RDF2TG: Towards Supporting RDF in TigerGraph Property Graph Database System

Lu Zhou, Jay Yu

Innovation and Development Center, Tigergraph, Inc., 3636 Nobel Dr. Suite 100 San Diego, CA 92122, USA

Abstract

Graph data technology adopters often face the challenge of choosing between flexible knowledge representation and reasoning on Resource Description Framework (RDF) and large-scale data processing performance on Property Graph (PG) models. In this paper, we propose a generic method to bring the best of both worlds together by supporting RDF data in TigerGraph, a massive parallel distributed native property graph database system. This method relies on a generic schema with mapping rules for loading RDF data while preserving the flexibility of the original RDF graphs. We use LDBC Semantic Publishing Benchmark (SPB) to demonstrate how this mechanism maps RDF data and SPARQL queries into TigerGraph and GSQL.

Keywords

RDF Knowledge Graph, Tigergraph Property Graph, SPARQL, GSQL

1. Introduction

RDF is a W3C standard model representing data in triple statements composed of subject and object as nodes, connected by predicate as edges. In contrast, PG allows labeled properties on both nodes and edges for additional information. Due to the inherent differences between RDF and PG databases and their associated query languages, users are forced to trade-off between the semantic expressivity of RDF and the performance scalability of PG. One approach to bridge the gap was by Neo4j plugin "neosemantics".[1] It maps datatype properties and values from triples to concrete node properties. This model transformation might have limitations for use cases like entity resolution, where attributes are represented as nodes to find similarity via graph connectivities. In this paper, we propose another approach to map RDF data to TigerGraph with a generic schema that preserves the flexibility of RDF graphs, meaning instead of mapping RDF knowledge graphs and ontologies explicitly, the generic schema can tolerate dynamic updates in RDF graphs without affecting the mapping process. We successfully load an RDF graph with about 32 million triples generated from LDBC SPB benchmark[2] to TigerGraph and execute 36 GSQL queries translated from SPARQL in the benchmark. A preliminary evaluation of a side-by-side comparison with an RDF database shows the generic PG model performs at a similar level without any optimization on the database engine level.

ISWC'22: International Semantic Web Conference, October 23–27, 2022, Hangzhou, China

lu.zhou@tigergraph.com (L. Zhou); jay.yu@tigergraph.com (J. Yu)
0000-0002-0453-9965 (L. Zhou)

[1] https://neo4j.com/labs/neosemantics/4.1/
[2] https://ldbcouncil.org/benchmarks/spb/

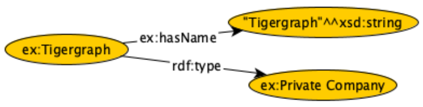

Figure 1: Example Triples in RDF Graphs

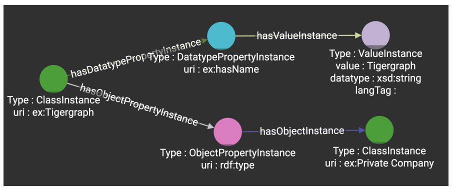

Figure 2: Example Triples in TigerGraph

2. RDF to PG Graph Mapping

We design a generic schema in TigerGraph to import RDF graphs[3] based on mapping rules. Figure 2 demonstrates an example graph after mapping two RDF triples depicted in Figure 1. There are four types of vertex: ClassInstance, ObjectPropertyInstance, DatatypePropertyInstance, and ValueInstance, and four types of directed edge: hasObjectPropertyInstance, hasObjectInstance, hasDatatypeProperyInstance, and hasValueInstance. ValueInstance has three properties (value, datatype, langTag), while other vertices have one property (uri). To evaluate the effectiveness of the schema and mapping rules, we utilize the LDBC SPB benchmark to generate an RDF graph with about 32 million triples. We load the RDF data into TigerGraph and result in a PG with about 39.8 million vertices and 127.1 million edges.

3. SPARQL to GSQL Translation

LDBC SPB Benchmark provides two types of queries - basic and advanced. Basic queries contain search, aggregate, geo-spatial, full-text search, and time-range, while advanced ones add analytical, drill-down, and faceted search. For this phase of the project, we manually translate 36 SPARQL to GSQL queries and verify the results are equivalent. We conduct a preliminary evaluation of query performance in TigerGraph. The results are promising and comparable to running the same benchmark on an RDF graph database using the same environmental configuration, without any optimization on the database engine level.

4. Conclusion and Future Work

In conclusion, we proposed a method to map RDF data and SPARQL queries to TigerGraph. Preliminary results from applying it to the LDBC SPB benchmark are promising. Codes to migrate RDF graphs to TigerGraph, mapping rules, queries, and performance are accessible in the GitHub repository.[4] We still have a few areas to expand our approach to cover more RDF features like blank nodes, named graphs, RDFS and OWL reasoning, as well as advanced SPARQL query capabilities to construct new graphs and perform updates. We will generalize the manual SPARQL to GSQL translation rules to automatically support no-code/low-code RDF data and query in TigerGraph.

[3]Supports RDF 1.1 with RDF Schema (RDFS) and Web Ontology Language (OWL).
[4]https://github.com/kbzhoulu/ldbc_spb_tigergraph