Introduction to Graph Databases, Cypher Query Language & ~~Examples in PHP~~

Sören Klein / Syndesi

What is a Graph Database?
Use Cases
Cypher Basics
Question Time
Sources, Tools, Credits & Next Steps

What is a Graph Database?

What is a Graph Database?

NoSQL: Everything except SQL: Document, column-family, key-value, time-series, ... and graph databases :D

Key feature of a graph database: It stores graphs and their components, nodes and edges.
Often optimized for traversing relations.

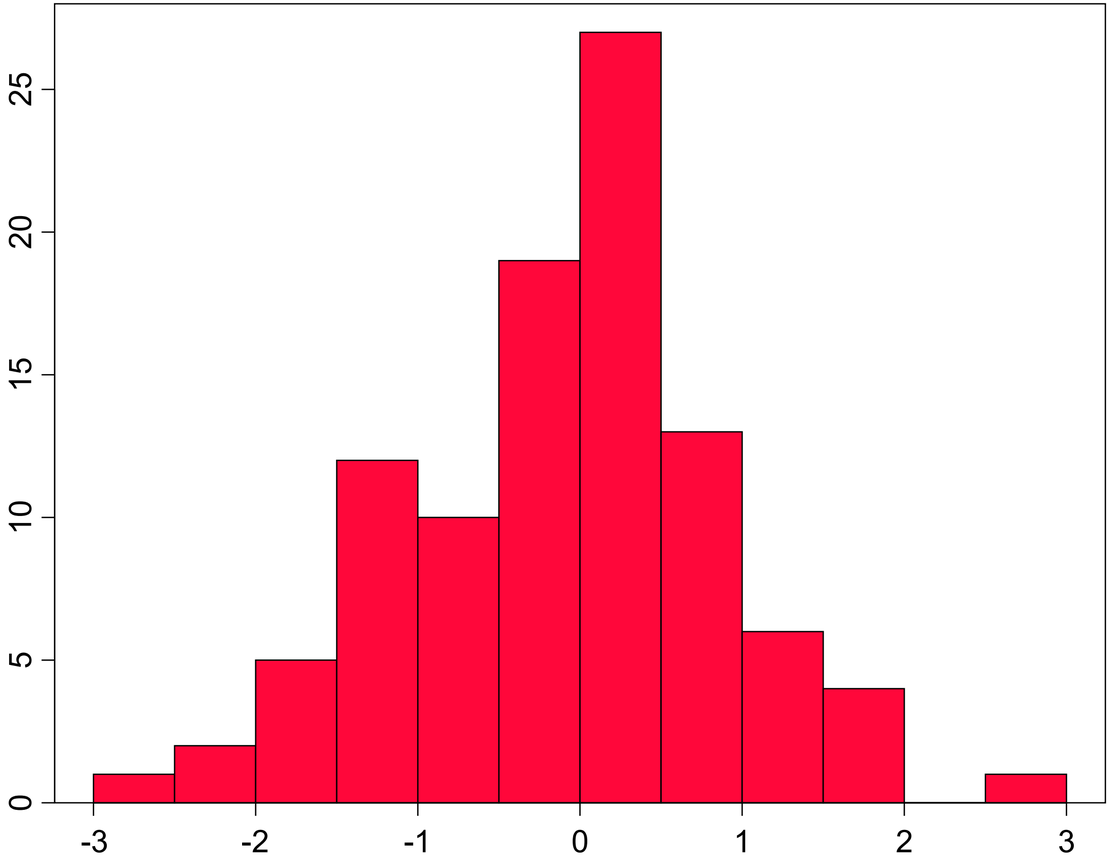
What is a Graph Database?

Different types of graph databases:

- Native vs non-native
- Multi-model vs specialized
- Query language support: Cypher, Gremlin, SPARQL, GQL, ...

What is a Graph Database?

What about charts?
They are not graphs.



Source: Image from Wikipedia

What is a Graph Database?

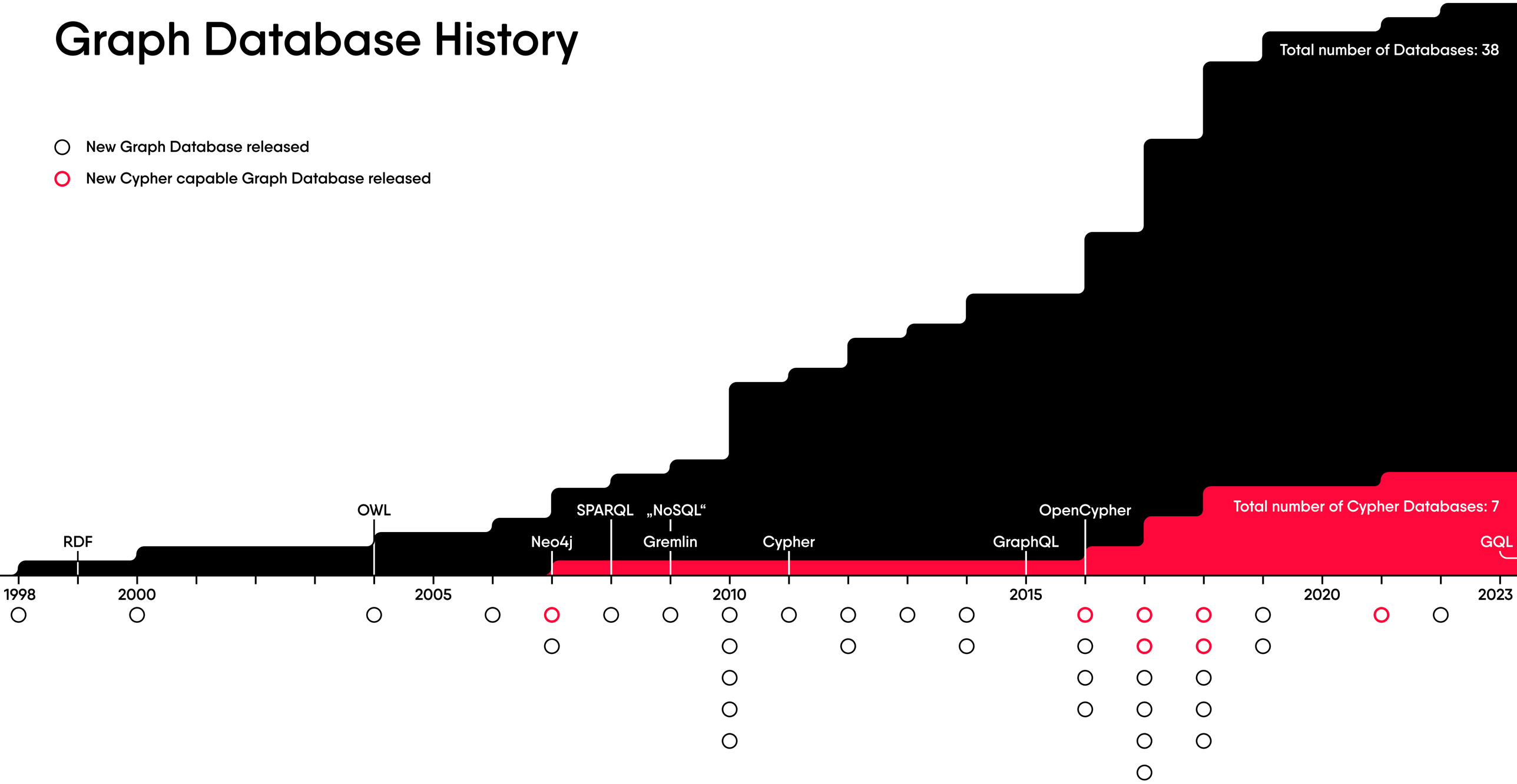
Benefits of using native graph databases:

- Schema-less like MongoDB, but with relations
- Traversing relations is fast
- Every node can relate to every other node - try that in SQL :P

Downsides:

