Научная статья
Original article
УДК 004.652

СРЕДСТВА ФИЗИЧЕСКОГО ХРАНЕНИЯ МЕТАГРАФОВОЙ МОДЕЛИ ДАННЫХ

PHYSICAL STORAGE OF METAGRAPH DATA MODEL

Дунин Иван Владимирович, магистр, аспирант кафедры ИУ-5 МГТУ им. Н.Э. Баумана, Московский Государственный Технический Университет им. Н.Э. Баумана, г. Москва, 2-я Бауманская ул., д. 5, стр. 1, почтовый индекс: 105005, тел. 8(909) 694-84-91, johnmoony@yandex.ru

Ivan V. Dunin, master of Computer Science, PhD student of Computer Science and Control Systems Department at Bauman Moscow State Technical University, Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow, postcode: 105005, tel. 8(909) 694-84-91, johnmoony@yandex.ru

Аннотация. В статье рассматривается метаграфовая модель данных. Описываются существующие средства хранения простых графов, гиперграфов. Дается описание метаграфовой модели с точки зрения физического хранения. Предлагается способ сохранения метаграфовой модели.

Abstract. The artilce describes metagraph data model. Existing storage solutions for simple graphs and hypergaphs are discussed. Storage of metagraph model is discussed. Method for metagraph storage is presented.

Ключевые слова: метаграф, метавершина, графовая база данных, гиперграф

Keywords: metagraph, metavertex, graph database, hypergraph

Введение

В настоящее время активно развиваются технологии хранения и обработки знаний, представленных в виде графов. Они используются в различных областях – от промышленных предприятий до анализа социальных сетей [1]. Широкое распространение получили и средства хранения графов – специализированные графовые СУБД и графовые надстройки в традиционных СУБД [2]. Существует ограниченный круг решений для работы с более сложными моделями – гиперграфами. Наиболее универсальная из графовых моделей, метаграф, пока не имеет специализированных инструментов для хранения. Ниже мы рассмотрим существующие решения в сфере баз данных для различных графовых моделей, а также рассмотрим возможности по физическому хранению метаграфов.

Графовые СУБД

Система управления базами данных графа (графовая база данных) - это онлайн-система управления базами данных с методами CRUD (create, remove, update, delete - создания, чтения, обновления и удаления), которая предоставляет графовую модель данных.

Графовые базы данных позволяют осуществлять запросы по обходу графа по связям между его вершинами, а также предоставляют графовые алгоритмы для поиска путей, расстояний, шаблонов, групп вершин на графах [3].

Графовые базы данных обычно создаются для использования с транзакционными системами, или OLTP-системами (Online Transaction Processing). Соответственно, они обычно оптимизированы для транзакционной работы и разработаны с учетом целостности транзакций и операционной доступности. Примеры графовых СУБД: Neo4j, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, Titan. Графовые базы данных как правило поддерживают ACID-транзакции, а также имеют собственные языки запросов для работы с графовыми данными, такие как Gremlin и Cypher (Neo4j).

Опишем характерные особенности реализации графовых баз данных: рассмотрим, чем они отличаются от других средств хранения и обработки сложных, непостоянных по структуре, сильно связанных данных. Несмотря на то, что сама модель достаточно схожа во всех реализациях графовых баз данных,

существует множество способов кодирования и представления графа в основной памяти движка базы данных. Не существует единого универсального шаблона архитектуры даже среди графовых баз данных, поэтому мы опишем распространенные архитектурные шаблоны и компоненты, которые можно найти в графовой базе данных.

Общей особенностью графовых СУБД является то, что они используют на физическом уровне способ представления данных, отражающий логическую (графовую) структуру, что позволяет использовать их наиболее эффективно для решения задач на графах по сравнению с традиционными решениями, например, реляционными СУБД. При использовании реляционной СУБД для моделирования графов потребуется создать таблицы для вершин и для ребер, а также соответствующие индексы (как минимум – индексы по полям начальной и конечной вершины в таблице ребер). При решении такой задачи как, например, поиск в глубину на графе, каждое обращение к следующей вершине будет означать поиск по индексу. В случае использования B-tree подобных индексов мы получим сложность O(m log n), где n – число вершин, m – число шагов в поиске. Это особенно существенно в таких ситуациях, как расширяющийся поиск нескольких уровней соседей какой-либо вершины (например, на графе социальных связей или на графе сетевой топологии). Графовые СУБД в первую очередь направлены на оптимизацию физического хранения для решения подобных задач.

Однако стоит отметить, что описанная проблема сложности задачи поиска теоретически решается и в реляционной СУБД при использовании не B-tree индекса с логарифмической сложностью, а хеш-индекса с временем поиска O(1).

