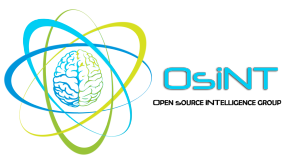
Neo4j: A Property Graph Database

Hemanta Baruah
Mala Das

Open Source Intelligence Lab
Indian Institute of Technology, Guwahati

April 8, 2018



Overview

Nosql Databases

Four Classes of NoSQL Databases:

1. Key-Value Store Database
   - Amazon's Simple DB
2. Column-family or Big table Database
   - Google's Big table.
   - Apache Cassandra Database
3. Document-Oriented Database
   - MongoDB Database.
4. **Graph Database**
   - **Neo4j Database** , OrientDB , TitanDB

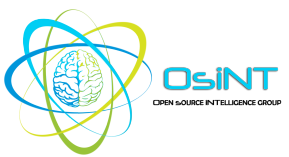
Difference between RDBMS and Graph Database

RDBMS	GraphDB
Tabular form	Graph form
Stores highly structured data	Maintains semi structured data
Depends on key constraints	Relationships are first-class citizens of the Graph Database model - Constraints can be represented using relationships
Data is normalised, meaning lots of joins, affecting speed	Better performance
Expensive with join operations	Eliminates the need for an expensive search / match computation.
Does not scale out horizontally	High scalability

Figure: RDBMS vs GraphDB

Introduction : Graph Database

- We are living in a connected World.
- Graph concept from Mathematics is used to build the Data Model.
- **Vertex/Node :**
  - Nodes are the entities in the Graph.
  - Can hold any number of attributes (key-value-pairs) called properties.
  - Nodes can be tagged with labels.
  - Label represents their different roles in the domain.
- **Edge/Relationship:**
  - Nodes are connected by Relationship.
  - Relationships can also hold properties.
  - Relationships are treated as first-class citizen in the data model.



