Поддержка контекстно-свободных ограничений в языке Cypher

Студент: Тимур Зиннатулин, 243

Научный руководитель: к.ф.-м.н. доц. С. В. Григорьев

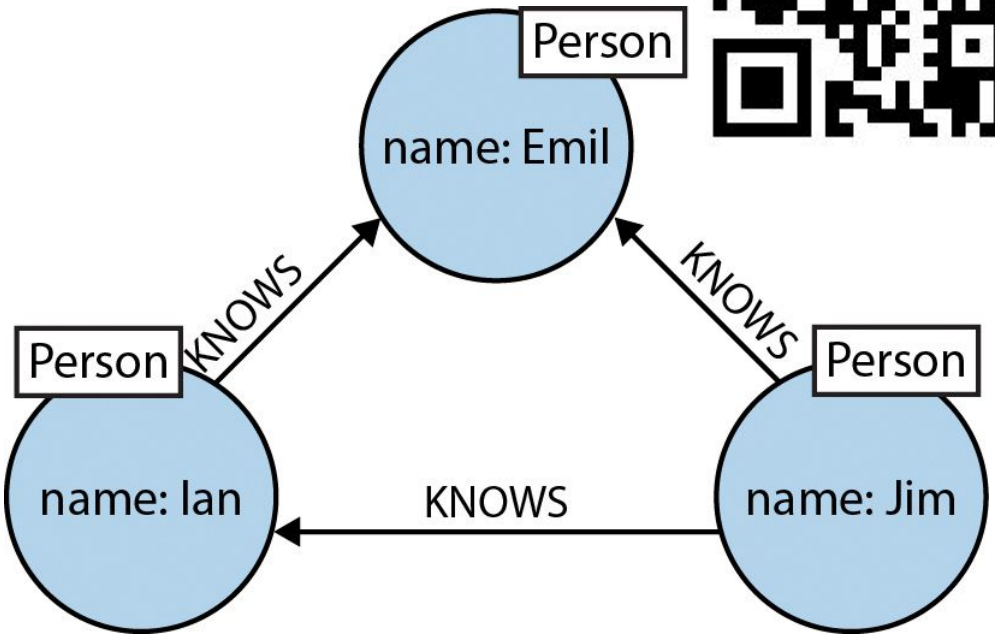
Групповой проект

Введение

Графовые базы данных

Предлагают более интуитивную форму представления данных

Используются в биоинформатике, социальных сетях и т.д.



Введение

Cypher Query Language

Декларативный язык для описания визуальных паттернов в графах с помощью синтаксиса ASCII-Art.

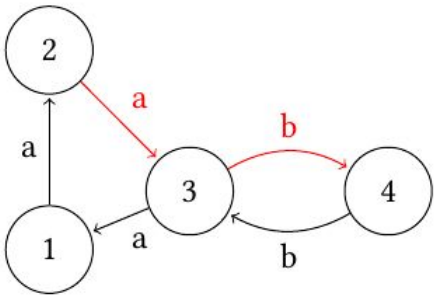
Введение

Контекстно-свободные (КС) запросы

Context-Free Path Querying

Запросы к графу - нахождение всех путей в графе, соответствующих заданным ограничениям.

Ограничения могут быть заданы с помощью грамматики.



S ->a S b | a b

Пример запроса всех путей $a^n b^n$ вида где n>0

Введение

КС-запросы

Context-Free Path Querying

За редким исключением нигде не поддерживаются.

Отсутствие возможности делать КС-запросы сильно ограничивает выразительную силу языка.

Введение

КС-запросы

Context-Free Path Querying

Командой лаборатории языковых инструментов JetBrains Research был создан быстрый алгоритм обработки КС-запросов.

Алгоритм реализован с помощью разреженных матриц.

Введение

RedisGraph

Графовая база данных, использующая для представления графов разреженные матрицы смежности.

Идеально подходит для внедрения разработанного алгоритма.

Использует Cypher в качестве языка запросов.

Введение

Libcypher-parser

Библиотека для синтаксического анализа языка Cypher.

Написана на языке C.

Используется RedisGraph в качестве синтаксического анализатора (парсера) Cypher.

