Neo4J Query Reference

Cypher is the declarative query language for Neo4j, the world's leading graph database.

Key principles and capabilities of Cypher are as follows:

- Cypher matches patterns of nodes and relationship in the graph, to extract information or modify the data.
- Cypher has the concept of identifiers which denote named, bound elements and parameters.
- Cypher can create, update, and remove nodes, relationships, labels, and properties.
- Cypher manages indexes and constraints.

You can try Cypher snippets live in the Neo4j Console at console.neo4j.org or read the full Cypher documentation in the Neo4j Manual. For live graph models using Cypher check out GraphGist.

The Cypher Refcard is also available in PDF format.

Note: `{value}` denotes either literals, for ad hoc Cypher queries; or parameters, which is the best practice for applications. Neo4j properties can be strings, numbers, booleans or arrays thereof. Cypher also supports maps and lists.

Syntax

Read Query Structure

```
[MATCH WHERE]
[OPTIONAL MATCH WHERE]
[WITH [ORDER BY] [SKIP] [LIMIT]]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

MATCH

```
MATCH (n:Person)-[:KNOWS]->(m:Person)
WHERE n.name = "Alice"
```
Node patterns can contain labels and properties.

```
MATCH (n)-->(m)
```
Any pattern can be used in `MATCH`.

```
MATCH (n {name: "Alice"})-->(m)
```
Patterns with node properties.

```
MATCH p = (n)-->(m)
```
Assign a path to `p`.

```
OPTIONAL MATCH (n)-[r]->(m)
```
Optional pattern, `NULL`s will be used for missing parts.

```
WHERE m.name = "Alice"
```
Force the planner to use a label scan to solve the query (for manual performance tuning).

WHERE

```
WHERE n.property <> {value}
```
Use a predicate to filter. Note that `WHERE` is always part of a `MATCH`, `OPTIONAL MATCH`, `WITH` or `START` clause. Putting it after a different clause in a query will alter what it does.

RETURN

```
RETURN *
```
Return the value of all variables.

```
RETURN n AS columnName
```
Use alias for result column name.

```
RETURN DISTINCT n
```
Return unique rows.

```
ORDER BY n.property
```
Sort the result.

```
ORDER BY n.property DESC
```
Sort the result in descending order.

```
SKIP {skipNumber}
```
Skip a number of results.

```
LIMIT {limitNumber}
```
Limit the number of results.

```
SKIP {skipNumber} LIMIT {limitNumber}
```
Skip results at the top and limit the number of results.

```
RETURN count(*)
```
The number of matching rows. See Aggregation for more.

WITH

```
MATCH (user)-[:FRIEND]-(friend)
WHERE user.name = {name}
WITH user, count(friend) AS friends
WHERE friends > 10
RETURN user
```
The `WITH` syntax is similar to `RETURN`. It separates query parts explicitly, allowing you to declare which variables to carry over to the next part.

```
MATCH (user)-[:FRIEND]-(friend)
WITH user, count(friend) AS friends
ORDER BY friends DESC
SKIP 1 LIMIT 3
RETURN user
```
You can also use `ORDER BY`, `SKIP`, `LIMIT` with `WITH`.

UNION

```
MATCH (a)-[:KNOWS]->(b)
RETURN b.name
UNION
MATCH (a)-[:LOVES]->(b)
RETURN b.name
```
Returns the distinct union of all query results. Result column types and names have to match.

```
MATCH (a)-[:KNOWS]->(b)
RETURN b.name
UNION ALL
MATCH (a)-[:LOVES]->(b)
RETURN b.name
```
Returns the union of all query results, including duplicated rows.