Social Network Analysis

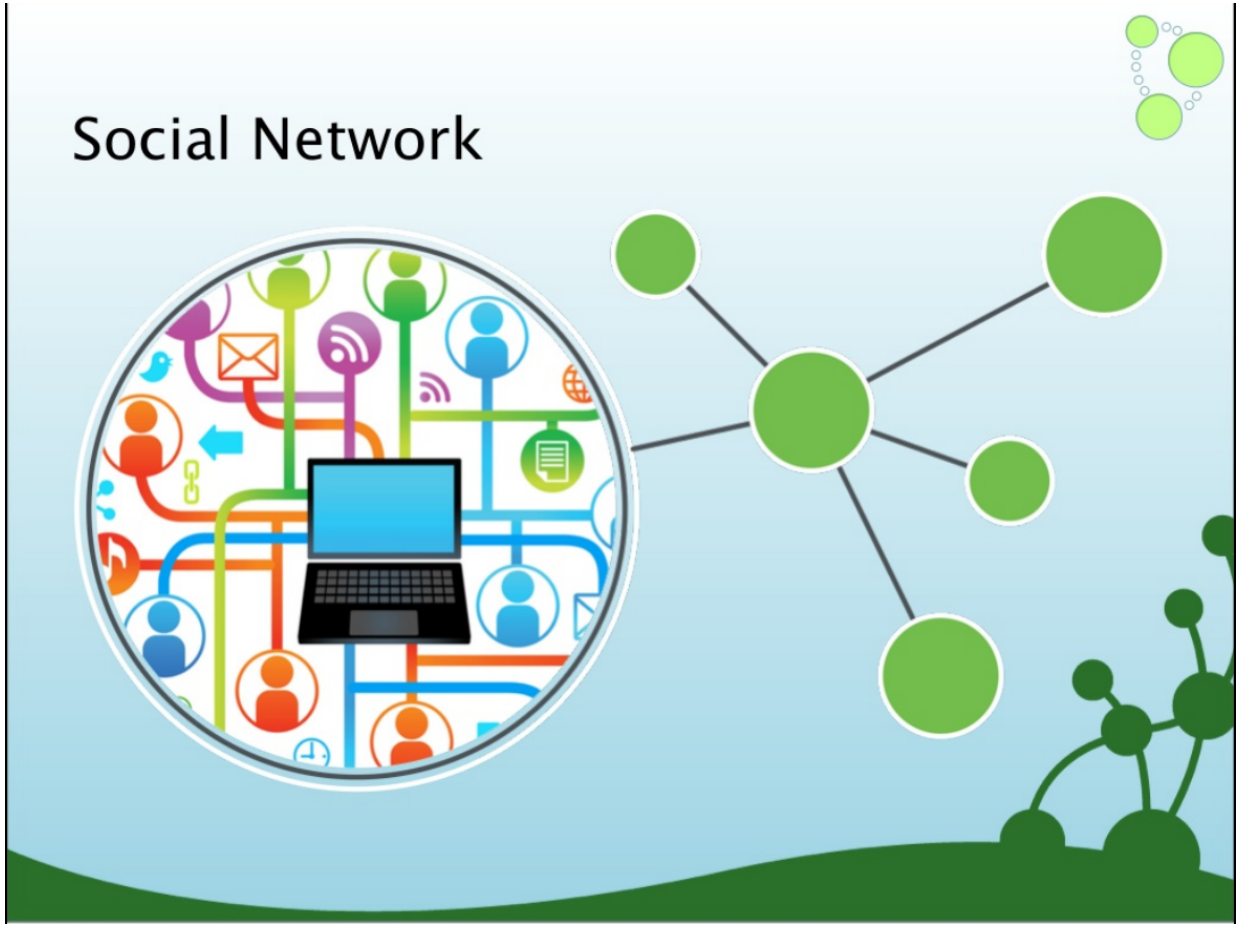


Figure: Social Network Analysis

Recommendation Engine

Figure: Recommendation System

Route Finding

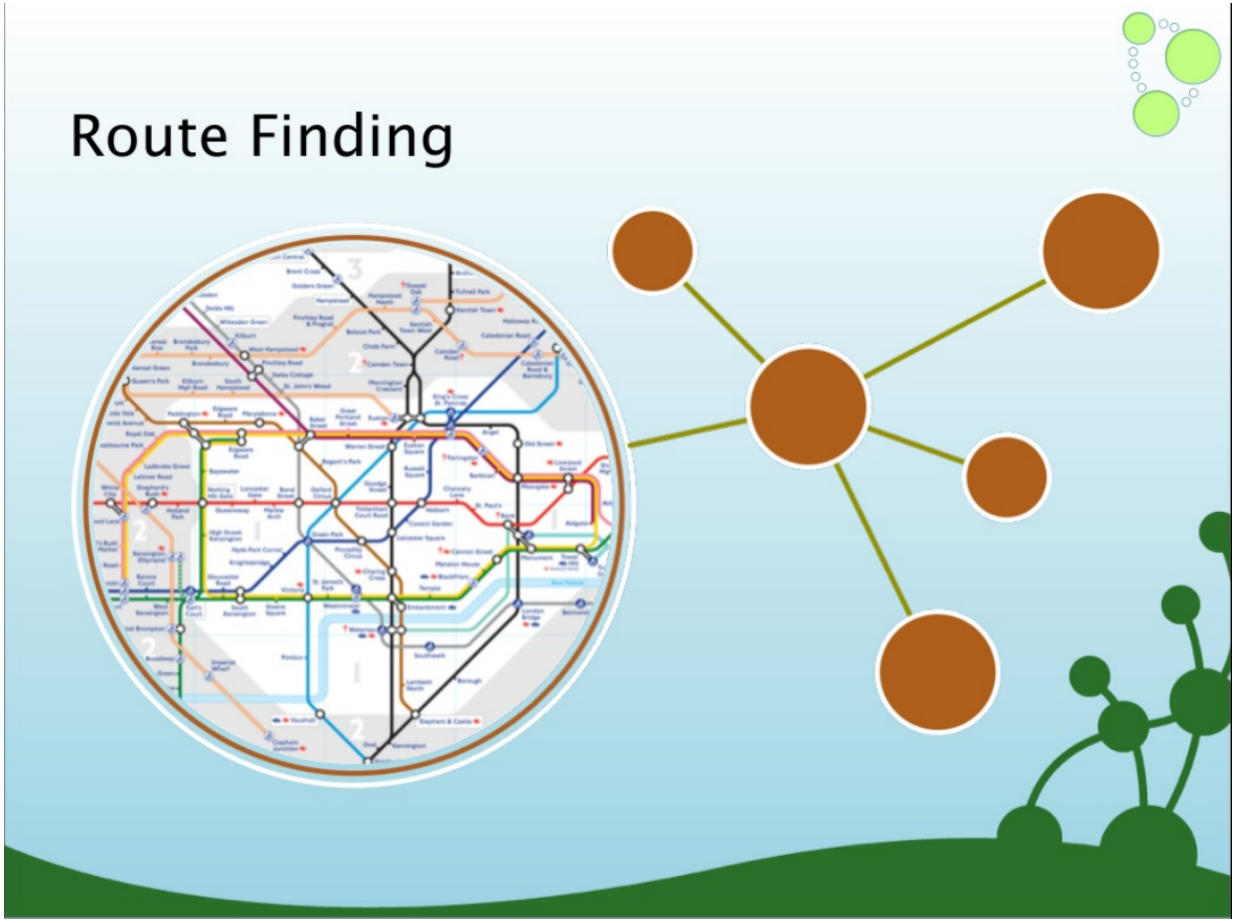


Figure: Route Finding

Logistics

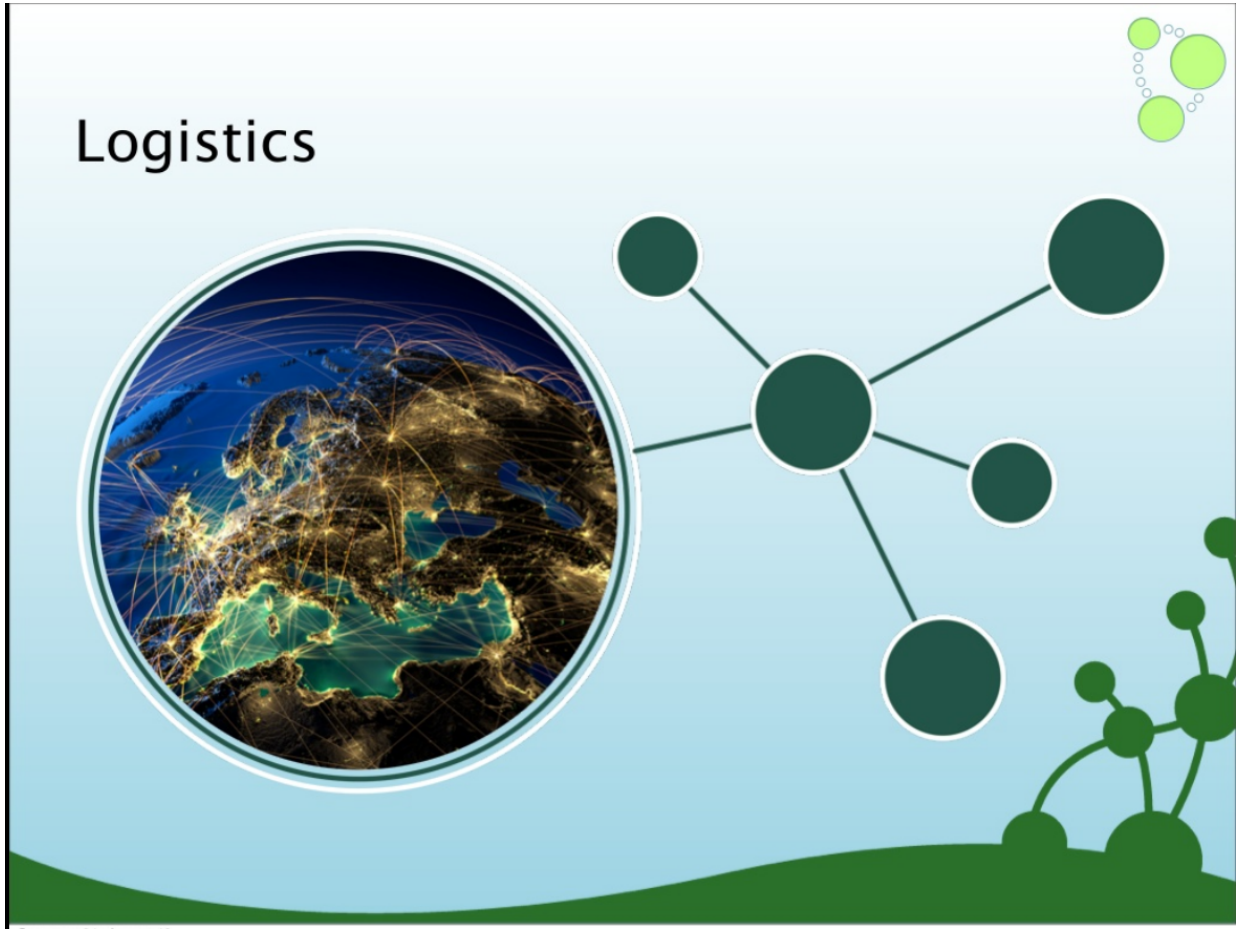


Figure: Used in Logistics

Fraud Analysis

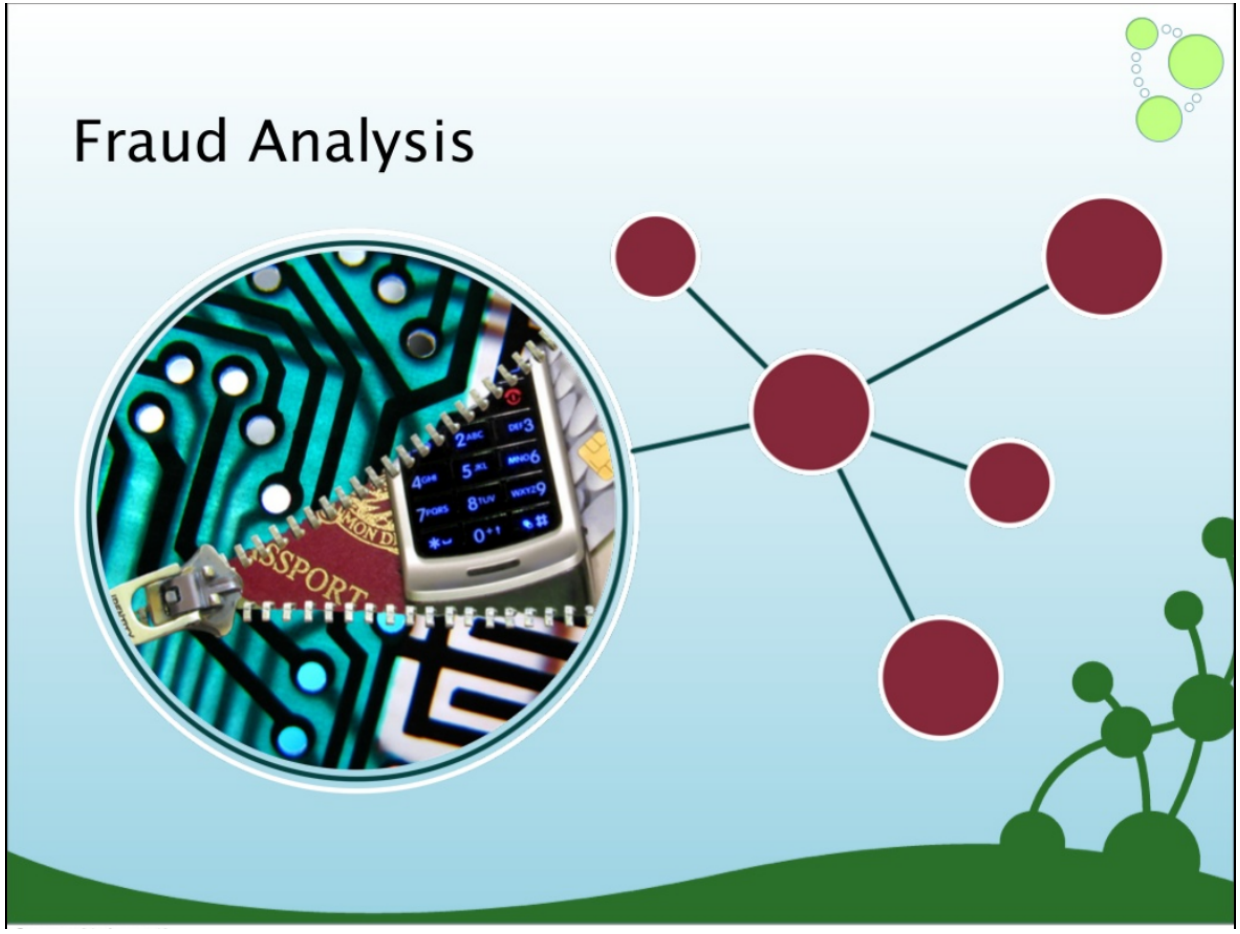


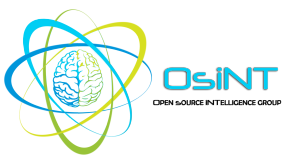
Figure: Fraud Analysis

Introduction To Neo4j Database

- World's leading Graph Database .
- Developed in Java by Neo Technologies.
- Open-source NoSQL native graph database.
- Store and query highly connected elements.
- White board friendly.
- Linear Operation runtime.
- Fully ACID transactional.

Features of Neo4j Database

