Визуализация и сравнение семантических деревьев, отражающих компонентную структуру патентуемого устройства

С.А. Фоменков[1], Д.М. Коробкин[1], Я.Д. Коробкин[2]

[1]Волгоградский государственный технический университет,
[2]МОУ Гимназия №5 Волгограда

Аннотация: В данной работе описываются подходы к визуализации и сравнению семантических деревьев, отражающих компонентную структуру патентуемого устройства и связи между ними, при помощи графовых баз данных. Данные СУБД используют графовые структуры для хранения, обработки и представления данных. Основные элементы графовой базы данных - узлы (nodes) и ребра (edges), которые в рамках поставленной задачи моделируют сущности 3-х типов (SYSTEM, COMPONENT, ATTRIBUTE) и 5 типов связей (PART-OF, LOCATED-AT, CONNECTED-WITH, ATTRIBUTE-FOR, IN-MANNER-OF). По результатам исследования можно заявить, что Neo4j демонстрирует наилучшие возможности для визуализации графов; ArangoDB, несмотря на правильно введённые запросы, осуществляет неполную визуализацию; AllegroGraph показал сложную работу с кодом, затрудненную настройку визуализации графового дерева. Апробированы 3 алгоритма сравнения графовых представлений информации: Graph Edit Distance, Topological Comparison, Subgraph Isomorphism. Алгоритмы реализован на python, сравнивает 2 графовых дерева, выводит на экран визуализацию и анализ общих структур и различий графов.
Ключевые слова: семантическое дерево, компонентная структура, патент, графовые БД, Neo4j, AllegroGraph, ArangoDB.

Введение

Графовые базы данных [1] представляют собой мощный инструмент для хранения, управления и анализа данных, структурированных в виде графов. Они отличаются от традиционных реляционных баз данных тем, что более естественно и эффективно моделируют сложные взаимосвязи и структуры, встречающиеся в реальном мире.

В рамках работ [2,3] были разработаны подходы к извлечению компонентов устройств и их взаимосвязей из естественно-языковых патентных документов, а также построению на их основе семантических деревьев. В продолжение данных работ необходимо разработать методику сравнения семантических деревьев, отражающих компонентную структуру патентуемого устройства и связи между ними, для выявления новизны изобретения.

Рассмотрим графовые БД с целью хранения в них семантических деревьев, описывающих сущности 3-х типов (SYSTEM, COMPONENT, ATTRIBUTE) и 5 типов связей (PART-OF, LOCATED-AT, CONNECTED-WITH, ATTRIBUTE-FOR, IN-MANNER-OF).

