UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Graph Generators
State of the Art and Open Challenges

Farzad Shami

Prof. Gianmaria Silvello

Introduction

Overview:

- *Importance of Benchmarking:*
- *Significance of RDF Data Benchmarking:*

Benchmark Selection Rationale:

- ***BSBM****:* Developed specifically for benchmarking RDF stores, Focuses on realistic e-commerce scenarios, providing insights into the performance of systems handling complex data structures.
- ***LUBM****:* Designed for ontology-based applications, particularly in the academic domain.
- ***WatDiv****:* A benchmark designed for generating diverse web-related data.

Objective of the Presentation:

- *Compare and Contrast*
- *Informed Decision-Making*

Structure of the Presentation:

  - Overview of BSBM, LUBM, and WatDiv.
  - Comparative analysis of their characteristics, performance metrics, and scenarios.
  - Evaluation of benchmark strengths and weaknesses.
  - Performance results and implications.

BSBM
Berlin SPARQL Benchmark

Berlin SPARQL Benchmark (BSBM) - Overview

Overview of BSBM:

Developed for benchmarking RDF stores.

Focus on realistic e-commerce scenarios.

Key Characteristics:

Diverse data, including products, reviews, and users.

Realistic data distribution.

Main Goals of BSBM:

Compare different stores that expose SPARQL endpoints

Have realistic use case motivated data sets and Query mixes

Test query performance (integration and visualization) against large RDF datasets rather than complex reasoning

Berlin SPARQL Benchmark (BSBM) - Dataset

BSBM Dataset

Built around an e-commerce use case

Dataset generator

Scales to arbitrary sizes (scale factor = # of products)

Data generation is deterministic

Dataset objects: Product, Product Type, Product Feature, Producer, Vendor, Offer, Review, Reviewer and Reviewing Site.

BSBM Query Mix

Simulates how customers browse, review and select items online

Operations include

Look for products with some generic features	Look for products without some specific features
Look for similar products	Look for reviews and offers
Pull up all information about a specific product	Find the best deal for a product

Berlin SPARQL Benchmark (BSBM) - Characteristics

Data Scale

Varying dataset sizes for scalability testing.

