# УДК 004.652

**Особенности преобразования метаграфа в модель плоского графа**

**Details of transforming a metagraph to a flat graph model**

1. **Дунин И.В. (Dunin I.V.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, johnmoony@mail.ru**
2. **Гапанюк Ю.Е. (Gapanyuk Yu.E.), доцент кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, gapyu@bmstu.ru**
3. **Ревунков Г.И. (Revunkov G.I.), доцент кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана (Bauman Moscow State Technical University), revunkov@bmstu.ru**

*Аннотация.* В статье рассматриваются модели физического хранения метаграфовой модели данных. Описывается преобразование иерархической метаграфовой модели к плоскому графу. Рассматриваются базовые операции над метаграфом и специфика их реализации при хранении метаграфа в виде плоского графа.

*Abstract.* The article discusses the models of physical storage of the metagraph data model. Transformation of metagraph model to flat graph model is covered. Basic metagraph operations and details of their usage in case of flat graph storage are described.

*Ключевые слова:* метаграф, метавершина, графовая база данных.

*Keywords:* metagraph, metavertex, graph database.

*Реферат.* Метаграфовая модель позволяет одновременно представлять связи между данными и их иерархические отношения. В связи с ростом объемов данных и увеличением сложности отношений, эта модель является перспективной. Одним из способов хранения и обработки метаграфов является преобразование к простому графу.

Метаграф состоит из метавершин, которые могут быть связаны ребрами с другими метавершинами, а также включать в себя другие метавершины и ребра. При преобразовании метаграфа к плоскому графу ребра метаграфа преобразуются в вершины плоского графа. Отношение принадлежности метавершины задается ребрами плоского графа.

Обработка метаграфа включает ряд характерных операций. Использование модели плоского графа и графовой СУБД позволяет получать вложенные вершины произвольной вложенности в одном запросе. При удалении метавершины должны удаляться связанные ребра и подвершины.

*Summary.* Metagraph model allows to describe both data connections and data hierarchical relations. Due to growth in data volumes and relations complexity, this model is promising. Transforming metagraph to simple graph is one of the ways to store and process it.

Metagraph includes metavertices, which can be interconnected via edges and include other metavertices and edges. When metagraph is transformed to a flat graph, edges of metagraph are transformed to graph vertices. Belonging to metavertex is stored as flat graph edge.

Metagraph processing includes several specific operations. Usage of flat graph model and graph database allows to fetch metavertex subvertices in a single query. Deletion of metavertex must lead to deletion of connected edges and subvertices.

1. **Введение**

По мере усложнения задач, решаемых при разработке программного обеспечения, возрастают требования к используемым моделям данных. Появление новых областей, таких как искусственный интеллект, веб-технологии, большие данные и распределенные вычисления требует разработки соответствующих решаемым задачам способов хранения данных. Использование неэффективных моделей данных может привести к неоправданному росту сложности программного обеспечения, а также негативно отразится на надежности и производительности разрабатываемых систем.

Примером модели данных, отвечающей требования функциональности современного программного обеспечения, является метаграфовая модель. Эта модель является обобщением графовой модели, использующейся в настоящее время во многих СУБД. Метаграф допускает хранения внутри каждой вершины произвольного количества вершин и связей с неограниченной вложенностью.

Графовые модели — один из основных способов представления данных. Эти модели приобрели особенную актуальность с распространением сети Интернет, поскольку многие данные, генерируемые пользователями сети, имеют естественное графовое представление. Характерным примером являются связи между пользователями в социальных сетях. Также графовые модели находят применение при обработке текстовой информации на естественных языках [1]. В большинстве практических реализаций используется именно простая модель с бинарными ребрами графа. Примерами СУБД, использующих такую модель, могут послужить Neo4j [2] и TitanDB [3]. Формальная модель метаграфа представлена в работе [4].

В статье рассматривается хранение метаграфа в графовой базе данных и описываются особенности такого подхода на примере базовых операций над метаграфом.

1. **Графовые СУБД**

Графовые базы данных применяются для задач, в которых данные имеют большое количество связей, например, социальные сети, выявление мошенничества, задачи по определению маршрутов и достижимости (в компьютерных сетях или для географических объектов). Примеры графовых СУБД: Neo4j, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, TitanDB. Графовые базы данных как правило поддерживают ACID, а также имеют собственные языки запросов для работы с графовыми данными, такие как Gremlin и Cypher (Neo4j).

