1. Анализ модели графа свойств на примере языка Cypher[[1]](#footnote-0)

Д. В. Мамонов

Студент бакалавр

Н. К. Самойлов

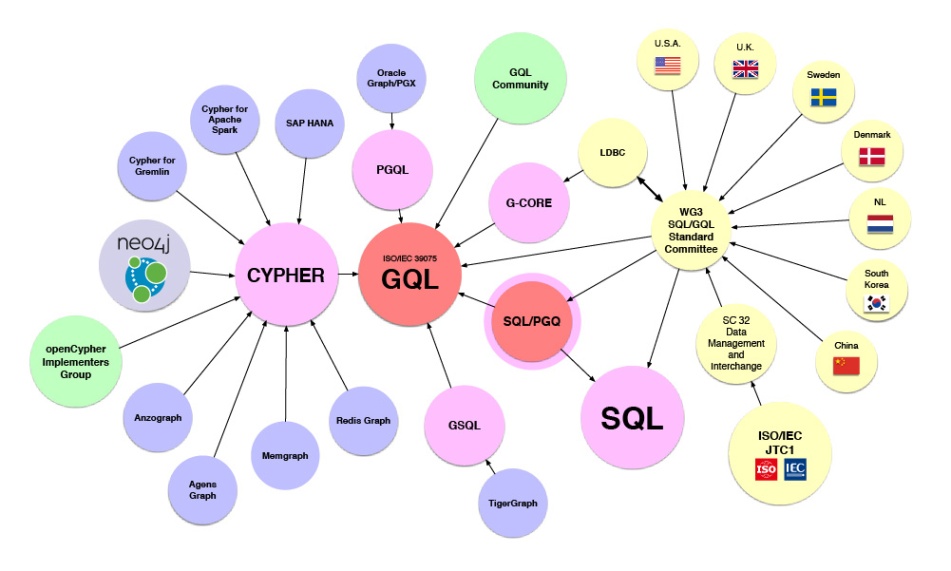
* 1. Старший преподаватель
  2. Введение

На сегодняшний день наиболее распространенными системы управления базами данных (СУБД) являются базы, основанные на реляционной модели отображения данных, например PostgreSQL, MySQL, Oracle. Несмотря на то, что они отлично справляются со структурированными данными и простыми отношениями, при работе с сильно взаимосвязанными данными или быстро меняющимися схемами они испытывают трудности. Поэтому во многих отраслях становятся популярными графовые базы данных, являющиеся NoSQL СУБД. Такие системы позволяют эффективно представлять и запрашивать сложные взаимосвязи между объектами данных. Узлы, ребра и свойства являются основными компонентами графовой базы данных, обеспечивая интуитивно понятную и наглядную модель данных. Но для таких СУБД язык запросов SQL, обычно используемый реляционными базами данных, неприменим в большинстве случаев. Поэтому разработчики придумали собственные варианты языков, один из которых – Cypher – и будет рассматриваться в данной работе.

* 1. История языка Cypher и попытки стандартизации

Cypher – специальный декларативный язык запросов, который был разработан инженером Андреасом Тейлором во время его работы в компании Neo4j.Inc в 2011 году. На этапе создания предполагался для взаимодействия только с графовой базой данных Neo4j, но в 2015 году появился проект openCypher, который позволил интегрировать его и с другими СУБД [1].

