Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем

Зиннатулин Тимур Раифович

Поддержка контекстно-свободных ограничений в языке Cypher

Отчёт по учебной практике

Научный руководитель:
к. ф.-м. н., доцент кафедры информатики Григорьев С. В.

Санкт-Петербург
2020

Оглавление

Введение

Графовые базы данных [12] — это тип баз данных, использующий графовые структуры для представления данных. Вершины графа представляют субъекты, имеющие определенные поля и свойства. Отношения между субъектами представляются в виде ребер графа. Во многих системах графовое представление данных является более интуитивным, и успешно используется в социальных сетях [4], биоинформатике [2], статическом анализе кода [15] и др.

Запросы к графу — нахождение всех путей в графе, соответствующих заданным ограничениям. Ограничения могут задаваться в том числе с помощью формальных грамматик. Большинство языков для запросов к графам поддерживают только регулярные грамматики как ограничения, что не позволяет им делать многие полезные запросы.

Примером такого запроса может послужить запрос всех потомков одного поколения [14]. Такой запрос может быть использован для анализа иерархии файлов в RDF-хранилищах [5]. Доказано, что этот запрос не представим в регулярных грамматиках [13]. Тем не менее, он естественно выражается с помощью контекстно-свободной грамматики.

Запросы к графу, выраженные с помощью контекстно-свободной (КС) грамматики называются КС-запросами. Языки запросов к графовым БД за редким исключением [5] не поддерживают КС-запросы.

Существует стандарт языка запросов к графам openCypher, который является open-source версией языка Cypher. Он используется во многих графовых БД в качестве языка запросов[1]. Cypher не поддерживает КС-запросы, что сильно ограничивает его выразительную силу.

Для обработки КС-запросов применяются различные алгоритмы синтаксического анализа, в том числе (G)LL, (G)LR, CYK [3, 5]. Тем не менее, данные решения не подходят для работы в реальном мире вследствие очень больших затрат алгоритмов по памяти и времени. В то же время, в статье [6] показано, что матричные алгоритмы для обработки КС-запросов имеют достаточно высокую скорость работы на реаль-

[1]Использование Cypher: https://www.opencypher.org/projects. Дата обращения: 21.05.2020

ных данных. Командой лаборатории языковых инструментов JetBrains Research был разработан алгоритм [1] обработки КС-запросов, реализованный с помощью разреженных матриц. Для тестирования алгоритма используется графовая БД RedisGraph, обрабатывающая разреженные матрицы смежности и использующая в качестве языка запросов Cypher. RedisGraph идеально подходит для внедрения данного алгоритма. Она использует синтаксический анализатор (далее — парсер) libcypher-parser [7] в качестве парсера языка Cypher.

Для того, чтобы иметь полное решение задачи эффективной обработки КС-запросов, необходимо, чтобы RedisGraph умел распознавать КС-запросы, написанные на языке Cypher. Для этого необходимо расширить возможности языка Cypher, чтобы в нем имелись языковые конструкции, позволяющие задавать КС-запросы. Это даст возможность полноценно взаимодействовать с RedisGraph и использовать алгоритм в обработке запросов. Данная работа посвящена решению проблемы поддержки контекстно-свободных ограничений в языке Cypher посредством расширения функциональности libcypher-parser.

1. Постановка целей и задач

Целью работы является расширение возможностей парсера языка Cypher libcypher-parser. Поставленные задачи перечислены ниже.

- Изучить синтаксис языка запросов к графовым базам данных Cypher и его реализацию в библиотеке libcypher-parser.
- Расширить парсер библиотеки libcypher-parser конструкциями, необходимыми для поддержки спецификации запросов с контекстно-свободными ограничениями.
- Поддержать построение новых типов узлов абстрактного синтаксического дерева в библиотеке libcypher-parser.
- Оформить запрос на внедрение полученных результатов в основную кодовую базу библиотеки libcypher-parser.
- Интегрировать модифицированный Libcypher-parser в RedisGraph.

2. Обзор

2.1. Парсер-комбинаторы для КС-запросов

В работе [9] исследуется применение парсер-комбинаторов как инструментов для задания КС-запросов на графах. Была расширена библиотека Meerkat, парсер-комбинатор для языка Scala. Модификация Meerkat позволяет создавать запросы к графам прямо в коде.