Ключевой особенностью графовых СУБД является то, что они на физическом уровне хранят данные приближенно к их логической структуре. Вершины графа содержат физические ссылки на другие вершины, с которыми их связывают ребра графа.

Результаты экспериментов по сравнению производительности графовых и реляционных СУБД является противоречивыми. Практическое подтверждение превосходства графовой базы данных можно найти в результатах экспериментов, проводившихся в работах [5] (на примере сравнения СУБД MySQL и Neo4j). При этом в схожих экспериментах в работе [6] большую производительность показал MySQL. В целом можно заключить, что производительность базы данных, в особенности нереляционной (в виду ее априорно не универсальной схемы данных), существенно зависит от решаемой задачи.

1. **Преобразование метаграфа к плоскому графу**

Графовые базы данных представляются очевидным решением для хранения метаграфов, поскольку метаграфы являются графами. Однако для использования графовых БД для построения метаграфового хранилища в общем случае потребуется преобразование метаграфа к плоскому графу. На рис. 1 показано представление метаграфа в виде плоского графа.

|  |
| --- |
|  |
| Рис. 1. Представление метаграфа в виде плоского графа |

Главной задачей отображения метаграфа в плоский граф является задание отношения «вложенная вершина» - «родительская вершина» для метавершин и их потомков. Единственным способом такого отображения является использование ребер графа для задания принадлежности вершины к метавершине. Метавершина и каждый из ее потомков становятся отдельными вершинами плоского графа, а между ними возникают ребра. Такая модель естественным образом допускает возможность принадлежности вершины к нескольким родительским метавершинам.

Для представления ребер метаграфа можно было бы использовать собственные ребра плоского графа, однако здесь стоит учитывать, что для ребер метаграфа также задано отношение принадлежности к метавершине. Если две вершины входят в родительскую вершину, то ребро между ними может в нее не входить, или вообще входить в другую вершину. Поэтому ребра метаграфа должны представляться как самостоятельные вершины. Начальная и конечная вершина исходного ребра при этом связывается с вершиной-ребром ребрами плоского графа.

Пример представления метаграфа сложной конфигурации в виде плоского графа представлен на рис. 2. Здесь показан сложный случай с пересечением нескольких метавершин по потомкам и реализация этой модели на графе.

|  |
| --- |
|  |
| Рис. 2. Пример представления метаграфа сложной конфигурации в виде плоского графа |

Следует отметить, что для различения ребер, ведущих к вершинам-ребрам, и ребер, связывающих родительские и дочерние метавершины, необходимо наличие у этих ребер какого-либо атрибута-признака. Тогда при обходе графа можно будет различать перемещение по путям в графе от перемещений по дереву потомков метавершин. Большинство графовых СУБД обеспечивают такую возможность [7].

1. **Реализация операций над метаграфом**

Выше было рассмотрено преобразование метаграфа к модели плоского графа. Помимо вопросов хранения метаграфа в виде плоского графа, существенным является изучение операций, которые можно совершать над метаграфом. Далее мы рассмотрим основные операции над метаграфом и определим их специфику при использовании графовой модели.

**4.1. Добавление вершины**

Добавление новой вершины (не имеющей ребер и не входящей в метавершины) к метаграфу представляет собой добавление одиночной записи в базу данных. Эта операция не содержит отличительных черт, характерных только для метаграфой модели.

**4.2. Добавление ребра**

Добавление ребра происходит для двух вершин, уже существующих в метаграфе. В связи с необходимостью задания самостоятельной принадлежности ребра метаграфа метавершинам, каждое ребро должно описываться отдельной записью в базе данных.

**4.3. Редактирование вершины**

При редактировании вершины мы удаляем, добавляем или редактируем ее атрибуты. Так как ребра метаграфа могут иметь атрибуты так же, как и вершины, а ребра на физическом уровне эквивалентны вершинами, данная операция для этих двух сущностей идентична. Отметим, что графовые СУБД являются нереляционными и допускают хранение данных без фиксированной схемы. В реляционной СУБД для добавления нового или удаления существующего атрибута придется производить редактирование схемы всей базы данных целиком.

**4.4. Добавление вершины в метавершину**

В этой операции устанавливается отношение принадлежности между двумя вершинами, существующими в метаграфе, либо между ребром и вершиной (что с точки зрения физической модели данных эквивалентно). В графовой модели мы добавляем ребро между метавершиной и ее потомком с соответствующей меткой.

