U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

JanusGraph

M.EIC - Non-Relational Databases 2022/2023

José Silva - up201904775@up.pt | Maria Carneiro - up201907726@up.pt | Sérgio Estêvão - up201905680@up.pt

Overview

What is JanusGraph ?

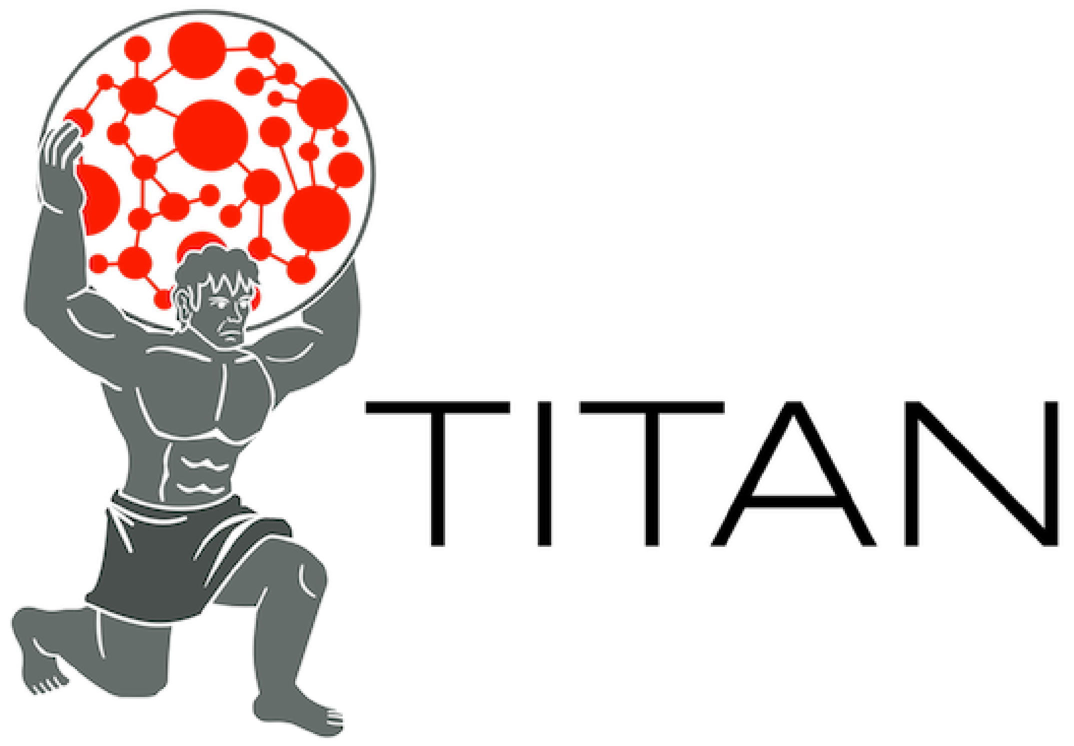
Graph Database

Distributed

Transactional

JanusGraph is an open-source **distributed graph database** project that began in 2017 as a clone of the TitanDB project, which had its development discontinued by the closure of the company that built it, Aurelius.

JanusGraph, from a TitanDB fork, was created by a group of developers from the open-source community to fill this void by offering a more actively maintained, community-driven alternative to TitanDB.



Overview

JanusGraph was brought under the open governance of the Linux Foundation in 2017 and was designed to be a scalable, adaptable, and distributed graph database that can manage enormous volumes of data and be utilized for a variety of applications.

- **Recommendation systems**
- **Fraud Detection**
- **Social Networks**

How is this versatility achieved?

Overview

JanusGraph main characteristic is its **modular architecture**, making it possible to use setup with different:

Storage Backends

Cassandra
HBase
BerkeleyBD
...