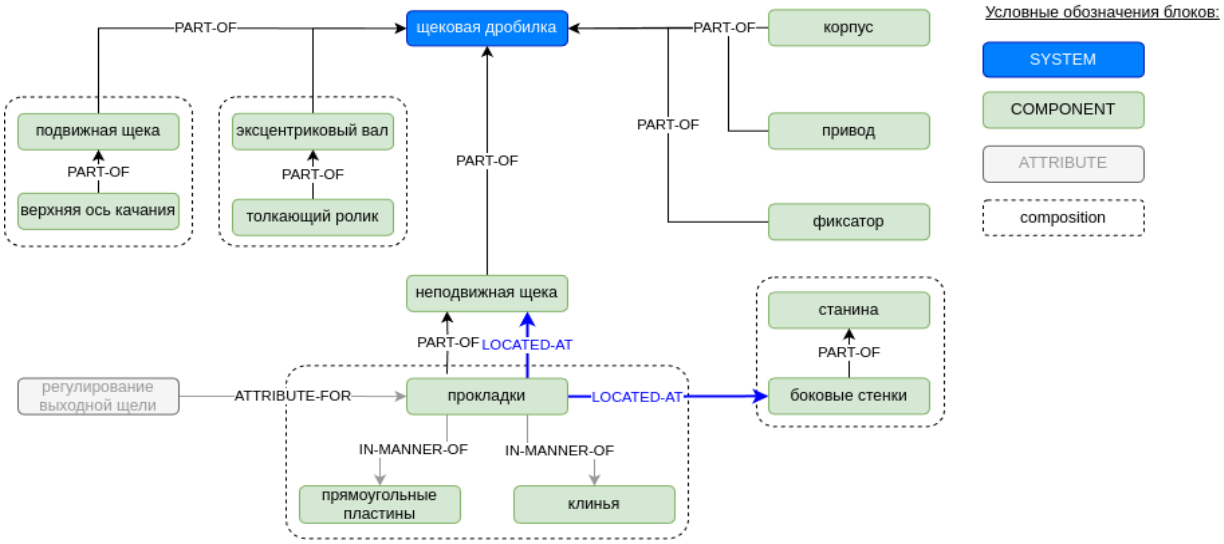

Рис. 1. – Пример семантического дерева, описывающего компонентную структуру запатентованного устройства

Анализ возможностей графовых баз данных

Было проведено исследование графовых баз данных, таких как Neo4j, AllegroGraph, ArangoDB, применительно к задаче представления семантических деревьев, содержащих описание компонентов запатентованных устройств и их взаимосвязей.

1. **Neo4j** - это графовая база данных [4], специализирующаяся на хранении данных в виде графовой структуры. Она предлагает множество возможностей благодаря своей ориентированности на работу с графами, приведем те из них, которые нам интересны в рамках нашей задачи:

– Гибкость модели данных: отличие от реляционных баз данных, Neo4j не требует строгой схемы данных. Вы можете добавлять новые типы узлов и связей (ребер) на лету, что упрощает адаптацию к изменяющимся требованиям бизнеса.

– Нативная поддержка графовых структур: Neo4j разработана с учетом работы с графами, что делает ее эффективной для запросов, требующих анализа связей и зависимостей между данными. Это особенно полезно в случаях, когда важно быстро находить пути между элементами или анализировать сети.

– Язык запросов Cypher: Neo4j использует язык запросов Cypher, который оптимизирован для работы с графовыми данными. Cypher позволяет легко выполнять сложные запросы на основе шаблонов графа, включая поиск путей, агрегацию данных и фильтрацию.

– Высокая производительность: Благодаря специализации на графовых структурах, Neo4j обеспечивает быстрый доступ к данным, особенно при выполнении запросов, связанных с глубоким поиском и анализом путей в графе.

– Визуализация и анализ: Neo4j имеет инструменты для визуализации графов, что упрощает анализ и понимание структуры данных. Это полезно для исследования связей между элементами и обнаружения паттернов.

– Масштабируемость: Neo4j поддерживает горизонтальное масштабирование, что позволяет обрабатывать большие объемы данных и повышать производительность системы при необходимости.

– Интеграция с другими технологиями: Neo4j предлагает API для интеграции с различными языками программирования и инструментами данных, такими как Python, Java, и другими. Это упрощает разработку приложений, использующих графовую базу данных.

2. **AllegroGraph** — это графовая база данных [5], специализирующаяся на хранении и анализе данных в виде графов, предлагает ряд особенностей и возможностей:

– Модель данных RDF: AllegroGraph базируется на стандарте RDF (Resource Description Framework), который используется для описания ресурсов в виде утверждений в форме троек (subject-predicate-object). Это делает AllegroGraph подходящей для хранения и обработки данных семантического веба, онтологий и связанных данных.

– Триплеты и SPARQL: В AllegroGraph данные хранятся в виде триплетов (троек), что упрощает представление сложных взаимосвязей между данными. Для запросов к данным используется язык SPARQL (SPARQL Protocol and RDF Query Language), который позволяет выполнять сложные запросы, включая агрегацию данных и работу с онтологиями.

– Гибкость и расширяемость: AllegroGraph поддерживает множество различных форматов данных и форматов сериализации RDF, включая RDF/XML, Turtle, N-Triples и другие. Это обеспечивает гибкость в импорте и экспорте данных, а также в интеграции с другими системами и инструментами.