**4.5. Удаление из метавершины**

Это обратная операция по отношению к операции добавления вершины в метавершину – мы убираем принадлежность вершины или ребра метавершине. С точки зрения СУБД это эквивалентно удалению ребра, соединяющего метавершину и вложенный элемент.

**4.6. Получение дочерних вершин**

Так как степень вложенности метавершин неограничена, и вложенные в метавершину вершины также могут быть не атомарными, операция получения дочерних вершин представляет собой получение дерева, как показано на рис. 3.

|  |
| --- |
|  |
| Рис. 3. Дерево подвершин метавершины |

Данная операция показывает одно из ключевых преимуществ графовой модели и графовых СУБД – удобство кодирования запросов. Графовая СУБД позволяет получить одним запросом произвольное количество уровней потомков. Если мы имеем вершину А, которая содержит вершину В, которая содержит вершину С, то в документной или реляционной СУБД для получения всего пути А-С нам потребуется делать две операции соединения аналогичные JOIN. В графовых СУБД существует поддержка операции «получить пути от А по ребрам типа Х», которые возвращают обе вершины без явного задания количества переходов. Например, так выглядит запрос на получение всех потомков вершины с идендификатором node\_id по ребрам типа submeta на языке AQL для СУБД ArangoDB:

**MATCH (n: { \_id: '{node\_id}' }) <-[:submeta\*..]- (subnode) RETURN subnode**

Аналогичный запрос на языке Cypher для Neo4j:

**FOR n,e,p IN 1..{max\_depth} INBOUND '{node\_id}' NodesConnections FILTER p.edges[\*].submeta ALL == true RETURN n**

Здесь *NodesConnections –* коллекция, хранящая ребра графа СУБД. Заметим, что в вопросе получения потомков между Neo4j и ArangoDB есть различие – ArangoDB не позволяет искать потомков на произвольной глубине, язык AQL требует явного задания максимального количества переходов (через параметр *max\_depth*). В Neo4j задание верхней границы для пути по ребрам не требуется.

Следует отметить, что функциональность иерархических запросов встречается не только в графовых СУБД. Рекурсивные запросы на основе так называемых common table expressions является частью стандарта SQL:1999 [8]. Возможность делать запросы графового типа с неопределенным числом переходов встречается также и в отдельных документных СУБД, в частности MongoDB [9] поддерживает операцию graphLookup с аналогичной функциональностью.

**4.7. Получение родительских вершин**

Получение родительских вершин и получение дочерних вершин для метавершины являются зеркальными операциями. Метавершина может иметь произвольное число родителей и произвольное число потомков, поэтому и родители, и потомки метавершины образуют дерево. В графовых СУБД вершины, связанные ребрами, образуют двусвязный список, по которому можно равноправно перемещаться в обе стороны [10], что делает операции получения родителей и потомков эквивалентными.

**4.8. Удаление ребра**

Удаление ребра в метаграфе может происходить либо по идентификатору конечной и начальной вершины, либо по идентификатору самого ребра. Стоит отметить, что при удалении ребра по идентификатору вершин может возникать неопределенность в том случае, если две вершины связаны более чем одним ребром (что допускается в метаграфе). При удалении ребра очевидно также должно происходить удаление принадлежности ребра к метавершинам.

**4.9. Удаление вершины**

Удаление вершины в метаграфе связано с рядом трудностей. Во-первых, при удалении вершины, очевидно, необходимо удалить связанные с ней ребра. Отдельные СУБД поддерживают каскадное удаление связанных сущностей, однако в случае метаграфа это осложняется тем, что ребра хранятся как самостоятельные элементы. Поэтому при удалении вершины требуется явно удалить все связанные с ней ребра (в которых искомая вершина является концом или началом). Во-вторых, при удалении вершины мы должны удалить записи о принадлежности вершины родительским метавершинам. Схематично удаление метавершины представлено на рис. 4.

|  |
| --- |
|  |
| Рис. 4. Удаление метавершины. |

Основной сложностью при удалении метавершины является удаление ее содержимого. Необходимо поддерживать два режима удаления, «поверхностый» и «глубокий». Потребность в двух вариантах удаления связана с тем, что метавершина может делить содержимое с другими метавершинами. Также можно предположить, что содержимое вершины семантически может существовать самостоятельно.

В поверхностном варианте удаления метавершина удаляется отдельно от своего содержимого. При этом должны удаляться связи между удаляемой вершиной и первым уровнем ее потомков. Удаление связей с потомками аналогично описанному выше удалению связей исходной удаляемой вершины с ее родителями.