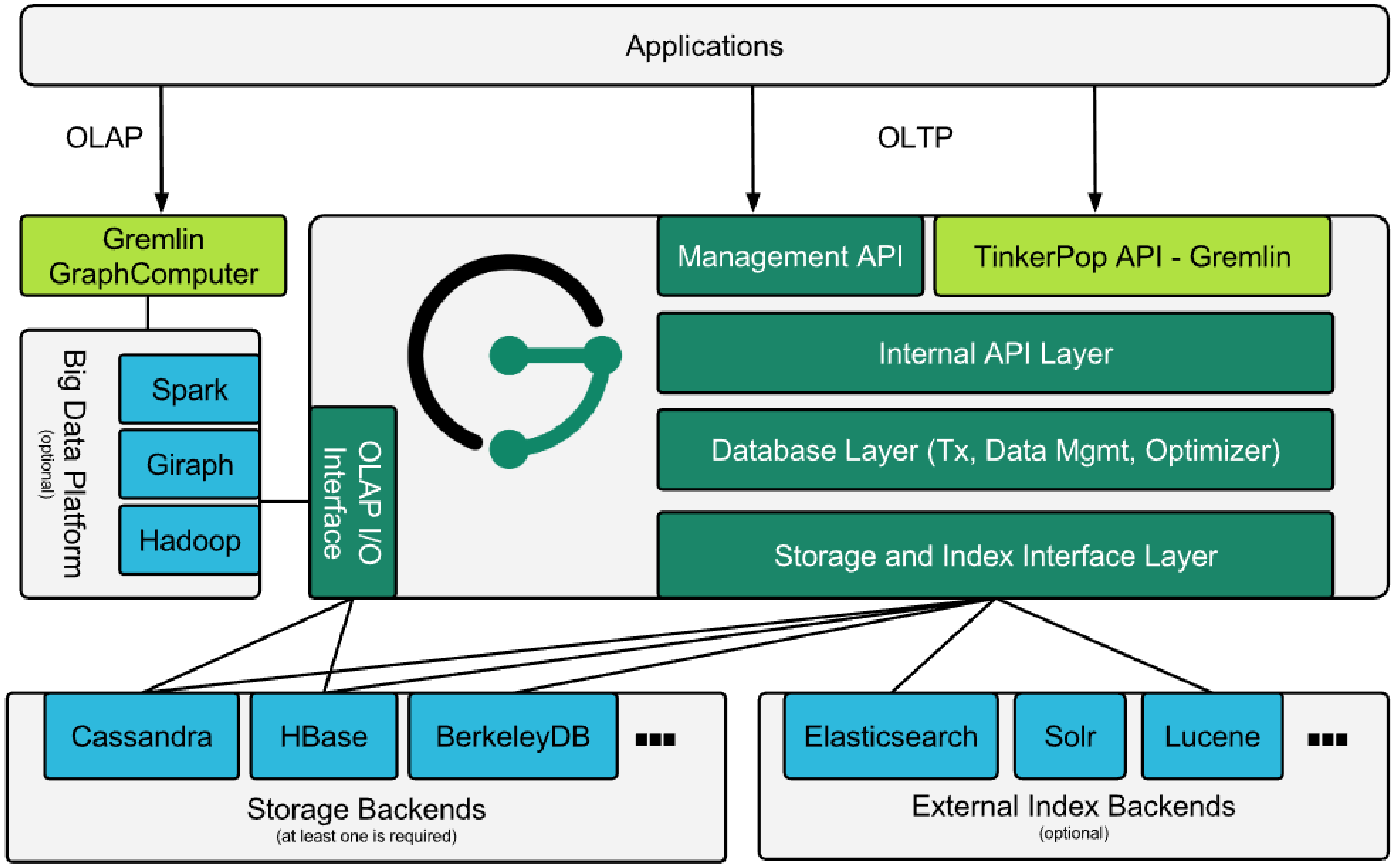
Indexing Backends

Solr
Lucene
Elasticsearch
...

Big Data Platforms

Spark
Giraph
Hadoop

JanusGraph Architecture



What is Gremlin?

Gremlin is a path-oriented language that expresses graph traversals and mutation operations.

Being an Apache TinkerPop's component, Gremlin is completely independent of JanusGraph and is supported by most graph databases which facilitate the migration of an application to different graph databases.

Installation and Administration

Installation

JanusGraph can be deployed using **Docker** containers. However, there's no docker image containing all of the JanusGraph architectures. They need to be run and managed independently.

JanusGraph provides a configuration file where the user specifies which components JanusGraph should use.

All of this can be easily run using **Docker-Compose**.

