Understanding Cypher Projections in Neo4j

What is a Cypher Projection?

A Cypher Projection is a type of graph projection in Neo4j Graph Data Science (GDS) that allows for the creation of highly customized in-memory graphs using Cypher queries. This approach is ideal for scenarios where the graph's structure does not directly align with the existing database schema, or when filtering and aggregating data is necessary.

Key Characteristics of Cypher Projections

1. **Customizable**: Allows full control over which nodes and relationships to include.

2. **Flexible**: Supports filtering, aggregation, and complex structures using Cypher.

3. **Dynamic**: Adjusts to the database's current state, reflecting updates when projected.

4. **Useful for Derived Graphs**: Enables the creation of graphs that differ significantly from the database schema.

How to Create a Cypher Projection

1. **Define Node and Relationship Queries**:

   Use Cypher queries to specify the nodes and relationships to include in the projection.

2. **Create the Projection**:

```
   CALL gds.graph.project.cypher(
       'cypherGraph',
       'MATCH (n:Person) RETURN id(n) AS id, n.name AS name',
           'MATCH (n:Person)-[r:FRIENDS_WITH]->(m:Person) RETURN id(n) AS source, id(m) AS target'
   );
```

   In this example:

   - Nodes are projected from `Person` nodes.

   - Relationships are projected from `FRIENDS_WITH` edges.

Example Cypher Projection

Suppose you want to create a graph of customers and their interactions based on orders:

- **Node Query**:

```
   MATCH (c:Customer) RETURN id(c) AS id, c.name AS name;
```

- **Relationship Query**:

```
   MATCH (c1:Customer)-[:PLACED_ORDER]->(:Order)<-[:PLACED_ORDER]-(c2:Customer)
   RETURN id(c1) AS source, id(c2) AS target, COUNT(*) AS weight;
```

   This creates a graph where customers are connected if they placed orders for the same products.

Run Algorithms on the Cypher Projection

Once the graph is projected, you can run algorithms like PageRank:

```
CALL gds.pageRank.stream('cypherGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC;
```

This ranks nodes (e.g., customers) based on their centrality in the graph.

Drop the Cypher Projection

When you're finished, drop the projection to free resources:

```
CALL gds.graph.drop('cypherGraph');
```

Benefits of Cypher Projections

1. **Customization**: Tailored to specific analysis needs using Cypher queries.

2. **Flexibility**: Works with complex or derived graph structures.

3. **Aggregation**: Allows for pre-aggregated relationships with weights or other metrics.

4. **Scalability**: Focuses on relevant parts of the graph, improving computational efficiency.

Conclusion

Cypher Projections in Neo4j GDS provide unparalleled flexibility for creating custom in-memory graphs. They are particularly useful for advanced analyses that require derived or highly filtered subgraphs.