При глубоком удалении метавершина удаляется вместе со всеми вершинами и ребрами, входящими в нее. Фактически, мы получаем все дерево потомков вершины как в операции «получение содержимого» и для каждой из полученных вершин выполняем простое удаление вершины. Под простым удалением здесь понимается удаление вершины, удаление связанных с ней ребер и удаление связей с другими метавершинами.

При удалении связанных с вершиной ребер их также нужно удалять из метавершин, к которым они могут принадлежать. Если вершина MV1 включала вершины V1, V2 и ребро между ними E12, то при удалении вершины V2 мы удаляем не только запись о принадлежности V2 к MV1 и ребро E12, но и запись о принадлежности ребра E12 к MV1.

Заметим, что во всех вариантах удаления метавершины с содержимым, удаление самой вершины должны быть заключительной операцией, так как в противном случае мы потеряем возможность получить ее потомков, могут удалиться связи с ними. Графовая СУБД позволяет одним запросом к БД получить всех потомков, а затем для каждого выполнить последовательно удаление связей с метавершинами, удаление ребер, и наконец, удаление самой вершины.

1. **Выводы и дальнейшие направления работы**

Было рассмотрено преобразование метаграфовой модели к модели плоского графа, пригодной для хранения в графовой СУБД. Также были рассмотрены базовые операции над метаграфом и их специфика в контексте использования модели плоского графа и графовой СУБД. Продолжением сформулированных идей будет являться программная реализация метаграфового хранилища на базе графовой СУБД и его применение к решению практических задач.

**Литература**

1. **Martin Junghanns, André Petermann. Management and Analysis of Big Graph Data: Current Systems and Open Challenges, Leipzig University Database Research Group, 2016, 47 p.**

**2. Официальная документация проекта Neo4j, режим доступа:** [**https://neo4j.com/docs/**](https://neo4j.com/docs/) **(дата обращения: 21.09.2018)**

**3. Официальная документация проекта TitanDB, режим доступа:** [**http://s3.thinkaurelius.com/docs/titan/1.0.0/**](http://s3.thinkaurelius.com/docs/titan/1.0.0/) **(дата обращения: 21.09.2018)**

**4. Basu A., Blanning R.: Metagraphs and Their Applications. Springer, Heidelberg, 2007.**

**5. Ataky Tsham Mpinda, Lucas Cesar Ferreira. Evaluation of graph databases performace through index techniques, International Journal of Artificial Intelligence & Applications (IJAIA) Vol. 6, 2015 – 12 p.**

**6. Peter Macko, Margo Seltzer - Performance Introspection of Graph Databases, Harvard University, 2013 – 10 p.**

**7. Ian Robinson, Jim Webber. Graph Databases, O’Reilly Media, Inc., 2015 – 238 p.**

**8. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006. — 1328 с.**

**9. Операция graphLookup в MongoDB, режим доступа** [**https://docs.mongodb.com/manual/reference/operator/aggregation/graphLookup/**](https://docs.mongodb.com/manual/reference/operator/aggregation/graphLookup/) **(дата обращения 21.09.2018).**

**10. Gaurav Vaswani, Anuradha Bhatia. Graph Databases - An Overview, International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014 – 4p.**

**References**

1. **Martin Junghanns, André Petermann. Management and Analysis of Big Graph Data: Current Systems and Open Challenges, Leipzig University Database Research Group, 2016, 47 p.**

**2. Neo4j Project documentation, available at :** [**https://neo4j.com/docs/**](https://neo4j.com/docs/) **(accessed: 21.09.2018)**

**3. TitanDB project documentation, available at:** [**http://s3.thinkaurelius.com/docs/titan/1.0.0/**](http://s3.thinkaurelius.com/docs/titan/1.0.0/) **(accessed: 21.09.2018)**

**4. Basu A., Blanning R.: Metagraphs and Their Applications. Springer, Heidelberg, 2007.**

**5. Ataky Tsham Mpinda, Lucas Cesar Ferreira. Evaluation of graph databases performace through index techniques, International Journal of Artificial Intelligence & Applications (IJAIA) Vol. 6, 2015 – 12 p.**

**6. Peter Macko, Margo Seltzer - Performance Introspection of Graph Databases, Harvard University, 2013 – 10 p.**

**7. Ian Robinson, Jim Webber. Graph Databases, O’Reilly Media, Inc., 2015 – 238 p.**

**8. C.J. Date. An Introduction to Database Systems. Moscow, Williams, 2006. — 1328 p.**

**9. Operation graphLookup in MongoDB, available at** [**https://docs.mongodb.com/manual/reference/operator/aggregation/graphLookup/**](https://docs.mongodb.com/manual/reference/operator/aggregation/graphLookup/) **(accessed 21.09.2018).**

