uc3m | Universidad Carlos III de Madrid

OpenCourseWare

Database

3.2. Introduction to Neo4j

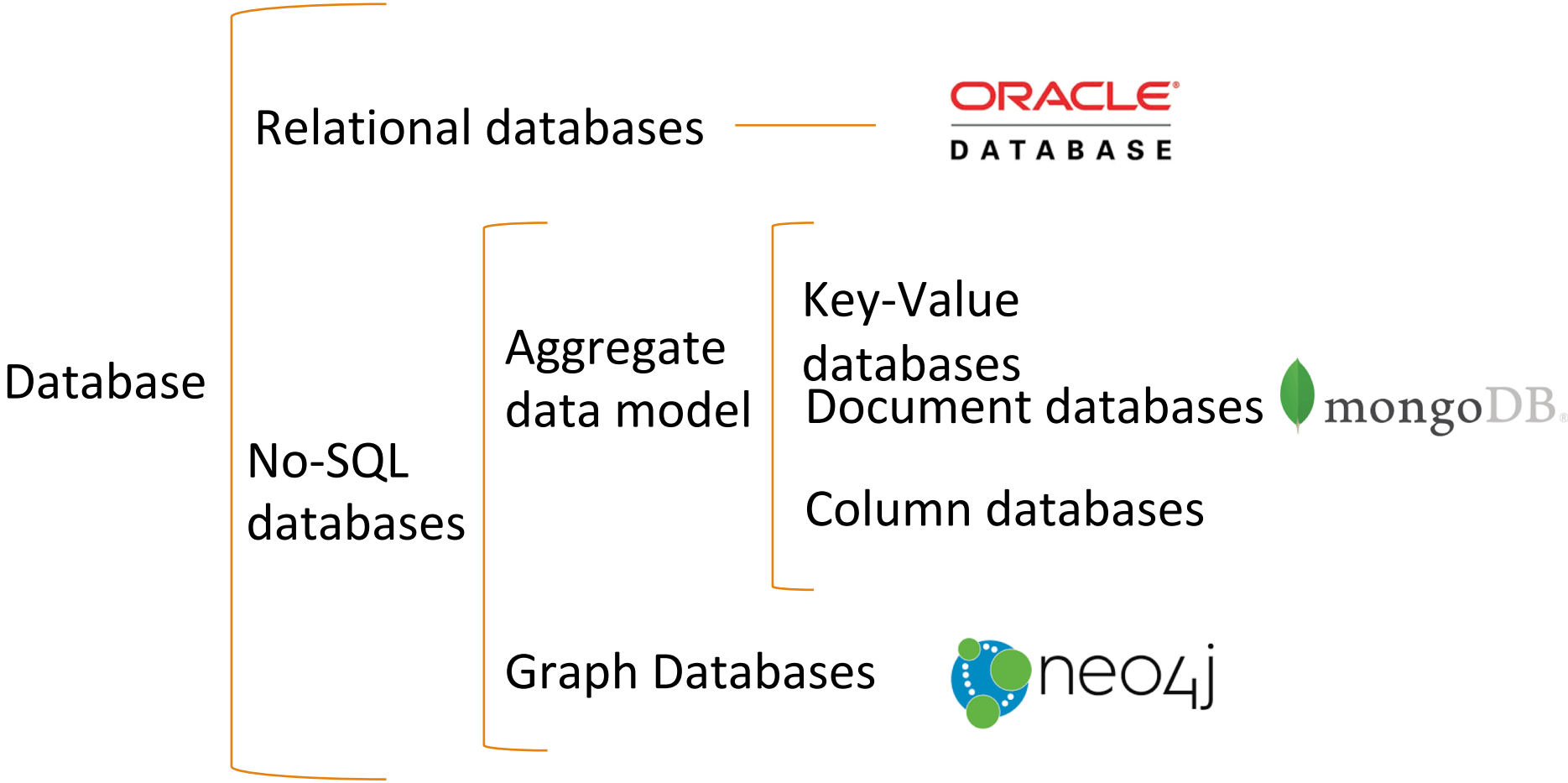
Lourdes Moreno López
Paloma Martínez Fernández
José Luis Martínez Fernández
Rodrigo Alarcón García

