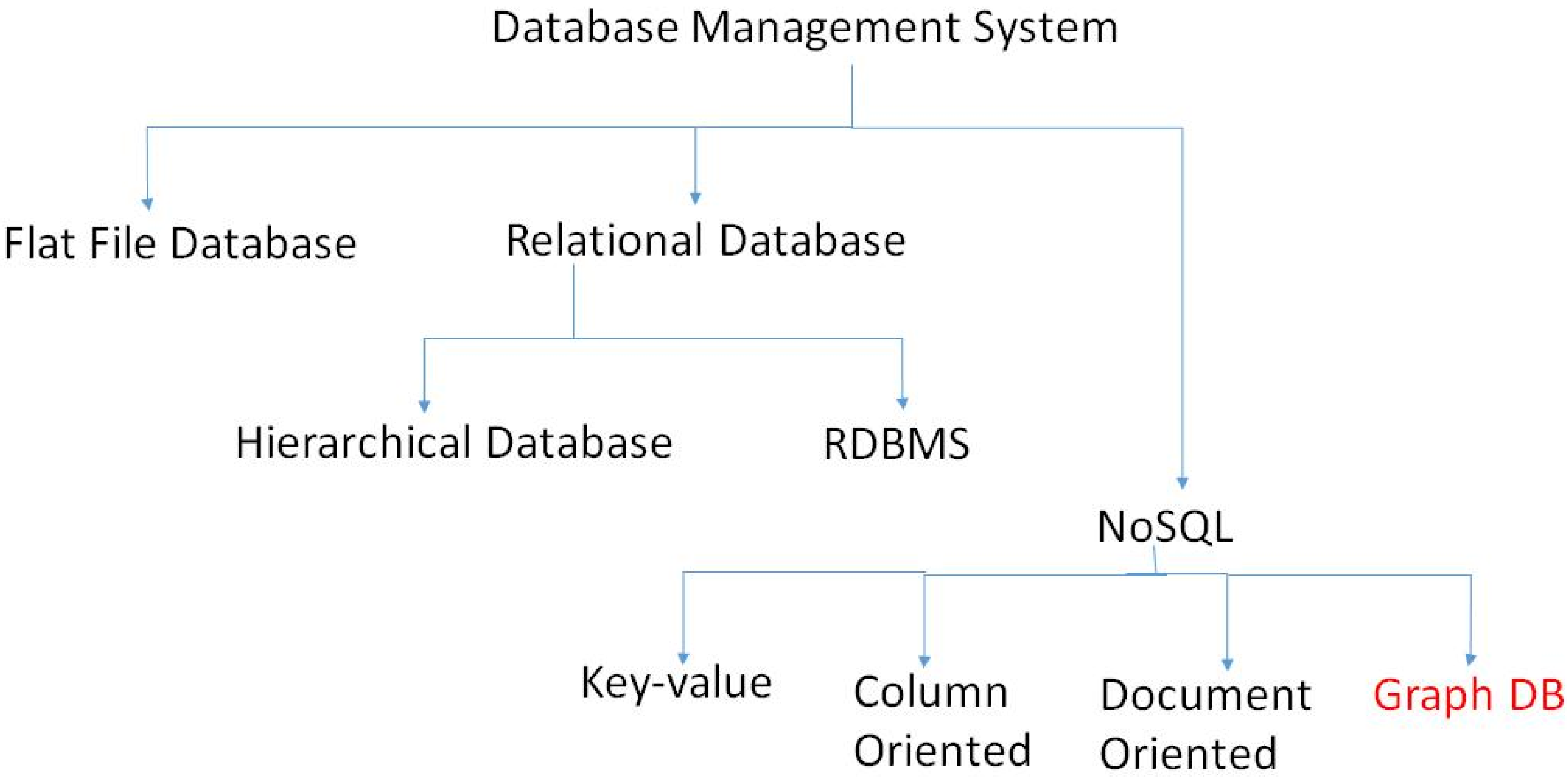
Graph Data Digest Document Format (GDF)