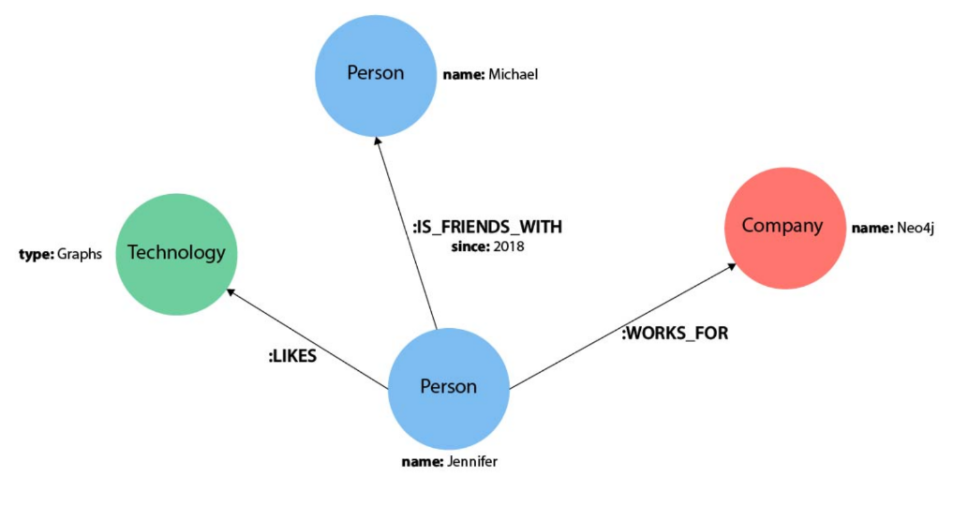
После создания проекта openCypher сообществом были предприняты попытки стандартизации языка в качестве базового языка обработки графовых данных, совместимого с языком запросов к реляционным СУБД SQL. Для этого были проведены 5 очных совещаний разработчиков openCypher. Первое совещание состоялось в феврале 2017 года в штаб-квартире SAP в Вальдорфе (Германия) и совпало с заседанием Совета по тестированию связанных данных. Последнее OCIM состоялось в Берлине одновременно с семинаром W3C по веб-стандартам управления графическими данными в марте 2019 года. По итогу работы в сентябре 2019 года был создан проект по разработке GQL (Graph Query Language) (рис. 1), который был одобрен голосованием национальных органов по стандартизации, которые являются членами Объединенного технического комитета ISO/IEC. На текущий момент стандарт находится в разработке и еще не является общедоступным.



Компоненты стандарта GQL

* 1. Модель графа свойств

Язык Cypher основан на использовании модели графа свойств (рис. 2). В теории графов он может быть определен как ориентированный, с помеченными узлами и ребрами мультиграф, где ребра могут быть уникально идентифицированы. В графе свойств понятие узла отражает вершину, а понятие отношения – ребра [2].



Пример графа свойств

Графовая модель включает в себя следующие понятия: сущность; путь в графе; токен; свойство:

* + 1. Сущность – уникальный объект, который можно сравнить на идентичность с другими аналогичными объектами. Она содержит в себе специальный контейнер свойств – словарь "ключ - значение". Каждое свойство (т.е. "ключ") является уникальным в пределах описания одной сущности. Сущность представлена узлом и отношением между узлами.
    2. Узел – базовая сущность графа с уникальными атрибутами, которая может существовать независимо от других сущностей. Он может хранить в себе набор свойств, который не является обязательным. Узел может иметь входящие и исходящие отношения (либо вообще не иметь отношений, связывающих его с остальными узлами).
    3. Отношение – сущность, обозначающая направленное соединение между двумя узлами (исходный и целевой узлы). Оно может быть исходящим и входящим. Исходящее отношение рассматривается со стороны исходного узла, входящее – со стороны целевого узла. Отношение может иметь только один тип (или метку).
    4. Путь в графе отображает обход в графе свойств и состоит из набора узлов, соединяемых отношениями между собой. Он всегда начинается с определенного узла и заканчивается узлом. Самым кратчайшим путем считается один узел, являющийся и начальным и конечным. Длина пути – характеристика, обозначающая количество отношений в нем.
    5. Токен – непустая строка символов в кодировке Unicode. Токенами представлены метки, типы отношений и ключи свойств. Метка присваивается только узлам, тип отношений - отношениям. Ключи свойств - названия уникальных идентификаторов полей в наборе свойств.
    6. Свойство – пара «ключ-значение», которая может быть инициализирована значением определенного типа или списком таких значений.
  1. Описание DML в Cypher

Cypher использует ASCII-представление для описания запросов, т.е. информация должна быть наглядно визуализирована для пользователя. Это делает язык очень наглядным и удобным для чтения, поскольку он как визуально, так и структурно представляет данные, указанные в запросе. Запросы Cypher объединяются с шаблонами узлов и связей с любой указанной фильтрацией по меткам и свойствам для создания, чтения, обновления, удаления данных, найденных в указанном шаблоне.

