Язык запросов Cypher.

Для работы с базой данной neo4j используется специальный язык Cypher. Cypher является декларативным языком запросов для графов, который позволяет выразительно и эффективно выполненить запросы и обновленить базу графов. Cypher является относительно простым, но также мощным языком. Сложные запросы к базе данных легко могут быть выражены через Cypher.

Cypher разработан как человекочитаемый язык запросов, подходящий как для разработчиков, так и специалистов аналитиков. Задачей языка состоит в том, чтобы сделать простые вещи легко, и сложные вещи возможными. Его конструкции основаны на английской языке и иконографии, которые помогают сделать запросы более понятными. Язык старается быть удобным как для чтения, так и для записи.

Cypher заимствует свою структуру из SQL - запросы строятся с использованием различных операций.

Операции могут быть соединены друг с другом, и они распределяют промежуточные результаты между собой. Например, совпадающие переменные из одного пункта MATCH имеют один контекст между операциями.

Язык запросов состоит из нескольких различных операций:

* MATCH: Шаблон графа для поиска. Наиболее распространенный способ получения данных из графа.
* WHERE: Используется как часть часть MATCH, OPTIONAL MATCH и WITH. Добавляет ограничения на шаблон, или фильтрет промежуточные результаты проходящий через WITH.
* RETURN: Что будет возвращено.

Использование neo4j в приложении.

Для текущего проекта будет использоваться следующая модель данных, представленная на рисунке 3.1

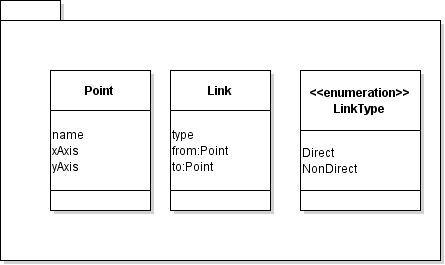


Рис.3.1 Модель данных приложения

Для создания в базе данных вершин графа используется следующий запрос:

CREATE (p:Point {name:'First', xAsis:15.15, yAxis:15.15})

CREATE (p:Point {name:'Second', xAsis:20.20, yAxis:20.20})

Для создания ребер графа использется следующий запрос:

MATCH (f:Point {name:'First'}) MATCH (s:Point{name:'Second'}) CREATE (f)-[d:DIRECT]->(s)

Для анализа работы с алгоритмами будет использоваться следующий граф.

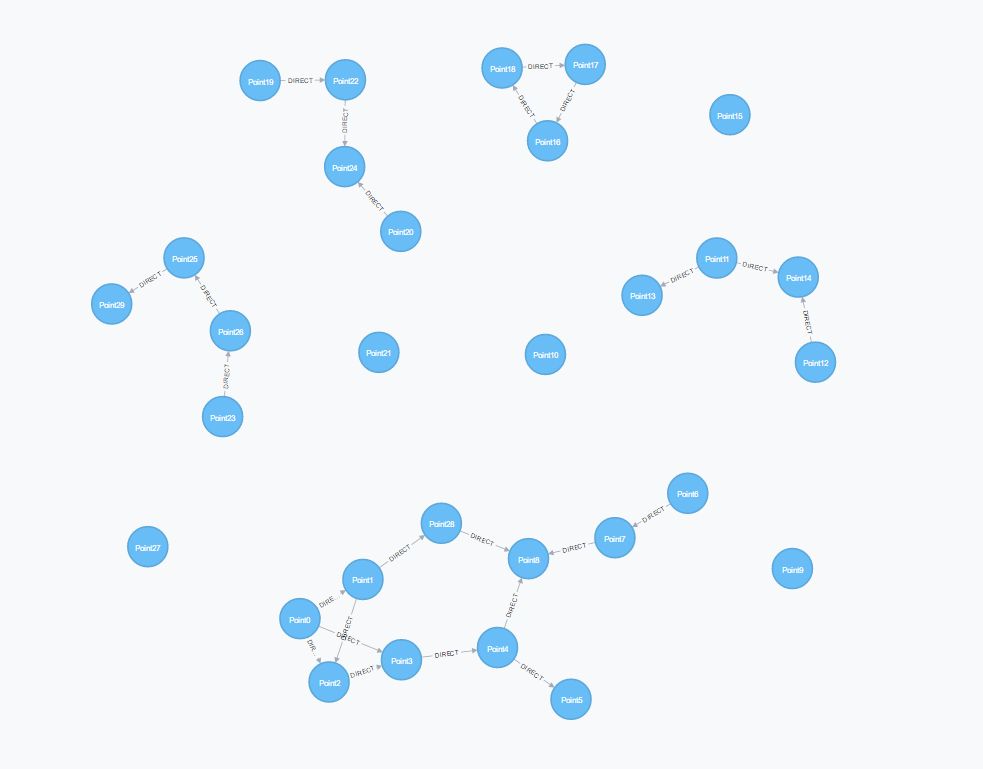


Рис. 3.2 – Тестовый граф

В базе данных neo4j также присутствуют реализации алгоритмов работы с графами, таких как поиск связанности вершин.

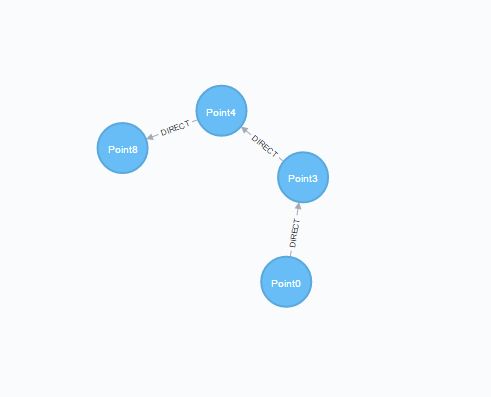
MATCH (f:Point{name:'Point0'})-[\*]->(s:Point{name:'Point8'}) RETURN count(\*)>0

Данный запрос вернет булево значение в зависимости связаны ли 2 вершины.

Другим встроенным алгоритмом является нахождение кратчайшего пути

MATCH (f:Point { name:"Point0" }),(s:Point { name:"Point8" }), p = shortestPath((f)-[\*]-(s)) RETURN p

Результатом выполнения будет являться подграф являющимся кратчайшим путем для данного графа.



3.3 – Результат нахождения кратчайшего пути для тетсового графа