- Flexible Schema.
- High Scalability and High Performance.
- Drivers for popular languages and frameworks.
- Powerful Cypher Query Language.
- Powerful Libraries are available to implement some of the popular Graph Algorithms.
  1. **Centrality Measure :** Page Rank , Betweenness Centrality, Closeness Centrality.
  2. **Community Detection:** Label Propagation , Connected Components , Triangle Count / Clustering Coefficient
  3. **Path Finding :** Minimum Weight Spanning Tree , All Pairs- and Single Source - Shortest Path



Graph Data Modeling

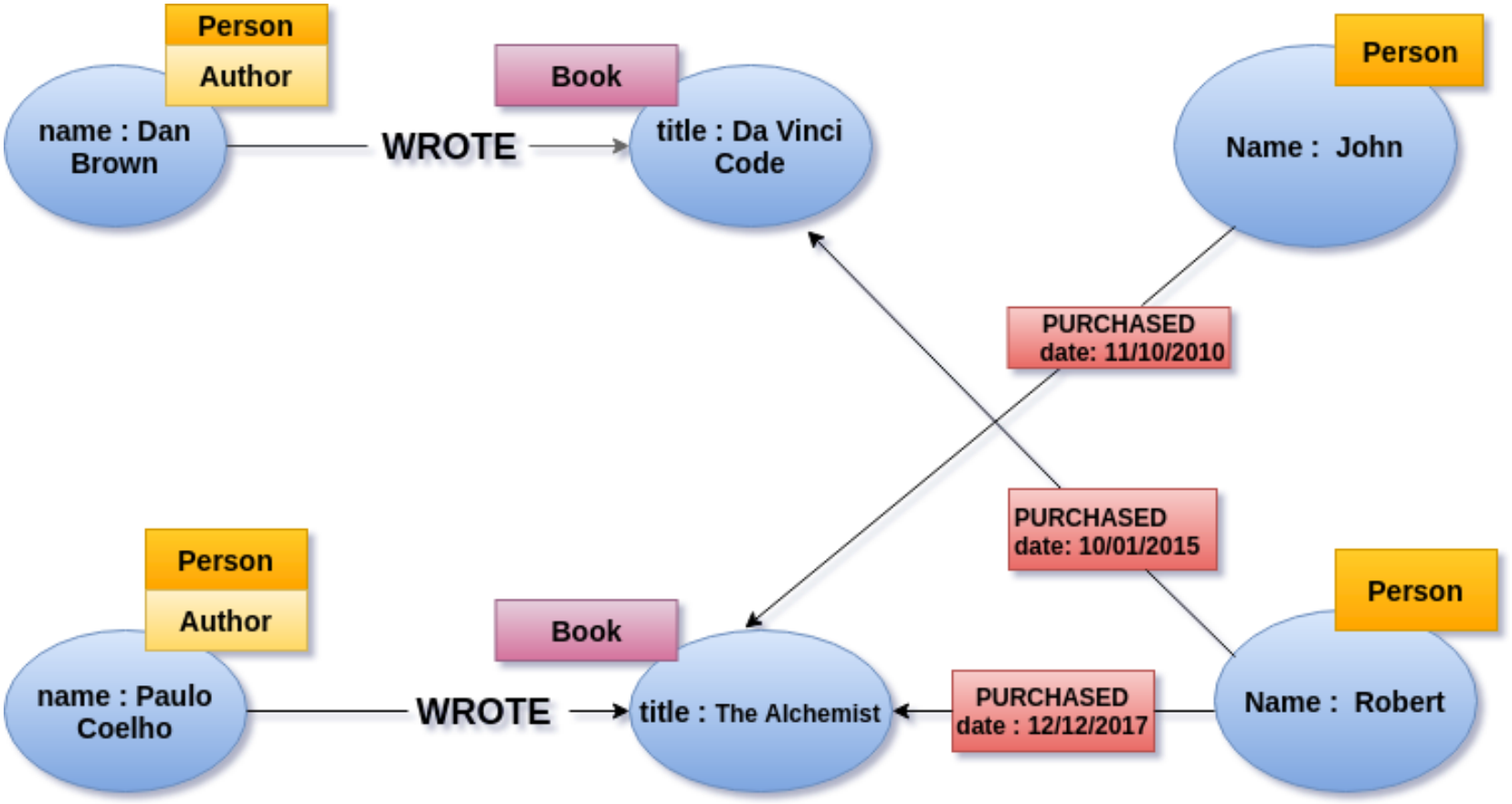


Figure: Property graph Data Model

Graph Data Modeling Building Blocks

Four Building Blocks

1. Nodes
2. Relationships
3. Properties
4. Labels

Nodes :