Характерной архитектурной особенностью графовых баз данных часто считают свойство, называемое смежностью без индекса [4]. СУБД, в которой используется смежность без индекса - это та, в которой каждый узел поддерживает прямые ссылки на свои смежные узлы. Следовательно, каждый узел действует как индекс соседних узлов. что намного эффективнее, чем использование глобальных индексов. Это означает, что время запроса не зависит от общего размера графа, а просто пропорционально размеру искомого графа.

Противоположностью этого подхода является использование глобальных индексов для связи узлов. Это значит, что при каждом обращении к соседнему узлу (смежной вершине графа) происходит поиск по индексу. Как и в описанном выше случае с использованием реляционных СУБД, при использовании обычных индексов это означает сложность обхода графа O(m log n), что скорее всего является неприемлемым для решения типичных задач на графах.

Использование индексов для моделирования связей становится проблематичным, когда требуется обход в противоположном направлении от того, для которого был создан индекс. В этом случае требуется или создавать обратный индекс для каждого сценария обхода, или выполнять перебор по исходному индексу, что является операцией стоимостью O(n), что невозможно в случае практического применения в онлайн-системах. В графовых базах данных, очевидно, требуется возможность свободного обхода графа в любом направлении с одинаковой производительностью.

На практике производительность может быть еще хуже из-за того, что граф и его глобальные индексы конкурируют за ресурсы, такие как кэши и ввод-вывод (например, когда возникает конфликт страниц между данными индекса и графа).

Отметим, что графовые СУБД имеют и свои ограничения. Как и большинство нереляционных СУБД, графовые базы данных являются ориентированными на конкретные задачи, другими словами, лишены универсальности. В частности, нередко отмечают, что графовые базы данных имеют более низкую производительность на типичных “табличных” операциях, например, при поиске среди всех вершин графа по значению поля (аналогично SQL-оператору SELECT … FROM … WHERE ...), сортировке большого количества записей, при операциях, выполняющих агрегирование (аналогично SQL-оператору GROUP BY).

Кроме того, для графовых СУБД особенно актуален вопрос партиционирования при размещения базы данных на нескольких узлах [5]. При размещении сильно связанных подграфов на одной машине использование внешней памяти кластера серверов будет неравномерным. При равномерном размещении возрастает время обхода графа. Оптимальное распределение вершин требует либо проверок при каждой вставке, либо периодических процедур

дефрагментации. Помимо вопросов производительности, это накладывает жесткие ограничения на максимальный размер базы данных. Обозначенная проблема, впрочем, в большей степени вызваны спецификой задач, решаемых с помощью графовых баз данных, чем их архитектурой. При шардировании реляционного представления графовых данных придется решать аналогичную проблему.

Гиперграфовая модель

Графовая модель имеет ряд обобщений, таких как гиперграф [6] и гиперсеть [7]. В гиперграфовой модели допускается наличие ребер, соединяющих более двух вершин – так называемых гиперребер. Гиперграфовая модель может использоваться для моделирования сложных промышленных процессов или химических реакций.

Из существующих систем в области хранения гиперграфов стоит упомянуть проект HyperGraphDB [8], представляющий собой базу данных для хранения гиперграфов. Единицей хранения в HyperGraphDB является кортеж, состоящий в свою очередь из нескольких кортежей. Авторы HyperGraphDB называют ее модель данных аналогичной реляционной, но с возможностью n-арных отношений, либо графовой, в которой ребра могут указывать на несколько вершин сразу (что в целом соответствует приведенному ранее формальному описанию гиперграфа). HyperGraphDB работает на основе key-value хранилища BerkeleyDB. Также как в рассмотренной ранее графовой базе данных Neo4j, HyperGraphDB использует раздельные хранилища для топологии гиперграфа и для данных – это соответственно хранилище LinkStore, в котором хранятся соответствия *ID -> [ID, … , ID]* (иными словами, гиперребра) и DataStore, в котором хранятся данные вершин гиперграфа в виде соответствий *ID -> byte[]*. Это то, что непосредственно помещается в key-value хранилище. Здесь ID – это идентификаторы вершин, 4-байтные UUID. Над уровнем хранения (LinkStore и DataStore) находится уровень модели данных. В HyperGraphDB и вершины, и гиперребра объединены в общую концепцию «атомов». Атом – это кортеж, содержащий «нагрузку» - 0 (для вершин) и более (для гиперребер) атомов. Уровень модели данных работает с примитивами атомов. Атомы определяются как:

AtomID -> [TypeID, ValueID, TargetID, ..., TargetID]

TypeID := AtomID

TargetID := AtomID

Т.е. атом является кортежом из идентификатора типа, идентификатора значения и нескольких идентификаторов других атомов. Также, как видно из этого определения, типы также являются атомами, что позволяет делать связи между типами. Авторы приводят в качестве возможного примера связанных типов типы «Покупка» и «Продажа». Кроме того, это делает систему типов HypegraphDB независимой от системы типов используемой платформы (СУБД или среды разработки – Java в случае HypegraphDB). В этой концепции можно увидеть аналогию с метаграфами.

