Neo4j

Graph database

By Michaël DOHR

Introduction

High business value in data relationships

Data is increasing in volume...

- New digital process
- More online transaction
- New Social networks
- More devices

... and is getting more connected

Customers, products, process, devices interact and relate to each other

Using data relationships unlocks value

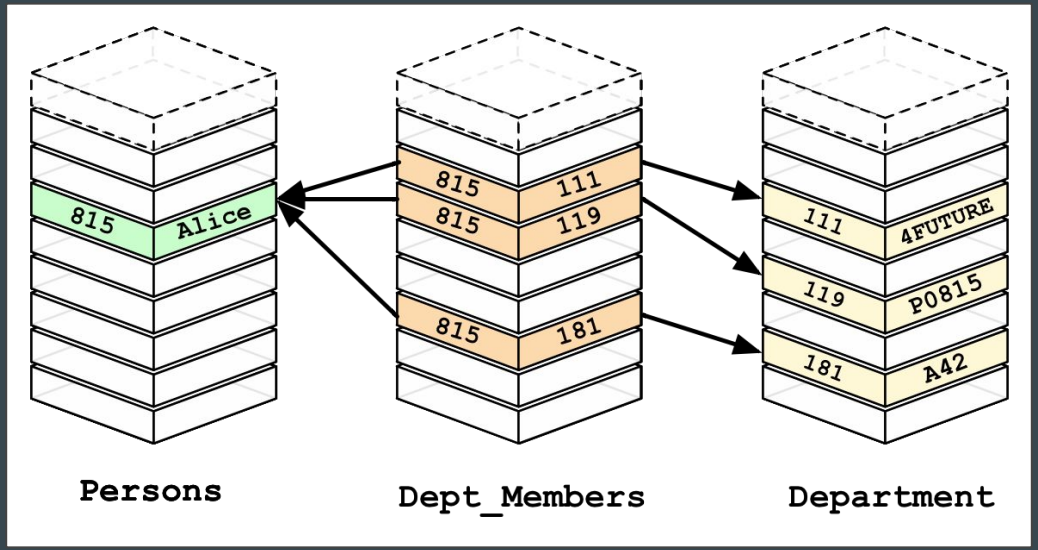
- Real-time recommendations
- Fraud detection
- Master data management
- Network and IT operations
- Identity and access management
- ...

Early adopters become industry leaders

Google LinkedIn facebook

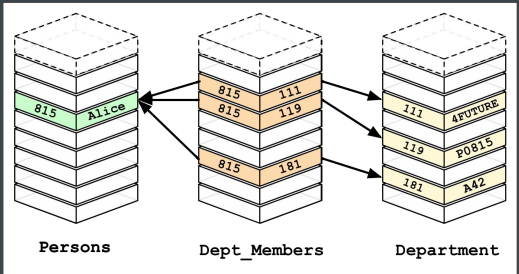
How to modelize relationships ?

Relational database



How to modelize relationships ?