Подход, описанный в статье, оказался успешным. Авторы работы доказали, что парсер-комбинаторы — надежный способ задания контекстно-свободных-запросов к графам. Недостатком подхода является то, что пользователям непривычно использовать парсер-комбинаторы для создания запросов к графам. Делать запросы с помощью специальных языков для создания запросов, как, например, Cypher, значительно удобнее.

2.2. Модификация языка SPARQL для поддержки КС-запросов

В статье [5] описывается реализация поддержки КС-запросов для языка SPARQL, работающего с RDF-данными. Разработанная модификация носит название cfSPARQL и имеет большую выразительность в плане запросов, которые можно написать, чем SPARQL.

В области Semantic Web[2] cfSPARQL является хорошим подходом к реализации КС-запросов. Но работа с RDF-данными не является популярной среди графовых БД. Поэтому реализация поддержки КС-запросов для языка, специализированного для работы с ними остается важной задачей.

2.3. КС-ограничения для openCypher

Предложение [8] представляет синтаксис для шаблонов, позволяющий задавать регулярные и КС шаблоны над метками путей.

[2]Семантическая паутина: https://www.w3.org/standards/semanticweb/. Дата обращения: 10.06.2020

Описанные продукции дают возможность задавать КС-грамматики над путями в графе. В пример можно привести запрос всех потомков одного поколения, описанный выше [14]. Это предложение было принято сообществом, но до сих пор не реализовано.

На языке Cypher с предложенными модификациями грамматики он может выглядеть следующим образом:

```
PATH PATTERN cousin = ()-/:PARENT> [ ~cousin | () ] <:PARENT/-()
MATCH (me)-/~cousin/-(my_cousin)
RETURN me, collect(my_cousin) AS cousins
```

Предложенный синтаксис был взят за основу для расширения синтаксического анализатора. С его помощью появляется возможность задавать запросы к путям графа, выражающиеся в формальных грамматиках, в том числе контекстно-свободных.

2.4. Libcypher-parser

Libcypher-parser [7] — библиотека для разбора языка Cypher в абстрактное синтаксическое дерево (АСД). Он написан на C и предназначен для создания инструментов и библиотек на различных языках.

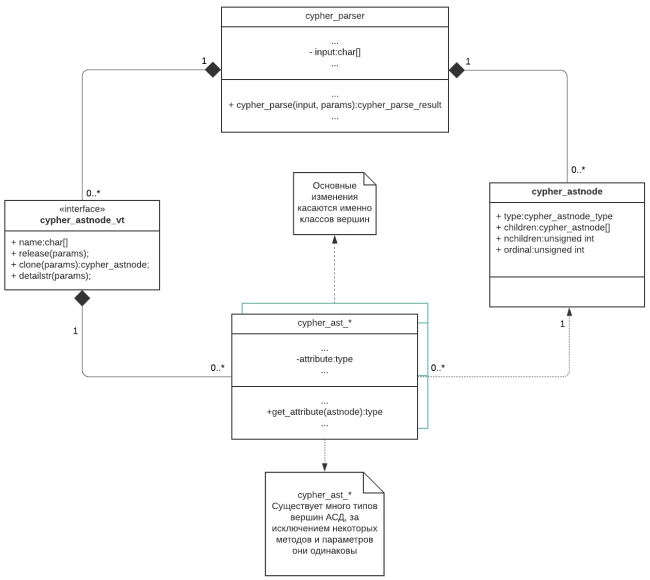

Рис. 1: Схематичное устройство библиотеки Libcypher-parser

Библиотека условно состоит из двух частей: лексического анализатора (лексера) и синтаксического анализатора (парсера). При получении запроса лексер разбивает его на лексемы согласно прописанной грамматике. После этого парсер строит по этим лексемам дерево разбора выражения, которое возвращаетя пользователю. Также библиотека умеет печатать наглядное представление дерева разбора в стандартный поток вывода.

На рис. 1 зеленым цветом выделена часть библиотеки, которую требуется модифицировать для данной работы. Библиотека не поддерживает КС-запросы ни на уровне лексера, ни на уровне парсера. Поэтому для реализации поддержки КС-запросов в парсере необходимо проделать работу в обоих частях парсера.

