СРАВНЕНИЕ СУБД ДЛЯ ХРАНЕНИЯ СТРУКТУР ДРЕВОВИДНЫХ ДАННЫХ

Мельник А. А.
Кафедра информационных технологий автоматизированных систем,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: a.melnik@bsuir.by

В статье рассматривается проблема эффективного хранения древовидных структур данных. Проводится сравнительный анализ методов решения этой задачи в реляционных (PostgreSQL) и нереляционных (MongoDB, Neo4j) базах данных. На основе практических замеров скорости выполнения запросов делается вывод, что графовая СУБД Neo4j является наиболее производительным решением благодаря своей природе, идеально соответствующей работе со связанными данными.

Введение

Современные информационные системы повсеместно используют древовидные структуры для организации иерархических данных. Эти структуры позволяют эффективно представлять отношения вложенности в таких областях, как бухгалтерский учет, управление организационными структурами, построение категорий товаров и социальных графов. С увеличением сложности и объемов данных задача оптимального хранения и обработки древовидных структур становится особенно актуальной.

Традиционные реляционные подходы к хранению иерархических данных часто сталкиваются с проблемами производительности при работе с глубокими древовидными структурами. Это стимулирует поиск альтернативных решений среди нереляционных баз данных, которые предлагают специализированные модели хранения и эффективные алгоритмы обхода. Каждый из подходов имеет свои преимущества и ограничения, требующие тщательного анализа.

Проведение сравнительного исследования различных методов хранения древовидных структур представляется важной задачей. Такой анализ должен охватывать как реляционные, так и нереляционные подходы и учитывать требования современных приложений к производительности и масштабируемости. Результаты исследования позволят выработать практические рекомендации по выбору оптимальной стратегии хранения иерархических данных.

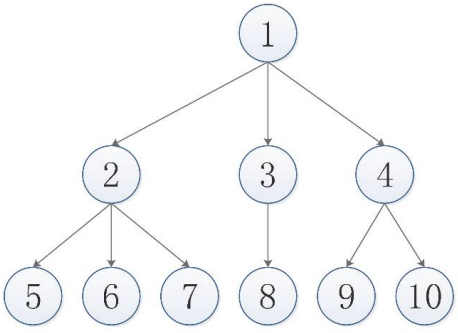

Рис. 1 – Пример дерева

I. Хранение в реляционных БД

Классические реляционные подходы могут сталкиваться с проблемами производительности при работе с глубокими иерархиями из-за необходимости использования множественных операций JOIN, что особенно заметно в системах без эффективного горизонтального масштабирования [1].

В реляционных БД можно выделить четыре способа хранения древовидной структуры данных: Список смежности (Adjacency List), Таблица связей (Closure Table), Вложенные множества (Nested sets), Материализованный путь (Materialized Path).

Список смежности позволяет хранить родительский узел для каждой вершины дерева. Для этого в таблицу добавляется еще одно поле, в котором хранится id родителя. Основная идея способа «Таблица связей» состоит в создании дополнительной таблицы для хранения id родителя и потомка. Основной недостаток состоит в том, что чем глубже в дереве находится узел, тем больше записей в таблице потребуется для описания связей, однако это позволяет обеспечить быстрый поиск. Материализованный путь позволяет хранить полный путь от корня дерева до нужного узла. Чтобы найти все дочерние элементы определенного узла, необходимо использовать LIKE с %, который работает медленнее, чем =, но в большинстве случаев этого достаточно. В схеме вложенных множеств каждый узел имеет присвоенные левые и правые значения, указывающие на его место в дереве.

II. Хранение в нереляционных БД

В нереляционных БД чаще всего используют: Array of Ancestors – сохраняет ссылки на «родительские» узлы и массив, в котором хранятся все предки [2], Materialized Path, Parent References – организует документы в древовидную структуру путем хранения ссылок на «родительские» узлы в «дочерних» узлах, Child References – организует документы в древовидную структуру путём

хранения ссылок на «дочерние» узлы в «родительских» узлах., Nested Sets.