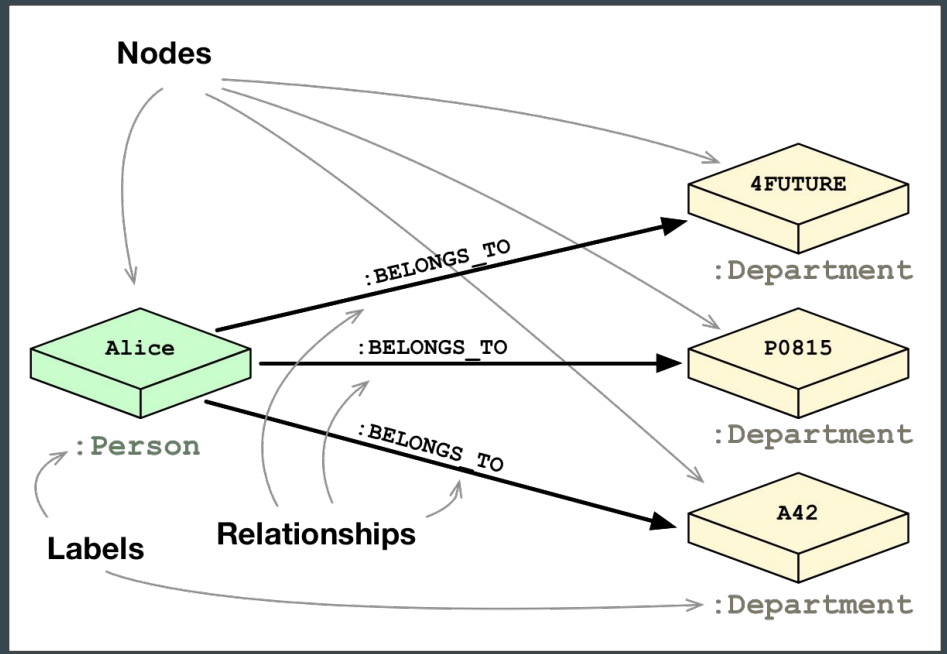
Relational database



- Cannot model and store data or relationships without complexity
- Performance degrades with number and level of relationships
- Query complexity grows with needs for JOINs

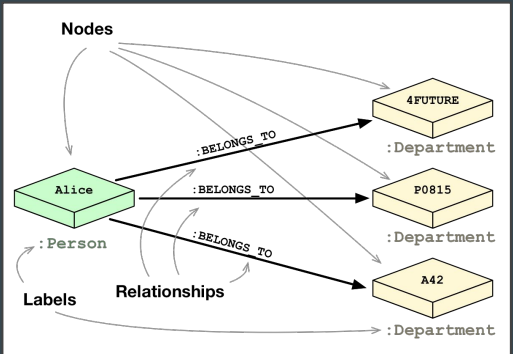
How to modelize relationships ?

Graph database



How to modelize relationships ?

Graph database



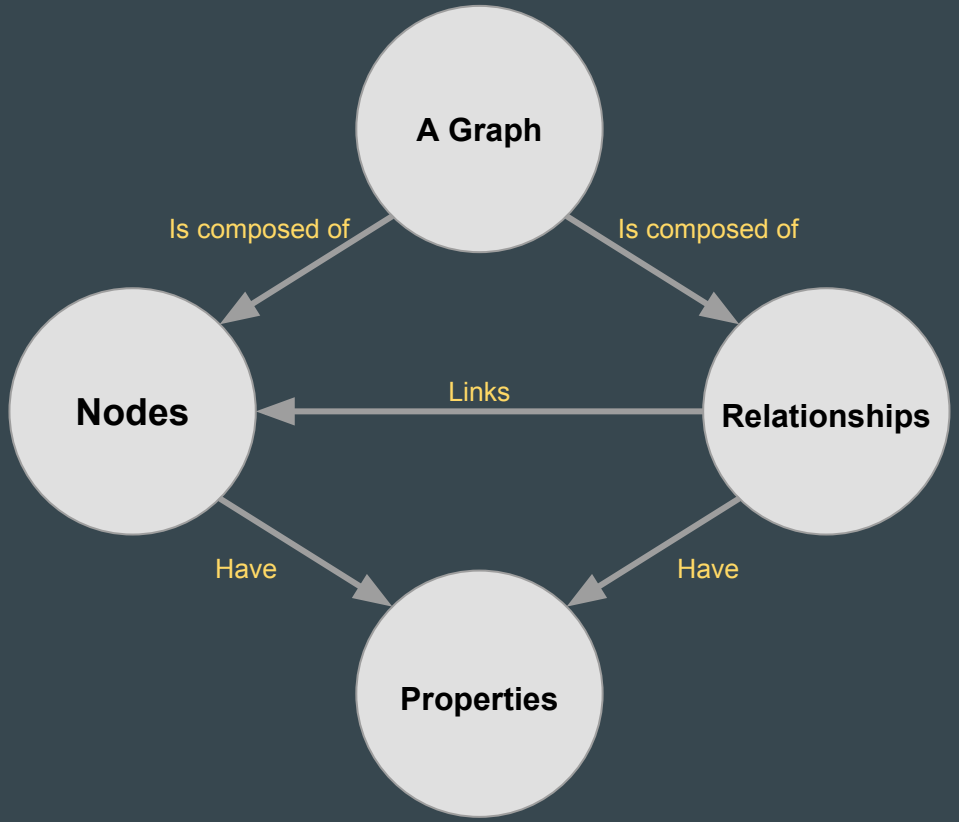
- Powerful data model, as general as RDB
- Fast, for connected data
- Easy to query