- Community is way smaller, but is growing fast

Graph Database History

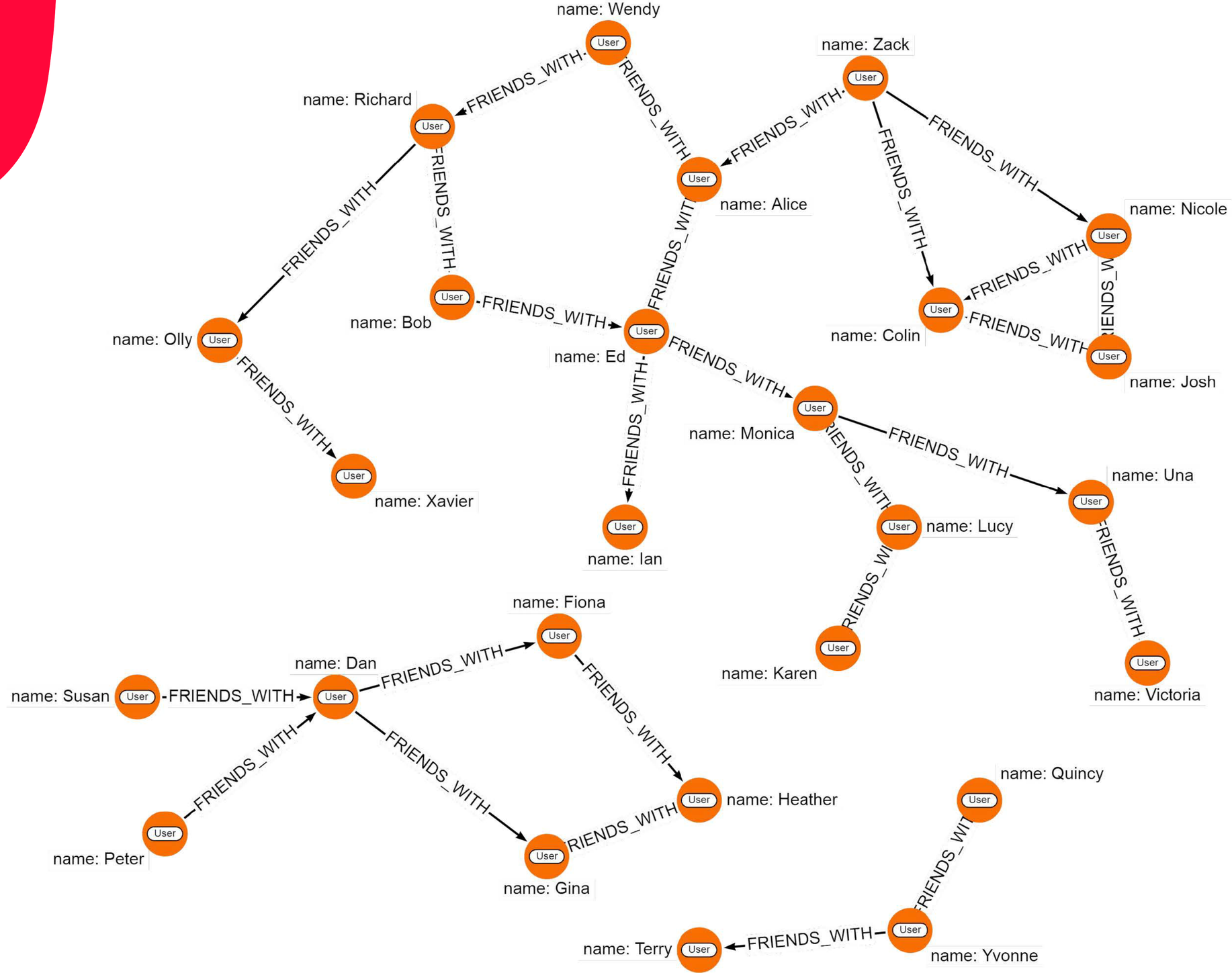


Created by Sören Klein / Syndesi. Source: db-engines.com

Use Cases

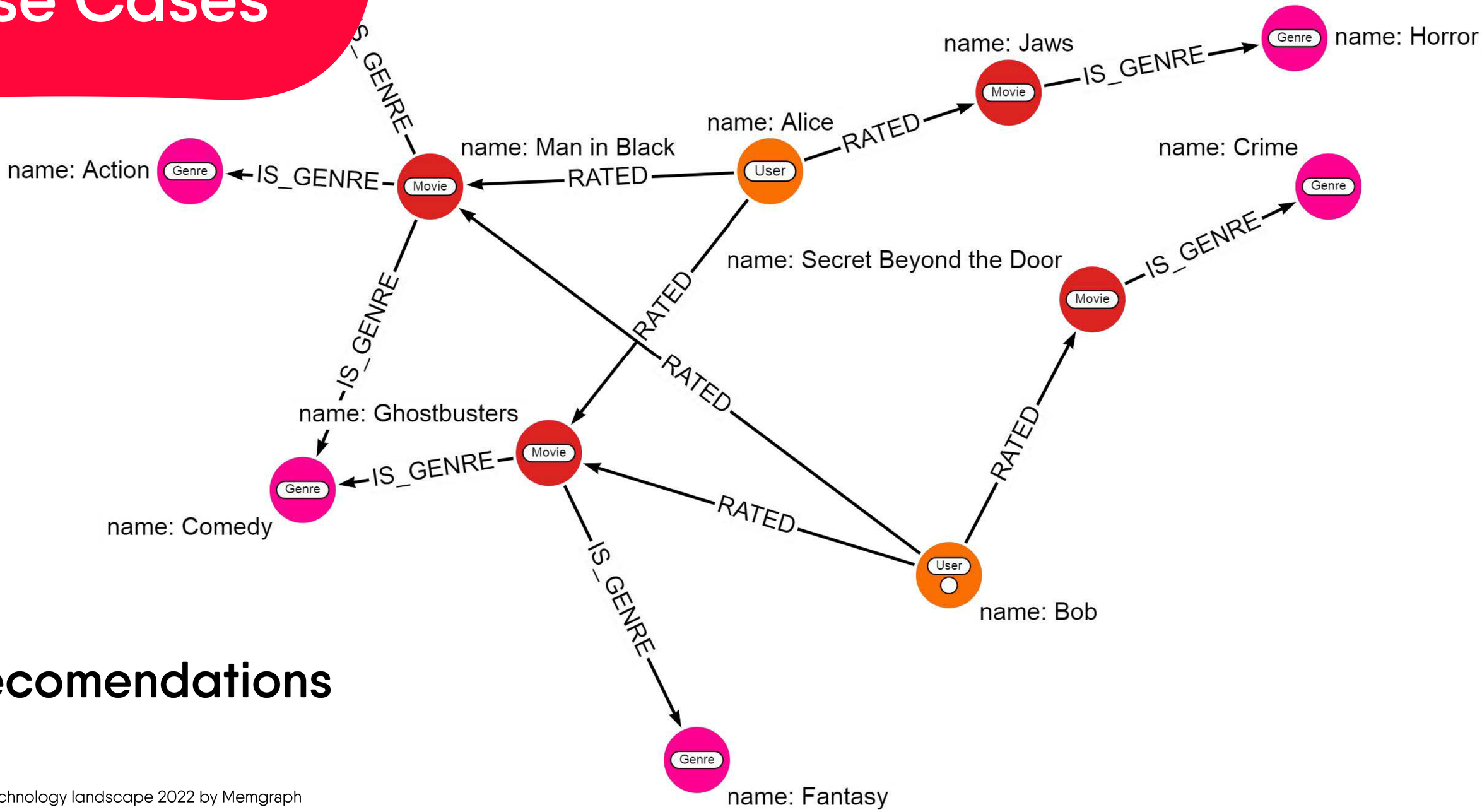
Use Cases

Social networks



Source: Graph technology landscape 2022 by Memgraph

Use Cases

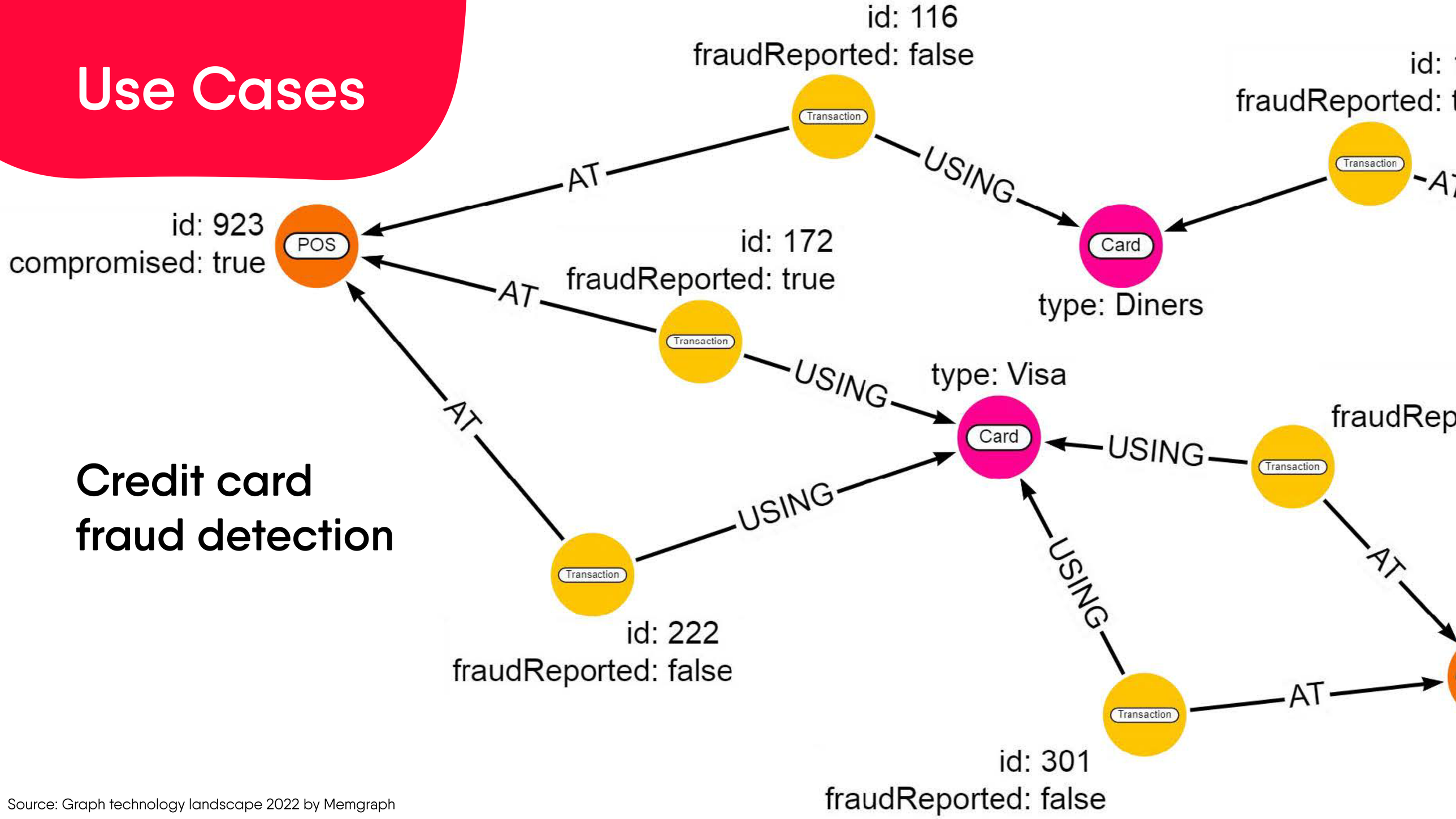


Recomendations

Source: Graph technology landscape 2022 by Memgraph

Use Cases

Credit card fraud detection