HULAT
Human Language & Accessibility Technologies

Departamento de Informática

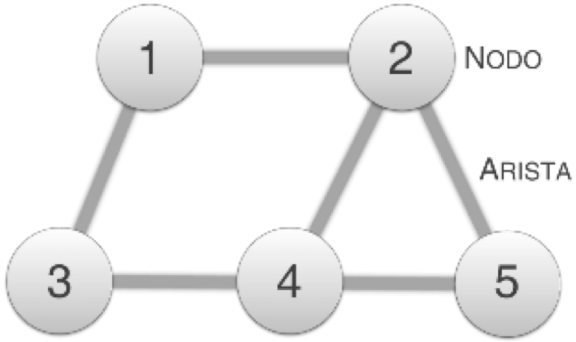
Review

DBMS to study in Course "Database"

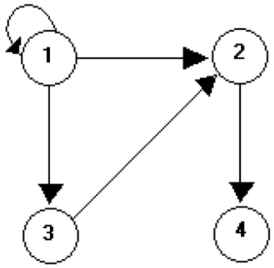


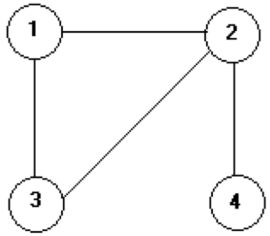
REVIEW
Graph oriented Data Model

- The graph model uses graph structures to represent and store the data



- The graphs have two basic elements:
  - Nodes: represent real-world concepts and objects
  - Edges: explicitly represent the relationships between nodes





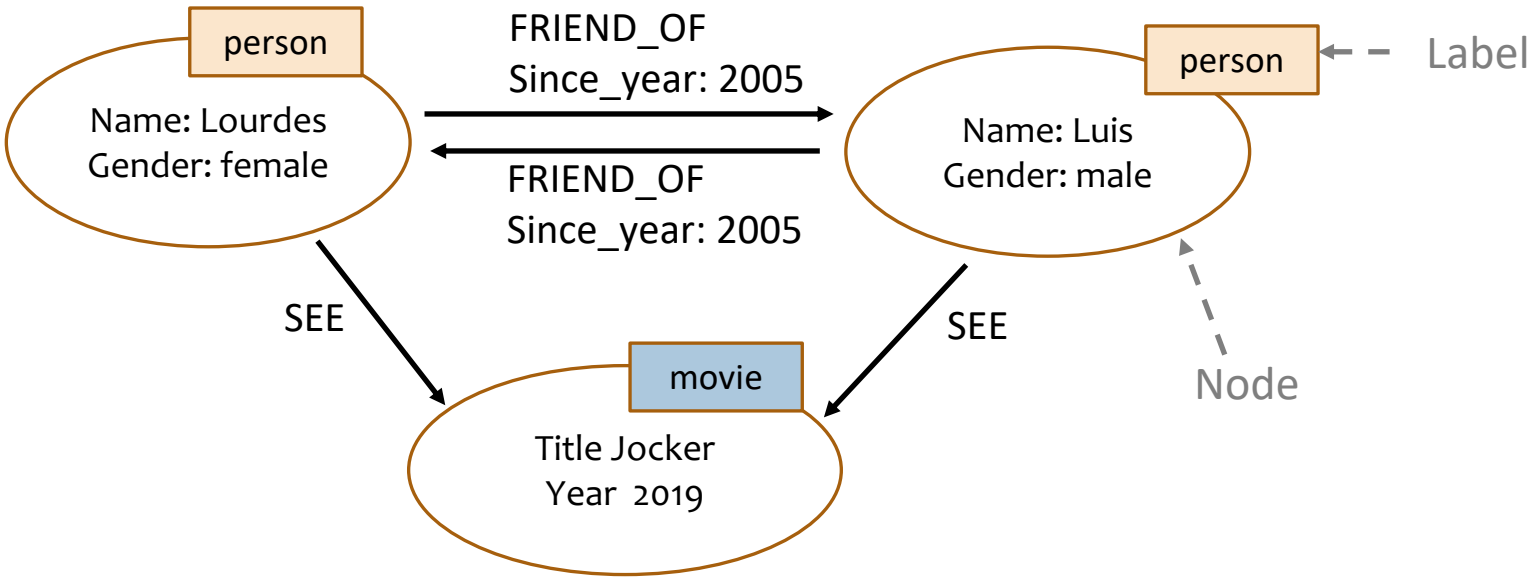
- Types: Directed, not directed

REVIEW
Graph oriented Data Model

- When to use a graph data model?
  - Highly related data: Useful model when the importance of data is its interrelationships (**there are few objects and many relationships**)
  - Useful when information can be represented as a network:
    - Networks (RRSS, logistics, maps, ...)
    - Semantic applications