- *Required conceptual shift*

Graph theory

- 1736 : Euler writes a paper on "The Seven Bridges of Konisberg"
- 1845 : Kirchoff publishes his electricals circuit laws
- 1852 : Guthrie poses the "Four colours problem"
- 1936 : Dénes König publishes a textbook a Graph Theory
- 1941 : Tamsey and Turan define Extremal Graph Theory
- 1959 : De Bruijin publishes a paper on Enumerative Graph Theory
- 1959 : Erdos, Renyi and Gilbert define Random Graph Theory
- 1969 : Heesch solves the "Four Color" problem

Graph theory



Neo4j Database

History

Developed by “Neo Technology © ”

- 2002 : First version of Neo4j
- 2003 : First 24/7 Neo4j production deployment
- 2007 : Open-sourced under GPL
- 2009 : Raised seed funding from Sunstone and Conor and continued development
- 2010 : Released Neo4j version 1.0
- 2011 : Moved headquarters to Silicon Valley.

Neo4j

- JAVA (Standalone / Embedded)
- Own filesystem (graph storage)
- Query with "Cypher"
- ACID (Atomicity, Consistency, Isolation, Durability)
- Schemaless
- REST
- Front-end
- CSV Loader
- *Clustered replication*
- *Cache Sharding*
- *Hot backup ...*

Neo4j with JDBC

https://github.com/neo4j-contrib/neo4j-jdbc

```
Connection con = DriverManager.getConnection("jdbc:neo4j://localhost:7474/");

String query = "MATCH (:Movie {title:{1}})<-[:ACTED_IN]-(a:Person) RETURN a.name as actor";

try (PreparedStatement stmt = con.prepareStatement(query)) {

  stmt.setString(1,"The Matrix");

  try (ResultSet rs = stmt.executeQuery()) {
    while(rs.next()) {
      System.out.println(rs.getString("actor"));
    }
  }
}
```

Neo4j with Spring Data

http://projects.spring.io/spring-data-neo4j/

```
@NodeEntity
public class Movie {

  @GraphId Long id;

  String title;

  Person director;

  @Relationship(type="ACTED_IN", direction = "INCOMING")
  Set<Person> actors = new HashSet<>();
}
```

```
interface MovieRepository extends GraphRepository<Movie> {

  @Query("MATCH (m:Movie)<-[rating:RATED]-(user)  " +
         "WHERE id(m) = {movieId} RETURN rating")
  Iterable<Rating> getRatings(Long movieId);

  Collection<Movie> findByTitle(name);
}
```

Neo4j with Scala

https://github.com/AnormCypher/AnormCypher

```
Cypher(
  """
   MATCH (:Movie {title:{title}})<-[:ACTED_IN]-(a:Person)
   RETURN a.name as name, actor
  """
).on("title" -> "The Matrix").
  asAsync {
    str("name") ~ node("actor") *
  }.map(items => {
  ???
})
```

Neo4j with NodeJS

https://github.com/brian-gates/cypher-stream

```
const cypher = require('cypher-stream')('http://localhost:7474');

cypher('match (user:User) return user')
  .on('data', function (result){
     console.log(result.user.first_name);
  })
  .on('end', function() {
     console.log('all done');
  });
```

Neo4j with Javascript

With fetch api

```
class Cypher {
  _auth;
  _url;
  constructor( auth, url ) {
    this._auth = auth;
    this._url = url;
  }
  process( query, params = {} ) {
    return fetch( this._url, {
      method: "POST",
      headers: {
        Authorization: `Basic ${this._auth}`
      },
      body: JSON.stringify({
        query:  query,
        params: params
      })
    } ).then( result => result.json() );
  }
}
```

```
var cypher = new Cypher( "bmVvNGo6YXplcnR5KjAw",
                         "http://localhost:7474/db/data/cypher" );


cypher.process(
  "MATCH (:Movie {title:{title}})<-[:ACTED_IN]-(a:Person) " +
  "RETURN a.name as name, a as actor"
).then( json => console.log( json ) );
```

Cypher

SQL-inspired language for describing patterns in graphs.