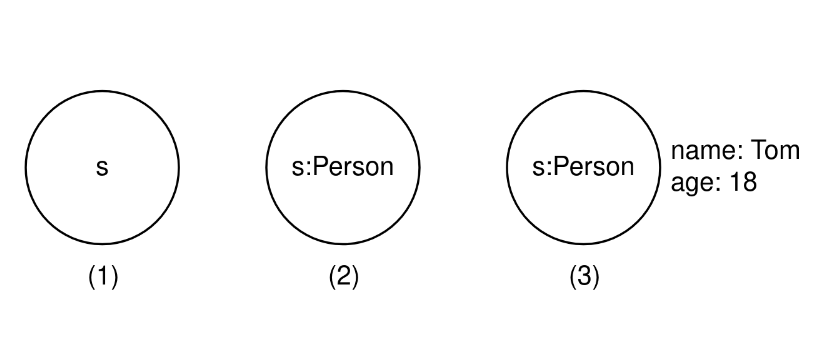
Самое простое описание узла это его имя, заключенное в круглых скобках. Дополнительно при описании узла могут указываться его метка (например, при выборе определенных узлов) и тело свойств (например, при создании нового отношения) (рис. 3) (листинг 1).

Описание узлов

(s) // 1. Простейшее описание

(s:Person) // 2. Описание с меткой

(s:Person{name: 'Tom', age: 18}) // 3. Описание со свойствами



Визуализация узлов графа

В случае если набор свойств отображен в выборке, то он накладывает дополнительное ограничение на ее результат. При выполнении запроса CREATE набор будет установлен по умолчанию в новых узлах и отношениях.

Для описания простейшего отношения между двумя узлами используется конструкция вида (a)-[]-(b). В этом случае такие отношения будут образовывать неориентированный граф. Для того чтобы указать направление отношения, используются знаки ->, представляющие собой «стрелочку». Как и узлы, отношения могут иметь имена и метки, обозначающие их тип (или класс, к которому они относятся). Описать отношение, которое имеет определенный тип, можно, указав в квадратных скобках идентификатор отношения и его тип через знак двоеточия. В отличие от узлов, при задании нового отношения или при построении шаблона имя отношения можно опустить и использовать только название метки (рис. 4) (листинг 2).

Описание отношений между узлами

// 1. Простейшее неориентированное отношение

(a)-[]-(b)

// 2. Простейшее ориентированное отношение

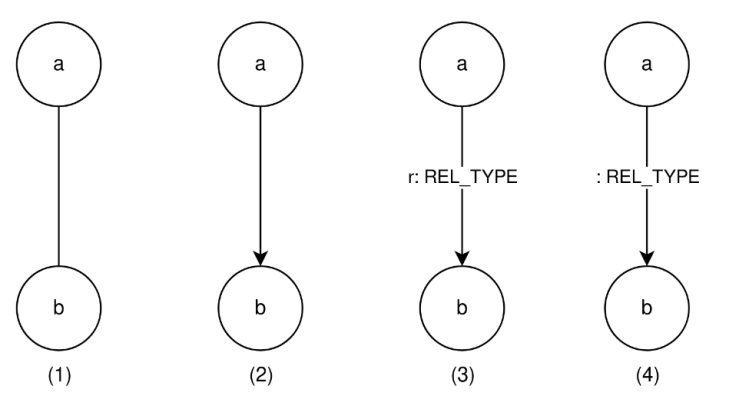
(a)-[]->(b)

// 3. Отношение с указанием его типа

(a)-[r:REL\_TYPE]->(b)

// 4. Отношение с указанием его типа и с опущением идентификатора

(a)-[:REL\_TYPE]->(b)



Визуализация отношений в графе

* 1. Запросы выборки данных в Cypher

Подобно другим языкам запросов, Cypher содержит множество ключевых слов для задания шаблонов, фильтрации шаблонов и возврата результатов. Наиболее распространенными для построения запросов являются MATCH и RETURN (рис. 5) (листинг 3).