Installation and Administration

Administration

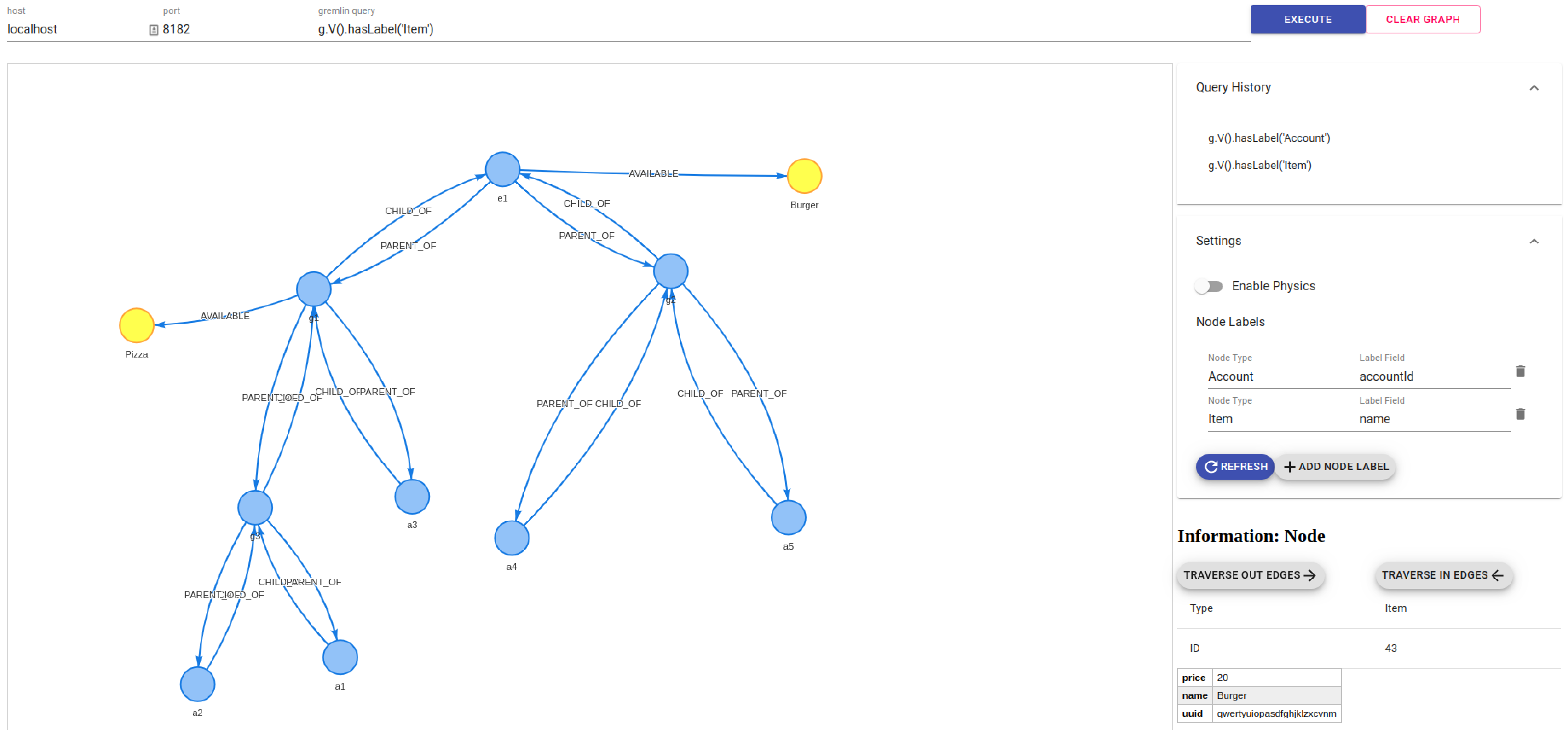
Although JanusGraph does not own a CLI to interact with the graph's data, we can interact with it using the gremlin server that comes in the installation files as well as in its docker image.

Gremlin provides the **Gremlin Visualizer**, making it possible to visualize and interact with the data.

Gremlin also has libraries in different programming languages, like Python and Java, making it possible to integrate into different servers.

Installation and Administration

Gremlin Visualizer



host	port	gremlin query
localhost	8182	g.V().hasLabel('Item')

Query History

g.V().hasLabel('Account')

g.V().hasLabel('Item')