Source: Graph technology landscape 2022 by Memgraph

Cypher Basics

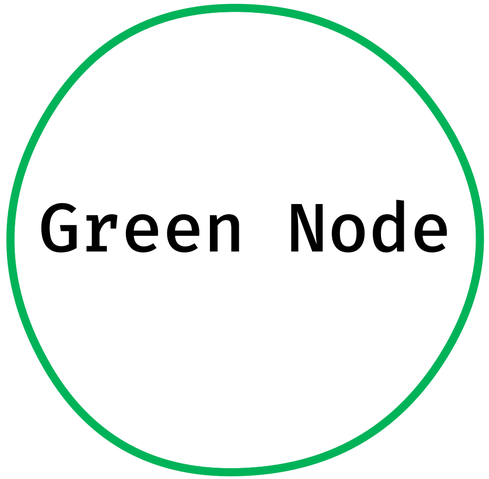
Cypher Basics

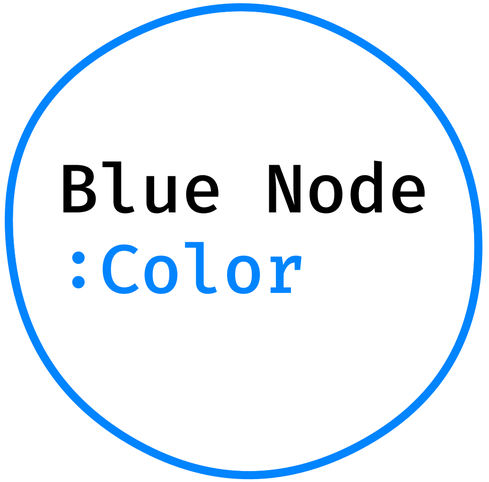
Nodes are the "rows" of graph databases

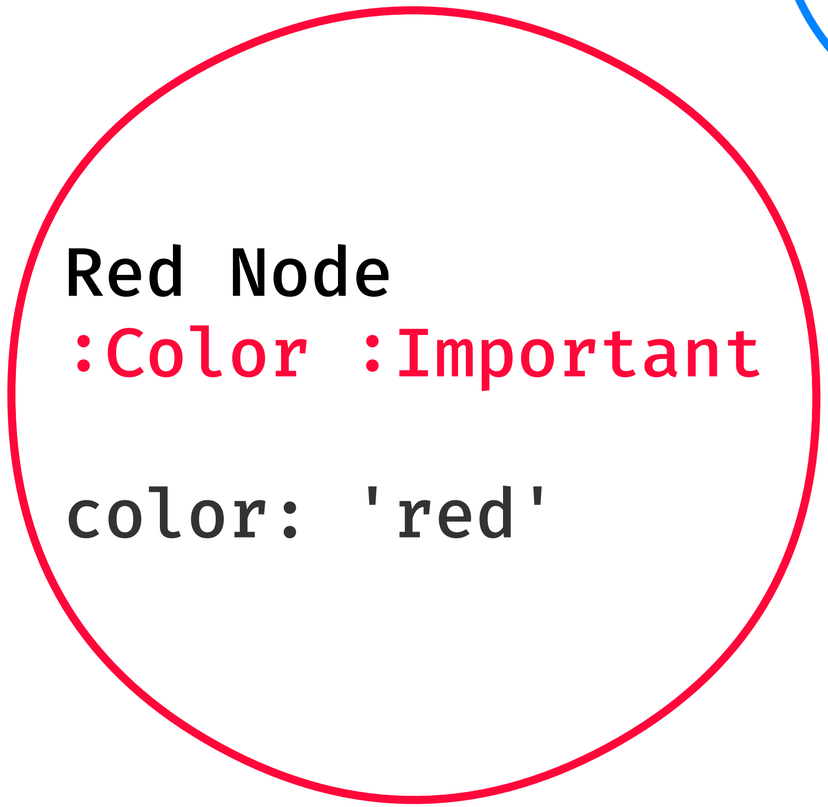
Nodes can contain Labels

Labels are written in CamelCase

Nodes can have properties







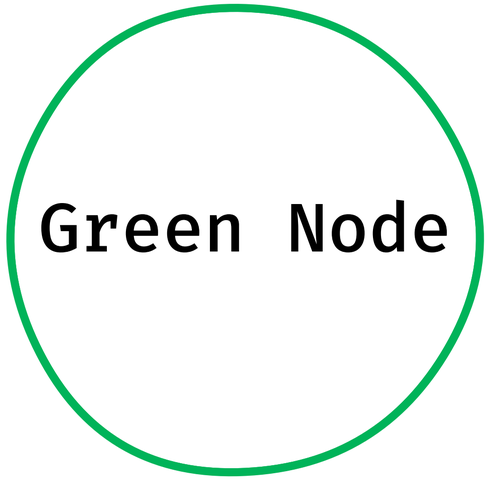
Cypher Basics

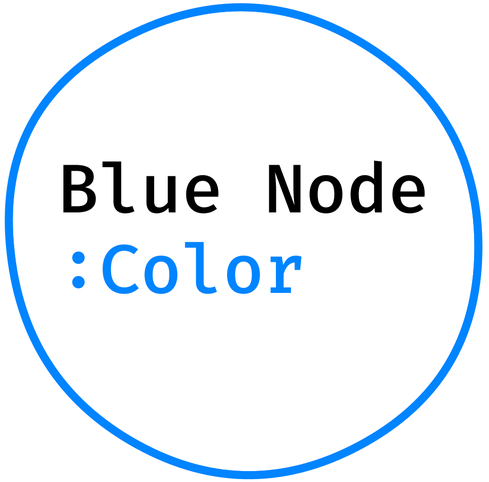
```
(variableName:Label1:Label2 {
  key1: 'value 1',
  key2: 'value 2'
})
```