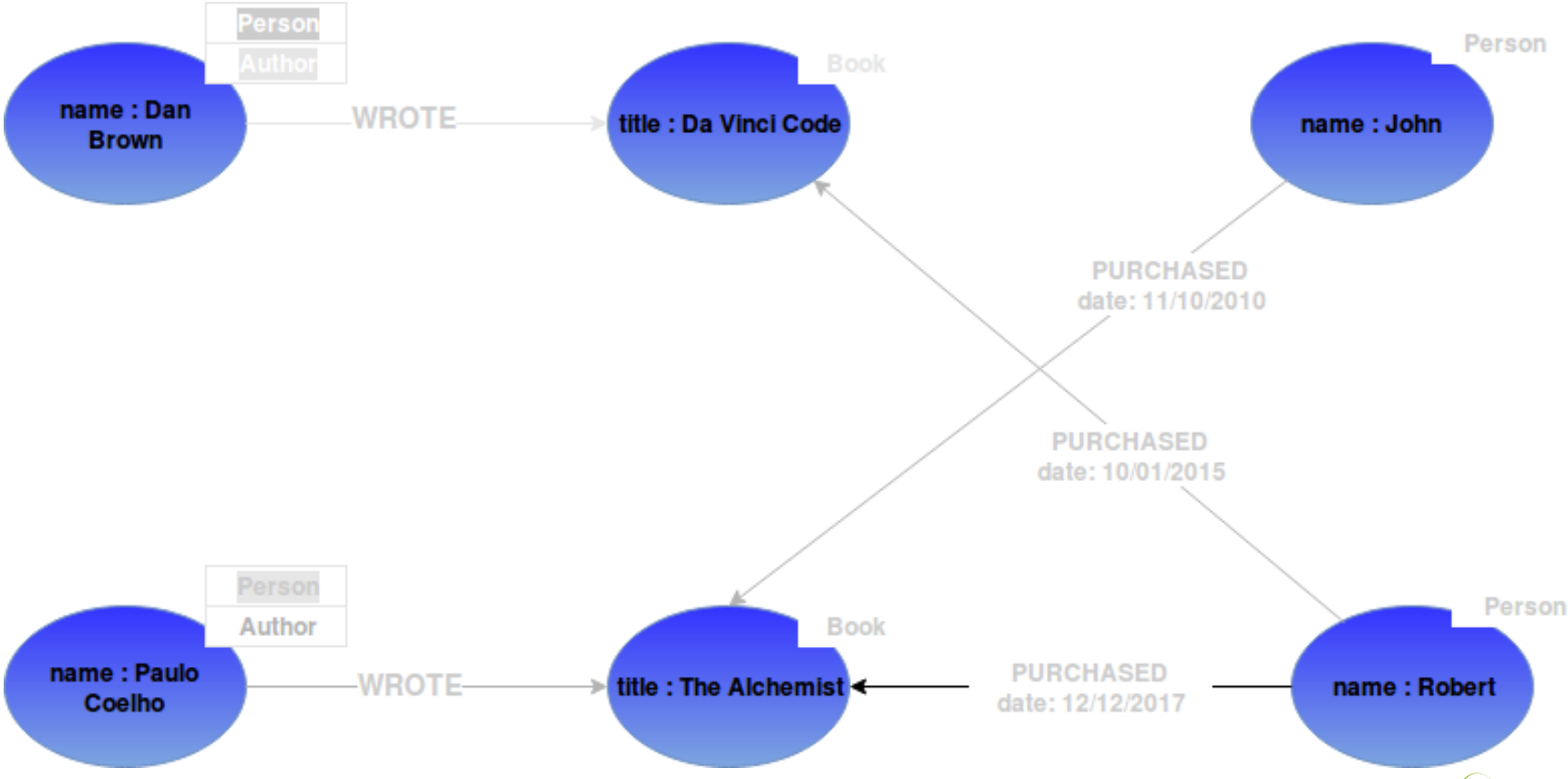


Figure: Nodes

Nodes :

Nodes:

- Represent all entities in the Domain.
- Similar to entities in RDBMS.
- Can have one or more labels.
- Can contain Properties
  - Represent entity attributes and/or meta data.
  - Key-Value pairs.
- Every node can have different properties.

Relationships :

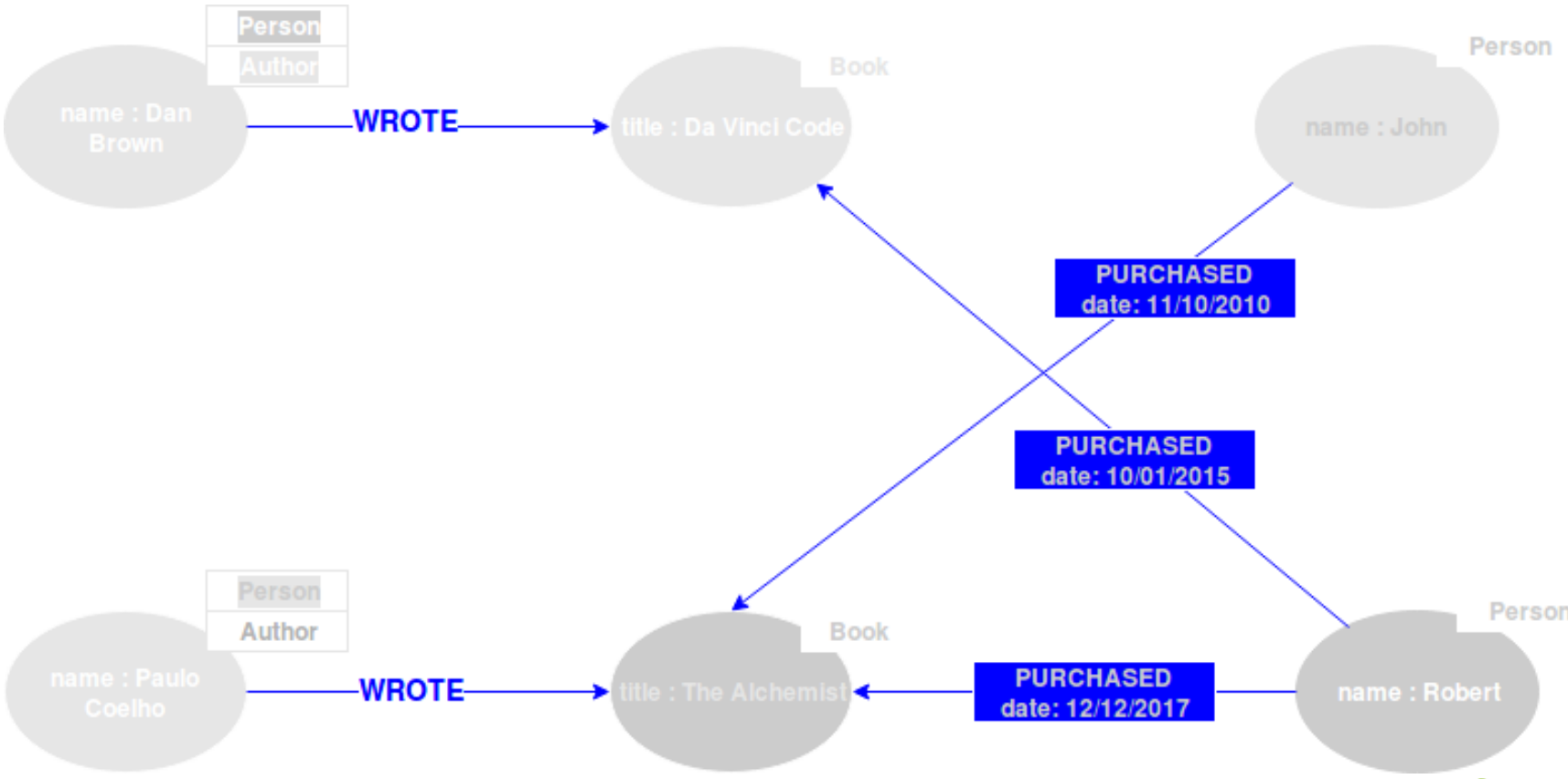


Figure: Relationships

Relationships :

Relationships:

- Relationship has a Name and a Direction.
  - Add Structure to the Graph.
  - Provide semantic context for nodes.
- Can contain Properties.
  - Represent quality or weight of Relationship.
  - Represent meta data
- Relationship must have a start node and a end node.
- Node Can have multiple Relationships.
- Multiple Relationship is possible between two same Nodes.

Properties :

Properties:

- Represent attributes and meta data.
- Associated with both Nodes and Relationships.
- Key-value pairs.

Labels :