By

Vipin Baswan	2017A7PS0429P
Suyash Raj	2017A7PS0191P
Yashdeep Gupta	2017A7PS0114P
Abhinava Arsada	2017A7PS0028P
Sreyas Ravinchandran	2017A7PS0275P



Database Management System

Flat File Database

Relational Database

Hierarchical Database

RDBMS

NoSQL

Key-value

Column Oriented

Document Oriented

Graph DB

What is GRAPH DB

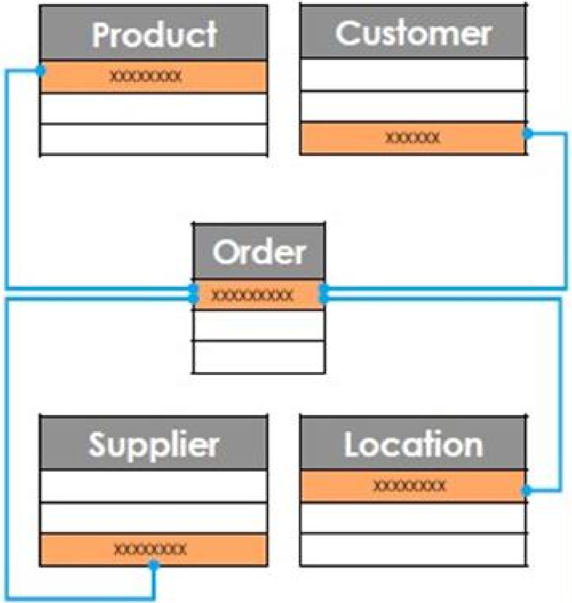
Uses a graphical model to represent and store the data.

Unlike relational database, in which data is stored in tables using a rigid structure with a predefined schema, in graph databases, there is no predefined schema.

It is a No-SQL database.

Prominent Graph DB : Neo4j, Blazegraph, and OrientDB.

Relational Database



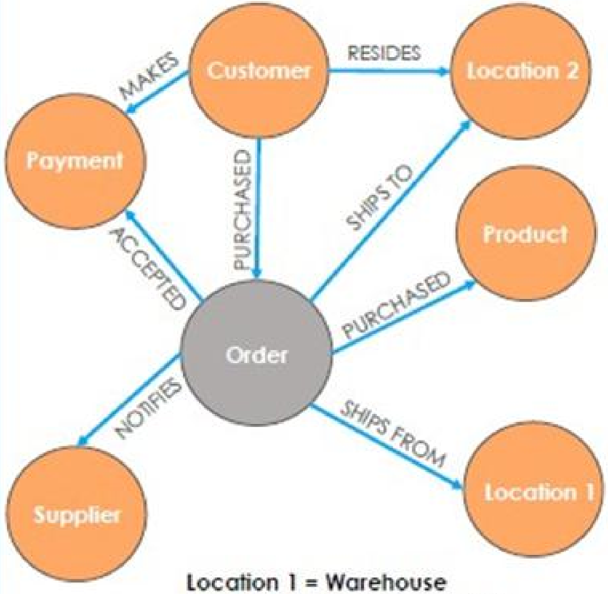
- Rigid Schema
- High Performance for transactions
- Poor performance for deep analytics

Key-Value Database

KEY		VALUE
Product	→	XXXXX
Customer	→	XXXXX
Supplier	→	XXXXX
Location	→	XXXXX

- Highly fluid schema/no schema
- High performance for simple transactions
- Poor performance deep analytics