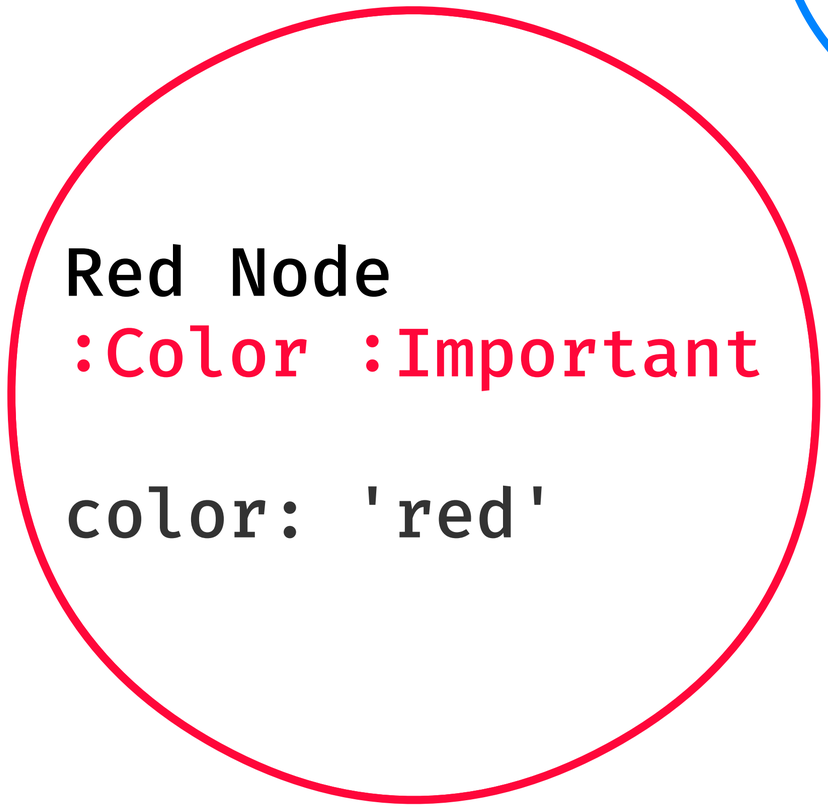
```
(greenNode)
```

```
(blueNode:Color)
```

```
(redNode:Color:Important {
  color: 'red'
})
```







Cypher Basics

Relations are the "foreign keys"

Relations have a direction

Relations must contain one Type

Types are written in SCREAMING_SNAKE_CASE

Relations can have properties

Relations always start and end at Nodes

Cypher Basics

```
()-[variableName:TYPE {
  key1: 'value 1',
  key2: 'value 2'
}]→()
```

```
()-[redRelation:RED_RELATION]→()
```

```
()-[greenRelation:GREEN_RELATION {
  color: 'green'
}]→()
```

Red Relation
:RED_RELATION

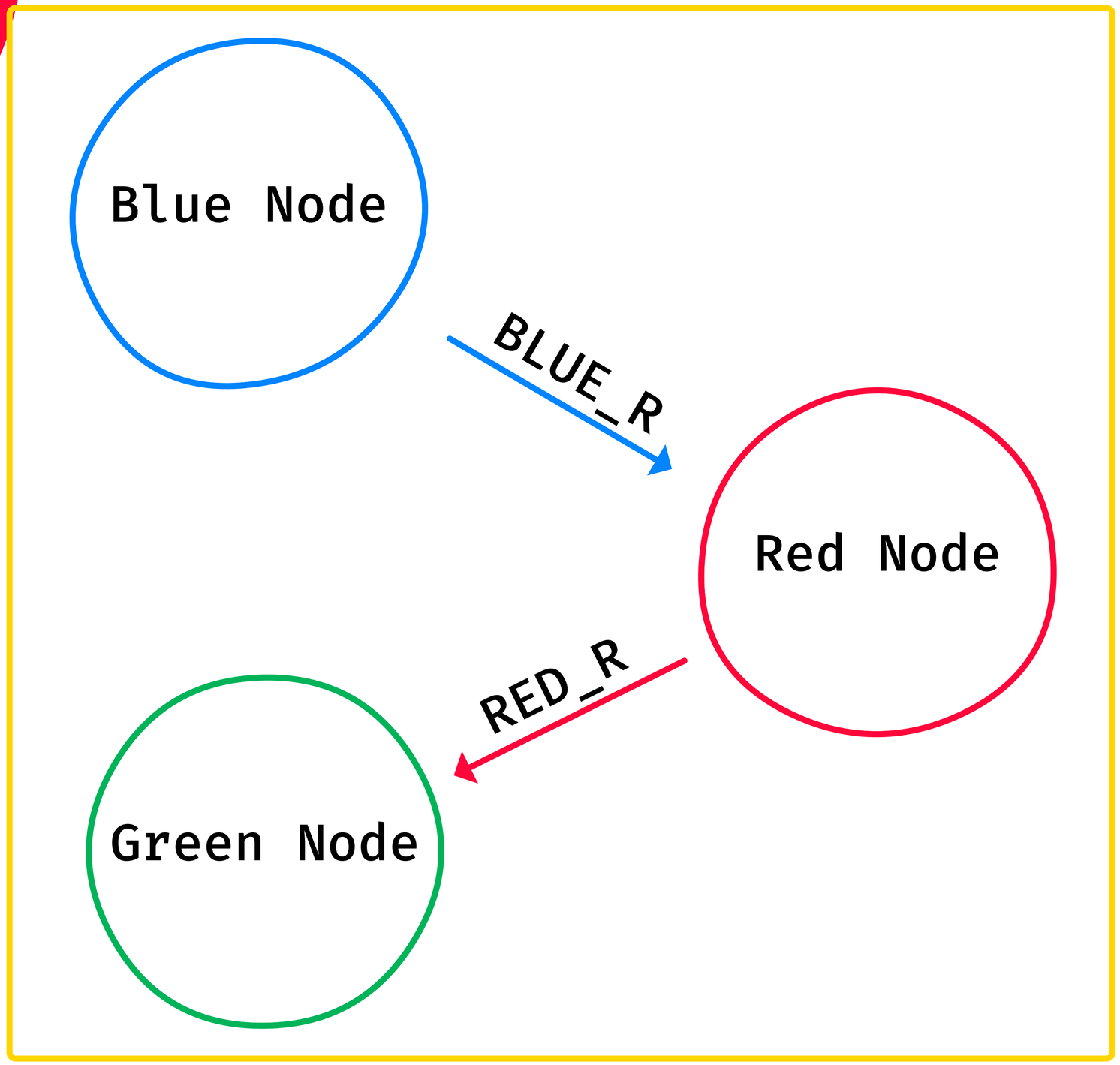
Green Relation
:GREEN_RELATION

color: 'green'