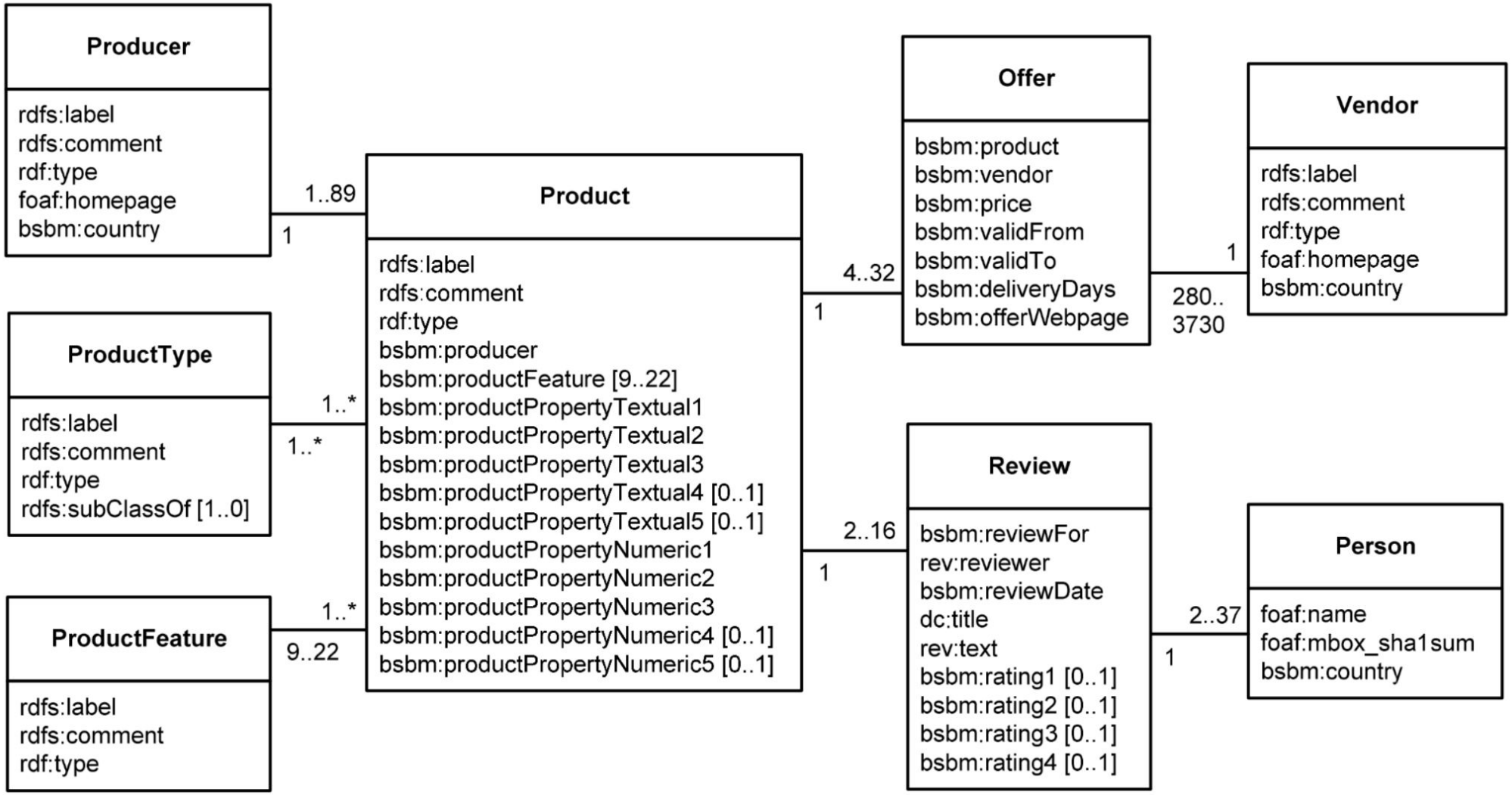
Query Mix

Complex SPARQL queries simulating real-world scenarios.

Performance Metrics

Throughput, response time, and other relevant metrics.

Berlin SPARQL Benchmark (BSBM) - Schema



Berlin SPARQL Benchmark (BSBM) - Setup & Run

Git repo https://github.com/lszeremeta/bsbmtools-json

There are different commands that you can generate the data + and for testing we use testdrive of this code.

we can get the output in the format of ttl, sql (mySql), n-triples or ….

we are allowed to put the dictionary for generation the data, it needs three dictionaries.

Default Dictionaries:

Dictionary 1: Words from set of product names (around 90.000 words)

Dictionary 2: Words from English text (todo: look for a corpus with English sentences, currently dictionary 1 is used)

Dictionary 3: Names of persons (around 90.000 names)

All the data are provided here

Based on the documents, Use-cases are around evaluating the performance of RDF stores in the context of e-commerce scenarios.

Berlin SPARQL Benchmark (BSBM) - Problems, Discussion

Importing the data to the PostgreSql (schema not compatible), Also I change a bit java versions to compile the project.

Connecting to postgres sometimes cause error + cannot automatically switch to SQL mode (I manually change code)

In my view, considering both my results and reported findings, opting for RDF stores when designing a system might not be the best choice. SQL-based databases show excellent performance.

My concern with this approach lies in the fully relational model presented. Logically, relational databases seem consistently superior. Additionally, there's a limitation in the ability to change the dataset model.

However, it's worth noting that for e-commerce, especially with a consistent schema, finding a good RDF store is feasible. The results, particularly in GraphDB and PostgreSQL, are promising according to the provided software versions in the article.

Berlin SPARQL Benchmark (BSBM) - Problems, Discussion

Importing the data to the PostgreSql (schema not compatible), Also I change a bit java versions to compile the project.

Connecting to postgres sometimes cause error + cannot automatically switch to SQL mode (I manually change code)

In my view, considering both my results and reported findings, opting for RDF stores when designing a system might not be the best choice. SQL-based databases show excellent performance.

My concern with this approach lies in the fully relational model presented. Logically, relational databases seem consistently superior. Additionally, there's a limitation in the ability to change the dataset model.

However, it's worth noting that for e-commerce, especially with a consistent schema, finding a good RDF store is feasible. The results, particularly in GraphDB and PostgreSQL, are promising according to the provided software versions in the article.

LUBM
The Lehigh University Benchmark

The Lehigh University Benchmark (LUBM) - Overview

Introduction to LUBM:

Primarily designed for ontology-based applications.

Academia-focused, representing university-related data.

LUBM Characteristics

Data Structure

Emphasis on university-related entities (professors, students, courses).

Query Complexity

SPARQL queries targeting ontology-based knowledge representation.

Evaluation Metrics

Efficiency in handling ontological queries.

The Lehigh University Benchmark (LUBM) - Generation

Generation:

LUBM generates synthetic RDF data representing a university domain using a data generator tool.

The dataset includes professors, students, courses, departments, and publications, with relationships between these entities.

The data generation process is parameterized, allowing users to control the size and complexity of the generated dataset.