Примерами систем управления нереляционными БД являются MongoDB и Neo4j. Neo4j – это графовая система управления базами данных с открытым исходным кодом, реализованная на Java. Она является ведущей графовой СУБД в мире. Для базы данных Neo4j был создан декларативный язык запросов Cypher, обеспечивающий эффективное чтение и запись данных в Neo4j. MongoDB – система управления базами данных, которая работает с документоориентированной моделью данных. В отличие от реляционных СУБД, MongoDB использует гибкую документо-ориентированную модель, не требующую строгой схемы, и обладает собственным богатым языком запросов.

III. Практическое исследование

В результате замеров можно увидеть, что наиболее эффективной и удобной СУБД для хранения древовидных данных является Neo4j – графовая СУБД. Это обусловлено тем, что объекты в такой базе данных связаны между собой как узлы в графах [3]. MongoDB же является документоориентированной базой данных, которая не была специально разработана для работы с древовидными структурами данных [4].

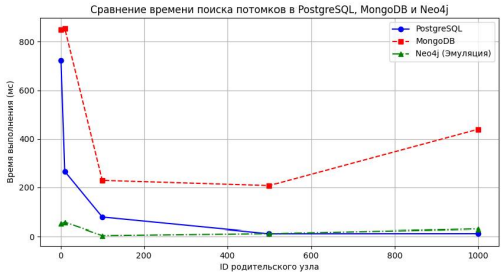

Рис. 2 – Результат замера скорости выполнения запросов поиска в PostgreSQL, MongoDB, Neo4j

Анализ результатов производительности выявил четкую закономерность: с увеличением глубины дерева и сложности запросов преимущество Neo4j становится более выраженным. В то время как реляционные подходы в PostgreSQL демонстрировали прогрессивное замедление при обработке многоуровневых иерархий [5], а MongoDB показывала средние результаты из-за необходимости выполнения дополнительных операций агрегации, графовая модель Neo4j обеспечивала стабильно высокую скорость выполнения даже для сложных рекурсивных запросов. Это подтверждает теоретическое положение о том, что специализированные графовые базы данных наиболее адекватно отвечают требованиям работы с древовидными структурами.

Заключение

Проведенное исследование позволяет сделать однозначный вывод о том, что выбор системы управления базами данных для работы с древовидными структурами напрямую определяет производительность и масштабируемость приложения.

Сравнительный анализ реляционных и нереляционных подходов показал, что классические реляционные СУБД, такие как PostgreSQL, несмотря на множество схем организации иерархических данных (список смежности, вложенные множества и др.), сталкиваются с фундаментальными ограничениями. К ним относятся низкая производительность операций JOIN, сложности с горизонтальным масштабированием и высокая нагрузка на сервер при работе с глубокими иерархиями.

Практические замеры скорости выполнения запросов подтвердили, что специализированные графовые базы данных демонстрируют превосходство в данной задаче. Neo4j показала себя как наиболее эффективное решение благодаря своей нативной модели данных, где связи между узлами являются объектами первого класса. Это позволяет ей напрямую и быстро обходить древовидные структуры, избегая дорогостоящих операций, присущих реляционным подходам.

Таким образом, для проектов, где работа с иерархическими и связанными данными является ключевой, графовые СУБД, и в частности Neo4j, следует рассматривать как оптимальный выбор, обеспечивающий высокую производительность и удобство разработки. В то время как документоориентированные СУБД, подобные MongoDB, предлагают гибкость, их использование для сложных древовидных структур может быть менее эффективным по сравнению со специализированными графовыми системами.

1. Maddipatla S. P. et al. Using databases to implement algorithms: Estimation of allan variance using b+-tree data structure //2024 American Control Conference (ACC). – IEEE, 2024. – С. 2164-2169.
2. Pan J. J., Wang J., Li G. Survey of vector database management systems // The VLDB Journal. – 2024. – Т. 33. – № 5. – С. 1591-1615.
3. Hodler A. E., Needham M. Graph data science using Neo4j //Massive Graph Analytics. – Chapman and Hall/CRC, 2022. – С. 433-457.
4. Dhanagari M. R. MongoDB and data consistency: Bridging the gap between performance and reliability // Journal of Computer Science and Technology Studies. – 2024. – Т. 6. – №. 2. – С. 183-198.
5. Makris A. et al. MongoDB Vs PostgreSQL: A comparative study on performance aspects // GeoInformatica. – 2021. – Т. 25. – № 2. – С. 243-268.