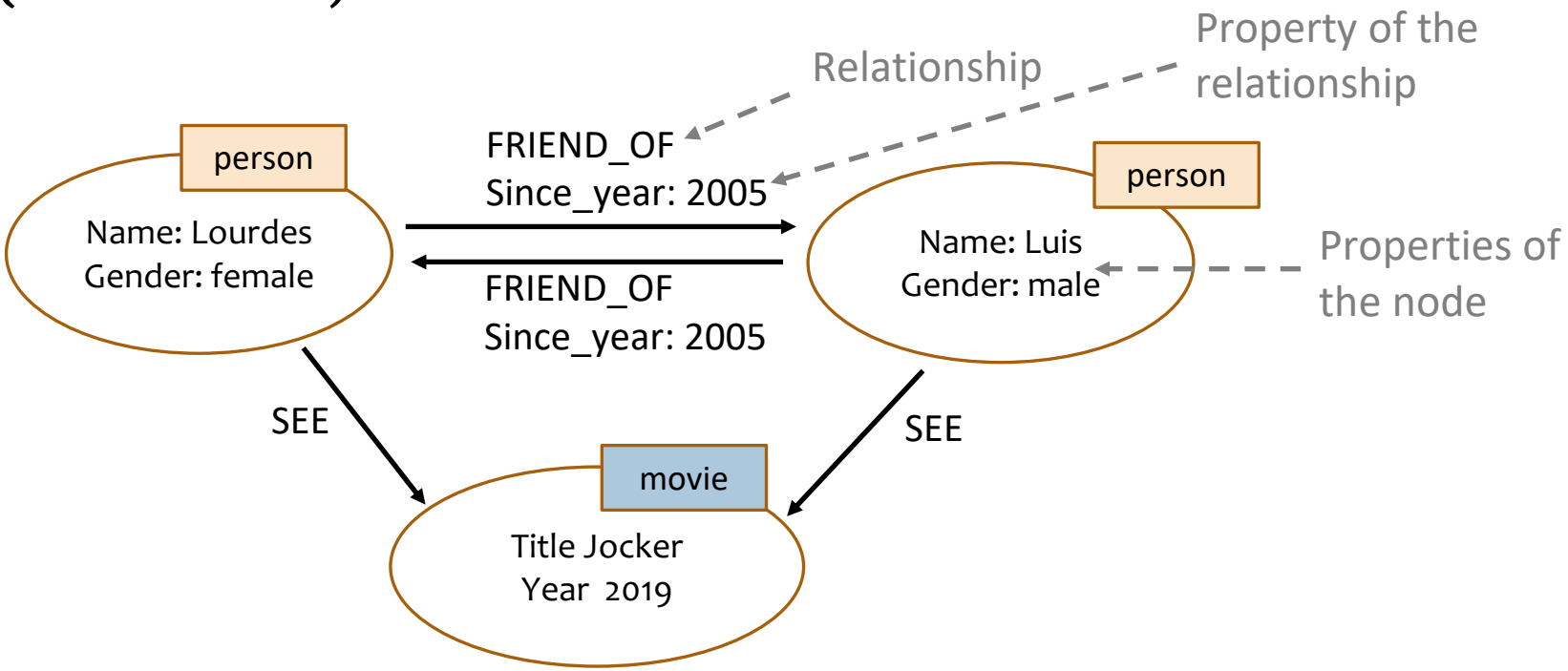
REVIEW
Graph oriented Data Model

- Tagged graphs: Semantics are provided by assigning **labels to nodes and edges**