– Высокая производительность и масштабируемость: AllegroGraph спроектирована для обработки больших объемов данных и поддерживает горизонтальное масштабирование. Она предлагает оптимизации для выполнения сложных SPARQL-запросов, что позволяет эффективно работать с графовыми данными в реальном времени.

– Поддержка стандартов и интеграция: AllegroGraph активно поддерживает различные стандарты и технологии, такие как OWL (Web Ontology Language) для работы с онтологиями, RDFS (RDF Schema) для описания схем данных и др. Она также интегрируется с другими технологиями и языками программирования через API и различные интерфейсы.

Благодаря своим возможностям, AllegroGraph широко используется в различных областях, в том числе в семантическом поиске.

3. **ArangoDB** — это многофункциональная база данных [6], которая поддерживает графовую модель данных, а также документную и ключ-значение модели. Основные возможности графовой части ArangoDB включают:

– Мульти-модельность: ArangoDB поддерживает не только графовую модель данных, но и документную (JSON) и ключ-значение модели. Это позволяет разработчикам выбирать наиболее подходящую модель для конкретных задач или даже использовать комбинацию различных моделей данных в рамках одной базы данных.

– Гибкий язык запросов: Для работы с графовыми данными в ArangoDB используется язык запросов AQL (ArangoDB Query Language), который поддерживает выполнение сложных запросов к графам. AQL обеспечивает возможность выполнять операции поиска путей, фильтрации, агрегации данных и многих других операций над графами.

– Хранение данных графа: Данные в графовой модели ArangoDB представлены как вершины (узлы) и ребра (связи) между ними. Это позволяет эффективно моделировать и анализировать сложные сетевые структуры и зависимости между данными.

– Транзакционная поддержка: ArangoDB обеспечивает поддержку транзакций для графовых операций, что позволяет выполнение множественных операций атомарно и обеспечивает целостность данных в графовой структуре.

– Высокая производительность и масштабируемость: Благодаря своей архитектуре, ArangoDB обеспечивает высокую производительность и масштабируемость. Она поддерживает как горизонтальное, так и вертикальное масштабирование, позволяя обрабатывать большие объемы данных и запросов.

– Графовые алгоритмы и аналитика: ArangoDB предлагает встроенные графовые алгоритмы, такие как поиск кратчайших путей, поиск подграфов, анализ центральности и другие. Эти алгоритмы позволяют проводить анализ данных и выявлять важные паттерны и структуры в графовых данных.

– Открытость и экосистема: ArangoDB является open-source продуктом с активным сообществом и разнообразными интеграционными возможностями. Она интегрируется с различными технологиями и языками программирования, что облегчает разработку и внедрение приложений, использующих графовые данные.

– Поддерживает визуализацию через ArangoDB Web UI и сторонние инструменты, такие как ArangoDB Oasis, которые предоставляют графический интерфейс для визуализации данных. Возможность визуализации зависит от используемого интерфейса и инструментов.

ArangoDB подходит для широкого спектра приложений, где важно учитывать и анализировать сложные взаимосвязи между данными.