Постановка задачи

Цель - расширить возможности библиотеки Libcypher-parser

Задачи:

1. Изучить синтаксис Cypher
2. Расширить libcypher-parser конструкциями, которые позволят задавать запросы с КС-ограничениями.
3. Оформить запрос на внедрение изменений в основной репозиторий libcypher-parser
4. Интегрировать модифицированный парсер в RedisGraph

Обзор

Parser Combinators for Context-Free Path Querying, 2018,
Ekaterina Verbitskaia, Ilya Nozhkin, Ilya Kirillov, Semyon Grigorev

- Парсер-комбинаторы применяются для задания КС-запросов
- Модифицикация Meerkat, парсер-комбинатора для Scala
- Можно задавать запросы прямо в коде

- Непривычно делать запросы с помощью парсер-комбинаторов
- Лучше использовать специальный язык запросов

Обзор

Context-Free Path Queries on RDF Graphs, 2016
Xiaowang Zhang, Zhiyong Feng, Xin Wang, Guozheng Rao, Wenrui Wu
cfSPARQL

- Расширение языка запросов SPARQL, поддерживающее CFPQ для обработки RDF-данных.

- Немногие графовые БД работают с RDF.

Обзор

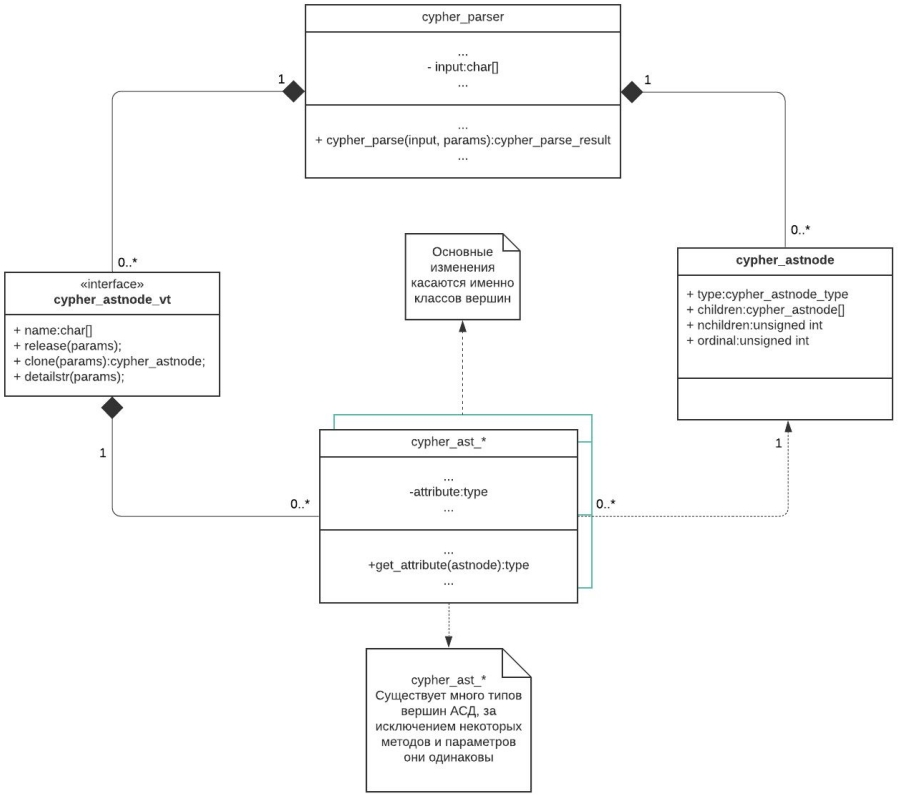
openCypher Path Patterns CIP, 2017

Tobias Lindaaker

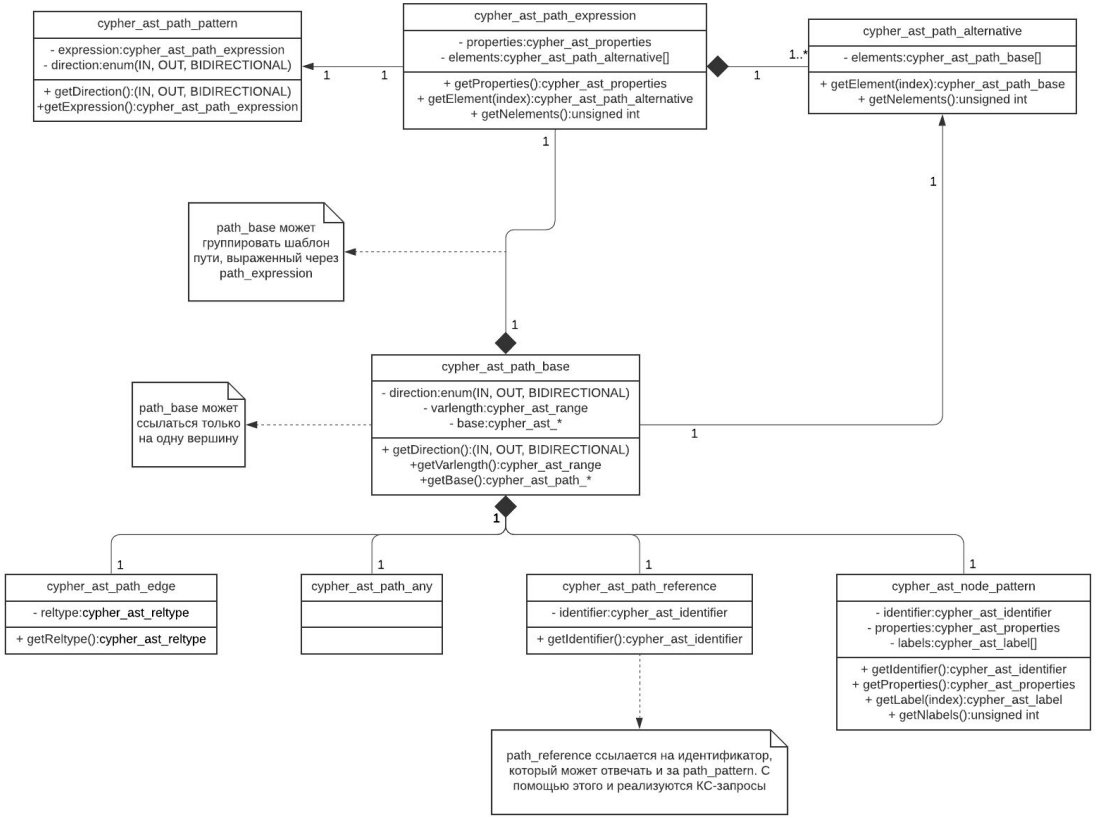
- Принятое предложение реализовать поддержку шаблонов над путями в языке Cypher.
- Не реализовано на практике, есть лишь грамматика.

- Был взят за основу для расширения парсера
- Синтаксис, описанный в документе, был реализован в ходе работы

Решение


cypher_parser
...
- input:char[]
...
...
+ cypher_parse(input, params):cypher_parse_result
...
1
1
0..*
0..*
«interface»
cypher_astnode_vt
+ name:char[]
+ release(params);
+ clone(params):cypher_astnode;
+ detailstr(params);
Основные изменения касаются именно классов вершин
cypher_astnode
+ type:cypher_astnode_type
+ children:cypher_astnode[]