Cypher Basics

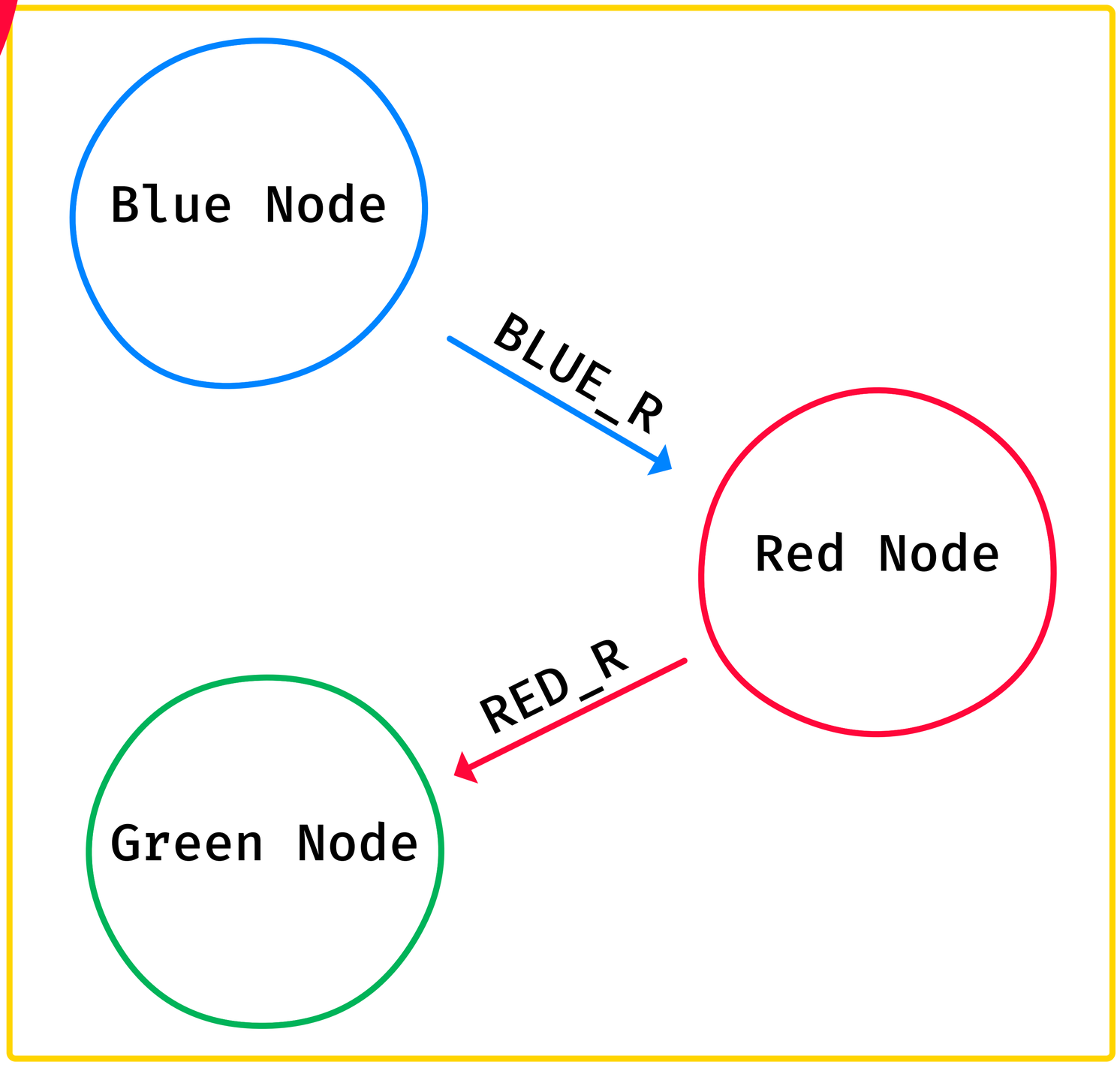
Paths are indirect data types

Paths contain Nodes and Relations



Cypher Basics

```
path = (blueNode)
  -[blueRelation]→
(redNode)
  -[redRelation]→
(greenNode)
```

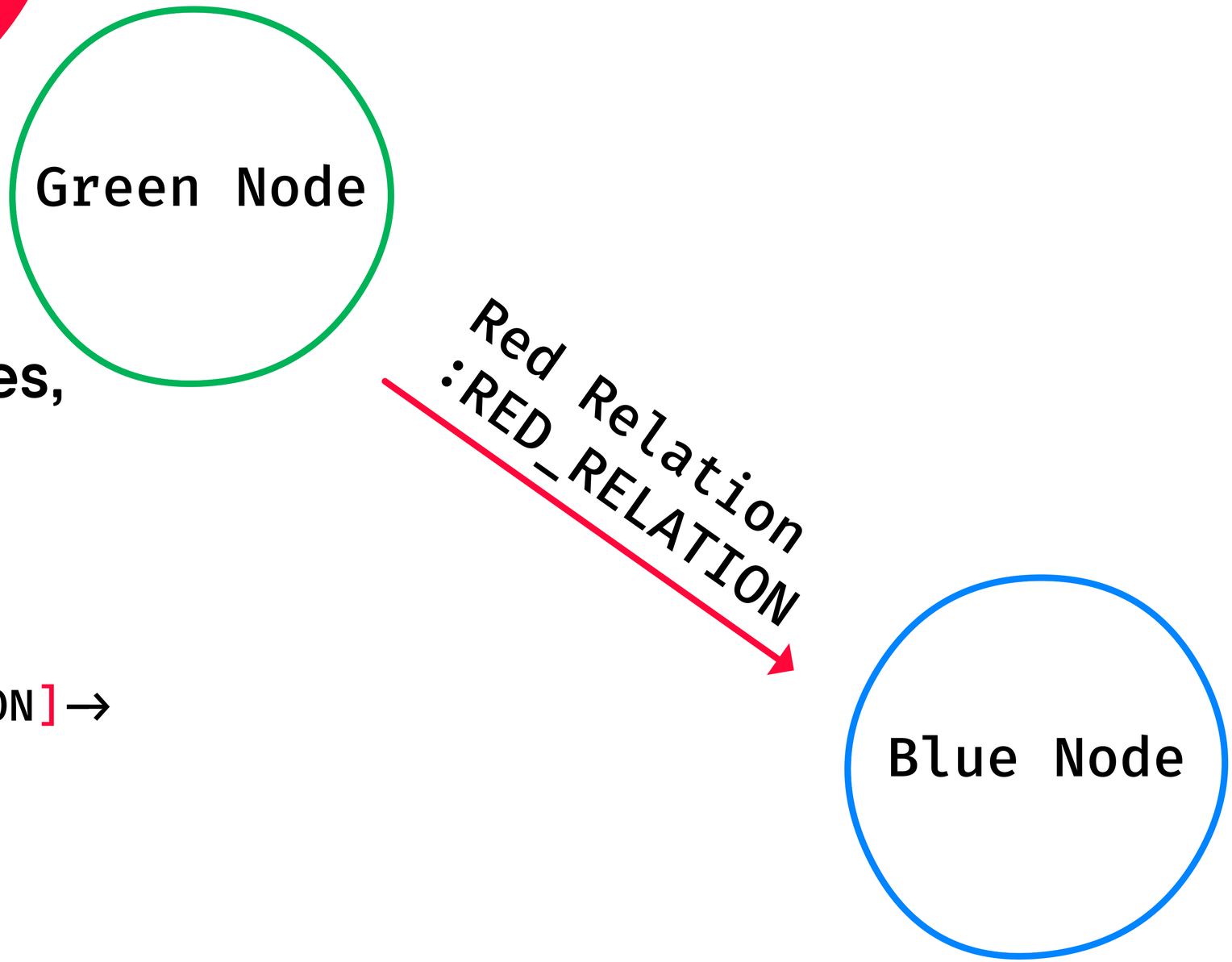


Cypher Basics

RETURN is used to get the value of variables and statements.

```
RETURN 1
```

Cypher Basics



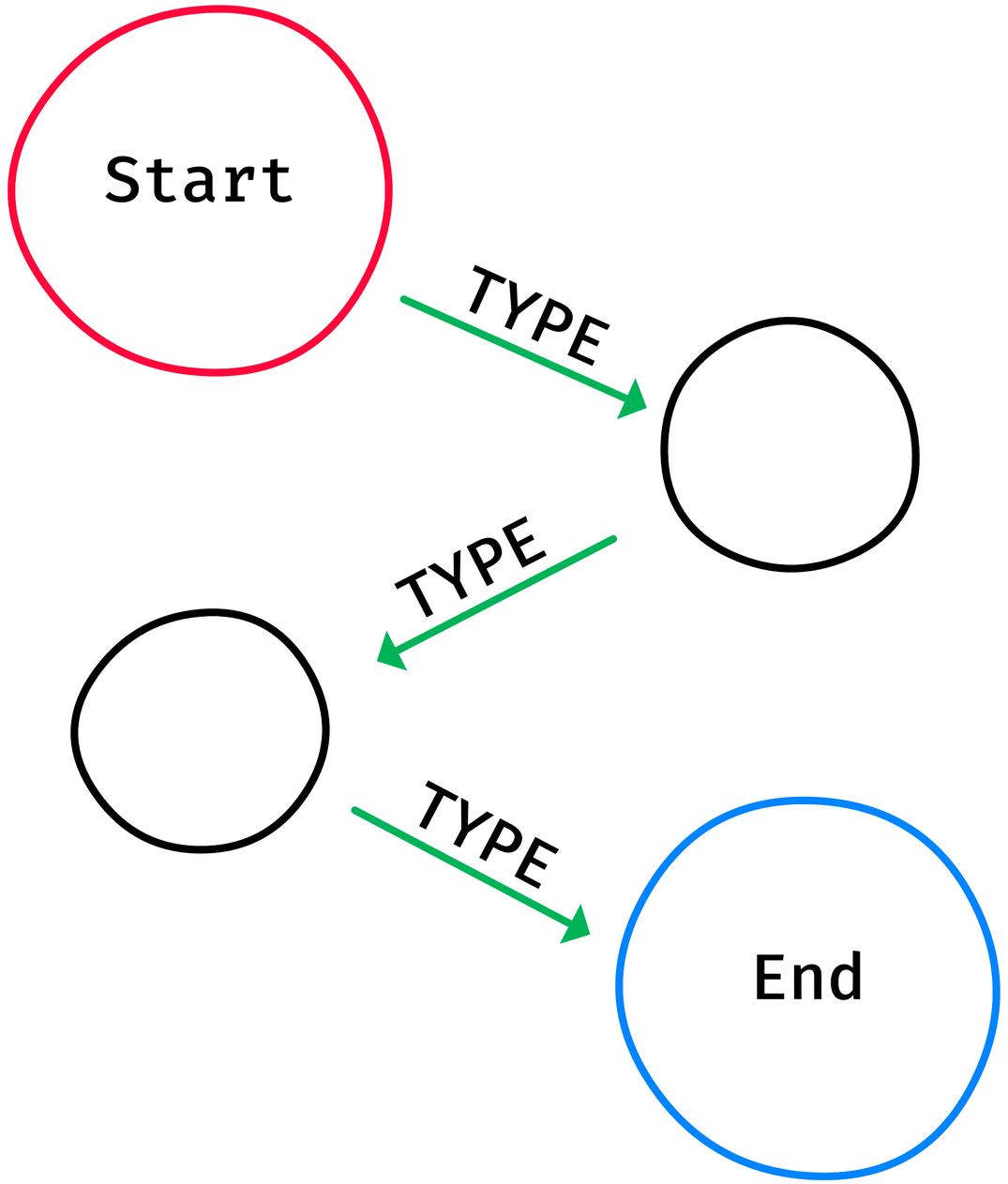
MATCH tries to find Nodes, Relations and Paths.

```
MATCH
  (greenNode)
  -[redRelation:RED_RELATION]→
  (blueNode)
RETURN
  greenNode,
  redRelation,
  blueNode
```