Хранение и визуализация семантических деревьев

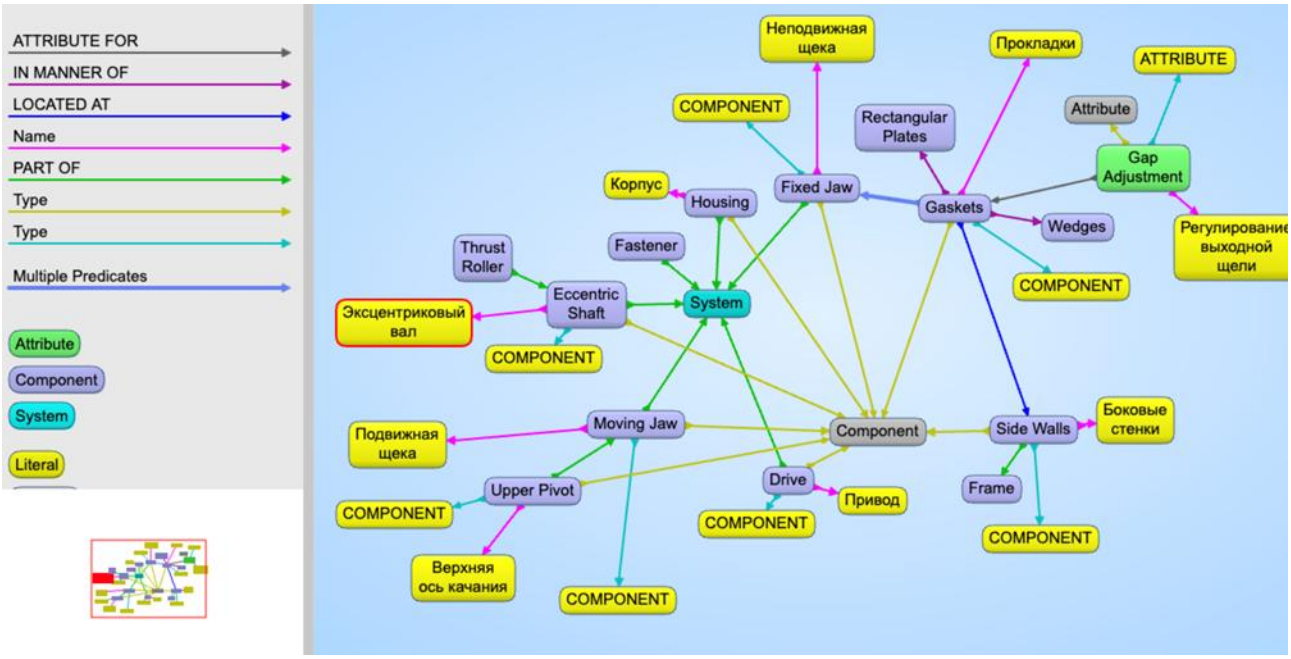

Рис. 2. – Графовое дерево, визуализированное при помощи СУБД AllegroGraph

Пример кода для внесения информации семантического дерева в СУБД AllegroGraph.

```
INSERT DATA {
  # Сущности
  ex:system a ex:System ;
    ex:name "Щековая дробилка" ;
    ex:type "SYSTEM" .

  ex:movingJaw a ex:Component ;
    ex:name "Подвижная щека" ;
    ex:type "COMPONENT" .
}
  # Связи
  ex:movingJaw ex:partOf ex:system .
  ex:upperPivot ex:partOf ex:movingJaw .
  ex:eccentricShaft ex:partOf ex:system .
…
  ex:gaskets ex:locatedAt ex:fixedJaw .
  ex:gaskets                    ex:inMannerOf
ex:rectangularPlates .
```