+ nchildren:unsigned int
+ ordinal:unsigned int
1
1
0..*
0..*
cypher_ast_*
...
-attribute:type
...
...
+get_attribute(astnode):type
...
cypher_ast_*
Существует много типов вершин АСД, за исключением некоторых методов и параметров они одинаковы

Решение


cypher_ast_path_pattern
- expression:cypher_ast_path_expression
- direction:enum(IN, OUT, BIDIRECTIONAL)
+ getDirection():(IN, OUT, BIDIRECTIONAL)
+getExpression():cypher_ast_path_expression
cypher_ast_path_expression
- properties:cypher_ast_properties
- elements:cypher_ast_path_alternative[]
+ getProperties():cypher_ast_properties
+ getElement(index):cypher_ast_path_alternative
+ getNelements():unsigned int
cypher_ast_path_alternative
- elements:cypher_ast_path_base[]
+ getElement(index):cypher_ast_path_base
+ getNelements():unsigned int
path_base может группировать шаблон пути, выраженный через path_expression
cypher_ast_path_base
- direction:enum(IN, OUT, BIDIRECTIONAL)
- varlength:cypher_ast_range
- base:cypher_ast_*
+ getDirection():(IN, OUT, BIDIRECTIONAL)
+getVarlength():cypher_ast_range
+getBase():cypher_ast_path_*
path_base может ссылаться только на одну вершину
cypher_ast_path_edge
- reltype:cypher_ast_reltype
+ getReltype():cypher_ast_reltype
cypher_ast_path_any
cypher_ast_path_reference
- identifier:cypher_ast_identifier
+ getIdentifier():cypher_ast_identifier
cypher_ast_node_pattern
- identifier:cypher_ast_identifier
- properties:cypher_ast_properties
- labels:cypher_ast_label[]
+ getIdentifier():cypher_ast_identifier
+ getProperties():cypher_ast_properties
+ getLabel(index):cypher_ast_label
+ getNlabels():unsigned int
path_reference ссылается на идентификатор, который может отвечать и за path_pattern. С помощью этого и реализуются КС-запросы