2.5. RedisGraph

RedisGraph [10] — это графовая база данных, использующая для представления графов разреженные матрицы смежности. В качестве языка запросов к базе RedisGraph использует Cypher.

После того, как парсер создает из запроса, написанного на Cypher, абстрактное дерево, встроенный транслятор преобразует его в выражения линейной алгебры, исполняемые библиотекой GraphBLAS[3]. Апробация показывает, что RedisGraph значительно производительнее конкуренции [11].

Благодаря матричному представлению данных RedisGraph идеально подходит для внедрения алгоритма обработки КС-запросов, предлагаемого в [1]. Для этого необходимо реализовать поддержку таких запросов на уровне языка Cypher, что и является целью данной работы.

[3]GraphBLAS: http://faculty.cse.tamu.edu/davis/GraphBLAS.html. Дата обращения: 10.06.2020

3. Решение

3.1. Синтаксис

В рамках учебной практики лексический анализатор парсера, реализованный с помощью инструмента для генерирования нисходящих анализаторов *leg*[4] был расширен грамматикой, идентичной описанной в [8].

В ходе работы в оригинальном предложении грамматики были найдены ошибки. Исправления, которые были предложены разработчикам синтаксиса, были приняты[5] как верные.

3.2. Реализация

Пример: реализация правила разбора шаблонов путей в грамматике:

```
path-pattern =
    < ( LEFT-ARROW-HEAD - DASH - DIV - e:path-expression - DIV - DASH
          ( - RIGHT-ARROW-HEAD > { $$ = path_pattern(e, BIDIRECTIONAL); }
          | _empty_ >            { $$ = path_pattern(e, INBOUND); }
          )
      | DASH - DIV - e:path-expression - DIV - DASH
          ( - RIGHT-ARROW-HEAD > { $$ = path_pattern(e, OUTBOUND); }
          | _empty_ >            { $$ = path_pattern(e, BIDIRECTIONAL); }
          )
      )
```

В фигурных скобках показан вызов функций построения определенных вершин АСД; в зависимости от того, как именно будет разобрана соответствующая лексема, могут быть построены вершины типа path_pattern с разными параметрами.

Были созданы и интегрированы в библиотеку новые типы вершин для абстрактного синтаксического дерева.

---

[4]leg: https://www.piumarta.com/software/peg/. Дата обращения: 21.05.2020

[5]Обсуждение: https://github.com/opencypher/openCypher/pull/187#discussion_r406155831. Дата обращения: 10.06.2020

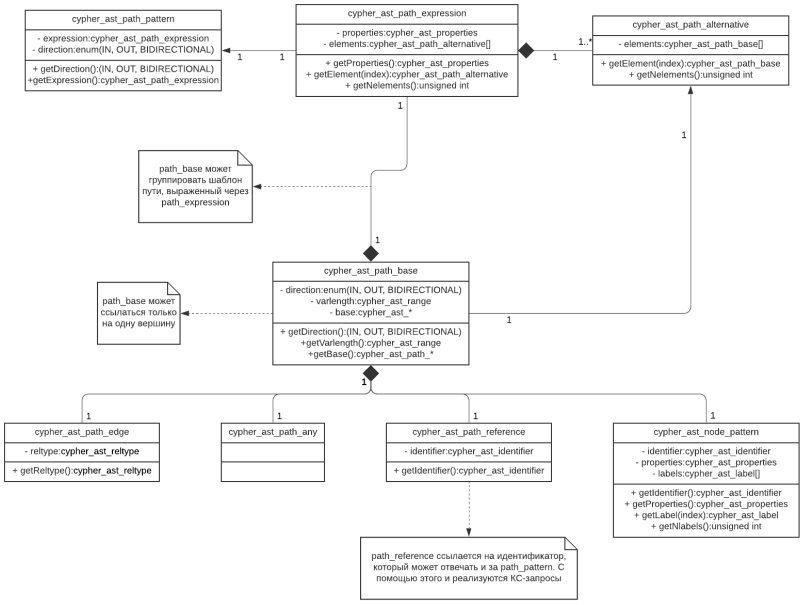

Рис. 2: Устройство добавленных типов вершин дерева разбора.