REVIEW
Graph oriented Data Model

- Property Graphs Tagged: Sometimes tags may be insufficient => **assign properties to nodes and edges** (name: value)



neo4j

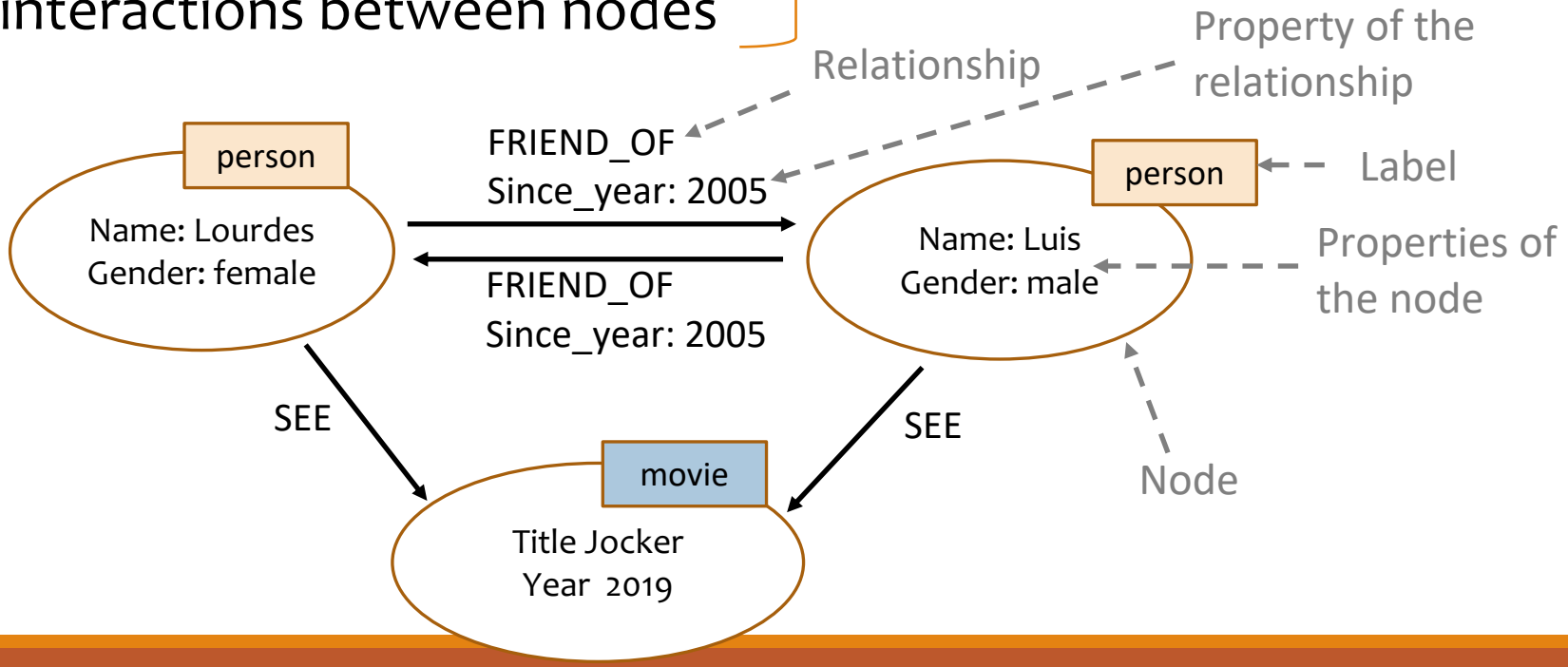
- Graphs oriented NoSQL database
- Store information using graph structures
- Developed by Neo technology, 2017
- It has three versions: Community (free), Enterprise and Government
- Own language of consultation and data manipulation: **Cypher**

Data Modeling Introduction

Data Modeling Introduction

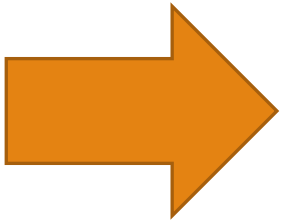
- Nodes: entities, concepts
  - Labels
- Relationships: Connections or interactions between nodes