Значением в атоме могут быть данные (byte) или кортеж идентификаторов:

ValueID -> [ID, …, ID] | byte[]

Также используется специфическое для модели данных индексирование – индекс атомов (AtomID), индекс типов (TypeID) и индекс значений вершин.

Помимо гиперграфовой модели данных HyperGraphDB также включается API для базовой работы с графами и реализацию основных графовых алгоритмов, а также специфический для гиперграфов обход с переходом между уровнями (hyper-traversal). Архитектура хранилища и индексов спроектирована с целью оптимизации обхода графов.

Авторы подчеркивают, что их проект не является графовой базой данных и по классификации находится ближе к объектным базам данных, поскольку предоставляет более гибкую модель данных. Также HyperGraphDB поддерживается типизацию хранимых данных. К особенностям данной системы авторы относят также возможности гибкой настройки – настраиваемые механизмы индексирования и типизации, настраиваемые алгоритмы партиционирования и возможность замены базового хранилища, на котором работает система.

Определение метаграфа

Наиболее общей и универсальной из существующих графовых моделей является метаграфовая модель. Впервые она была предложена в работе Базу и Блэннинга [9].

Рассмотрим формальное описание метаграфа, приведенное в работе [10].

$$MG=(V,VM,E)$$

Где V — множество вершин метаграфа, VM — множество метавершин метаграфа, E — множество ребер метаграфа.

Каждая вершина метаграфа может характеризоваться произвольным числом атрибутов:

$$v_i = \{atr_k\}, v_i \in V$$

Тогда ребро метаграфа можно описать следующим образом.

$$e_i = \left(V_S, V_E, eo, \{atr_k\}\right), e_i \in E, eo=true,false$$

Где e_i — метавершина метаграфа, V_S — вершина-источник (метавершина), V_E — вершина-приемник, eo — флаг направленности ребра, atr_K — атрибут.

Фрагмент метаграфа:

$$MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV)$$

Где ev — элемент, принадлежащий множеству вершин, ребер или метавершина.

Метавершина метаграфа:

$$mv_i = \left(\{atr_k\}, MG_j\right), mv_i \in MV$$

Где atr_k — атрибут, MG_i — фрагмент метаграфа.

Таким образом метавершина, помимо атрибутов, включает фрагмент метаграфа. Наличие у метавершины скрытых атрибутов и связей является отличительной особенностью метаграфов [10].

Специализированных решений по хранению метаграфов на данный момент не существует. Существует несколько способов физического отображения метаграфа. Нам представляется, что наиболее эффективным методом является преобразование метаграфа к простому графу, как показано на рисунке 1.

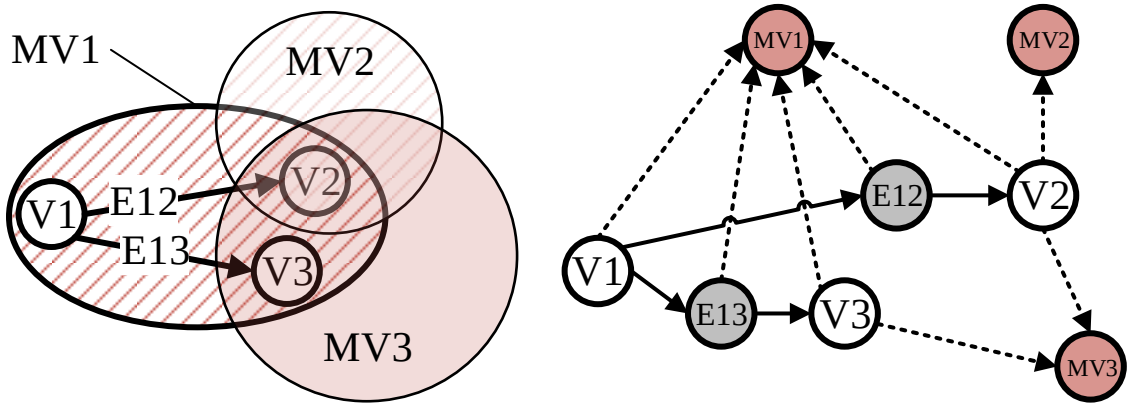


Рисунок 1 – Графовое представление метаграфа

Здесь вершины, метавершины и ребра графа представляются самостоятельным вершинами графа. Это отражает гибкость метаграфовой модели – как видно из рисунка, вершины и ребра могут независимо входить в родительские метавершины, что соответствует модели метаграфа.