Settings

Enable Physics

Node Labels

Node Type	Label Field
Account	accountId
Item	name

Information: Node

Type	Item
ID	43

price	20
name	Burger
uuid	qwertyuiopasdfghjklzxcvnm

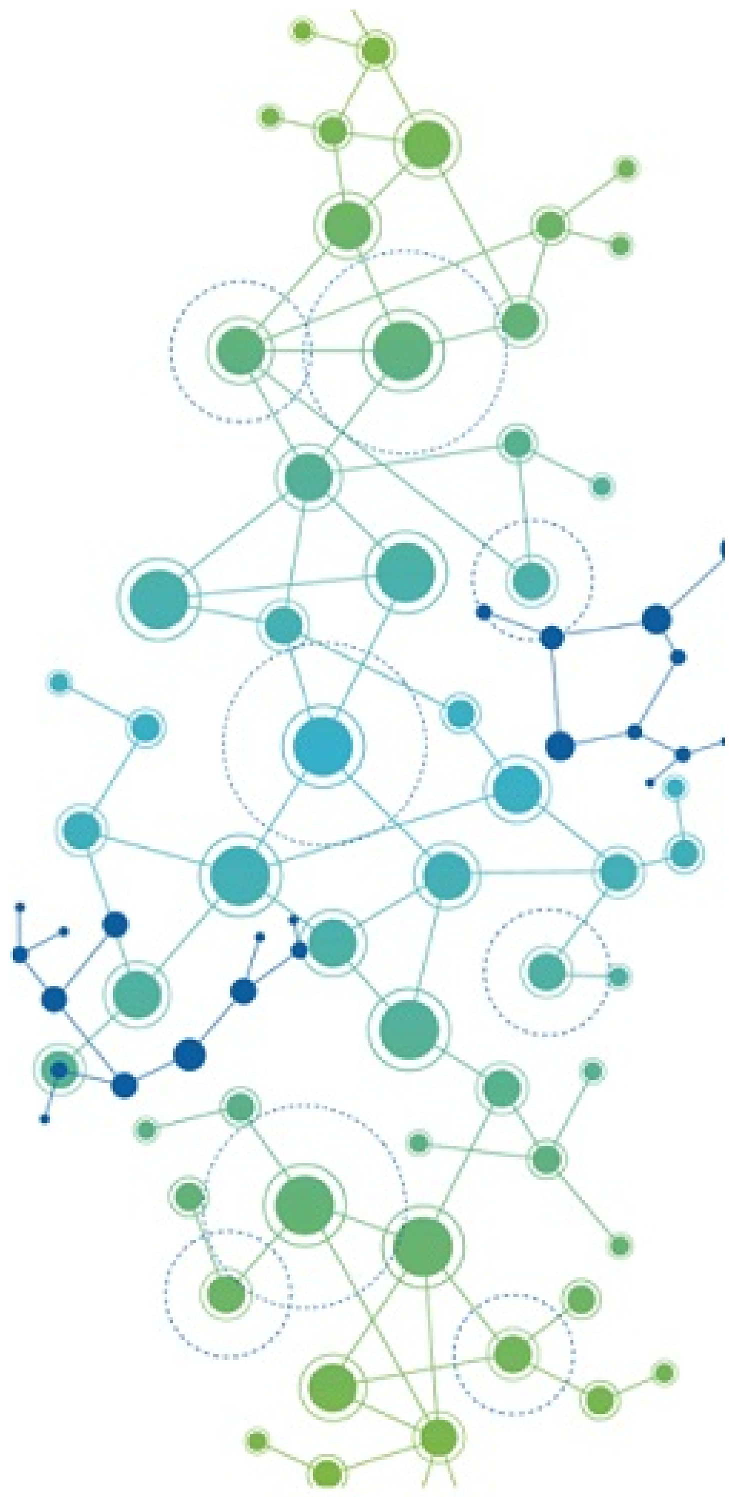
Data Model and Data Operations

Data Model

JanusGraph adopts a property graph data model, which consists of vertices, edges, and properties associated with both vertices and edges.

This model allows for representing complex relationships and storing rich data within the graph.

The vertices represent entities or objects, while the edges define the relationships between these entities.



Data Model and Data Operations

Data Operations

Regarding operations, JanusGraph provides a rich set of operations for traversing, querying, and modifying graph data.

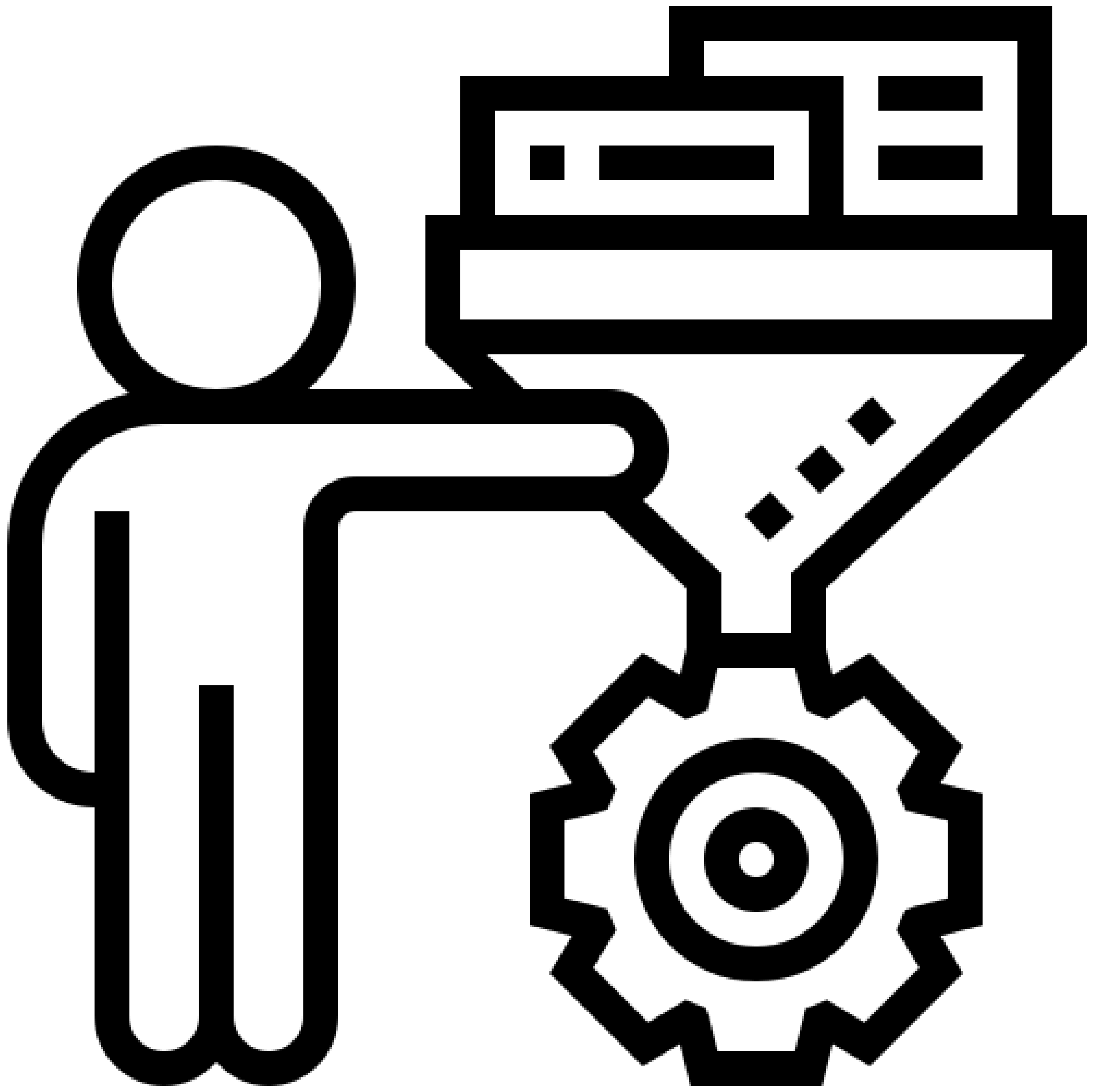
It provides the typical CRUD operations, and graph traversal, by using the integrated Gremlin traversal language, flexible graph querying, supporting property-based and index-based queries, and ACID transactions.