They can have properties



Data Modeling Introduction

Relational Model
Database
Table, view
Row
Column
Primary key
Foreign Key
Tables join



neo4J
Graph
Node label
Node
Property
Property and uniqueness constraint
Relationship
Graph trasversing

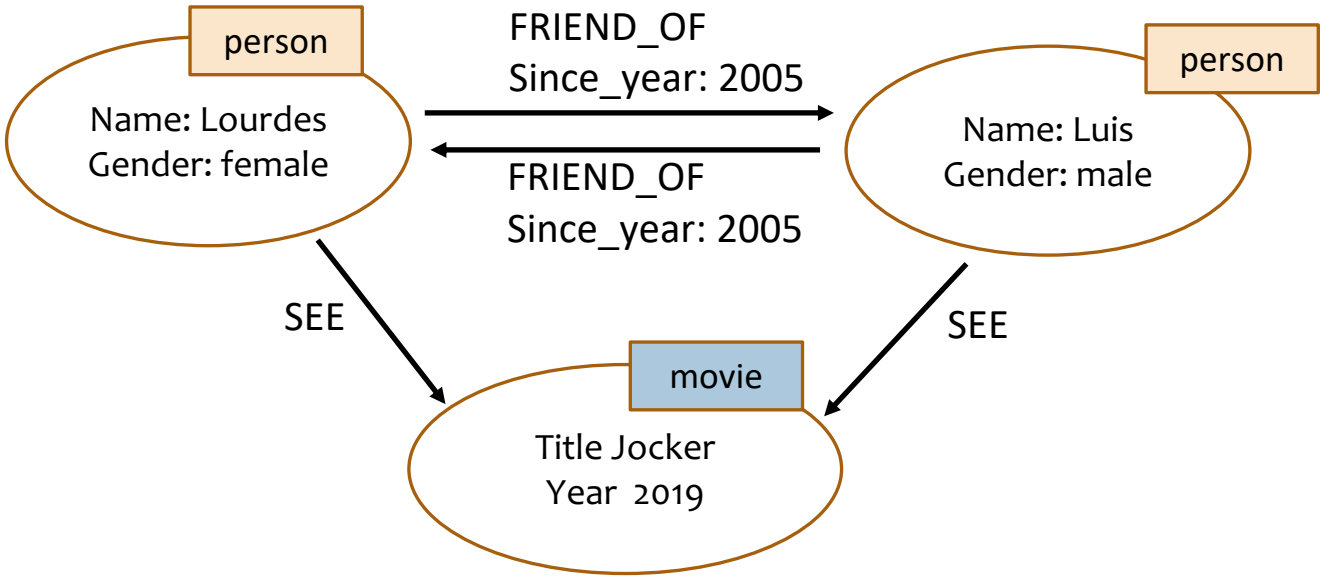
Data Modeling Introduction

- How to create a graph structure that describes the information we want to retrieve from the database?
  - What information do we want to retrieve?
  - What entities or concepts (will be the nodes) and relationships (will be the relationships) do we need to retrieve the information?
  - Queries: **Cypher**

Cypher
Neo4j's graph query language

- Cypher is Neo4j's graph query language that allows users to store and retrieve data from the graph database.
- It is a declarative, SQL-inspired language for describing visual patterns in graphs.
- It is open-source, it an open language specification

Cypher
Example



Cypher
Example

- The CREATE clause allows you to create new nodes.

```
CREATE (LourdesNode: person {name: 'Lourdes', gender: female})
CREATE (LuisNode: person {name: 'Luis', gender: male})
```

- The CREATE statement allows you to create relationships between nodes

```
CREATE (LuisNode)-[:FRIEND_OF {since_year:[2005]}]->(Lourdesnode),
(LourdesNode)-[:FRIEND_OF {since_year:[2005]}]->(LuisNode)
```

Cypher
Example

- The MATCH clause allows you to make queries with conditions about the data of the nodes and relationships.
- Recover data from all nodes.

```
MATCH(p) RETURN p
```

- Retrieve all nodes labeled as "person“

```
MATCH(p: person) RETURN p
```

- Retrieve the name and gender of all data labeled 'person'

```
MATCH(p: person) RETURN p.Name, p.Gender
```