Реализация новых вершин была написана на языке C. Для каждой вершины была описана ее структура, т.е. какие данные она может хранить в себе и какие типы вершин могут от нее наследоваться. Также было написано явное представление вершины в абстрактном синтаксическом дереве в общем виде.

Для корректной работы парсера с новыми вершинами были вставлены их описания в нескольких файлах libcypher-parser, отвечающих за разбор выражения, построение дерева разбора и взаимодействие с клиентом.

3.3. Тестирование

Для разработанного синтаксиса были созданы тесты на правильность разбора запроса в АСД.

На рис. 3 показан пример разбора следующего запроса:

```
MATCH (me)-/~cousin/-(my_cousin)
RETURN me, collect(my_cousin) AS cousins
```

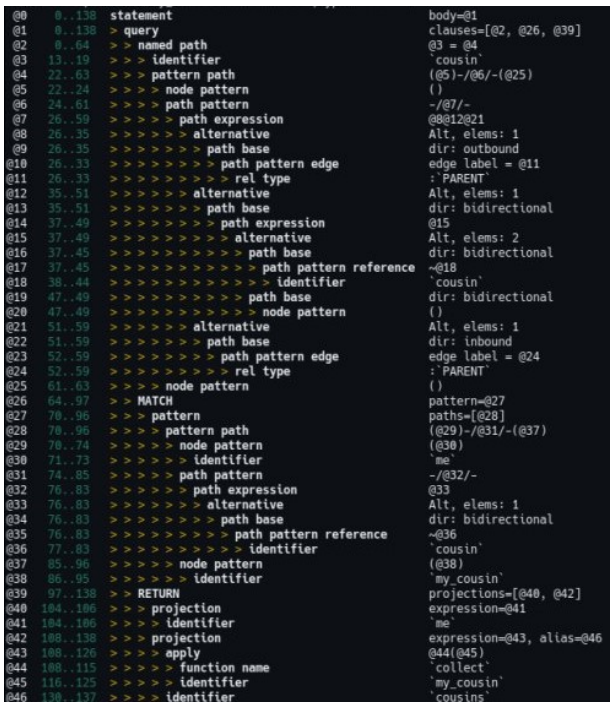

Рис. 3: Пример работы libcypher-parser.

После разбора запроса на лексесмы было построено абстрактное синтаксическое дерево, представление которого было выведено в консоль.

3.4. Запрос на внесение изменений в Libcypher-parser

По окончании работы был составлен запрос на внесение изменений в основной репозиторий GitHub[6]. Запрос находится на стадии просмотра кода.

3.5. Интеграция с RedisGraph

Рабочая версия libcypher-parser была интегрирована с версией RedisGraph, используемой в лаборатории языковых инструментов JetBrains Research[7].

[6]Pull request: https://github.com/cleishm/libcypher-parser/pull/26. Дата обращения: 10.06.2020

[7]Рабочая ветвь RedisGraph: https://github.com/YaccConstructor/RedisGraph/tree/path_patterns. Дата обращения: 10.06.2020

Заключение

В ходе данной работы мы смогли реализовать поддержку контекстно-свободных запросов в языке Cypher, а именно:

- Изучить синтаксис языка запросов к графовым базам данных Cypher и его реализацию в библиотеке libcypher-parser. Для реализации поддержки КС-запросов в языке Cypher было необходимо изучить устройство языка и его возможности. Также для работы с парсером было необходимо изучить его работу и взаимодействие между его файлами.

- Расширить синтаксичесий анализатор библиотеки libcypher-parser с помощью конструкций, необходимых для поддержки спецификации запросов с контекстно-свободными ограничениями. Было добавлено 13 новых синтаксических правила, позволяющие распознавать КС-запросы.

- Реализовать построение новых типов узлов абстрактного синтаксического дерева в библиотеке libcypher-parser. Было добавлено 9 новых типов вершин абстракстного синтаксического дерева.

- Оформить запрос на внедрение полученных результатов в основную кодовую базу библиотеки libcypher-parser. Текущее состояние запроса можно увидеть на GitHub[8].

- Интегрировать библиотеку libcypher-parser в RedisGraph[9]. На уровне транслятора были реализованы базовые конструкции шаблонов путей.