Cypher Basics

Variable-length pattern matching can be used to create powerful queries across many relations. Simple recursion is possible.

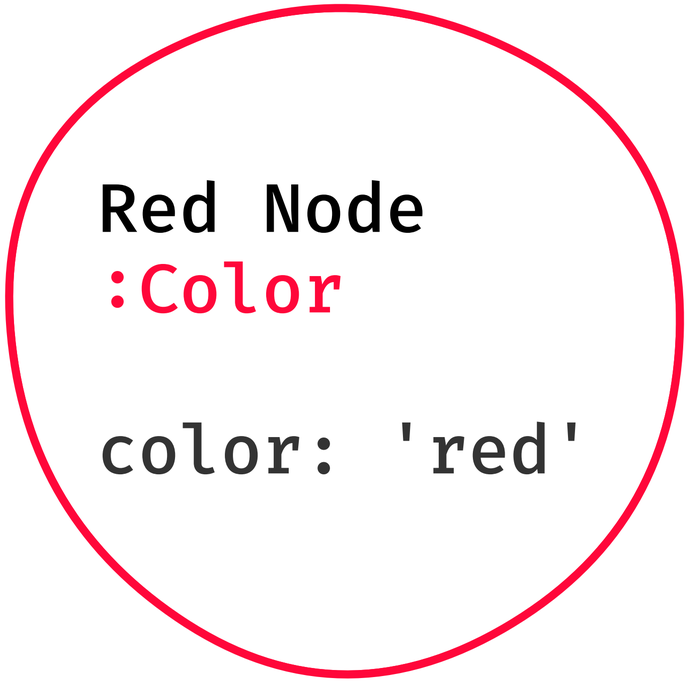
```
MATCH
  path = (start)-[:TYPE*]→(end)
RETURN
  path
```



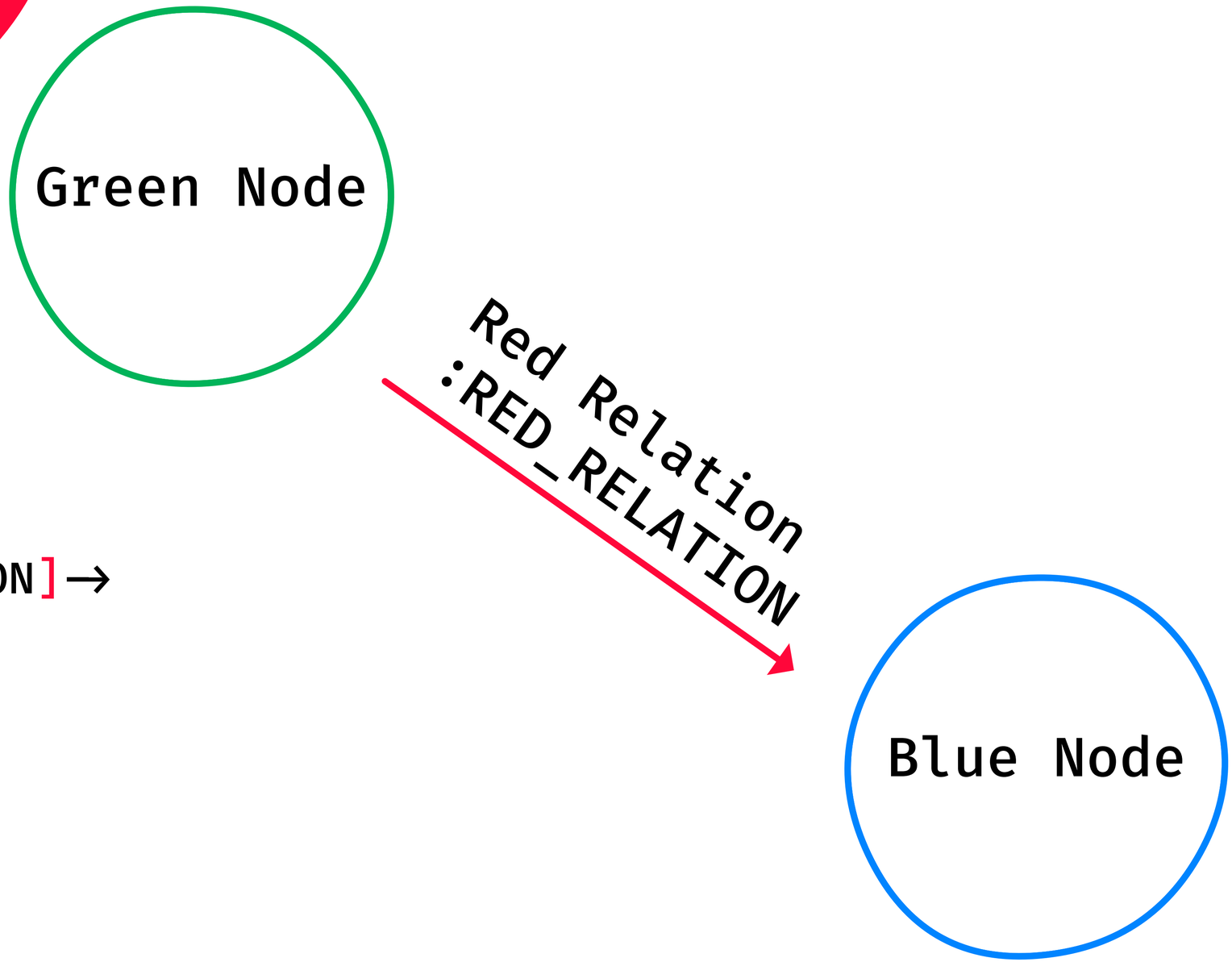
Cypher Basics

CREATE always creates a new Node or Relation

```
CREATE (redNode:Color {
  color: 'red'
})
RETURN redNode
```



Cypher Basics

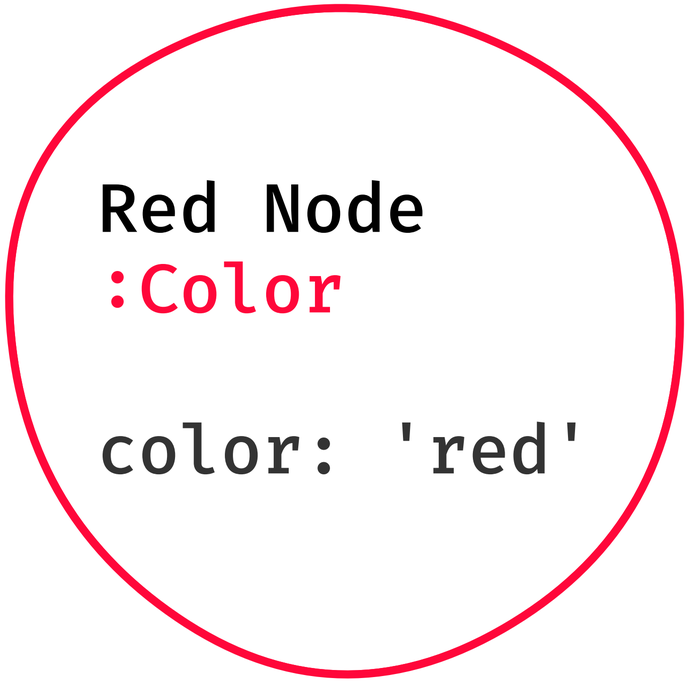



```
CREATE
  (greenNode)
  -[redRelation:RED_RELATION]→
  (blueNode)
RETURN
  greenNode,
  redRelation,
  blueNode
```