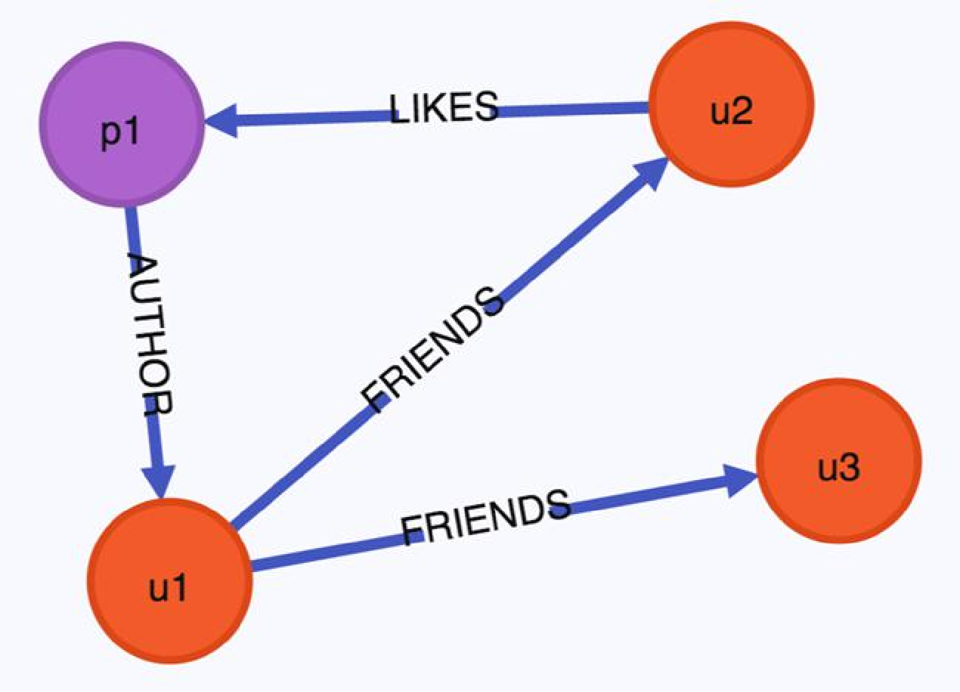
Graph Database



Location 1 = Warehouse
Location 2 = Delivery Location

- Flexible schema
- High performance for complex transactions
- High performance for deep analytics

Example



Why GRAPH DB

Better performance – Since there are no joins, recursive queries are localized to only a part of the graph.

Flexibility – New relationships, subgraphs can be added without disturbing existing database structure and queries, since schema is not fixed.

Maintenance – Application interface can be modified without any change to data definition schema.

Our Project..

Background and Objective

-Many queries are often cheap and efficient, when done on a graph.

-The goal is to represent data of any kind as a graph.

-Modelled using Resource Description Framework(RDF).

  -Stores data in the form of triples(Yashdeep|likes|iceCream)each having a globally unique Resource ID.

-Uses platforms like GitHub, Docker ,Bash and Python etc.

Various stages of the Project

1)Deciding the structure of our GDF files.

-Any kind of data is ultimately reduced to this form.

-We are using a 7-column format.(1 hash,3 resources,3 qualifiers)

-Hash is a unique ID given to each triple.

-Resources are a reference to any object/entity.

-Qualifiers are like adjectives.(blue sky).

Various stages of the Project

2)Generation of globally unique IDs.(using hashes or salted hash)

-SHASUM can be used to generate a unique ID for any string.

Generation of GDF files from given structured data(e.g. SQL table)

-Creation of a metadata structure which maintains constraints like foreign key or primary key etc.

-Creating a Parser which converts any data format to GDF.

Various stages of the Project

3)Developing a query language (based on SPARQL) for information retrieval. Implemented using Bash for quick runtime.

-Some queries we plan to implement are:-

-about: Lists all resources present in a GDF file.

