GraphQL

Mimanshu Maheshwari

December 4, 2024

Contents

0.1 Setting up Neo4j with docker . . . 3
0.2 Creating Nodes . . . 3
0.3 Creating Relationships . . . 3
0.4 Defining a constraint . . . 4
0.5 MATCH . . . 4
0.6 Optional Match . . . 4
0.7 WHERE . . . 5
0.8 Merge . . . 5
0.9 UNION, UNION ALL . . . 5
0.10 ORDER BY, SKIP, LIMIT, DISTINCT . . . 5
0.11 SET . . . 5
0.12 FOREACH . . . 6
0.13 REMOVE . . . 6
0.14 DELETE . . . 6
0.15 DETACH DELETE . . . 6
0.16 Bulk Load of Data . . . 7
0.17 Functions . . . 8
0.17.1 toLower() . . . 8
0.17.2 toUpper() . . . 8
0.17.3 substring() . . . 8
0.17.4 replase() . . . 8
0.17.5 reverse() . . . 8
0.17.6 Aggregate function . . . 8
0.17.7 List functions . . . 9
0.18 User defined functions(UDF) in Neo4J . . . 9
0.18.1 UDFs in Neo4J . . . 9
0.18.2 Deploying UDF . . . 10
0.18.3 Calling a UDF . . . 10
0.19 Optional Schema Indexes and constraints . . . 10
0.19.1 Analyze query performance using EXPLAIN . . . 10
0.19.2 Improve performance using indexes . . . 10
0.20 Working with constraints . . . 11
0.20.1 Unique node property constraint . . . 11
0.21 Using Neo4J with Java . . . 11

List of Figures

Listings

1 get neo4j image . . . 3
2 Starting neo4j container . . . 3
3 Create Node . . . 3
4 Create Node Example . . . 3
5 Create Relationships . . . 3
6 Create Relation Example . . . 4
7 constraint . . . 4
8 Match syntax . . . 4
9 Match example . . . 4
10 Match example . . . 4
11 Match example . . . 4
12 Match example . . . 5
13 Neo4J with java . . . 12

Chapter 1

Why and What is GraphQL?

1.1 Why GraphQL

We all know that REST contributes more towards API development. In recent years, REST has become the dominant style for creating web services that are reusable and manageable. Everything in REST is represented as a resource which in turn has a unique URI to get identified. Also, REST works with a set of HTTP methods/verbs to define the operations being performed on the REST resources. Each one of us, the developers, will accept that REST is a great approach to proceed with API development as

- It leverages the usage of the standard HTTP protocol
- It is stateless
- It supports caching
- It is light-weight
- It supports multiple representations for resources and so on