**Graphs are commonly used to represent linked data,**

- Efficient algorithms have been proposed to consume and mine graphs.
- Existing approaches rely on main-memory structures.
- Graph engines manage, store, and query graphs, e.g., DEX, Neo4j, HyperGraphDB, RDF-3X.
- There is no clear understanding of how existing graph database engines perform on graph-based tasks for consuming and mining linked data.

**Goals:**

- Understand the parameters that impact on the complexity of graph-based tasks.
- Visualize trends and patterns exhibited by existing engines in tasks of creation, reachability, adjacency, graph mining and pattern matching.

**Discussion**

- **Graph size**, density, and number of labels negatively impact on the performance of all the engines.
- **Graph summarization** seems to be more affected by the graph density and the number of labels.
- **Dense graph** is more influenced by the size of the graphs.
- RDF-3X outperforms the rest of the engines in pattern matching and graph creation.
- DEX seems to overcome the rest of the engines when the graphs are dense.
- Neo4j exhibits better performance in sparse graphs whenever they have a large number of labels.
- Internal implementations are able to exploit the properties of the data structures and indices implemented by each engine.

**Conclusions and Future Work**

Summarizing,

- Graph Engines implement a wide variety of structures to efficiently represent large graphs.
- RDF engines implement special-purpose structures to efficiently store RDF graphs.
- General-purpose graph database engines provide APIs to manage and query data.
- RDF engines efficiently support general data management operations.

In the Future,

- GRAPHIUM will visualize behavior of additional state-of-the-art graph engines and a wider variety of linked datasets.
- Extend GRAPHIUM to understand the impact of graph traversal strategies in query execution.