Data Model and Data Operations

Processing and Data Handling

JanusGraph also provides features to facilitate operations in data handling:

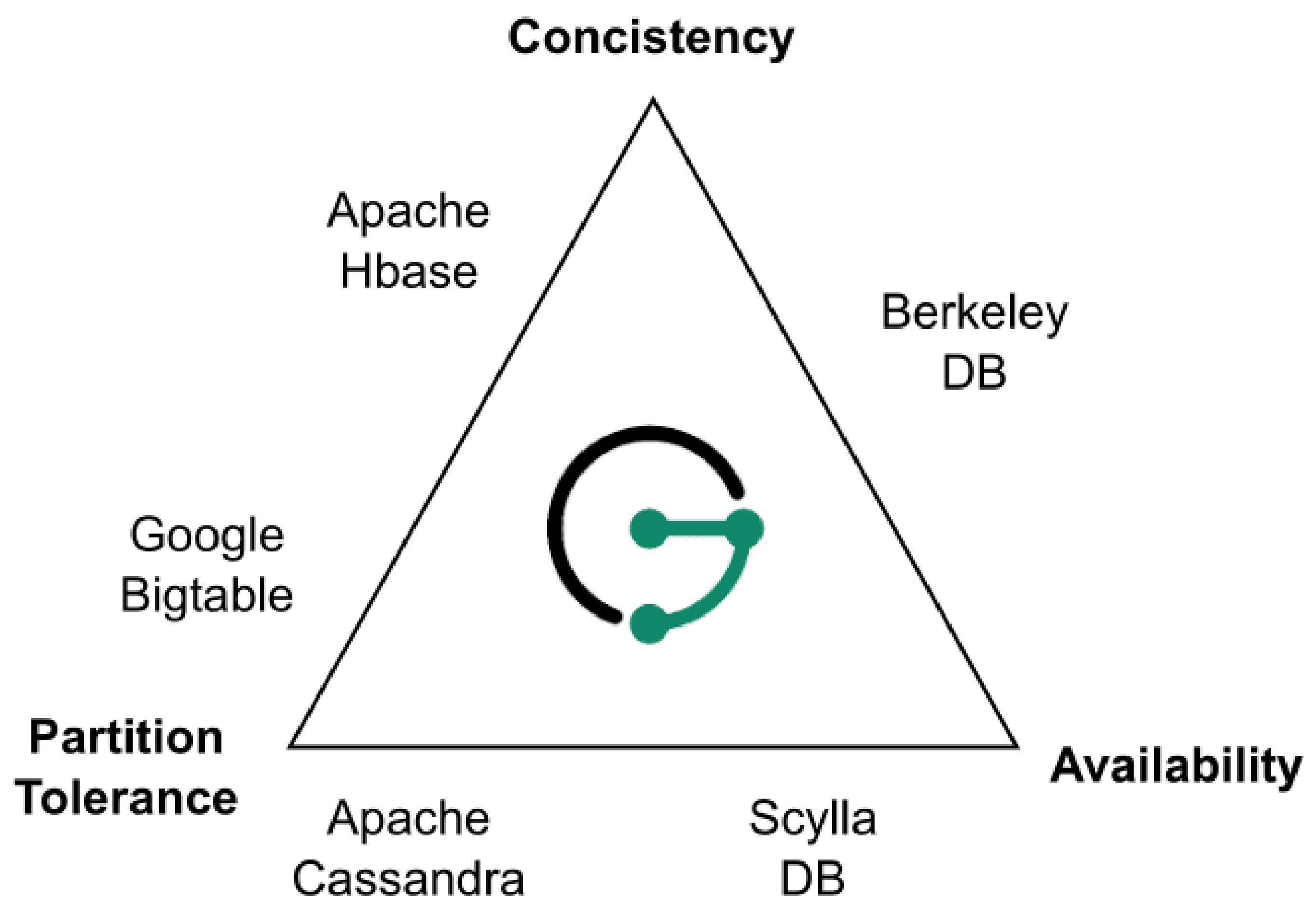
- Bulk Loading
- Caching for frequently accessed
- Transaction failure handling
- System recovery in cluster systems
- Migrations to TitanDB and Apache Thrift



Features Highlight

The choice of backend storage technology has a direct influence on how the CAP theorem is achieved in the context of JanusGraph.

Due to the nature of JanusGraph, depending on the selected backend different features of the CAP theorem may be emphasized.