(A)-[:LIKES]->(B)

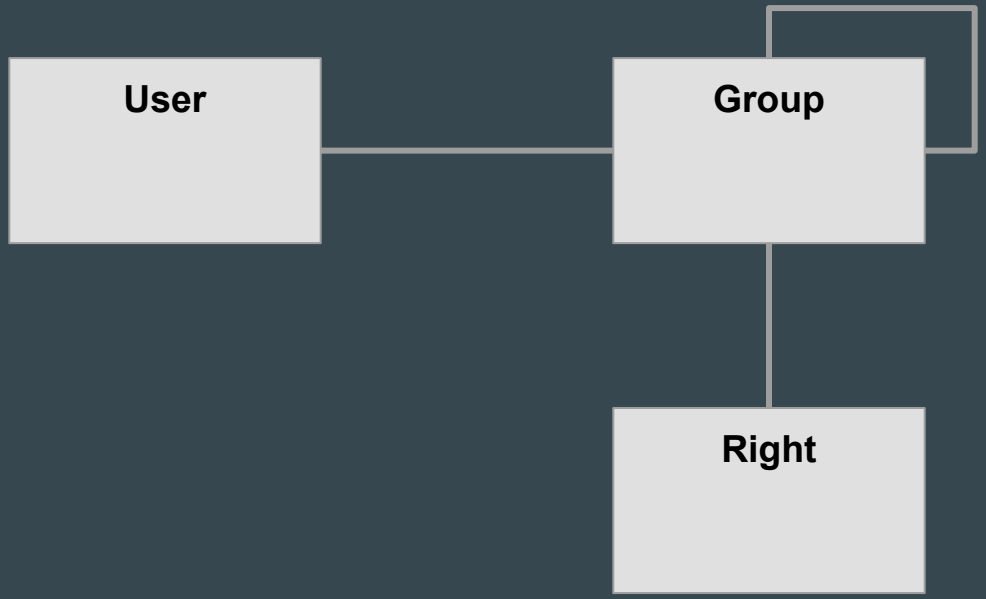
Cypher

- Node : Surround node with parentheses (looks like circle)
  - MATCH (node) RETURN node.property
  - MATCH (node1)-->(node2) RETURN node1.property, node2.property
- Relationships :
  - MATCH (node1)-[rel]->(node2) RETURN rel.property
  - MATCH (node1)-[:IS_LINKED]->(node2) RETURN node1, node2
  - MATCH ()-[is :IS_LINKED]->() RETURN is
- Label :
  - MATCH (node: TYPE1) RETURN node
  - MATCH (node :TYPE1 :TYPE2) RETURN node
- Properties:
  - MATCH (node { name: 'Toto'}) RETURN node
  - MATCH (node)-[r {idx: 0}] RETURN node

