NeoJ4

The (very) basics…

Labels

`(a:Person)` a Person

`(a:Person {name:"Keanu Reeves"})` a Person with properties

`(a:Person)-[:ACTED_IN]->(m:Movie)` a Person that ACTED_IN some movie

`(a)` actors

`(m)` movies

`( )` some anonymous node

Relationships

`-[r]->` a relationship referred to as "r"

`(a)-[r]->(m)` actors having a relationship referred to as "r" to movies

`-[:ACTED_IN]->` the relationship type is ACTED_IN

`(a)-[:ACTED_IN]->(m)` actors that ACTED_IN some movie

`(d)-[:DIRECTED]->(m)` directors that DIRECTED some movie

Nodes with Properties

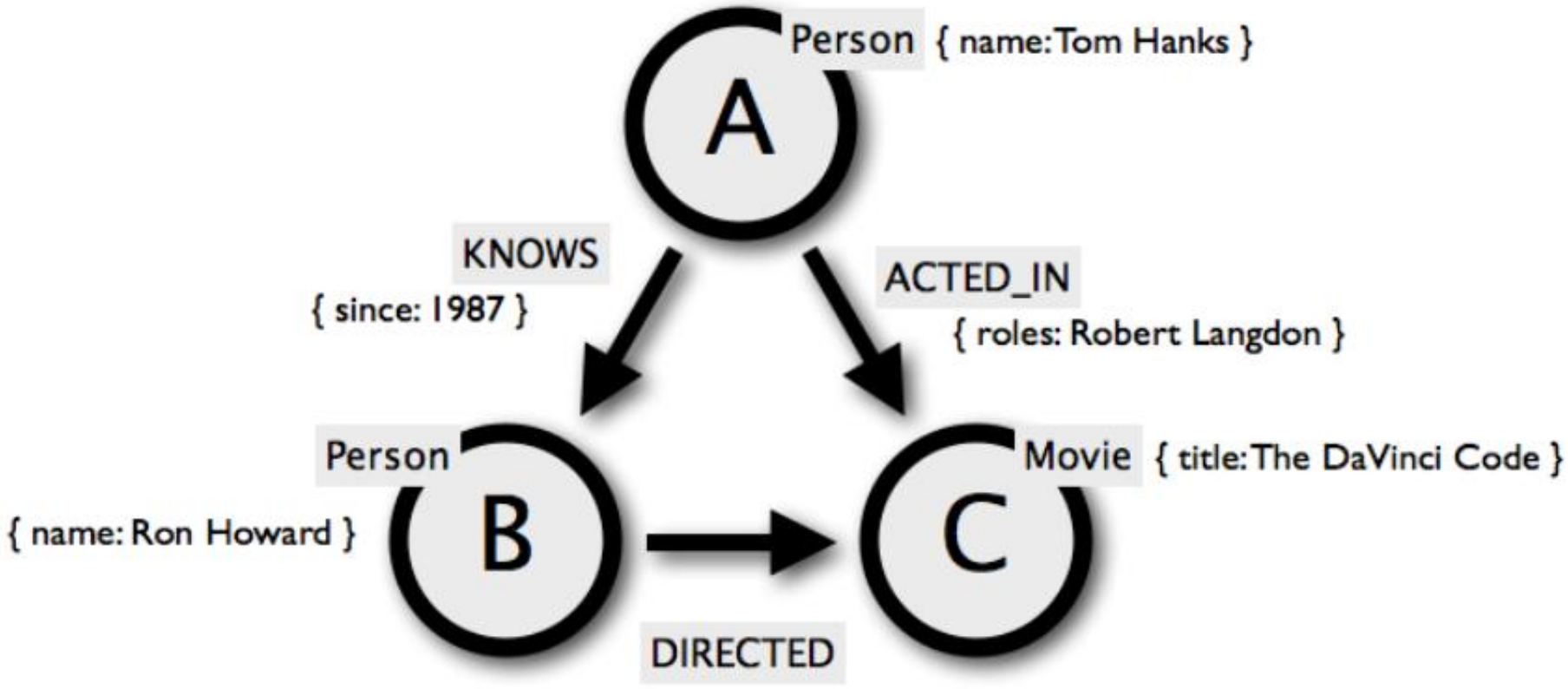
`(m {title:"The Matrix"})` Movie with a title property

`(a {name:"Keanu Reeves",born:1964})` Actor with name and born property

Relationship with Properties

`(a)-[:ACTED_IN {roles:["Neo"]}]->(m)` Relationship *ACTED_IN* with *roles* property (an array of character names)

Property Graph



Queries:

```
MATCH (actor:Actor { name: "Tom Hanks" })
RETURN actor;
```

```
MATCH (actor:Actor)
WHERE actor.name = "Tom Hanks"
CREATE (movie:Movie { title:'Sleepless IN Seattle' })
CREATE (actor)-[:ACTED_IN]->(movie);
```

A live example…

```
private static void createData() {
    GraphDatabaseAPI db = server.getDatabase().getGraph();
    ExecutionEngine executionEngine = new ExecutionEngine(db);
    String query = "CREATE (:Movie {title:'The Matrix', released: 1999, tagline: 'The one and only'})" +
            " <-[:ACTED_IN {roles:['Neo']}]-" +
            " (:Person {name:'Keanu Reeves',born:1964})";

    executionEngine.execute(query).dumpToString();
}
```

```
public Map findMovie(String title) {
    if (title==null) return Collections.emptyMap();
    return IteratorUtil.singleOrNull(cypher.execute(
            "MATCH (movie:Movie {title:{title}})" +
                    " OPTIONAL MATCH (movie)<-[r]-(person:Person)\n" +
                    " RETURN movie.title as title, collect({name:person.name, job:head(split(lower(type(r)),'_')), role:r.role
            map("title", title)));
}
```