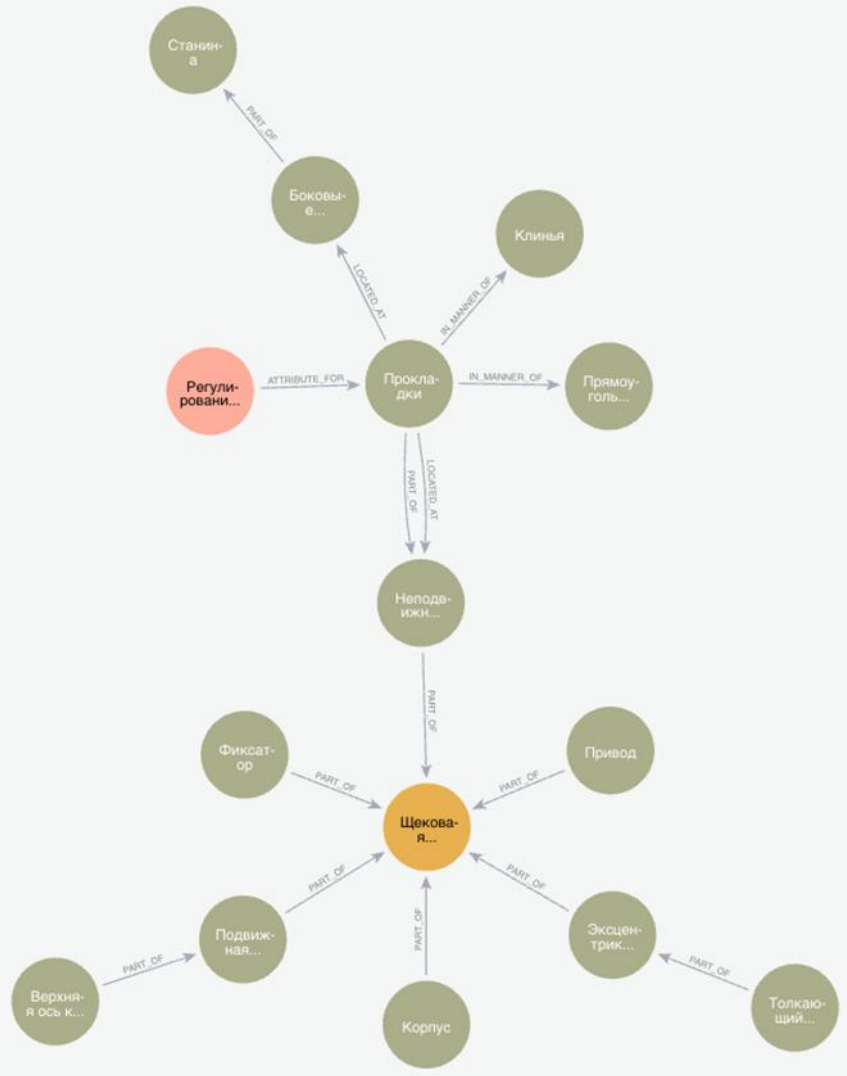

Рис. 3. – Графовое дерево, визуализированное при помощи СУБД Neo4j

Пример кода для внесения информации семантического дерева в СУБД Neo4j.

```
// Создание сущностей
CREATE (system:System {name: 'Щековая дробилка', type: 'SYSTEM'})
CREATE (movingJaw:Component {name: 'Подвижная щека', type: 'COMPONENT'})
CREATE (upperPivot:Component {name: 'Верхняя ось качания', type: 'COMPONENT'})
CREATE (eccentricShaft:Component {name: 'Эксцентриковый вал', type: 'COMPONENT'})
CREATE (thrustRoller:Component {name: 'Толкающий ролик', type: 'COMPONENT'})
…
CREATE (gapAdjustment:Attribute {name: 'Регулирование выходной щели', type: 'ATTRIBUTE'})
```

```
// Создание связей
CREATE (movingJaw)-[:PART_OF]->(system)
CREATE (upperPivot)-[:PART_OF]->(movingJaw)
CREATE (housing)-[:PART_OF]->(system)
…
CREATE (gaskets)-[:LOCATED_AT]->(fixedJaw)
CREATE (gaskets)-[:IN_MANNER_OF]->(rectangularPlates)
CREATE (gaskets)-[:IN_MANNER_OF]->(wedges)
CREATE (gaskets)-[:LOCATED_AT]->(sideWalls)
CREATE (sideWalls)-[:PART_OF]->(frame)
CREATE (gapAdjustment)-[:ATTRIBUTE_FOR]->(gaskets)
```