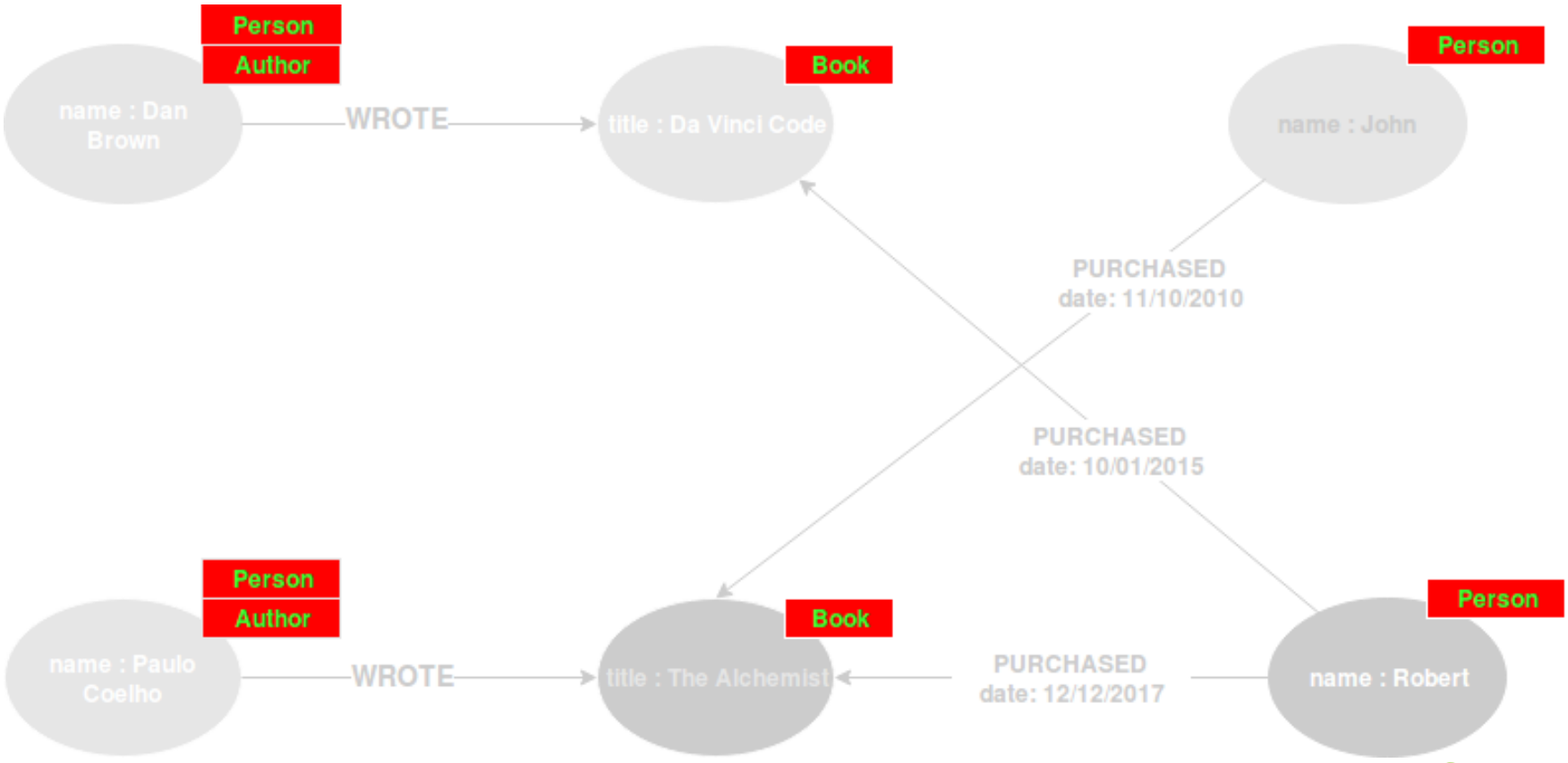


Figure: Node Labels

Labels :

Labels:

- Nodes can have zero or More labels.
- Used to represent roles.
  - e.g : User , Product, Company

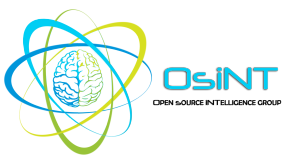
Cypher Basics

- Cypher is a declarative, SQL-inspired language.
- Very Powerful Graph Query language.
- Uses an ascii-art syntax.

Basic Cypher Syntax

CREATE STATEMENT : Used to create a Node and a Relation.

1. To Create a Node :
   CREATE (user:USER {Name:"Hemanta", Location:"India"})
   - Here , one Node is created i.e. "user";
   - One Label is created i.e. "USER";
   - Two Properties are created i.e. Name , Location.
2. To Create a Relation:
   CREATE (n:USER)-[p:POSTS {Date:'2018-03-22'}]→(m:TWEET)
   - Relationship 'POSTS' is created with the given type, direction and properties.
   - Bind a variable to it , i.e. 'p'.



Basic Cypher Syntax

DELETE STATEMENT : Used to Delete a Node and a Relationship.

1. **DELETE n, r**
   - Delete a node and a Relationship.
2. **DETACH DELETE n**
   - Delete a node and all relationships connected to it.
3. **MATCH (n)**
   DETACH DELETE n
   - Delete all nodes and relationships from the database.

Basic Cypher Syntax

MATCH STATEMENT : Like Select Statement in RDBMS

1. **MATCH (n:USER)-[:POSTS]→(m:TWEET)**
   WHERE n.author_name = 'Hemanta'

   - Node patterns can contain labels and properties.
   - Select the Graph Pattern in The Network

Basic Cypher Syntax

WHERE Statement : Anchor Pattern .

1. **WHERE n.property = value**
   - Use a Predicate to filter.
   - WHERE is always part of a MATCH Clause.

Basic Cypher Syntax

RETURN STATEMENT:

1. **RETURN $\star$**
   - Return the value of all variables.
2. **RETURN n AS columnName**
   - Use alias for matching result as columnName
3. **ORDER BY n.property**
   - Sort the result.
4. **ORDER BY n.property DESC**
   - Sort the result in descending order.

Basic Cypher Syntax

Aggregating Functions:

1. **count(⋆)**
   - Return the number of matching rows.
2. **collect(n.property)**
   - List from the values, ignores null.

Basic Cypher Syntax

CONSTRAINT: Used to create Constraint on the Properties of Nodes and Relationships .

1. **CREATE CONSTRAINT ON (n:USER) ASSERT n.author_name IS UNIQUE**
   - Create a unique Property constraint on the label **USER** and Property **author_name** .
   - If any other node with that label i.e **USER** is updated or created with a **author_name** that already exists, the write operation will fail .

Neo4j Data Model: With an example

Strategies to follow :

- Identify application/end user goals.
- What questions to Ask.
- For each possible questions.
  - Identify the entities in each question.
  - Identify the relationships between entities.
- Convert entities and relationships to paths.
  - These become the basis of the data model.
- Express questions as graph pattern.
  - These become the Basis for queries.

Neo4j Data Model Cycle

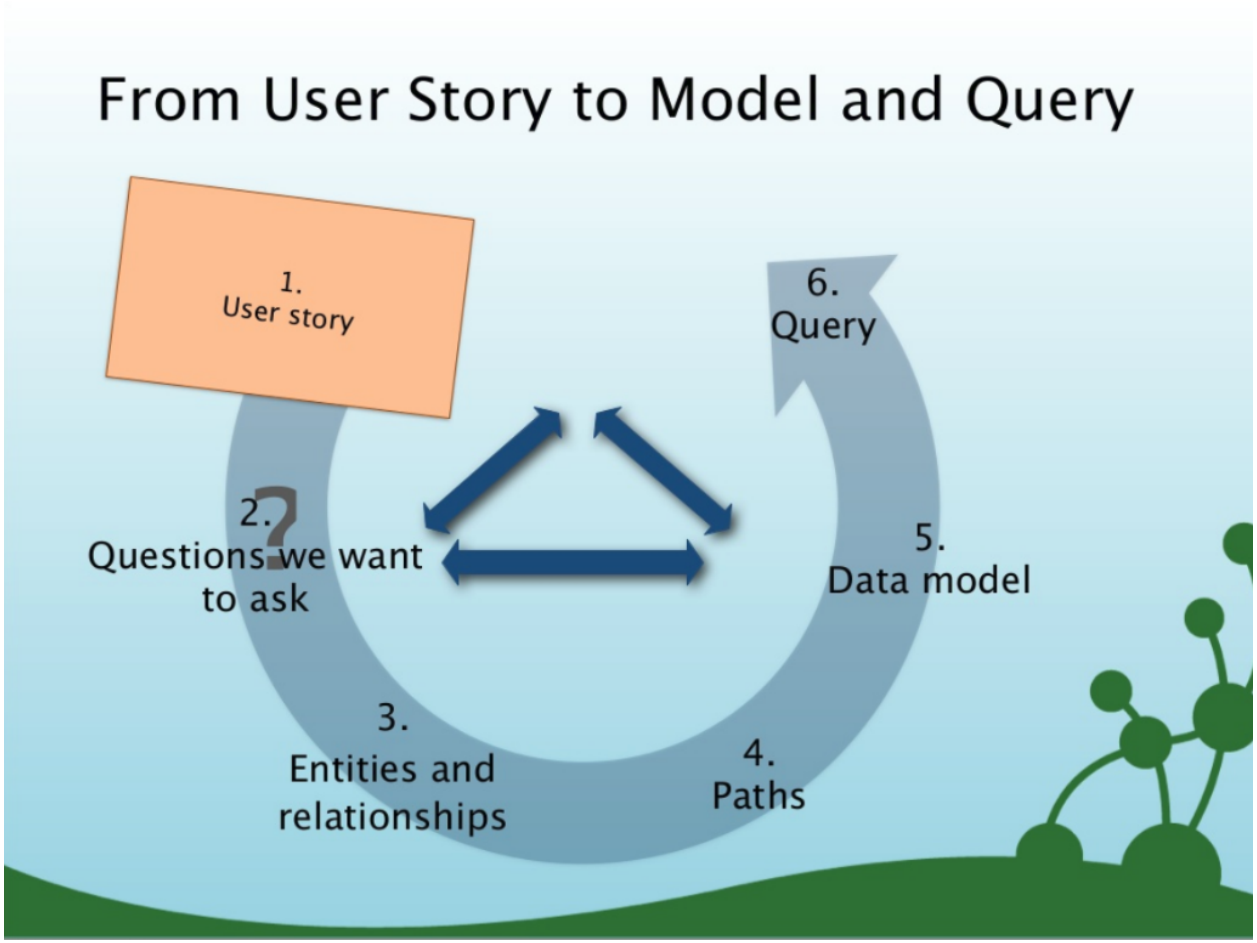


Figure: Data Modeling Cycle

1. Application/End User Goals

As a **Social Media Analyst**

I want to see who are the top people whom Narendra Modi frequently mentions in his tweets.