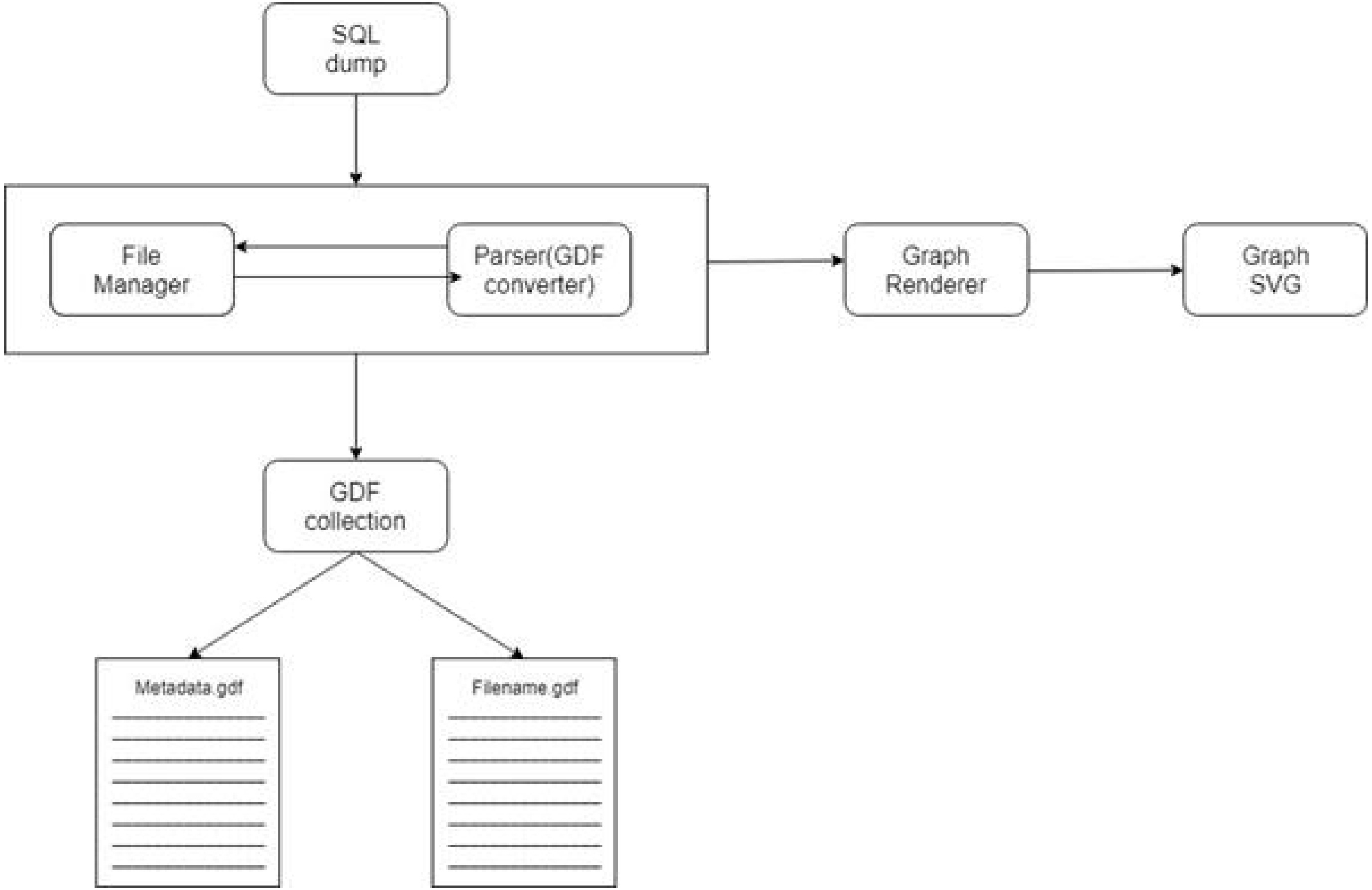
-seperate: Segregates all the subjects into separate files along with their associated triplets.

Various stages of the Project

4)Developing a renderer which generates a SVG file(visual format) representing the underlying graph structure of any GDF file.

-D3 stands for Data-Driven Documents. D3.js is a JavaScript library for data manipulation.

-We will use D3.js to generate SVGs for our GDF files.