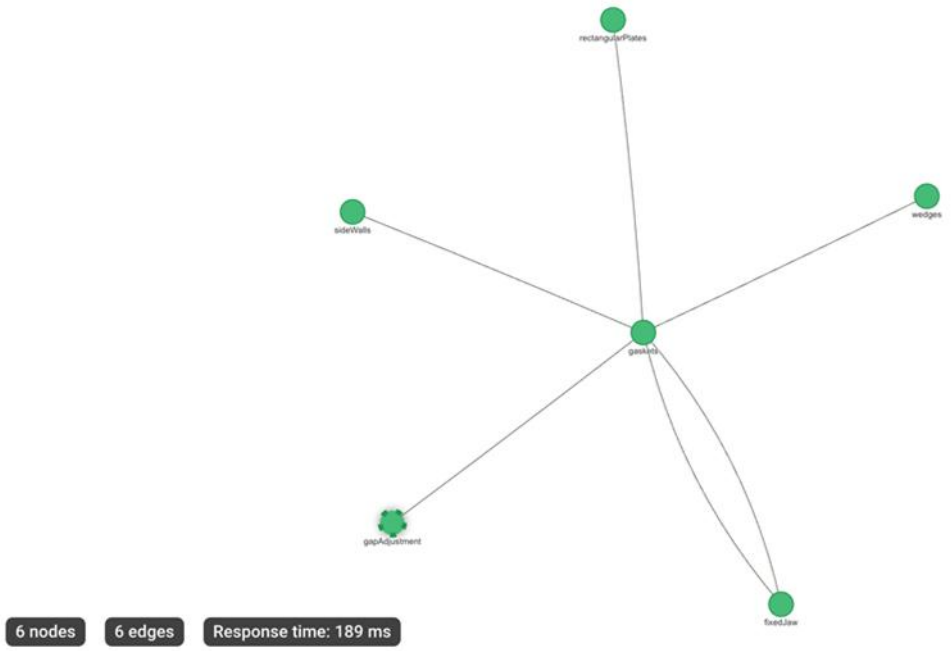

Рис. 4. – Графовое дерево, визуализированное при помощи СУБД ArangoDB

Пример кода для внесения информации семантического дерева в СУБД ArangoDB.

```
INSERT {
  "_key": "system", "name": "Щековая дробилка", "type": "SYSTEM"
} INTO nodes
INSERT {
  "_key": "movingJaw", "name": "Подвижная щека", "type": "COMPONENT"
```

```
INSERT {
  "_from": "nodes/movingJaw", "_to": "nodes/system", "type": "PART_OF"
} INTO edges
INSERT {
  "_from": "nodes/upperPivot", "_to": "nodes/movingJaw", "type": "PART_OF"
```

```
} INTO nodes
INSERT {
  "_key": "upperPivot", "name":
"Верхняя ось качания", "type":
"COMPONENT"
} INTO nodes
…
INSERT {
  "_key": "gapAdjustment", "name":
"Регулирование выходной щели", "type":
"ATTRIBUTE"
} INTO nodes
```

```
} INTO edges
…
INSERT {
  "_from":   "nodes/gaskets",   "_to":
"nodes/sideWalls",            "type":
"LOCATED_AT"
} INTO edges
INSERT {
  "_from":  "nodes/sideWalls",  "_to":
"nodes/frame", "type": "PART_OF"
} INTO edges
```

Алгоритмы сравнения графовых данных

Алгоритм 1: Graph Edit Distance

Graph Edit Distance (GED) [7] - метрика, используемая для измерения сходства между двумя графами. Она определяется как минимальное количество операций редактирования (удаление узлов, добавление узлов, удаление ребер, добавление ребер) для преобразования одного графа в другой.

Принцип работы:

1. Инициализация: Задаются два графа, которые необходимо сравнить.

2. Редактирование: На каждом шаге определяется, какая операция редактирования (удаление, добавление или замена узла/ребра) минимизирует различия между графами.

3. Подсчет: Общее количество операций редактирования, необходимых для преобразования одного графа в другой, определяет Graph Edit Distance.

Алгоритм 2: Topological Comparison

Топологическое сравнение [8] фокусируется на структурных особенностях графов, таких как количество узлов, количество ребер, типы сущностей и типы связей. Оно не требует сложных вычислений и предоставляет более интуитивное сравнение.

Принцип работы:

1. Извлечение характеристик: для каждого графа извлекаются основные характеристики: количество узлов, количество ребер, типы сущностей и типы связей.

2. Сравнение характеристик: сравниваются извлеченные характеристики обоих графов. Если они совпадают, графы считаются топологически равными.

Алгоритм 3: Subgraph Isomorphism

Изоморфизм подграфов [9] определяет, можно ли один граф рассматривать как подграф другого, то есть существует ли отображение одного графа на подграф другого, сохраняя структуры и связи.

Принцип работы:

1. Инициализация: задаются два графа, где один рассматривается как возможный подграф другого.