So that we can identify the important people in **Narendra Modi's** network.

2. Questions to Ask of the Domain

Who are the top twitter users , whom the user Narendra Modi(@narendramodi) frequently mentions in the tweet that he posts?

3. Identify Entities

Who are the top twitter **users** , whom the **user** Narendra Modi(@narendramodi) frequently mentions in the **tweet** that he posts?

- **USER**
- **TWEET**

4. Identify Relationships Between Entities

Who are the top twitter users , whom the user Narendra Modi(@narendramodi) frequently **mentions** in the tweet that he **posts**?

- **USER POSTS TWEET**
- **TWEET MENTIONS USER**

5. Convert To Cypher Paths

USER POSTS TWEET

$\Downarrow$

(:USER) - [:POSTS] $\rightarrow$ (:TWEET)

TWEET MENTIONS USER

$\Downarrow$

(:TWEET) - [:MENTIONS] $\rightarrow$ (:USER)

OsINT
OPEN SOURCE INTELLIGENCE GROUP

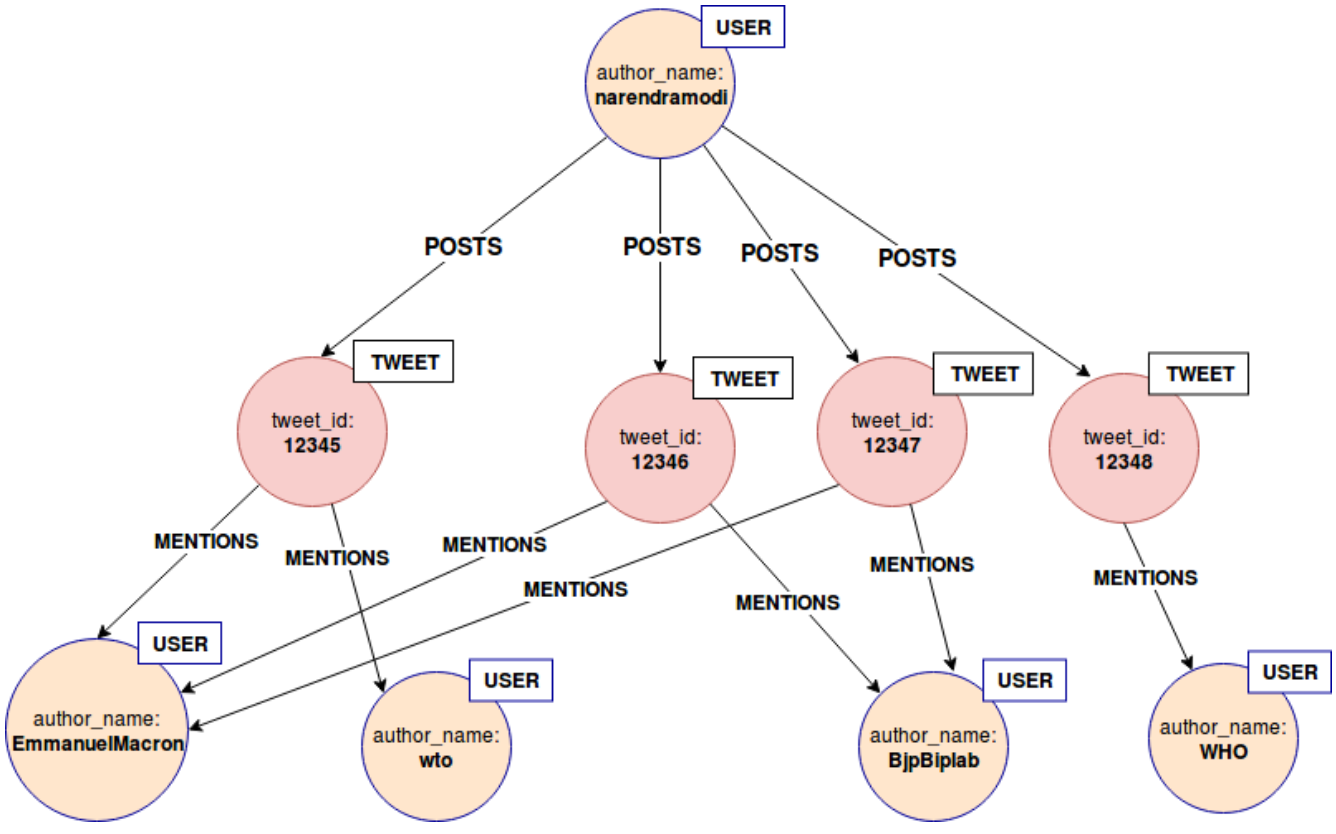
6. Consolidate Paths

(:USER) - [:POSTS] → (:TWEET) ,
(:TWEET) - [:MENTIONS] → (:USER)

(:USER) - [:POSTS] → (:TWEET) - [:MENTIONS] →
(:USER)

7. Candidate Data Model

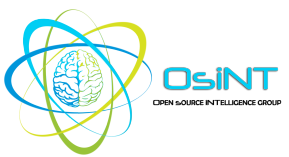
(:USER) - [:POSTS] → (:TWEET) - [:MENTIONS] → (:USER)



Cypher Query

Who are Top twitter users , whom the user Narendra Modi(@narendramodi) frequently mentions in the tweet that he posts?

```
$ MATCH (u1:USER)-[:POSTS]->(t:TWEET)-[:MENTIONS]->(u2:USER)
  WHERE u1.author_name='narendramodi'
  RETURN u1.author_name AS First_Author, u2.author_name AS Second_Author,
  COLLECT(t.tweet_id) AS tweet_ids ,
  COUNT(*) AS Total_Mention_Count ORDER BY Total_Mention_Count DESC LIMIT 10
```



Graph Pattern

Who are Top twitter users , whom the user Narendra Modi(@narendramodi) frequently mentions in the tweet that he posts?

```
$ MATCH (u1:USER)-[:POSTS]->(t:TWEET)-[:MENTIONS]->(u2:USER)
  WHERE u1.author_name='narendramodi'
  RETURN u1.author_name AS First_Author, u2.author_name AS Second_Author,
  COLLECT(t.tweet_id) AS tweet_ids ,
  COUNT(*) AS Total_Mention_Count ORDER BY Total_Mention_Count DESC LIMIT 10
```

Anchor Pattern

Who are Top twitter users , whom the user Narendra Modi(@narendramodi) frequently mentions in the tweet that he posts ?

```
$ MATCH (u1:USER)-[:POSTS]->(t:TWEET)-[:MENTIONS]->(u2:USER)
  WHERE u1.author_name='narendramodi'
  RETURN u1.author_name AS First_Author, u2.author_name AS Second_Author,
  COLLECT(t.tweet_id) AS tweet_ids ,
  COUNT(*) AS Total_Mention_Count ORDER BY Total_Mention_Count DESC LIMIT 10
```