Advantages & Drawbacks

Advantages	Drawbacks
Scalability	Complexity
Flexibility	Learning Curve and Effort
High Performance	Lack of Maturity
Unique Ecosystem	
Transactions	

Real Use Cases

JanusGraph is ideal for scenarios that involve complex networks (Graph problems), distributed and highly available environments, and graph analysis applications.

Cloud Applications ⁎

Kubernetes

Google BigTable

Graph Analytics

⁎ Architecture used by **Google**

A Network of Thrones: Overview

Relationships Conflicts Wars and Kinship in Monarchies

Original Dataset: Kinship And Conflict In Europe, from 1418 to 1918

Expanded Royals from 283 to 58674 by parsing a Geneological Tree file

Simplification over original dataset

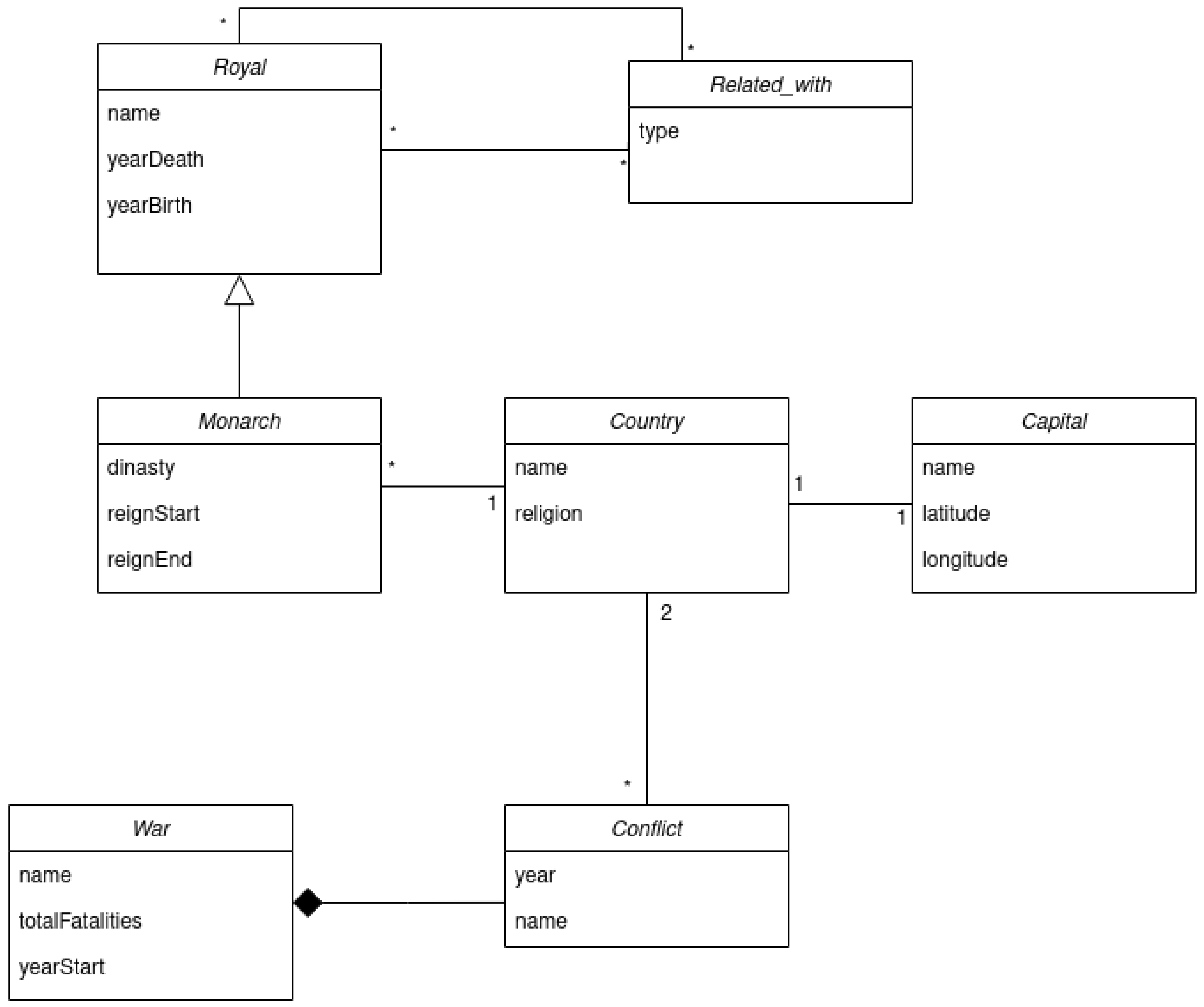
Aggregated only relevant information regarding royals, countries, wars and conflicts