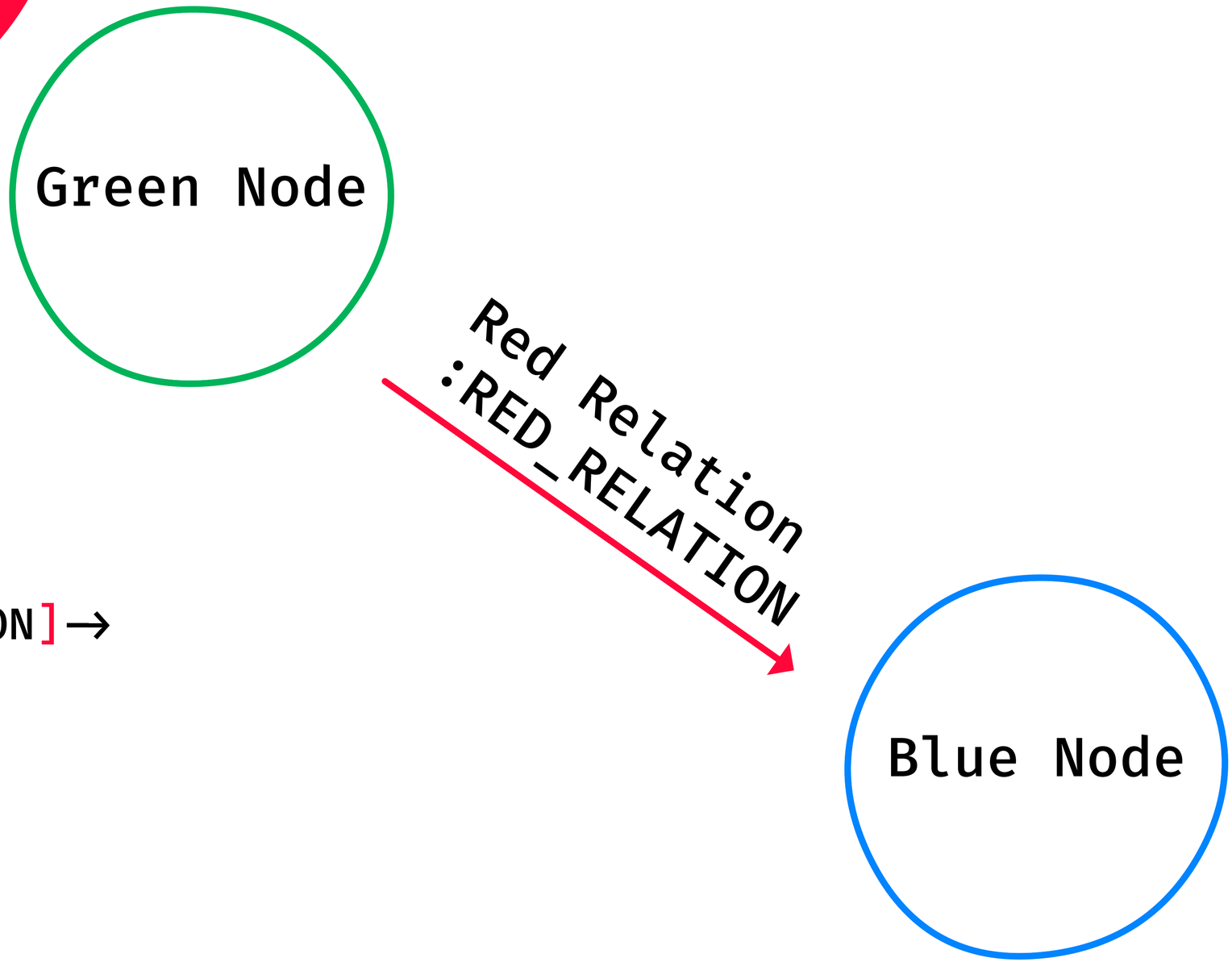
Cypher Basics

MERGE tries to find existing Nodes or Relations. If nothing is found, it creates them.

```
MERGE (redNode:Color {
  color: 'red'
})
RETURN redNode
```



Cypher Basics

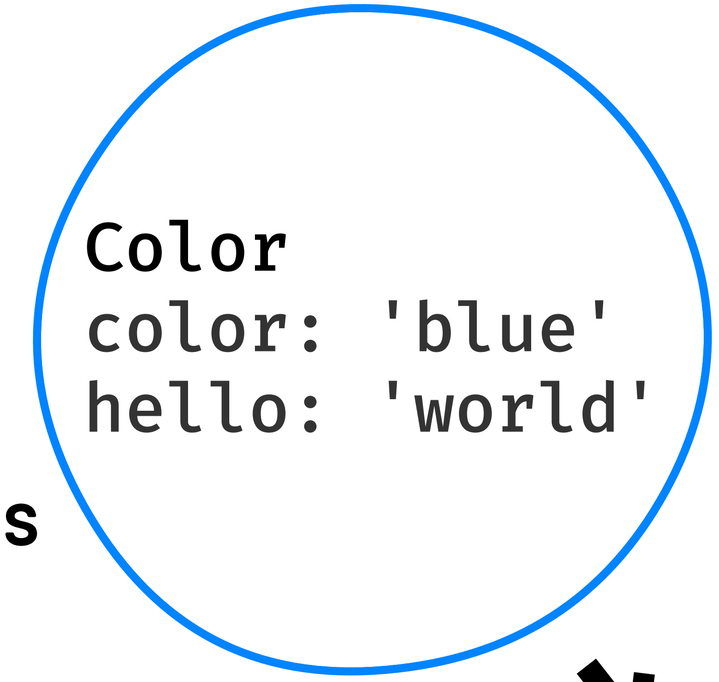


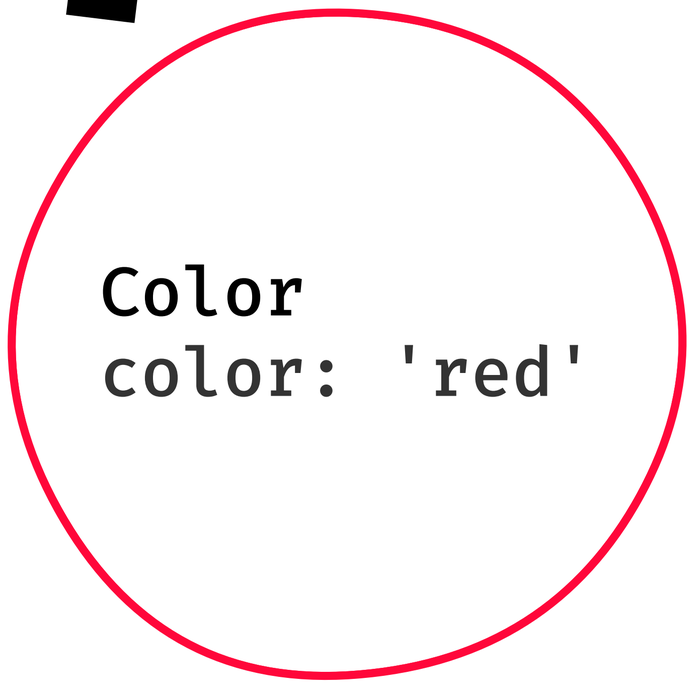

```
MERGE
  (greenNode)
  -[redRelation:RED_RELATION]→
  (blueNode)
RETURN
  greenNode,
  redRelation,
  blueNode
```

Cypher Basics

SET can add or replace properties of elements.
Setting properties to NULL removes them.

```
MATCH
  (color)
SET
  color.color = 'red',
  color.hello = NULL
```