SQL dump
File Manager
Parser(GDF converter)
Graph Renderer
Graph SVG
GDF collection
Metadata.gdf
Filename.gdf

7-COLUMN FORMAT

"Yashdeep likes to eat ice-cream"

↓

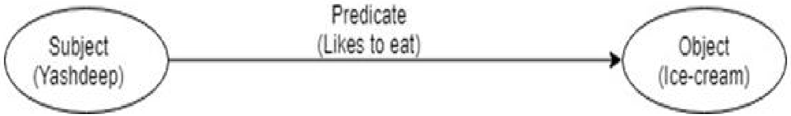
Parse in the seven column format : UID | Subject | Subject_Qualifier | Predicate | Predicate_Qualifier | Object | Object_Qualifier

↓

1242353353|Yashdeep|Person|LikesToEat|Preferences|Ice-cream|Dessert

↓

Represented in graph as :

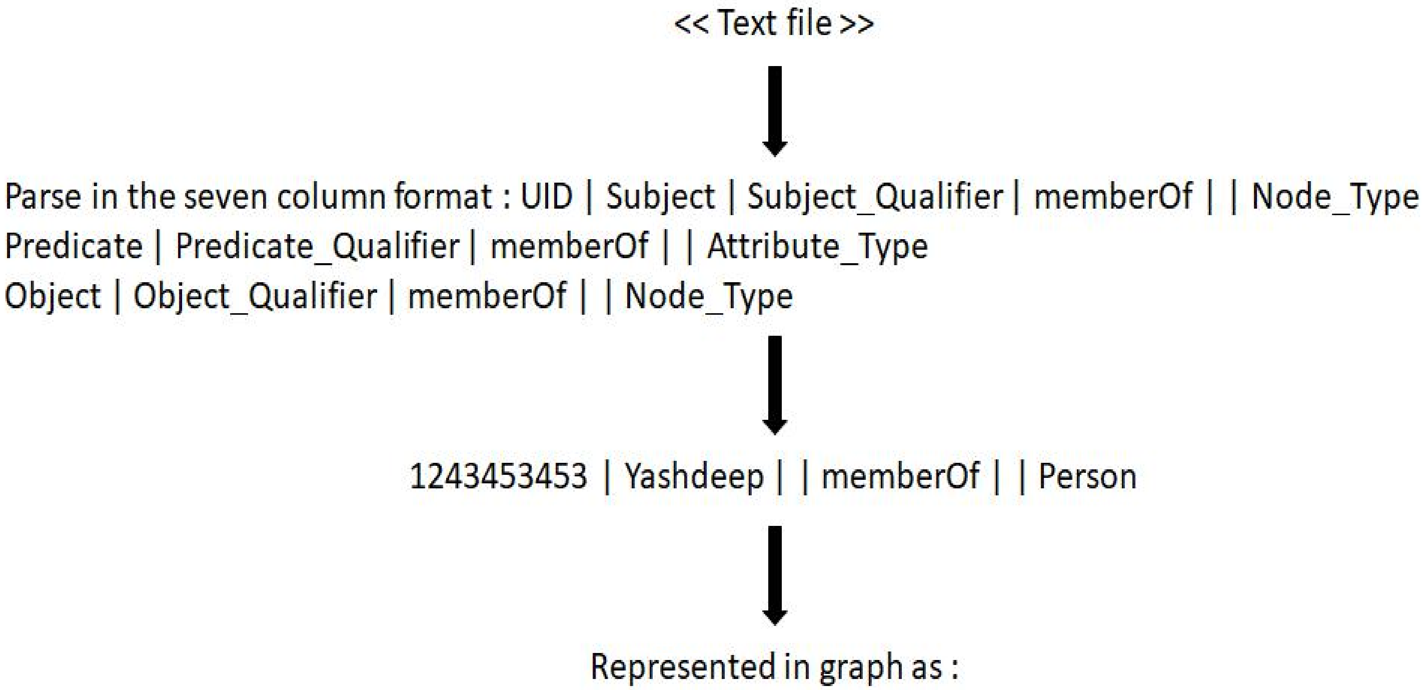


Qualifiers and UID

- UID is used to uniquely identify each tuple in the GDF file
- UID is calculated by using an in-built hash in bash: **md5sum**
- So, UID is ***md5sum of (Subject Predicate Object)***

- Qualifiers are used to give more information about Subject/Predicate/Object
- **Example:**
  Yashdeep is an honest boy. Then, “honest” and “boy” are qualifiers of Yashdeep
- Qualifiers can also be a URL.

