**Графовая база данных** **Neo4j.** **Язык Cypher.**

Neo4j – графовая СУБД открытым кодом. Написана на языке Java компанией Neo Technology в 20003 году. Максимальный размер графа до 32 миллиардов узлов и 64 миллиардов свойств. Поддерживает транзакции ACID, индексацию данных, хорошо масштабируется. Может работать на кластере серверов. Улучшена работа фреймворка для обеспечения быстрого обхода узлов.

Может работать как обособленный сервер с доступом через REST API, так и в качестве встраиваемой системы реализующей функции работы с графом в пользовательских программах.

Работает с языками Clojure, Erlang, Gremlin, Groovy, Java, PHP, Python, Ruby и Scala.

**Язык Cypher.**

Это основной язык графовой СУБД Neo4j. Используется не только в ней. Декларативность языка Cypher, как всегда, позволяет задать свойства результата, не вдаваясь в подробности работы алгоритма.

Многие ключевые слова, такие как CREATE, WHERE, ORDER BY, DELETE заимствованы из SQL. Сопоставление с образцом перенесено из SPARQL. Часть семантики коллекций заимствована из языков Haskell и Python. Запрос в Cypher, также как и в SQL, составлен из нескольких фраз. Они передают промежуточные результирующие наборы данных друг другу.

Запрос может включать одну или несколько фраз MATCH, задающих часть графа соответствующую записанному шаблону. Эти фразы создают контекст, в котором работают следующие фразы, например, RETURN, возвращающая значения.

**Описания шаблонов в языке Cypher.**

В Cypher для описания шаблонов извлекаемой из графа информации, используется синтаксис ascii-art. Основные компоненты шаблонов:

* Узлы (вершины, nodes) изображают сущности (объекты), могут быть помечены одной или любым количеством меток, могут иметь одно или несколько поименованных свойств.
* Рёбра (отношения, связи, relationshis) имеют тип и направление, соединяют два узла – один начальный и один конечный. Также могут содержать свойства.
* Свойства представляются парами Key : Value, где Key – строка, а Value любого типа данных.
* Метки (labels). Узлы, помеченные одной меткой, образуют набор. Метки упрощают создание запросов. Используются для задания ограничений и индексов.

Изобразительные средства:

|  |  |
| --- | --- |
| Обозначение в языке Cypher | Описание |
| () | Узел. Сущность или объект. Изображается парой круглых скобок, внутри которых задаются тип вершины и/или значения свойств, например, (a:User) - вершина типа User. |
| [] | Рёбрo изображается парой квадратных скобок, внутри них могут задаваться условия. |
| : | Метка. Тип узла или ребра. Например, [:FRIEND] - связь типа FRIEND. |
| -> или <- или - | Направление связи. Например, ()-[]->(). Если можно не указывать ограничения на связи, опускаются прямые скобки и знаки направления () - -(). |
| {} | Свойство узла или ребра. Например, (mike{name:’Mike’}). |

Обозначение () определяет любой узел. Если необходимо обратиться к конкретному узлу, записываем в него переменную, например, (p). Несколько узлов можно пометить, записав метку после двоеточия. Например, (p: Person).

У рёбер, представляющих отношения, также могут задаваться имена и типы, например, **- [rel1:KNOWS] ->** . При этом типов может быть несколько, например, **- [:KNOWS|:LIKE]->** .

Двунаправленные отношения представляются парой однонаправленных.

Для извлечения данных из хранилища используется запрос, содержащий шаблон фильтрации, позволяющий получать:

* **(n)—>(m)** — все направленные ребра из вершины n в вершину m.
* **(n)—(m)** — все ненаправленные ребра между вершинами n и m.
* **(n:Person)** — все вершины с меткой Person.
* **(n:Person:Russian)** — все вершины, имеющие обе метки Person и Russian.
* **(n:Person {name:{value}})** — все вершины с меткой Person и отфильтрованные по дополнительному свойству.
* **(n:Person)- ->(m)** — ребра между вершинами n с меткой Person и вершиной m.
* **() - [r {name: “some data”}] - ()** – находим все отношения с указанными свойствами.
* **(n) - [\*1 .. 5] -> (m)** – путь переменной длины от 1 до 5 ребер между вершинами n и m.
* **(n) - [\*] -> (m)** –путь любой длины между вершинами n и m.
* **shortestPath((n1:Person) - [\*..6] - (n2:Person) )** – единственный короткий путь длиной до 6 ребер между вершинами n1 и n2 с метками Person.
* **allShortestPaths((n1:Person)-[\*..6]->(n2:Person))** – все короткие пути до 6 ребер между вершинами n1 и n2 с метками Person.
* **size((n) --> () --> () )** – подсчитывает пути между вершинами, указанными в шаблоне.

**Язык Cypher. Типы данных**

|  |  |
| --- | --- |
| **Тип данных CQL** | **Использование** |
| **Boolean** | Представляет булевы литералы: true, false |
| **byte** | Представляет 8-bit целые |
| **short** | Представляет 16-bit целые |
| **int** | Представляет 32-bit целые |
| **long** | Представляет 64-bit целые |
| **float** | Представляет 32-bit числа с плавающей запятой |
| **double** | Представляет 64-bit числа с плавающей запятой |
| **char** | Представляет 16-bit символы |
| **String** | Представляет строковые значения |

Простейшие конструкции **create, return, match и delete**.

**CREATE (n:МеткаN{ключ1:”значение1”})-[r:LOVES {since: }]->(m:МеткаМ)**

**return n,r,m**

В результате будет созданы вершины **n** и **m** соединенные направленным ребром **r**. Вершины **n** и **m** будут помечены метками **МеткаN** и **МеткаM** соответственно, a ребро **r** – меткой **LOVES**. Кроме того вершине n будет присвоено свойство с ключом – **ключ1** со значением **”значение1”**, а ребру **r** будет присвоено свойство с ключом **since** с пустым значением. После создание будут возвращены в точку вызова значения переменных **n**, **m** и **r**.

**MATCH (n)**

**RETURN n**

Найти все вершины и вернуть их в точку вызова.

**MATCH (n:Person)-[:KNOWS]->(m:Person)**

**WHERE n.name = ”Alice”**

Найти все вершины **n** помеченные меткой **Person** соединенные с вершинами **m** помеченные меткой **Person** ребром с меткой **KNOWS**, причем у вершины **n** должно быть свойство с ключом **name** и со значением **”Alice”**.

**DELETE n, r**

Удалить вершины и ребра связанные с переменными n и r.

**DETACH DELETE n**

Удалить вершины, связанные с переменной n. Кроме того автоматически будут удалены ребра, ведущие в данную вершину.

**MATCH (n)**

**DETACH DELETE n**

Полное удаление базы данных. Будут удалены все вершины и ребра с ними связанные.