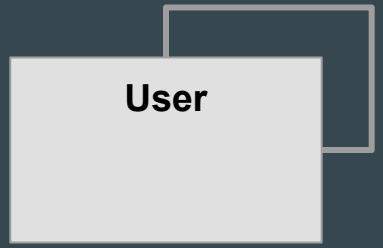
Examples

https://github.com/dohr-michael/neo4j_sid

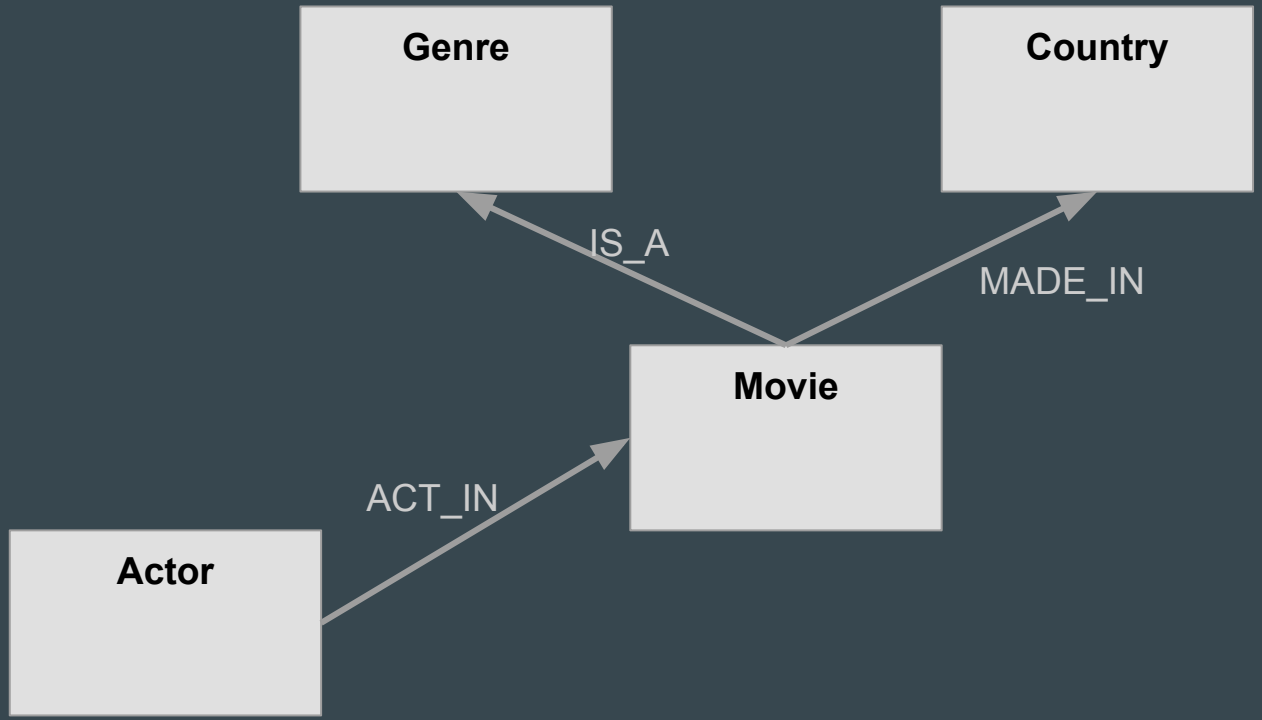
User management



Person relation



Movie



Integrations

- **Spark**
  - Preprocessing to import data into Neo4j
    - http://www.markhneedham.com/blog/2015/04/14/spark-generating-csv-files-to-import-into-neo4j/
  - Perform analysis in larger dataset
    - https://github.com/kbastani/neo4j-mazerunner
- **MongoDB**
  - One-way synchronization from MongoDB to Neo4j
    - https://github.com/neo4j-contrib/neo4j_doc_manager
- **Docker**
  - https://hub.docker.com/_/neo4j/

Other graph DB

- **OrientDB** : Open Source, ACID, Http / REST, SQL Like
  - http://orientdb.com/
- **Titan DB** : Open Source, Scalable Database, use backend storage (Cassandra) integration with Spark, Hadoop ...
  - http://thinkaurelius.github.io/titan/
- AllegroGraph : Closed Source, RDF Store (Linked Data)
  - http://franz.com/

Other articles

- http://neo4j.com/blog/neo4j-doc-manager-polyglot-persistence-mongodb/
  - Real time recommendation
- https://www.airpair.com/neo4j/posts/modelling-game-economy-with-neo4j
  - MMORPG game economy
- https://devcenter.heroku.com/articles/graphenedb
  - Neo4j on Heroku
- http://neo4j.com/graphgists/
  - Example of graph model

Questions ?