META DATA FORMAT



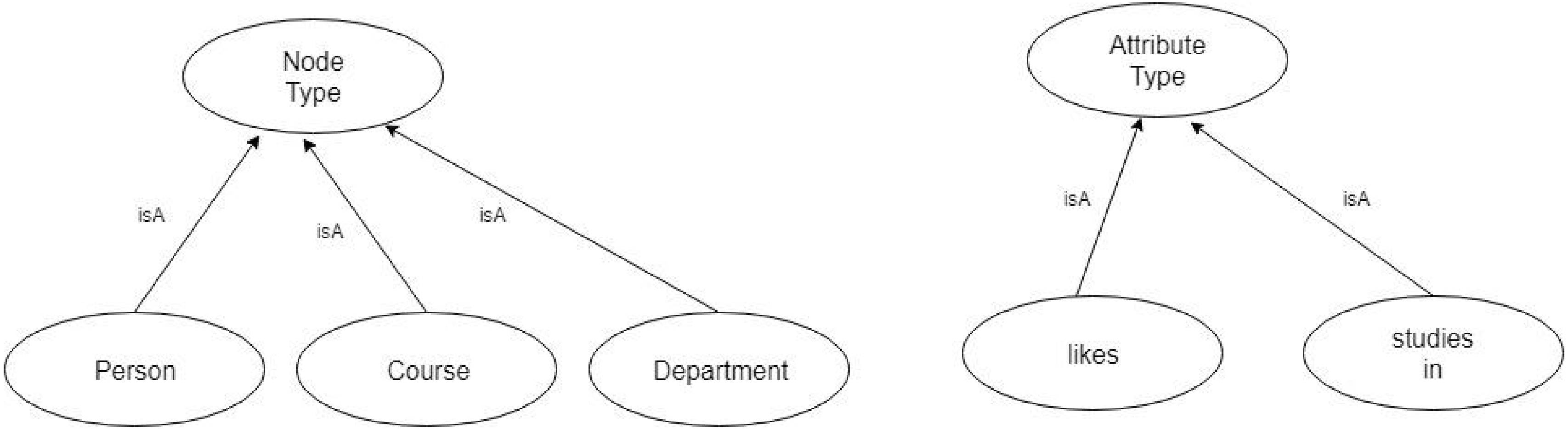
<< Text file >>

Parse in the seven column format : UID | Subject | Subject_Qualifier | memberOf | | Node_Type
Predicate | Predicate_Qualifier | memberOf | | Attribute_Type
Object | Object_Qualifier | memberOf | | Node_Type

1243453453 | Yashdeep | | memberOf | | Person

Represented in graph as :

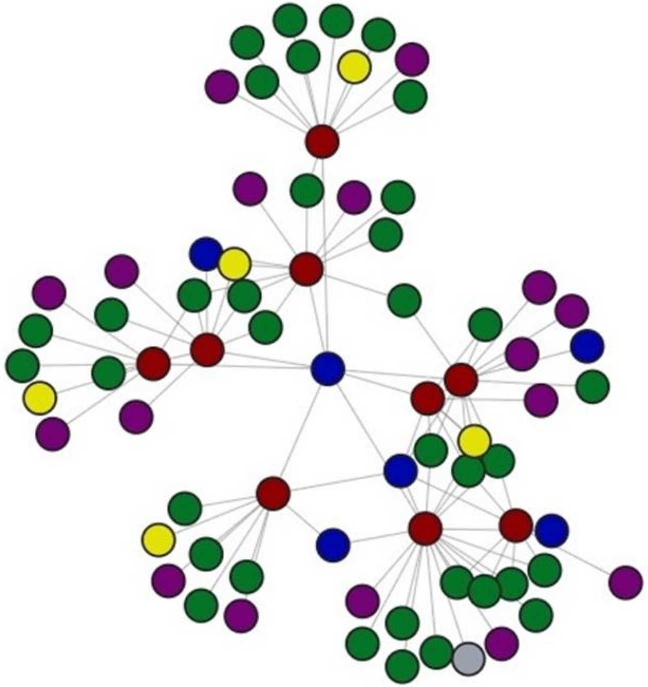
META DATA: Defining Nodes and Edges



D3.JS renderer

- ✓ Renderer is used to display the graphical output.

