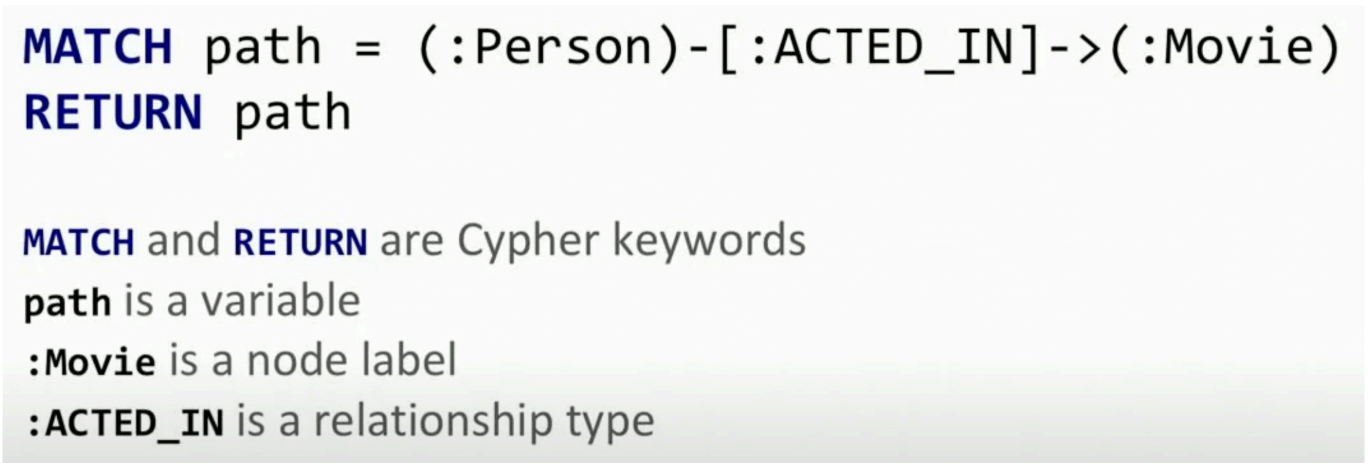
Cypher Query



```
MATCH path = (:Person)-[:ACTED_IN]->(:Movie)
RETURN path
```

MATCH and **RETURN** are Cypher keywords
path is a variable
:Movie is a node label
:ACTED_IN is a relationship type

Case sensitivity

Case sensitive

1. Node labels (:Person)
2. Relationship types (:ACTED_IN)
3. Property keys (name)

Case insensitive

1. Cypher keywords (MATCH, return)

Aggregates

Aggregate queries in Cypher are a little bit different than in SQL as we don't need to specify a grouping key.

We implicitly group by any non aggregate fields in the RETURN statement.

```
// implicitly groups by p.name
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, count(*) AS numberOfMovies
```

Other aggregate functions

https://neo4j.com/docs/cypher-refcard/current/

Constraints and Indexes

Unique Constraints

We create unique constraints to:

1. ensure uniqueness
2. allow fast lookup of nodes which match label-property pairs.

```
CREATE CONSTRAINT ON (label:Label)
ASSERT label.property IS UNIQUE
```

There are three types of unique constraints:

1. Unique node property constraint
2. Node property existence constraint
3. Relationship property existence constraint

```
CREATE CONSTRAINT ON (label:Lable)
ASSERT EXISTS(label.name)
```

```
CREATE CONSTRAINT ON ()-[rel:REL_TYPE]->()
ASSERT EXISTS(rel.name)
```

Indexes

We create indexes to:

1. allow fast lookup of nodes which match label-property pairs.

```
CREATE INDEX ON :Label(property)
```

The following predicates use indexes:

1. Equality
2. STARTS WITH
3. CONTAINS
4. ENDS WITH
5. Range searches
6. (Non-) existence checks

The MERGE Clause

```
MERGE (p:Person {name:'Tom Hanks', oscar: true})
RETURN p
```

There is not a :Person node with name:'Tom Hanks' and oscar:true in the graph, but there is a :Person with name:'Tom Hanks'.

What do you think will happen here?

```
MERGE (p:Person {name:'Tom Hanks'})
SET p.oscar = true
RETURN p
```

Write queries

The CREATE Clause

```
CREATE (m:Movie {title:'Mystic River', released:2003})
RETURN m
```

The SET Clause

```
MATCH (m:Movie {title:'Mystic River'})
SET m.tagline = 'We bury our sins here, Dave. We wash them clean.'
RETURN m
```

The CREATE Clause

```
MATCH (m:Movie {title:'Mystic River'})
MATCH (p:Person {name:'Kevin Bacon'})
CREATE (p)-[r:ACTED_IN {role:['Sean']}]->(m)
RETRUN p,r,m
```

ON CREATE and ON MATCH

```
MERGE (p:Person {name: 'Your Name'})
```

```
ON CREATE SET p.created = timestamp(), p.updated= 0
On MATCH SET p.updated = p.updated + 1
RETURN p.created, p.updated
```

Reading data from CSV with Cypher

```
[USING PREIODIC COMMIT] // optional transaction batching
LOAD CSV // load csv data
WITH HEADERS // optionally use first header row as keys in 'row' map
FROM "url" // file:// URL relative to $NEO4J_HOME/import or http://
AS row // return each row of the CSV as list of strings or map
FIELDTERMINATOR ";" // alternative field delimiter

... rest of the Cypher statement ...
```

Tabular CSV records to Graph structure

Type the following command into the query pane in the browser:

```
:play http://guides.neo4j.com/fundamentals/import.html
```

Developing Applications with Neo4j and Cypher

Available APIs

- Remote with Drivers
  - Bolt Binary Protocol
  - Officially Supported Drivers
  - Cypher transactional HTTP Endpoint
  - neo4j.com/developer/language-guides
- Native Java API
  - User Defined Procedures
  - Execute Cypher
  - Core Java API

관련 문서

- Cypher