2. Поиск отображения: алгоритм ищет отображение узлов и ребер одного графа на подграф другого графа.

3. Проверка изоморфизма: если отображение найдено, графы считаются изоморфными.

Заключение

После детального изучения и сравнения графовых баз данных Neo4j, ArangoDB и AllegroGraph, можно сделать следующие выводы относительно их возможностей по визуализации и хранению графов:

– Neo4j демонстрирует наилучшие возможности для визуализации графов. Благодаря таким инструментам, как Neo4j Browser и Neo4j Bloom, пользователи могут легко и интуитивно исследовать свои данные. Эти инструменты позволяют выполнять запросы, моментально видеть результаты и проводить визуальный анализ графов, что делает Neo4j отличным выбором для тех, кто ценит удобство и наглядность в работе с графовыми данными.

– Визуализация графов в ArangoDB менее совершенна. Хотя ArangoDB предоставляет инструменты для визуализации через ArangoDB Web UI и ArangoDB Oasis, они не всегда позволяют полностью и удобно отображать сложные графовые структуры. Пользователям может не хватать некоторых функций и возможностей для глубокого анализа и наглядного представления данных, которые имеются в Neo4j.

– AllegroGraph, несмотря на свои мощные возможности для работы с RDF-данными и онтологиями, сталкивается с трудностями в визуализации. Инструменты визуализации, такие как Gruff, могут показаться сложными для использования, особенно при работе с многочисленными типами связей и сущностей. Это может привести к путанице и затруднению в понимании и анализе данных. Пользователи могут испытывать трудности в правильной настройке и отображении графов, что делает работу с AllegroGraph менее интуитивной по сравнению с Neo4j и ArangoDB.

Благодарности

Исследование выполнено за счет гранта Российского научного фонда № 24-21-20140, rscf.ru/project/24-21-20140/ , и Администрации Волгоградской области.

Литература (References)

1. Besta M., Gerstenberger R., Peter E., Fischer M., Podstawski M., Barthels C., Alonso G., Hoefler T. Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries. ACM Computing Surveys, 2023. DOI: 56. 10.1145/3604932.

2. Vasiliev S., Korobkin D., Fomenkov S. "Extracting the Component Composition Data of Inventions from Russian Patents using Dependency Tree Analysis," 2023 International Conference on Industrial Engineering, Applications

and Manufacturing (ICIEAM), Sochi, Russian Federation, 2023, pp. 1030-1034, DOI: 10.1109/ICIEAM57311.2023.10139170.

3. Korobkin D., Fomenkov S., Vasiliev S., Kolesnikov S.G. Methods of Russian Patent Analysis. Statistical Methodologies, IntechOpen, 2020. DOI: 10.5772/intechopen.90136

4. Holzschuher F., Peinl R. Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j. ACM International Conference Proceeding Series, 2013. DOI: 10.1145/2457317.2457351.

5. Fernandes D., Bernardino J. Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. Proceedings of the 7th International Conference on Data Science, Technology and Applications, 2018, pp. 373-380. DOI: 10.5220/0006910203730380.

6. Belgundi R., Kulkarni Y., Jagdale B. Analysis of Native Multi-model Database Using ArangoDB. Lecture Notes in Networks and Systems. vol 587, 2023. DOI: 10.1007/978-981-19-7874-6_68.

7. Serratosa F. Redefining the Graph Edit Distance. SN Computer Science. SN COMPUT. SCI. 2, 438, 2021. DOI: 10.1007/s42979-021-00792-5.

8. Qin Y., Fasy B., Wenk C., Summa B. Rapid and Precise Topological Comparison with Merge Tree Neural Networks. IEEE transactions on visualization and computer graphics, 2024. DOI: 10.1109/TVCG.2024.3456395.

9. Ullmann J. An Algorithm for Subgraph Isomorphism. Journal of the ACM (JACM). vol. 23, 1976, pp. 31-42. DOI: 10.1145/321921.321925.

Дата поступления: 22.09.2024

Дата публикации: 31.10.2024