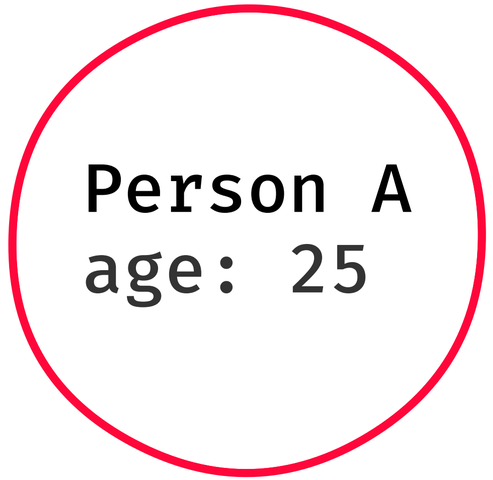


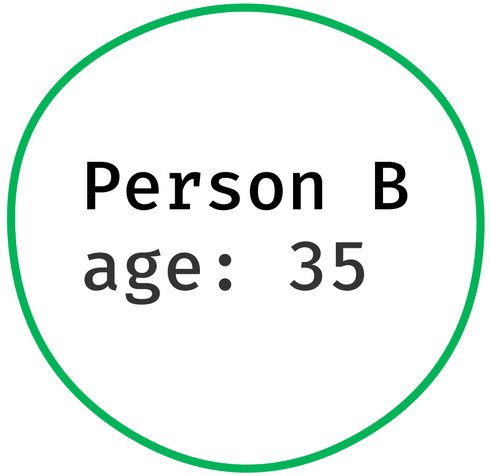


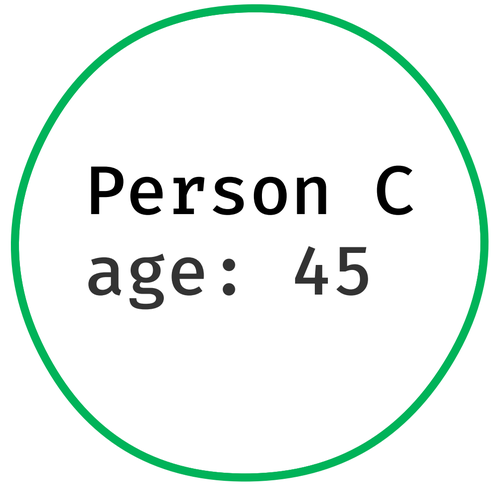
Cypher Basics

WHERE can filter results.

```
MATCH
  (people:Person)
WHERE
  people.age > 30
RETURN
  people
```



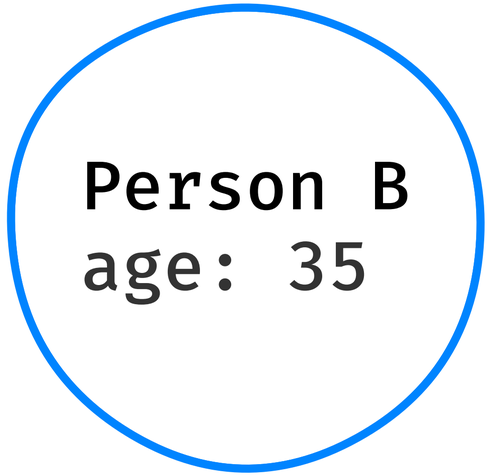


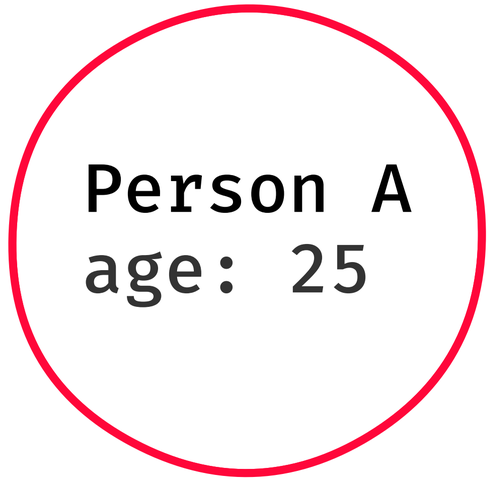


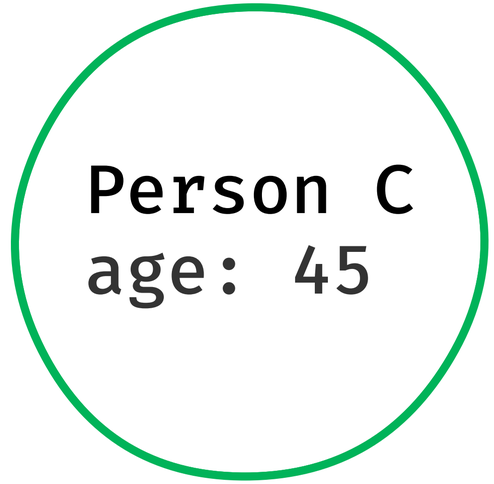
Cypher Basics

ORDER BY is used to order results.

```
MATCH
  (people:Person)
RETURN
  people
ORDER BY
  people.age DESC
```



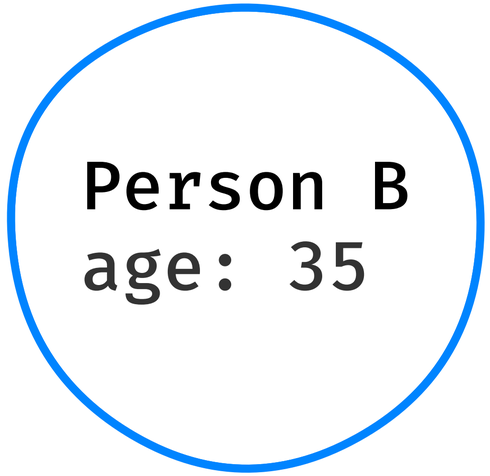


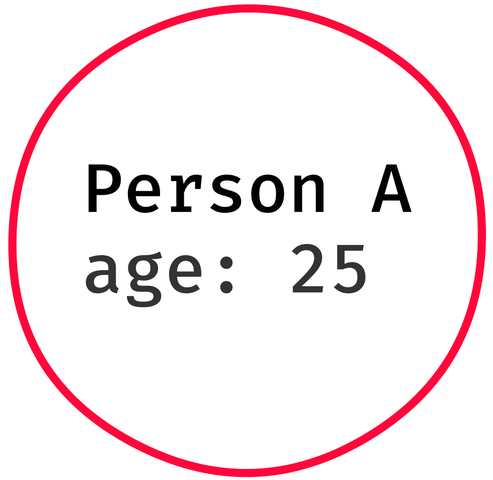


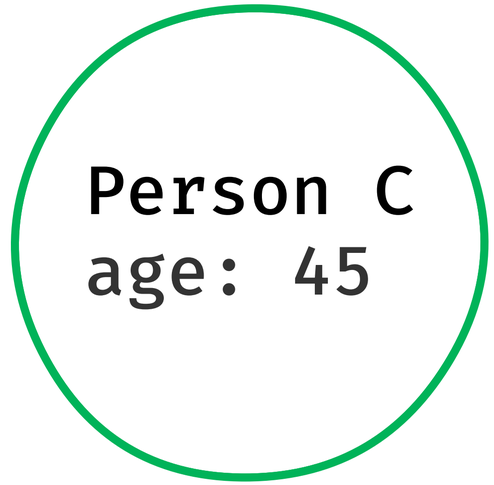
Cypher Basics

LIMIT returns just the first n elements of the result.
SKIP jumps over the first n elements of the result.
Note: ORDER BY is required.

```
MATCH (people:Person)
RETURN people
ORDER BY people.age
SKIP 1
LIMIT 1
```







Cypher Basics

DELETE removes elements from the database.
Relations which are connected to Nodes can be automatically deleted to.

```
MATCH (node)
DELETE node
```

```
MATCH (n)-[relation]→(m)
DELETE relation
```

```
MATCH (node)
DETACH DELETE node
```

Cypher: Going deeper

Constraints & Indexe can be used to improve performance drastically. These features are not part of the OpenCypher specification, and every database handles them differently.

EXPLAIN & PROFILE can be used to debug queries and to identify bottlenecks.

Sources, Tools, Credits & Next Steps

Sources

OpenCypher specification: opencypher.org

Neo4j Cypher Manual: neo4j.com/docs/cypher-manual

Presentations made by Memgraph: memgraph.com

DB Engines: db-engines.com

Tools

Illustrator

MagicPattern: Blob Generator

ChatGPT for initial feedback - cross checked

Credits

Yolande Poirier, Neo4j

Katarina Šupe, Memgraph

Michael Hunger, Neo4j

ChemnitzerWebDevs

Friends

Next Steps

Learn: neo4j.com/docs/getting-started
memgraph.com/docs/memgraph

Discord: discord.gg/neo4j
discord.gg/memgraph

Be awesome! :D