Ключевое слово MATCH обозначает поиск существующих узлов, отношений, меток, свойств или шаблонов в СУБД. Оно является аналогичным слову SELECT в SQL. В запросах, построенных с MATCH, можно производить поиск всех меток и типов отношений, узлов и отношений на основе шаблонов и т.д.

Ключевое слово RETURN в Cypher указывает, какие значения или результаты можно вернуть из запроса. В качестве объектов, которые можно возвращать, могут использоваться узлы, отношения, свойства узлов и отношений, шаблоны. RETURN не требуется при выполнении запросов записи, но необходим для выборки. Названия переменных узла и отношения становятся необходимыми при использовании RETURN. Чтобы вернуть узлы, отношения, свойства или шаблоны, в запросе MATCH должны быть указаны переменные для данных, которые надо вернуть.

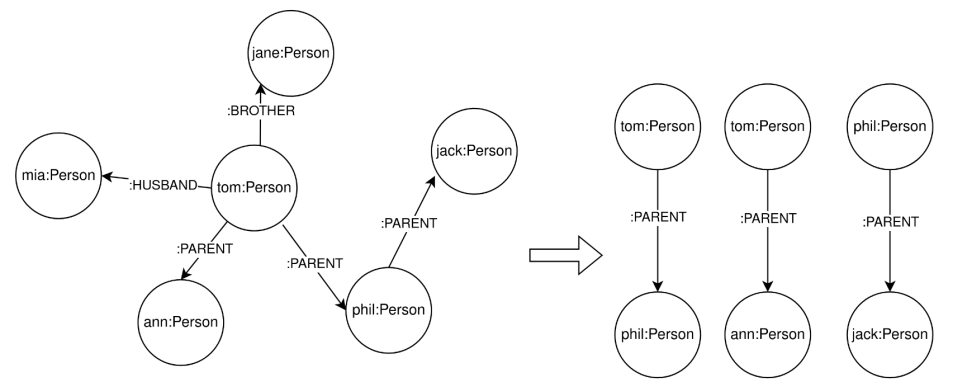
Примеры запросов на шаблонах с MATCH и RETURN

MATCH (p1:Person)-[:PARENT]->(p2:Person)

RETURN p1, p2

MATCH ans = (:Person {name: "Pam"})-[:PARENT]->(:Person {name: "Bob"})

RETURN ans



Визуализация выполнения запроса выборки

Язык поддерживает фильтрацию выборки с помощью ключевого слово WHERE, которое имеет аналогичный принцип работы как в языке SQL (рис. 6) (листинг 4).

Примеры запросов с фильтрацией через WHERE

MATCH (p1:Person)-[:PARENT]->(p2:Person)

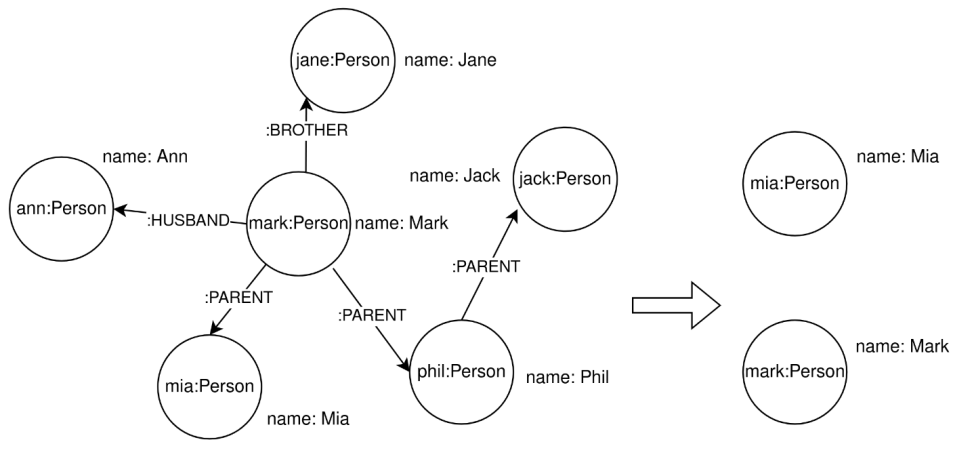
WHERE p1.name="Bob"

RETURN p2

MATCH (p:Person)

WHERE p.name STARTS WITH 'M'

RETURN p.name



Визуализации фильтрации выборки по свойству

* 1. Запросы изменения базы данных в Cypher

Редактирование базы данных (создание новых узлов и отношений, их изменение и удаление) обеспечивается при помощи ключевых слов CREATE, SET и DELETE.