Дальнейшим развитием данной работы является:

- Дальнейшая интеграция с RedisGraph. Для полной реализации возможностей, предоставленных модифицированным libcypher-parser

[8]Pull request: https://github.com/cleishm/libcypher-parser/pull/26. Дата обращения: 10.06.2020

[9]Рабочая ветвь RedisGraph: https://github.com/YaccConstructor/RedisGraph/tree/path_patterns. Дата обращения: 10.06.2020

необходимо реализовать более низкоуровневые конструкции, лежащие в основе шаблонов путей.

- Запрос на внесение изменений в RedisGraph. По окончании интеграции libcypher-parser с рабочей ветвью RedisGraph следующим шагом будет внедрение этих модификаций в основной репозиторий, чтобы предоставить возможность выполнять КС-запросы всем пользователям.

Текущее состояние парсера можно увидеть на GitHub[10].

[10]Репозиторий: https://github.com/YaccConstructor/libcypher-parser/tree/path-patterns-support. Дата обращения: 21.05.2020

Список литературы

[1] Azimov Rustam, Grigorev Semyon. Graph Parsing by Matrix Multiplication // CoRR. — 2017. — Vol. abs/1707.01007. — 1707.01007.

[2] BioGraph: a web application and a graph database for querying and analyzing bioinformatics resources / Antonio Messina, Antonino Fiannaca, Laura La Paglia et al. // BMC Systems Biology. — 2018. — Vol. 12, no. S5. — URL: https://doi.org/10.1186/s12918-018-0616-4 (online; accessed: 21.05.2020).

[3] Bradford Phillip G. Quickest Path Distances on Context-Free Labeled Graphs // Proceedings of the 6th WSEAS International Conference on Information Security and Privacy. — ISP'07. — Stevens Point, Wisconsin, USA : World Scientific and Engineering Academy and Society (WSEAS), 2007. — P. 22–29.

[4] Chaudhary Anoop, FAISAL ABDUL. Role of graph databases in social networks. — 2016. — 06.

[5] Context-Free Path Queries on RDF Graphs / Xiaowang Zhang, Zhiyong Feng, Xin Wang, Guozheng Rao // CoRR. — 2015. — Vol. abs/1506.00743. — 1506.00743.

[6] Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication / Nikita Mishin, Iaroslav Sokolov, Egor Spirin et al. // Proceedings of the 2Nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA'19. — New York, NY, USA : ACM, 2019. — P. 12:1–12:5. — URL: http://doi.acm.org/10.1145/3327964.3328503 (online; accessed: 10.06.2020).

[7] Leishman Chris. libcypher-parser. — https://github.com/cleishm/libcypher-parser. — 2016.

[8] Lindaaker Tobias. CIP2017-02-06 Path Patterns // GitHub. — 2016. — URL: https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc (online; accessed: 21.05.2020).

[9] Parser Combinators for Context-free Path Querying / Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, Semyon Grigorev. — 2018. — 11 p. — URL: http://doi.acm.org/10.1145/3241653.3241655 (online; accessed: 21.05.2020).

[10] RedisGraph - A graph database module for Redis. — https://github.com/RedisGraph/RedisGraph. — 2016.

[11] RedisGraph GraphBLAS Enabled Graph Database / Pieter Cailliau, Tim Davis, Vijay Gadepally et al. // CoRR. — 2019. — Vol. abs/1905.01294. — 1905.01294.

[12] Robinson Ian S., Webber Jim, Eifrém Emil. Graph Databases: New Opportunities for Connected Data. — 2015.

[13] S. Abiteboul P. Buneman, Suciu D. Foundations of Databases. — Addison-Webley, 1995.

[14] Suzuki Susumu, Kishi Masasuke, Ibaraki T. Query evaluation of the same generation problem with many variables // Systems and Computers in Japan. — 1993. — Vol. 24. — P. 1–14.

[15] Urma Raoul-Gabriel, Mycroft Alan. Source-code queries with graph databases—with application to programming language usage and evolution // Science of Computer Programming. — 2015. — Vol. 97. — P. 127–134. — URL: https://doi.org/10.1016/j.scico.2013.11.010 (online; accessed: 21.05.2020).