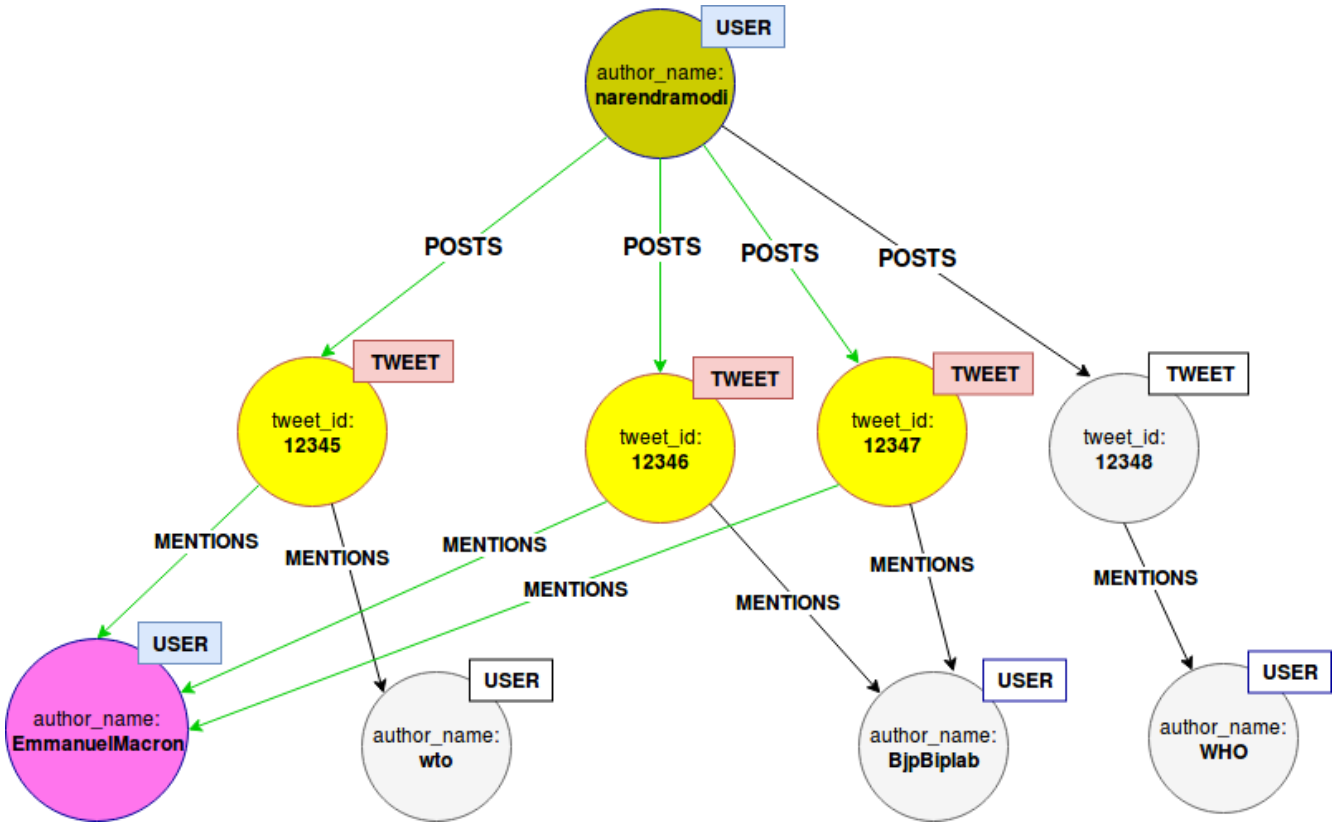
Projection Pattern

Who are Top twitter users , whom the user Narendra Modi(@narendramodi) frequently mentions in the tweet that he posts ?

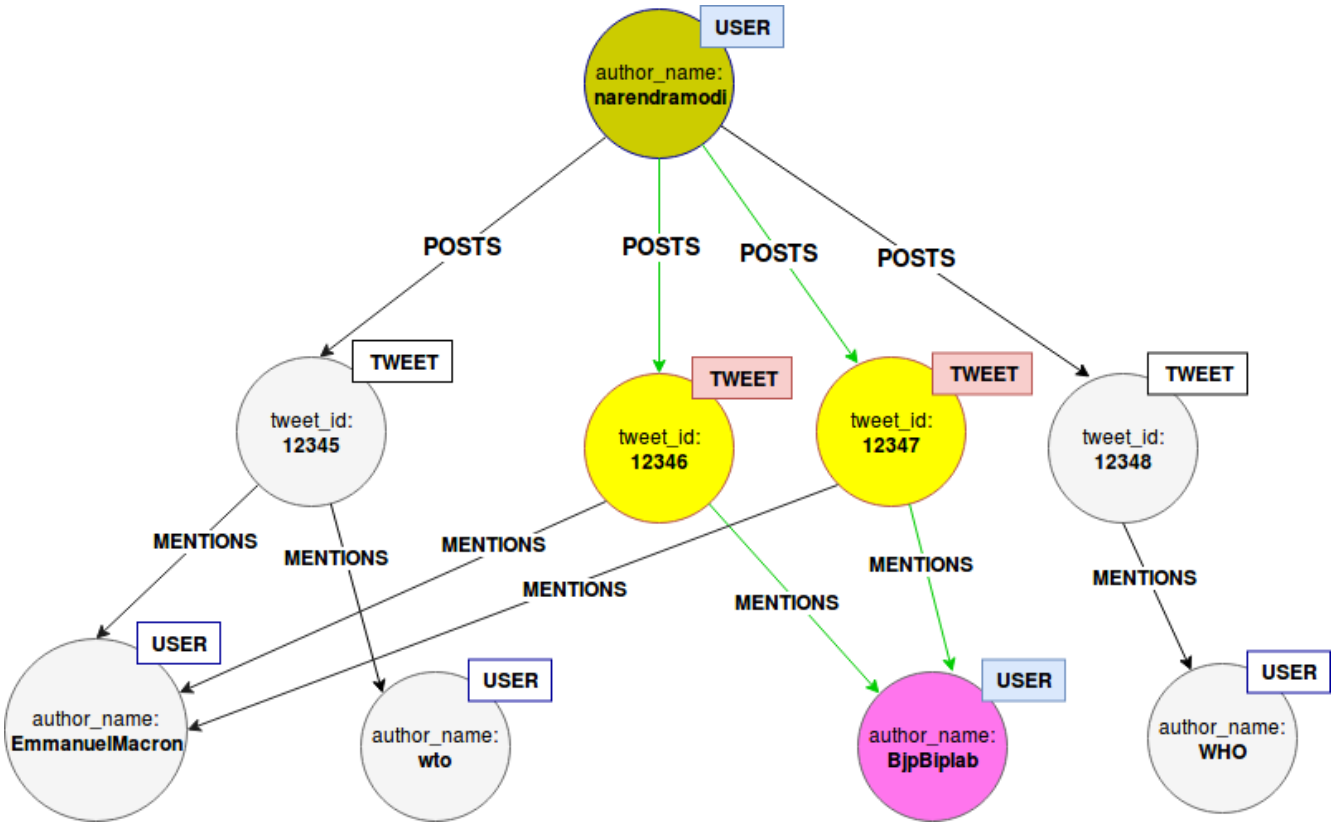
N.B: Here the **RETURN** statement GROUP all the relationships one after the other to project the output result. Its like **GROUP BY** statement in MySQL .

```
$ MATCH (u1:USER)-[:POSTS]->(t:TWEET)-[:MENTIONS]->(u2:USER)
  WHERE u1.author_name='narendramodi'
  RETURN u1.author_name AS First_Author, u2.author_name AS Second_Author,
  COLLECT(t.tweet_id) AS tweet_ids ,
  COUNT(*) AS Total_Mention_Count ORDER BY Total_Mention_Count DESC LIMIT 10
```

