Cypher is the declarative query language for Neo4j, the world-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, named 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](http://console.neo4j.org) or read the full Cypher documentation in the Neo4j Manual. For live graph models using Cypher check [GraphStream](http://graphstream.dev).

The Cypher Refcard is also available in PDF format.

### Syntax

#### Read Node Query Structure

```
MATCH [MATCH_LEVEL] [WHERE] [ON] [OPTIONAL] [WHERE] [ON] [AGGREGATE] [WHERE] [ON] [ORDER BY] [LIMIT] [SKIP] [RETURN]
```

- A match pattern is a graph pattern that identifies the nodes and relationships to be matched.
- A WHERE clause can be used to filter nodes or relationships.
- An ORDER BY clause can be used to order the results.
- A LIMIT clause can be used to limit the number of results returned.
- A SKIP clause can be used to skip a certain number of results.

#### Write-Only Query Structure

```
CREATE [CREATE_LEVEL] [ON] [WHERE] [ON] [ORDER BY] [AGGREGATE] [ON] [RETURN]
```

- A create pattern is a graph pattern that identifies the nodes and relationships to be created.
- A WHERE clause can be used to filter nodes or relationships.
- An ORDER BY clause can be used to order the results.
- An AGGREGATE clause can be used to perform aggregations on the results.

#### Read-Write Query Structure

```
MATCH [MATCH_LEVEL] [WHERE] [ON] [OPTIONAL] [WHERE] [AGGREGATE] [WHERE] [ON] [ORDER BY] [LIMIT] [RETURN]
```

- A match pattern is a graph pattern that identifies the nodes and relationships to be matched.
- A WHERE clause can be used to filter nodes or relationships.
- An ORDER BY clause can be used to order the results.
- An AGGREGATE clause can be used to perform aggregations on the results.

### Mathematical Functions

- `atan2` (x, y) returns the angle whose tangent is y/x.
- `cos`, `sin`, `tan` (x) returns the cosine, sine, and tangent of x (in radians).
- `exp` (x) returns e^x.
- `log` (x) returns the natural logarithm of x.
- `pow` (x, y) returns x^y.
- `sqrt` (x) returns the square root of x.
- `abs` (x) returns the absolute value of x.
- `ceil` (x) returns the smallest integer greater than or equal to x.
- `floor` (x) returns the largest integer less than or equal to x.
- `round` (x) returns the nearest integer to x.
- `sign` (x) returns 1 if x is positive, 0 if x is zero, and -1 if x is negative.
- `random` () returns a random number between 0 and 1.

### Null Value

- `null` is used to represent undefined/missing values.
- `is_null()` checks if a value is null.
- `is_defined()`, `is_defined()` checks if a value is defined.

### Paths

A path is a sequence of connected nodes. Paths can be used to traverse the graph, starting from a specific node and following a set of relationships.

```
MATCH <start_label> -[<relationship>]-> <end_label> RETURN <start_property>, ..., <end_property>
```

- `MATCH` specifies the pattern to match.
- `RETURN` specifies the properties to return.

### Relationships

```
relationship: relationship property = value
```

- `relationship` specifies the relationship type.
- `property` specifies the relationship property.
- `value` specifies the value of the relationship property.

### Properties

```
property: property name = value
```

- `property` specifies the property name.
- `name` specifies the name of the property.
- `value` specifies the value of the property.

### Aggregation

```
aggregate: aggregate function on property
```

- `aggregate` specifies the aggregation function to apply.
- `function` specifies the function to apply.

### Cypher Specific Syntax

```
OPTIONAL MATCH [WHERE] [ON] [AGGREGATE] [WHERE] [ON] [ORDER BY] [LIMIT] [RETURN]
```

- An optional match pattern is a graph pattern that identifies the nodes and relationships to be matched, but only if they exist.

### Cypher Examples

```
MATCH (a:Person) WHERE a.age > 18 RETURN a
```

`MATCH` specifies the pattern to match.

```
MATCH (a:Person)-[:FRIENDS]->(b:Person) RETURN a, b
```

- `MATCH` specifies the pattern to match.
- `RETURN` specifies the properties to return.

### Neo4j Cypher Refcard 2.3.12

 achievable in a dynamically computed property name.

### Properties

```
property: property name = value
```

- `property` specifies the property name.
- `name` specifies the name of the property.
- `value` specifies the value of the property.

### Aggregation

```
aggregate: aggregate function on property
```

- `aggregate` specifies the aggregation function to apply.
- `function` specifies the function to apply.

### Cypher Specific Syntax

```
OPTIONAL MATCH [WHERE] [ON] [AGGREGATE] [WHERE] [ON] [ORDER BY] [LIMIT] [RETURN]
```

- An optional match pattern is a graph pattern that identifies the nodes and relationships to be matched, but only if they exist.

### Cypher Examples

```
MATCH (a:Person) WHERE a.age > 18 RETURN a
```

`MATCH` specifies the pattern to match.