Основной особенностью такого способа отображения является то, что оно дает возможность напрямую сохранять метаграф в графовой СУБД. Поскольку метаграф является частным случаем графа, использование графовой СУБД представляется целесообразным, так как позволит использовать графовые языки запросов и реализации графовых алгоритмов, существующие в графовых СУБД.

Заключение

В настоящее время получили распространение графовые СУБД, которые обеспечивают производительную обработку простых графов. Также существуют некоторые решения для хранения данных в модели гиперграфа. Существующих решений по хранению метаграфовой модели на данный момент не существует. Представляется, что в настоящее время самым эффективным способом хранения

метаграфа является преобразования метаграфа в простой граф и сохранение в графовой СУБД.

Литература

1. D. Reut, S. Falko, E. Postnikova. About scaling of controlling information system of industrial complex by streamlining of big data arrays in compliance with hierarchy of the present lifeworlds. International Journal of Mathematical, Engineering and Management Sciences, 4(5):1127-1139, October 2019.
2. N. Marz, J. Warren. Big Data. Principles and best practices of scalable realtime data systems. Manning, New York, 2015
3. Ian Robinson, Jim Webber. Graph Databases, O'Reilly Media, Inc., 2015 – 238 p.
4. Gaurav Vaswani, Anuradha Bhatia. Graph Databases - An Overview, International Journal of Computer Science and Information Technologies, Vol. 5 (1) , 2014 – 4p.
5. Peter Macko, Margo Seltzer - Performance Introspection of Graph Databases, Harvard University, 2013 – 10 p.
6. V.I. Voloshin. Introduction to Graph and Hypergraph Theory. Nova Science Publishers, Inc, 2009.
7. J. Johnson. Hypernetworks in the Science of Complex Systems. London, Imperial College Press, 2013.
8. Официальная документация проекта HypergraphDB, режим доступа: http://www.hypergraphdb.org/ (дата обращения: 7.12.2021)
9. Basu A., Blanning R.: Metagraphs and Their Applications. Springer, Heidelberg (2007)
10. Черненький В.М., Гапанюк Ю.Е., Ревунков Г.И., Терехов В.И., Каганов Ю.Т. Метаграфовый подход для описания гибридных интеллектуальных информационных систем. Прикладная информатика. 2017. Т. 12. № 3 (69). С. 57-79.

References

1. D. Reut, S. Falko, E. Postnikova. About scaling of controlling information system of industrial complex by streamlining of big data arrays in compliance with

hierarchy of the present lifeworlds. International Journal of Mathematical, Engineering and Management Sciences, 4(5):1127-1139, October 2019.

2. N. Marz, J. Warren. Big Data. Principles and best practices of scalable realtime data systems. Manning, New York, 2015
3. Ian Robinson, Jim Webber. Graph Databases, O'Reilly Media, Inc., 2015 – 238 p.
4. Gaurav Vaswani, Anuradha Bhatia. Graph Databases - An Overview, International Journal of Computer Science and Information Technologies, Vol. 5 (1) , 2014 – 4p.
5. Peter Macko, Margo Seltzer - Performance Introspection of Graph Databases, Harvard University, 2013 – 10 p.
6. V.I. Voloshin. Introduction to Graph and Hypergraph Theory. Nova Science Publishers, Inc, 2009.
7. J. Johnson. Hypernetworks in the Science of Complex Systems. London, Imperial College Press, 2013.
8. HypergraphDB, available at: http://www.hypergraphdb.org/ (accessed: 7.12.2021)
9. Basu A., Blanning R.: Metagraphs and Their Applications. Springer, Heidelberg (2007)
10. Chernenkiy V.M., Gapanyuk Yu.E., Revunkov G.I., Terekhov V.I., Kaganov Yu.T. Metagrafoviy podhod dlya opisaniya gibridnyh intellectualnih informatsionnyh system [Metagraph approach for hybrid intelligent information systems description]. Prikladnaya Informatika – Journal of Applied Informatics, 2017, vol. 12, no. 3(69), pp. 57–79.

© Дунин И.В., 2022 Научно-образовательный журнал для студентов и преподавателей «StudNet» №1/2022.

Для цитирования: Дунин И.В., СРЕДСТВА ФИЗИЧЕСКОГО ХРАНЕНИЯ МЕТАГРАФОВОЙ МОДЕЛИ ДАННЫХ // Научно-образовательный журнал для студентов и преподавателей «StudNet» №1/2022.