Запрос CREATE является отдельно выполняемым запросом (как и INSERT в SQL), но при создании отношений может являться частью MATCH. Он позволяет создавать как отдельные узлы и отношения из уже имеющихся узлов, так и целый граф, в котором будут содержаться новые данные (рис. 7) (листинг 5).

Запросы на создание сущностей через CREATE

CREATE (n:Person)

CREATE (n:Person {name: 'Andres', title: 'Developer'})

MATCH (a:Person), (b:Person)

WHERE a.name = 'Node A' AND b.name = 'Node B'

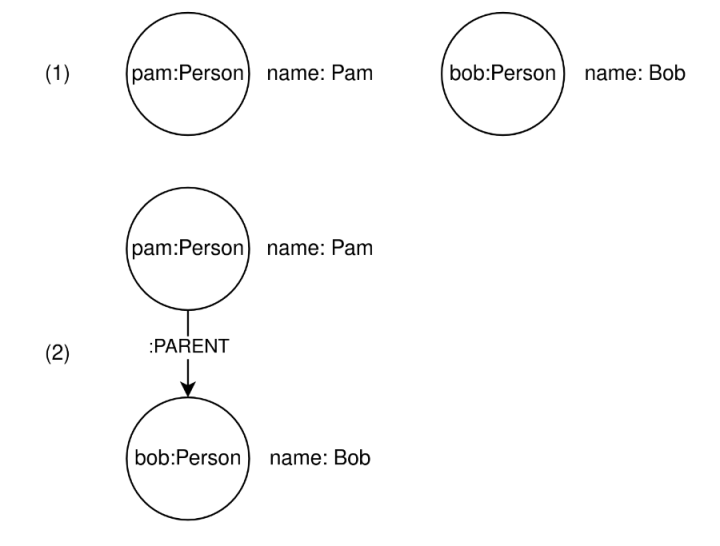
CREATE (a)-[r:RELTYPE]->(b)

RETURN r

CREATE pam: Person{name: "Pam"},

bob: Person{name: "Bob"},

(pam: Person)-[:PARENT]->(bob: Person)



Визуализация процесса создания сущностей в базе данных

Запросы SET и DELETE выполняются только в теле запроса MATCH и могут проводиться как над всеми выбранными данными, так и над результатом их фильтрации. Узлы разрешается удалять только после того, как они не будут включены ни в одно отношение.

Запросы на изменение данных в графе

MATCH (n {name: 'Andres'})

SET n.surname = 'Taylor'

RETURN n

MATCH (n:Useless)

DELETE n

MATCH (p1:Person)-[r]->(p2:Person)

DELETE r

* 1. Заключение

В данной статье было представлено описание модели графа свойств и относящихся к ней понятий, таких как сущность, узел, отношение, токен, путь и свойство. Также была рассмотрена история разработки языка запросов Cypher и варианта стандартизации подобных языков для обращения к графовым базам данных GQL. Показаны базовые варианты определения в языке узлов, отношений, их свойств, меток и типов, а также шаблонов, связывающих компоненты в единое целое. В работе были приведены примеры полноценных запросов на выборку, создание и изменение данных.

* 1. Список литературы
     + 1. Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker. Cypher: An Evolving Query Language for Property Graphs / Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker // SIGMOD’18 Proceedings of the 2018 International Conference on Management of Data. – Houston, United States, June, 2018. – P. 2-3
       2. Property Graph Model [Электронный ресурс] – Режим доступа: https://github.com/opencypher/openCypher/blob/master/docs/property-graph-model.adoc
       3. Cypher Query Language Reference Version 9 [Электронный ресурс] – Режим доступа: https://s3.amazonaws.com/artifacts.opencyphe r.org/openCypher9.pdf

1. 1. Мамонов Д. В., Самойлов Н. К., 2024

   [↑](#footnote-ref-0)