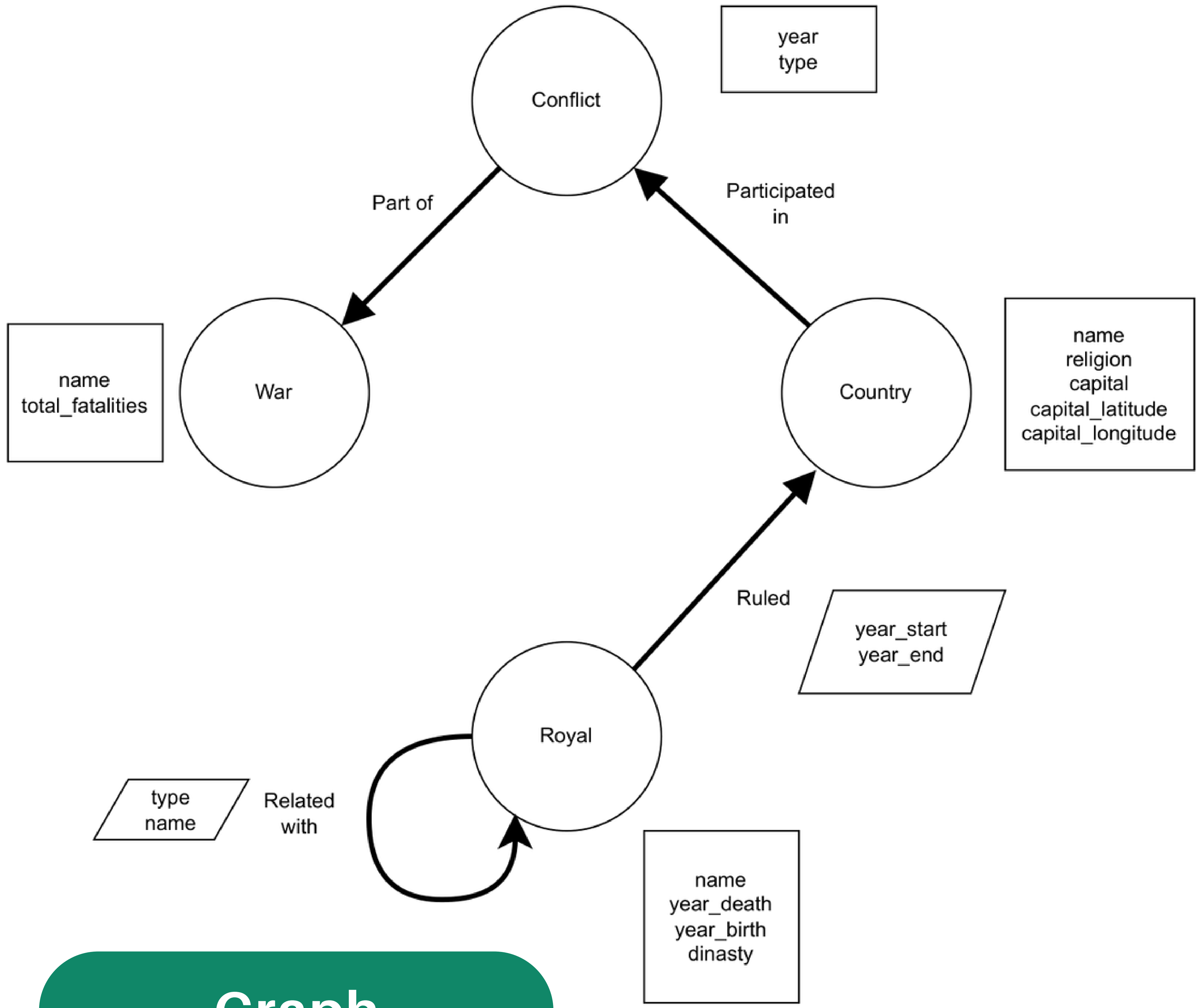
Besides the 58674 royals, we gathered 101 wars, 55 countries and 2614 conflicts