- ✓ We are using D3.js renderer which will generate the graph database for our GDF data



An example of the output of D3.js

About SPARQL

SPARQL is the standard language to query graph data represented as RDF triples.

- **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
- One of the three core standards of the Semantic Web, along with RDF and OWL.
- Became a W3C standard January 2008.
- SPARQL 1.1 now in Working Draft status.

Types of SPARQL queries

- **SELECT**
  Return a table of all X, Y, etc. satisfying the following conditions ...

- **CONSTRUCT**
  Find all X, Y, etc. satisfying the following conditions ... and substitute them into the following template in order to generate (possibly new) RDF statements, creating a new graph.

- **DESCRIBE**
  Find all statements in the dataset that provide information about the following resource(s) ... (identified by name or description)

- **ASK**
  Are there any X, Y, etc. satisfying the following conditions ...

Structure of a SPARQL query

```
PREFIX rov: <http://www.w3.org/TR/vocab-regorg/>

SELECT ?name

WHERE
  { ?x rov:legalName ?name }
```

Definition of prefixes

Type of query

Variables, i.e. what to search for

RDF triple patterns, i.e. the conditions that have to be met

SPARQL Update

Can be used for...

- Adding data (INSERT)
- Deleting data (DELETE)
- Loading RDF Graph (LOAD / LOAD .. INTO)
- Clearing an RDF Graph (CLEAR GRAPH)
- Creating RDF Graphs (CREATE GRAPH)
- Removing RDF Graphs (DROP GRAPH)
- Copying RDF Graphs (COPY GRAPH ... TO GRAPH)
- Moving RDF Graphs (MOVE GRAPH ... TO GRAPH)
- Adding RDF Graphs (ADD GRAPH TO GRAPH)

SELECT: Return the name of an organization with particular URI

Sample data

```
comp:A rov:haslegalName “Niké” .  comp:A org:hasRegisteredSite site:1234 .
Comp:B rov:haslegalName “BARCO” .

site:1234 locn:fullAddress “Dahliastraat 24, 2160 Wommelgem .
```

Query

```
PREFIX comp: < http://example/org/org/>
PREFIX org: < http://www.w3.org/TR/vocab-regorg/ >  PREFIX site: <http://example.org/site/>
PREFIX rov: <http://www.w3.org/TR/vocab-regorg/>

SELECT ?name  WHERE
  { ?x org:hasRegisteredSite site:1234 .
     ?x rov:haslegalName ?name .}
```

Result

name
“Niké”

Our Progress

We have completed the following:

a) Text to GDF file conversion

b) Generation of meta-data.gdf file

c) Initial steps for a querying engine.

FUTURE PROSPECTS

- Conversion of our GDF file into JSON objects
- Understand the working of D3.js renderer and use it to generate graph SVGs from the JSON objects generated from (a)
- Complete the building of querying engine (based on SPARQL)

APPLICATIONS

- Social networks
- Network diagrams
- Fraud detection
- Access management
- Graph based search of digital asset etc.

REFERENCES

https://github.com/Sreyas-108/GDF : Our complete work is present on this remote repo

CONCLUSION

- Graph databases show enormous promise in terms of efficiency, by one or more orders of magnitude.
- Graph databases have a very flexible data model and a mode of delivery conforming to modern methods.
- We aim tobuild a database engine to Create, Read, Update and Delete(CRUD).
- The paucity of time may not allow us to finish what we've started but we hope to build a foundation on which further progress can be made

THANKYOU