Cypher
Example

- Retrieve the name and gender of people named ‘Lourdes’

```
MATCH(p: person) WHERE p.Name=‘Lourdes’ RETURN p.Name, p.Gender
```

- Retrieve the name and gender of people named 'Lourdes' or male gender

```
MATCH(p: person) WHERE p.Name=‘Lourdes’ OR p.Gender=‘male’
RETURN p.Name, p.Gender
```

- Retrieve the name and gender of people named 'Lourdes’ and female gender

```
MATCH(p: person) WHERE p.Name=‘Lourdes’ AND p.Gender=‘female’
RETURN p.Name, p.Gender
```

Cypher
Example

- Retrieve the name and gender of people named ‘Lourdes’

```
MATCH(p: person) WHERE p.Name=‘Lourdes’ RETURN p.Name, p.Gender
```

- Retrieve the name and gender of people named 'Lourdes' or male gender

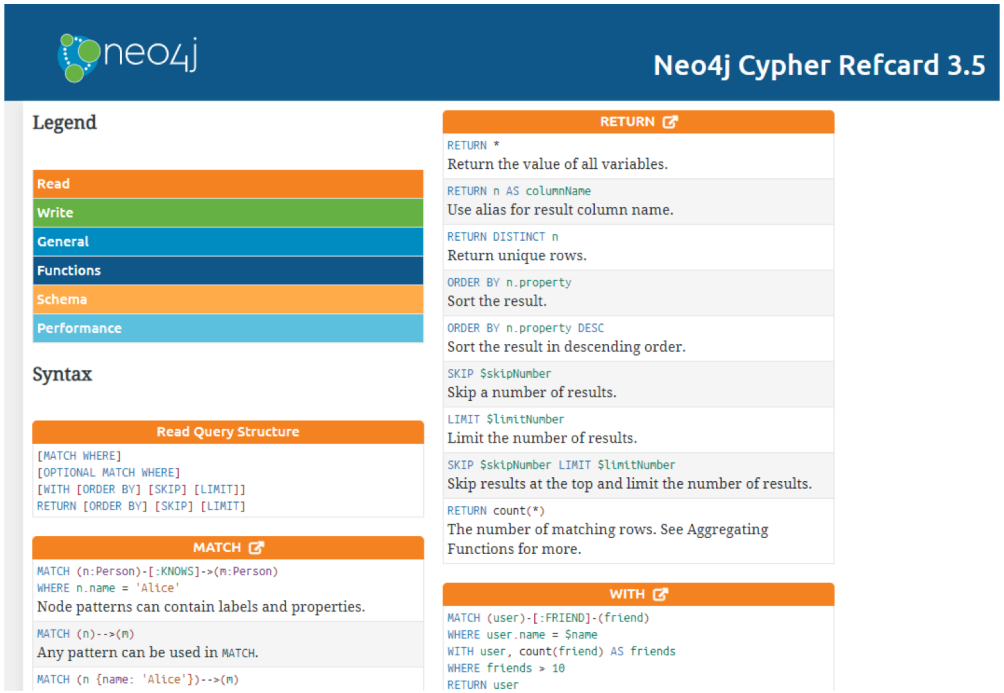
```
MATCH(p: person) WHERE p.Name=‘Lourdes’ OR p.Gender=‘male’
RETURN p.Name, p.Gender
```

- Retrieve the name and gender of people named 'Lourdes’ and female gender

```
MATCH(p: person) WHERE p.Name=‘Lourdes’ AND p.Gender=‘female’
RETURN p.Name, p.Gender
```

Cypher

- More ….
  - Neo4j Cypher Refcard 3.5: https://neo4j.com/docs/cypher-refcard/current/



neo4j

Neo4j Cypher Refcard 3.5

Legend

Read
Write
General
Functions
Schema
Performance

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)
```

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 Aggregating Functions for more.

WITH

```
MATCH (user)-[:FRIEND]-(friend)
WHERE user.name = $name
WITH user, count(friend) AS friends
WHERE friends > 10
RETURN user
```

© neo4j

References

- The Neo4j Cypher Manual v3.5. Copyright © 2021 Neo4j, Inc. https://neo4j.com/docs/cypher-manual/3.5/