Neo4j

# Graph database

Causal Cluster of Neo4j

**Causal Cluster Members**

| Roles | Addresses | Groups | Database | Actions |
|---|---|---|---|---|
| FOLLOWER | bolt://localhost:7687, http://localhost:7474, https://localhost:6477 | - | default | Open |
| LEADER | bolt://localhost:7688, http://localhost:7475, https://localhost:6478 | - | default | Open |
| FOLLOWER | bolt://localhost:7689, http://localhost:7476, https://localhost:6479 | - | default | Open |
| READ_REPLICA | bolt://localhost:7690, http://localhost:7477, https://localhost:6480 | - | default | Open |
| FOLLOWER | bolt://localhost:7691, http://localhost:7478, https://localhost:6481 | - | default | Open |

Setting up Neo4j database with relations.

```
//RELATION product → category
MATCH (p:Product), (c:Category)
WHERE p.category = c.name
MERGE (p)-[:CATEGORIZED_AS]→(c)
```

```
//RELATION review → product
MATCH (r:Review), (p:Product)
WHERE r.uniq_id = p.id
MERGE (r)-[:REVIEWS]→(p)
```

```
//RELATION brand → product
MATCH (b:Brand), (p:Product)
WHERE b.name = p.manufacturer
MERGE (b)-[:OWNS]→(p)
```

```
-- Load Product nodes
LOAD CSV WITH HEADERS FROM 'file:///modified_data.csv' AS row

-- Create Product node
MERGE (product:Product {id: row.uniq_id})
SET product.name = row.product_name,
    product.price = substring(row.price, 2),
    product.manufacturer = row.manufacturer,
    product.category = row.amazon_category_and_sub_category;

-- Load Category nodes
LOAD CSV WITH HEADERS FROM 'file:///modified_data.csv' AS row
MERGE (category:Category {name: row.amazon_category_and_sub_category});

-- Load Reviews with sublists as properties
LOAD CSV WITH HEADERS FROM 'file:///modified_data.csv' AS row

-- Process customer_reviews sublist
FOREACH (reviewList IN split(substring(row.customer_reviews, 1, size(row.customer_reviews) - 2), "], [") |
    -- Remove extra quotes and split sublist into individual properties
    CREATE (review:Review {
      review: REPLACE(SPLIT(reviewList, "', '")[0], "'", ""),
      rating: toFloat(REPLACE(SPLIT(reviewList, "', '")[1], "'", "")),
      date: REPLACE(SPLIT(reviewList, "', '")[2], "'", ""),
      username: REPLACE(SPLIT(reviewList, "', '")[3], "'", ""),
      uniq_id: row.uniq_id
    })
);

-- Load Brand nodes
LOAD CSV WITH HEADERS FROM 'file:///modified_data.csv' AS row

-- Create Brand node
MERGE (brand:Brand {name: row.manufacturer});
```

Graph projection of all nodes with all relations

```
// Create graph projection with all nodes/edges
CALL gds.graph.project(
    'myGraph',  // Graph name
    '*',        // Node projection (selects all node labels)
    '*',        // Relationship projection (selects all relationship types)
    {}          // Configuration (empty in this example)
)
YIELD graphName, nodeProjection, nodeCount, relationshipProjection, relationshipCount, projectMillis
RETURN graphName, nodeProjection, nodeCount, relationshipProjection, relationshipCount, projectMillis;
```

Used to show top 10 most popular categories using the degree centrality algoritm

```
# Shows top 10 most popular categories using the degree centrality algorithm
def popular_categories():
    query = """
    MATCH (c:Category)
    OPTIONAL MATCH (p:Product)-[:CATEGORIZED_AS]->(c)
    WITH c, count(p) AS degreeCentrality
    ORDER BY degreeCentrality DESC
    RETURN c.name AS name, degreeCentrality
    LIMIT 10
    """

    with driver.session() as session:
        result = session.run(query)
        categories = [{"name": record["name"], "degreeCentrality": record["degreeCentrality"]} for record in result]

    return jsonify({"categories": categories})
```