Добавленная грамматика

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

Пример использования добавленной грамматики.
Данный КС-запрос реализует запрос всех потомков одного поколения

Подобный запрос может быть использован для анализа иерархии файлов в RDF-хранилищах.
Не представим в регулярных грамматиках.

Тесты

```
@0    0..138  statement                                  body=@1
@1    0..138  > query                                    clauses=[@2, @26, @39]
@2    0..64   > > named path                             @3 = @4
@3   13..19   > > > identifier                           `cousin`
@4   22..63   > > > pattern path                         (@5)-/@6/-(@25)
@5   22..24   > > > > node pattern                       ()
@6   24..61   > > > > path pattern                       -/@7/-
@7   26..59   > > > > > path expression                  @8@12@21
@8   26..35   > > > > > > alternative                     Alt, elems: 1
@9   26..35   > > > > > > > path base                     dir: outbound
@10  26..33   > > > > > > > path pattern edge            edge label = @11
@11  26..33   > > > > > > > > rel type                   :`PARENT`
@12  35..51   > > > > > > alternative                     Alt, elems: 1
@13  35..51   > > > > > > > path base                     dir: bidirectional
@14  37..49   > > > > > > > > path expression            @15
@15  37..49   > > > > > > > > alternative                 Alt, elems: 2
@16  37..45   > > > > > > > > > path base                 dir: bidirectional
@17  37..45   > > > > > > > > > path pattern reference  ~@18
@18  38..44   > > > > > > > > > > identifier             `cousin`
@19  47..49   > > > > > > > > > path base                 dir: bidirectional
@20  47..49   > > > > > > > > > > node pattern           ()
@21  51..59   > > > > > > alternative                     Alt, elems: 1
@22  51..59   > > > > > > > path base                     dir: inbound
@23  52..59   > > > > > > > path pattern edge            edge label = @24
@24  52..59   > > > > > > > > rel type                   :`PARENT`
@25  61..63   > > > > node pattern                       ()
@26  64..97   > > MATCH                                  pattern=@27
@27  70..96   > > > pattern                              paths=[@28]
@28  70..96   > > > > pattern path                       (@29)-/@31/-(@37)
@29  70..74   > > > > > node pattern                     (@30)
@30  71..73   > > > > > > identifier                     `me`
@31  74..85   > > > > > path pattern                     -/@32/-
@32  76..83   > > > > > > path expression                @33
@33  76..83   > > > > > > > alternative                   Alt, elems: 1
@34  76..83   > > > > > > > > path base                   dir: bidirectional
@35  76..83   > > > > > > > > > path pattern reference  ~@36
@36  77..83   > > > > > > > > > > identifier             `cousin`
@37  85..96   > > > > > node pattern                     (@38)
@38  86..95   > > > > > > identifier                     `my_cousin`
@39  97..138  > > RETURN                                 projections=[@40, @42]
@40 104..106  > > > projection                           expression=@41
@41 104..106  > > > > identifier                         `me`
@42 108..138  > > > projection                           expression=@43, alias=@46
@43 108..126  > > > > apply                              @44(@45)
@44 108..115  > > > > > function name                    `collect`
@45 116..125  > > > > > identifier                       `my_cousin`
@46 130..137  > > > > identifier                         `cousins`
```

Запрос на внедрение в Libcypher-parser

Интеграция с RedisGraph

Итоги работы

- Изучен синтаксис языка Cypher и его реализация в Libcypher-parser.
- Добавлены новые синтаксические конструкции.
- Реализованы новые типы узлов АСД.
- Оформлен запрос на внедрение в основную библиотеку.
- Произведена интеграция с RedisGraph.