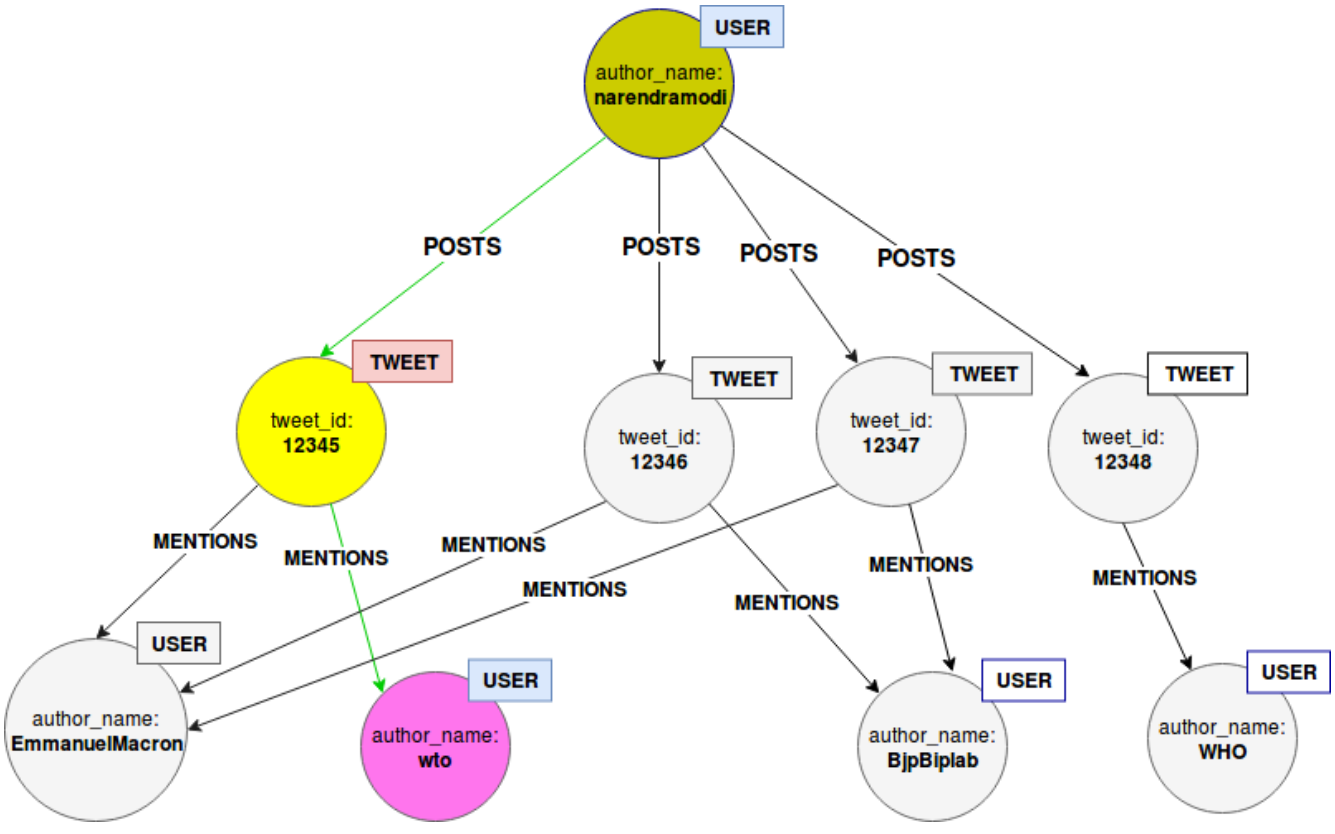
First Match


USER
author_name:
narendramodi
POSTS
POSTS
POSTS
POSTS
TWEET
tweet_id:
12345
TWEET
tweet_id:
12346
TWEET
tweet_id:
12347
TWEET
tweet_id:
12348
MENTIONS
MENTIONS
MENTIONS
MENTIONS
MENTIONS
MENTIONS
MENTIONS
USER
author_name:
EmmanuelMacron
USER
author_name:
wto
USER
author_name:
BjpBiplab
USER
author_name:
WHO

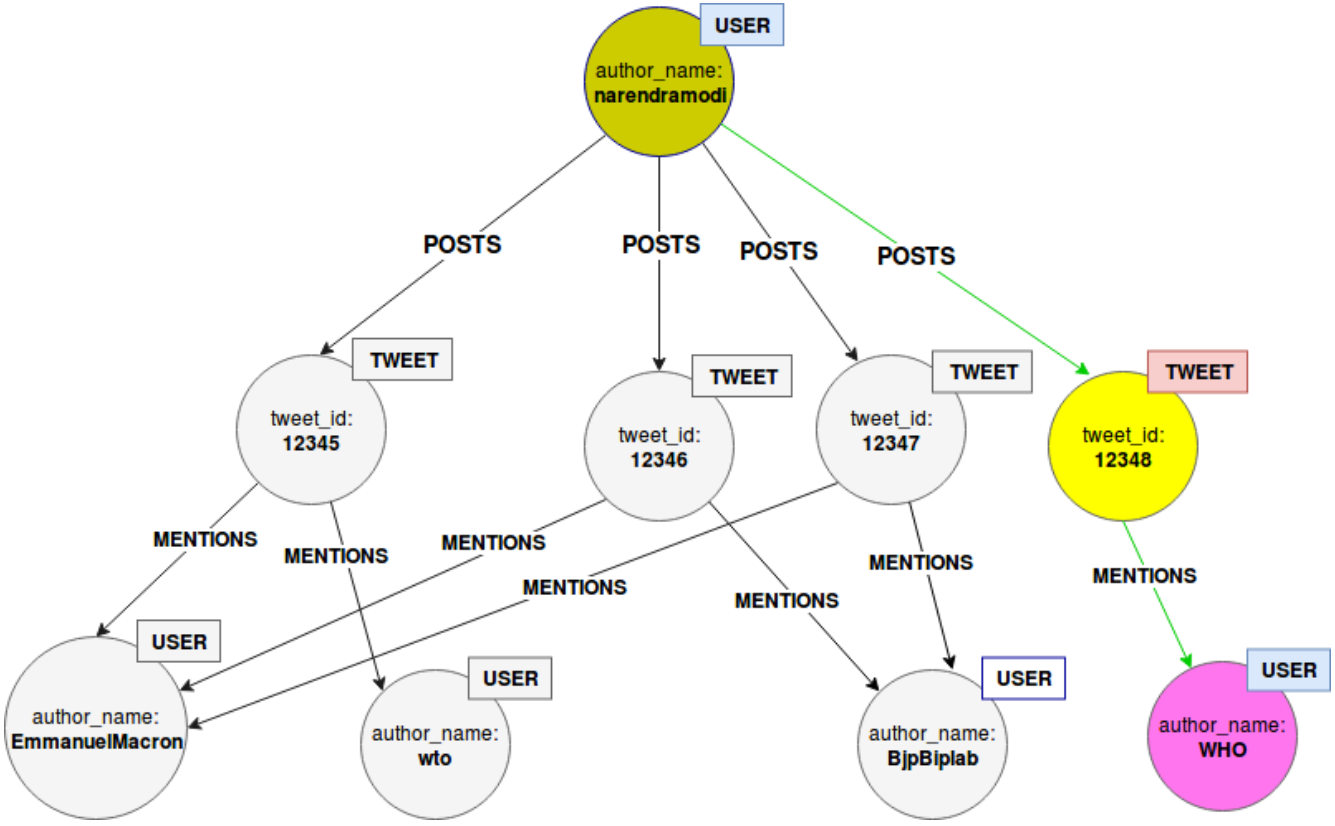
Second Match



Third Match


USER
author_name:
narendramodi
POSTS
POSTS
POSTS
POSTS
TWEET
tweet_id:
12345
TWEET
tweet_id:
12346
TWEET
tweet_id:
12347
TWEET
tweet_id:
12348
MENTIONS
MENTIONS
MENTIONS
MENTIONS
MENTIONS
MENTIONS
USER
author_name:
EmmanuelMacron
USER
author_name:
wto
USER
author_name:
BjpBiplab
USER
author_name:
WHO

Fourth Match



Running The Query

Cypher Query Result:

First_Author	Second_Author	tweet_ids	Total_Mention_Count
"narendramodi"	"EmmanuelMacron"	["12345" , "12346" , "12347"]	3
"narendramodi"	"BjpBiplab"	["12346" , "12347"]	2
"narendramodi"	"wto"	["12345"]	1
"narendramodi"	"WHO"	["12347"]	1

Total 4 rows

References

1. https://neo4j.com/docs/cypher-refcard/current/?ref=browser-guide
2. https://neo4j.com/sandbox-v2/
3. https://www.slideshare.net/neo4j/data-modeling-with-neo4j-25767444
4. https://neo4j.com/developer/get-started/

Thank you