Allows filtering in order to analyze relationships between entities

Data Models

Conceptual

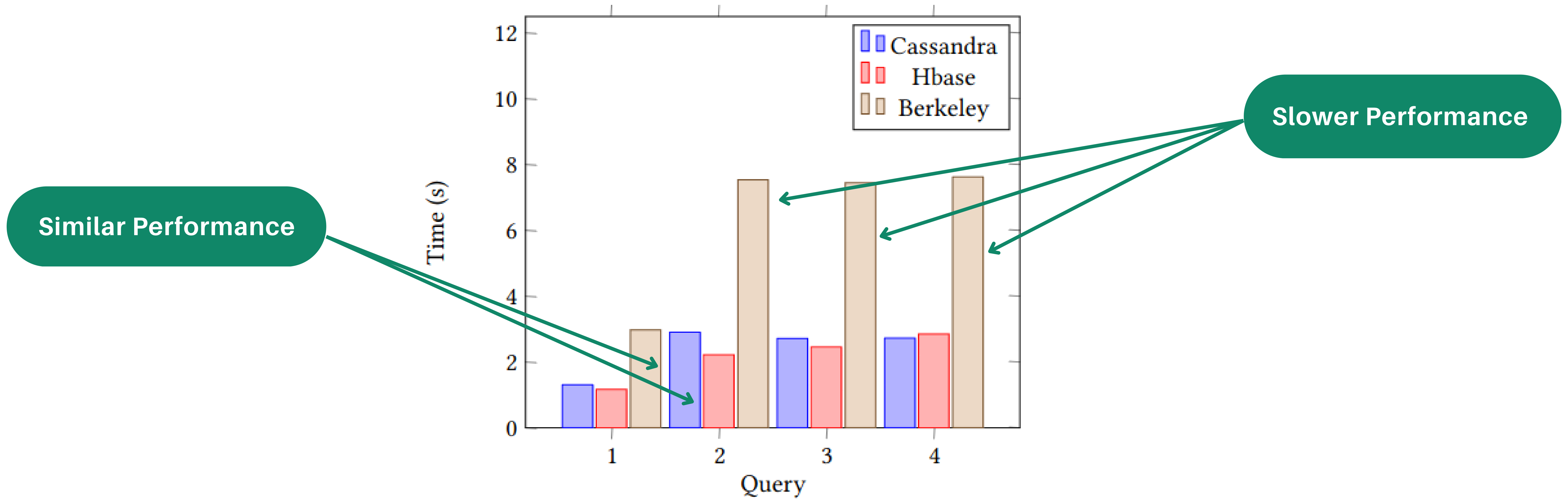




Graph

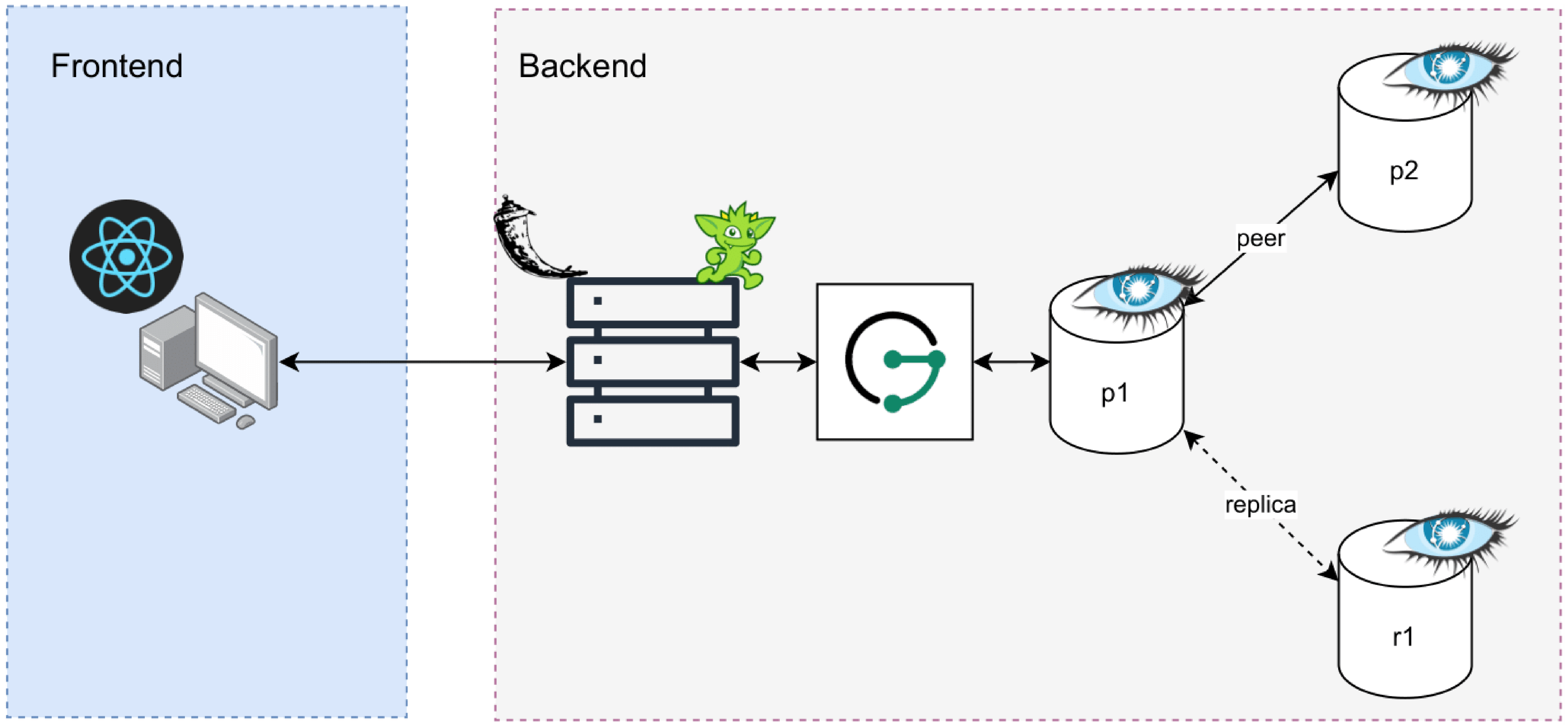
Storage Backend Benchmarking

4 distinct queries using the 3 most common data storage solutions: Cassandra, Hbase and Berkeley



Prototype Architecture

Emulate a Cluster System at a Small Scale



We chose **Cassandra** in order to value **Availability** over Consistency, due to the small scale of the project.

Demo

Questions?