**10. Gaurav Vaswani, Anuradha Bhatia. Graph Databases - An Overview, International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014 – 4p.**

**Сведения об авторах**

**Дунин Иван Владимирович**

*Год рождения:* 1995

*Год окончания вуза и его название:* 2018, МГТУ им. Н.Э. Баумана

*Ученая степень:* магистр

*Место работы, должность:* аспирант кафедры ИУ-5 МГТУ им. Н.Э. Баумана

*Полный адрес организации:* Московский Государственный Технический Университет им. Н.Э. Баумана, г. Москва, 2-я Бауманская ул., д. 5, стр. 1, почтовый индекс: 105005

*Количество опубликованных работ:* 5

*Область научных интересов:* обработка больших данных, анализ данных, графовые базы данных

*Электронная почта: johnmoony@mail.ru*

*Контактный телефон:* 8 909 694 84 91

**Гапанюк Юрий Евгеньевич**

*Год рождения:* 1974

*Год окончания вуза и его название:* 1998, МГТУ им. Н.Э. Баумана

*Ученая степень:* к.т.н.

*Место работы, должность:* доцент кафедры ИУ-5 МГТУ им. Н.Э. Баумана

*Полный адрес организации:* Московский Государственный Технический Университет им. Н.Э. Баумана, г. Москва, 2-я Бауманская ул., д. 5, стр. 1, почтовый индекс: 105005

*Количество опубликованных работ:* около 40

*Область научных интересов:* проектирование автоматизированных систем

*Электронная почта:* gapyu@bmstu.ru

*Контактный телефон:* 8 916 558 94 30

**Ревунков Георгий Иванович**

*Год рождения:* 1948

*Год окончания вуза и его название:* 1971, МВТУ им. Н.Э. Баумана

*Ученая степень:* к.т.н.

*Место работы, должность:* доцент кафедры ИУ-5 МГТУ им. Н.Э. Баумана

*Полный адрес организации:* Московский Государственный Технический Университет им. Н.Э. Баумана, г. Москва, 2-я Бауманская ул., д. 5, стр. 1, почтовый индекс: 105005

*Количество опубликованных работ:* более 50

*Область научных интересов:* базы данных, проектирование автоматизированных систем

*Электронная почта:* revunkov@bmstu.ru

*Контактный телефон:* 8 499 263 62 16

**Information about authors**

**Dunin Ivan Vladimirovich**

*Год рождения:* 1995

*Год окончания вуза и его название:* 2018, Bauman Moscow State Technical University

*Ученая степень:* master (Computer Science)

*Место работы, должность:* PhD student of Computer Science and Control Systems Department at Bauman Moscow State Technical University

*Полный адрес организации:* Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow, postcode: 105005

*Количество опубликованных работ:* 5 publications

*Область научных интересов:* big data processing, data mining, graph databases

*Электронная почта: johnmoony@mail.ru*

*Контактный телефон:* 8 909 694 84 91

**Gapanyuk Yuriy Evgenievich**

*Год рождения:* 1974

*Год окончания вуза и его название:* 1998, Bauman Moscow State Technical University

*Ученая степень:* Ph.D. (Computer Sciences)

*Место работы, должность:* associate professor of Computer Science and Control Systems Department at Bauman Moscow State Technical University

*Полный адрес организации:* Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow, postcode: 105005

*Количество опубликованных работ:* about 40 publications

*Область научных интересов:* designing of automated systems, designing of hybrid intelligent information systems, complex graph models

*Электронная почта:* gapyu@bmstu.ru

*Контактный телефон:* 8 916 558 94 30

**Revunkov Georgiy Ivanovich**

*Год рождения:* 1948

*Год окончания вуза и его название:* 1971, Moscow Higher Technical School Named After N.E. Bauman

*Ученая степень:* Ph.D. (Engineering)

*Место работы, должность:* associate professor of Computer Science and Control Systems Department at Bauman Moscow State Technical University

*Полный адрес организации:* Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow, postcode: 105005

*Количество опубликованных работ:* more then 50 publications

*Область научных интересов:* databases, designing of automated systems

*Электронная почта:* revunkov@bmstu.ru

*Контактный телефон:* 8 499 263 62 16