Cypher (Neo4J's Query Language)

Ian Moore

What is Cypher?

- Cypher is “a declarative graph query language that allows for expressive and efficient querying and updating of the graph store”.

- Designed to make simple things easy, and complex things possible

What is Cypher?

- Cypher is based on English prose and iconography
  - This helps to make queries more self-explanatory
- Focuses on *what* to retrieve from a graph, as opposed to *how* to retrieve it
  - Contrasts with Java, Gremlin, etc

What is Cypher?

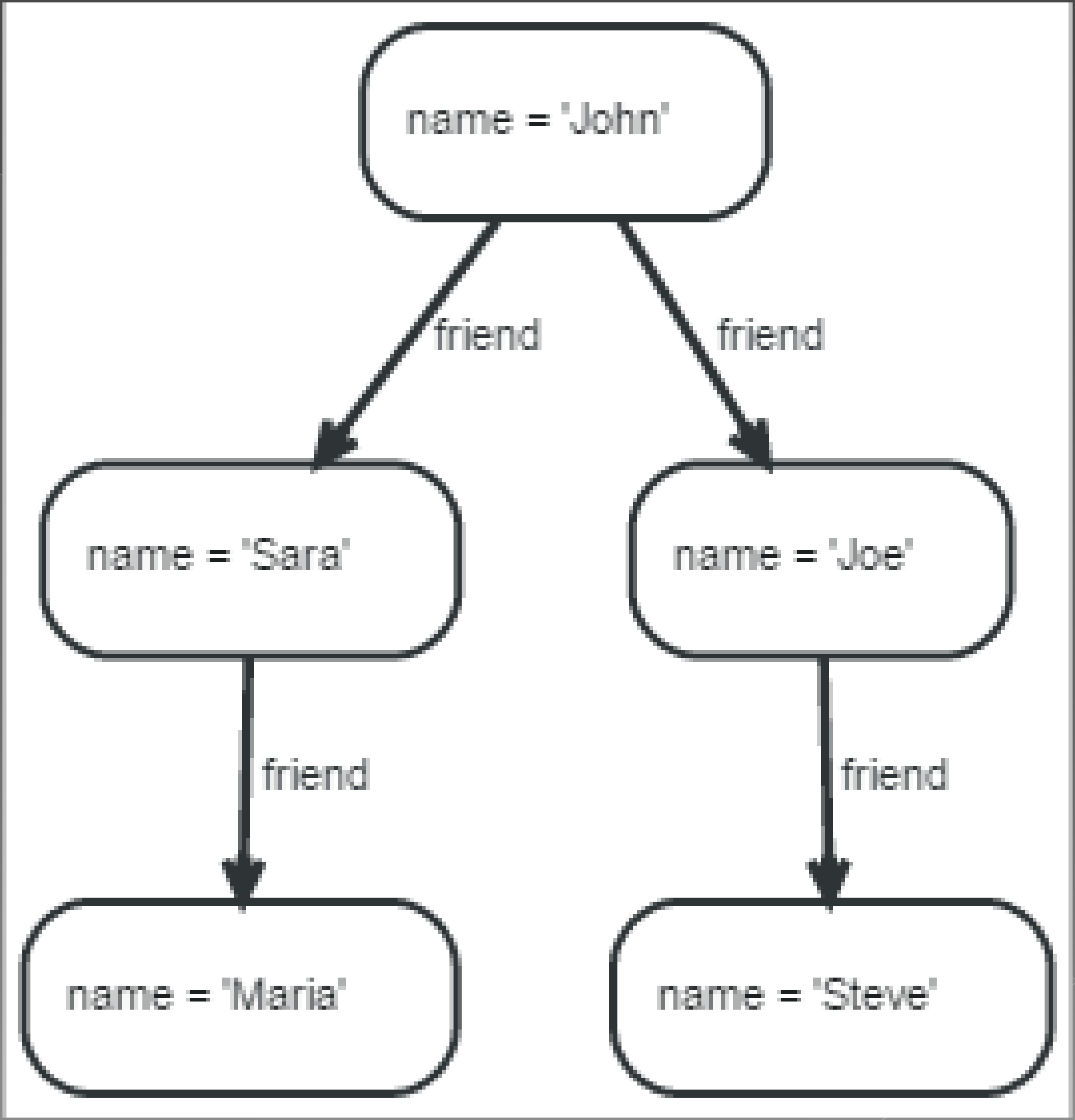
- Cypher as a language has been inspired and adapted through a number of approaches, and builds upon established querying practices
  - Keywords, such as WHERE and ORDER BY inspired by SQL
  - Pattern matching borrows expressions from SPARQL
  - Collection semantics from Python, etc

Cypher's Structure

- Borrows structure from SQL
  - Built up using a variety of clauses
- These clauses are chained together and feed intermediate result sets between each other

Cypher's Structure

- A few clauses to read from the graph:
  - MATCH: The graph pattern to match
  - WHERE: Part of MATCH, OPTIONAL MATCH, and WITH. Adds constraints to a pattern, or filters the intermediate result passing through WITH
  - RETURN: What to return
- Example graph on next slide →


name = 'John'
friend
friend
name = 'Sara'
name = 'Joe'
friend
friend
name = 'Maria'
name = 'Steve'

- MATCH (john {name: ‘John’})-[:friend]->()-[:friend]->(fof)
  RETURN john.name, fof.name

- This query finds a user John and John’s friends before returning both John and any friends-of-friends (fof) of John

- This returns:

```
John.name          fof.name
  "John"           "Maria"
  "John"           "Steve"
 2 rows
```

- More filtering can be set to see parts in motion →

- Take a list of user names, find all nodes with names from that list, match their friends, and return only those followed users who’s *name* property starts with S

```
MATCH (user)-[:friend]->(follower)
WHERE user.name IN [‘Joe’, ‘John’, ‘Sara’, ‘Maria’, ‘Steve’]
AND follower.name =~ ‘S.*’
RETURN user.name, follower.name
```

- This results in:

```
user.name                     follower.name
   "John"                          "Sara"
   "Joe"                           "Steve"
2 rows
```

Cypher's Structure

- CREATE/DELETE: Creates and deletes nodes and relationships
- SET/REMOVE: Sets values to properties and adds/removes labels on nodes
- MERGE: Match existing or create new nodes and patterns

- These are examples of clauses to update a graph

Features of Cypher

- While pattern matching, Cypher makes sure to not include matches where the same graph relationship is found multiple times in a single pattern
- This removes repetition and improves results

Features of Cypher

- Cypher supports querying with parameters
  - Developers don’t have to resort to string building
  - Makes caching of execution plans easier for Cypher
- Cypher is still changing rapidly
  - New pattern matchers, aggregators, and optimizations to make your queries faster
  - Allows use of older parsers even when syntax changes with updates

Neo4J's Cypher

- Graphing Language
- Efficient Querying and Updating of Graphs
- Neat iconography → Self-explanatory queries
- Still building and evolving, becoming even better!