Query Set:

LUBM provides a set of SPARQL queries that are representative of queries related to university data. These queries involve retrieving information about professors, students, courses, publications, and relationships between entities.

The Lehigh University Benchmark (LUBM) - Results, Discussion

They have tested Jena only with the smallest dataset. Unsurprisingly, when its RDFS reasoning was turned on, Jena' s performance was exactly the same as Sesame' s in terms of query completeness and soundness. However, Jena was much slower in answering most of the queries than Sesame.

For some of the queries, Jena did not terminate even after being allowed to run for several hours. In this experiment we have used a timeout of 2 hours.

When Jena was used with its OWL inferencing, it could answer even smaller num-ber of queries within the time limit. We speculate that the poor performance of Jena is because its rule-based reasoners are less optimized for a semantically complex ontology like Univ-Bench.

In my view, this is better than the previous benchmark, even though they deal with the data distribution almost the same way, but this benchmark is a more acceptable approach if you have a similar scenario considering that it pays attention to the inference system.

PS: I don't go ahead with testing on PostgreSql or Graphdb and left the results to the paper results, but i test the repository that generate data.

The Lehigh University Benchmark (LUBM) - Results, Discussion

They have tested Jena only with the smallest dataset. Unsurprisingly, when its RDFS reasoning was turned on, Jena' s performance was exactly the same as Sesame' s in terms of query completeness and soundness. However, Jena was much slower in answering most of the queries than Sesame.

For some of the queries, Jena did not terminate even after being allowed to run for several hours. In this experiment we have used a timeout of 2 hours.

When Jena was used with its OWL inferencing, it could answer even smaller num-ber of queries within the time limit. We speculate that the poor performance of Jena is because its rule-based reasoners are less optimized for a semantically complex ontology like Univ-Bench.

In my view, this is better than the previous benchmark, even though they deal with the data distribution almost the same way, but this benchmark is a more acceptable approach if you have a similar scenario considering that it pays attention to the inference system.

PS: I don't go ahead with testing on PostgreSql or Graphdb and left the results to the paper results, but i test the repository that generate data.

WatDiv

Waterloo SPARQL Diversity Test Suite

Waterloo SPARQL Diversity Test Suite (WatDiv) - Overview

WatDiv Overview:

Web Data Generator Benchmark.

Emphasizes diverse web-related data.

WatDiv Characteristics

Data Diversity:

Simulates various types of web data (e.g., social networks, publications).

Query Challenges:

Mix of queries testing capabilities in handling diverse data structures.

Benchmarking Objectives:

Evaluate performance on realistic web data scenarios.

Waterloo SPARQL Diversity Test Suite (WatDiv) - benefits

Diverse Query Workloads:

WatDiv provides a diverse set of SPARQL queries that cover various aspects of query patterns, complexities, and data access patterns. This diversity allows for a more comprehensive evaluation of SPARQL query engine performance.

Realistic Data Generation:

WatDiv includes a data generator that produces synthetic RDF datasets based on a realistic domain model. The generated data is designed to reflect the complexity and diversity of real-world RDF datasets.

Parameterization for Scalability:

The benchmark allows users to control the scale of the generated datasets using parameters, enabling the evaluation of SPARQL engines across different data sizes and complexities. This parameterization supports scalability testing.

Community Adoption:

WatDiv has gained recognition and adoption within the Semantic Web and database research communities. Its usage is supported by researchers and developers seeking a standardized benchmark for SPARQL engine evaluations.

Waterloo SPARQL Diversity Test Suite (WatDiv) - benefits

Challenges Common Assumptions:

WatDiv is designed to challenge common assumptions and limitations of SPARQL engines. By including diverse and complex queries, it helps identify strengths and weaknesses in query engine implementations.

Reflects Real-World Challenges:

The benchmark is designed to reflect real-world challenges in querying large and complex RDF datasets. This makes it a valuable tool for researchers and practitioners aiming to understand and improve the performance of SPARQL engines in realistic scenarios.

Waterloo SPARQL Diversity Test Suite (WatDiv) - Problems, Discussion

TBH, there isn't any problem with running and setup this tool, as community provide docker image, that you can easily setup the project in whatever system you have.

Waterloo SPARQL Diversity Test Suite (WatDiv) - Problems, Discussion

TBH, there isn't any problem with running and setup this tool, as community provide docker image, that you can easily setup the project in whatever system you have.

There are 4 sections:

1- Declaring Namespaces

2- Declaring Entities

3- Declaring Literal Properties

4- Declaring Non-Literal Properties

Documents are available here.

Waterloo SPARQL Diversity Test Suite (WatDiv) - System design