ISAD353: Advanced Databases and Data Management

Seminar 4: Graph databases

Marco Palomino

Module Aims

To expose students to the challenges of solutions for managing, processing, analysing and interpreting large amounts of unstructured data within relational and **non-relational database** environments.

Assessed Learning Outcomes

Critically evaluate current and emerging approaches for analysing and interpreting data using data mining and business intelligence techniques.

Coursework (NoSQL)

“... Netflix is **considering** the use of Cassandra—an open source, NoSQL distributed database—for promptly writing high volumes of data into storage, and Redis—another open source, NoSQL database—for rapidly reading high volumes of data.”

“.. you are to write a 3,000-word report discussing the particular NoSQL database product that you have chosen for handling Netflix's **recommendation system**.”

Recap

- NoSQL databases:
    - NOT based on the relational model.
    - Support distributed database architectures.
    - Provide high scalability, high availability and fault tolerance.
    - Geared towards performance.

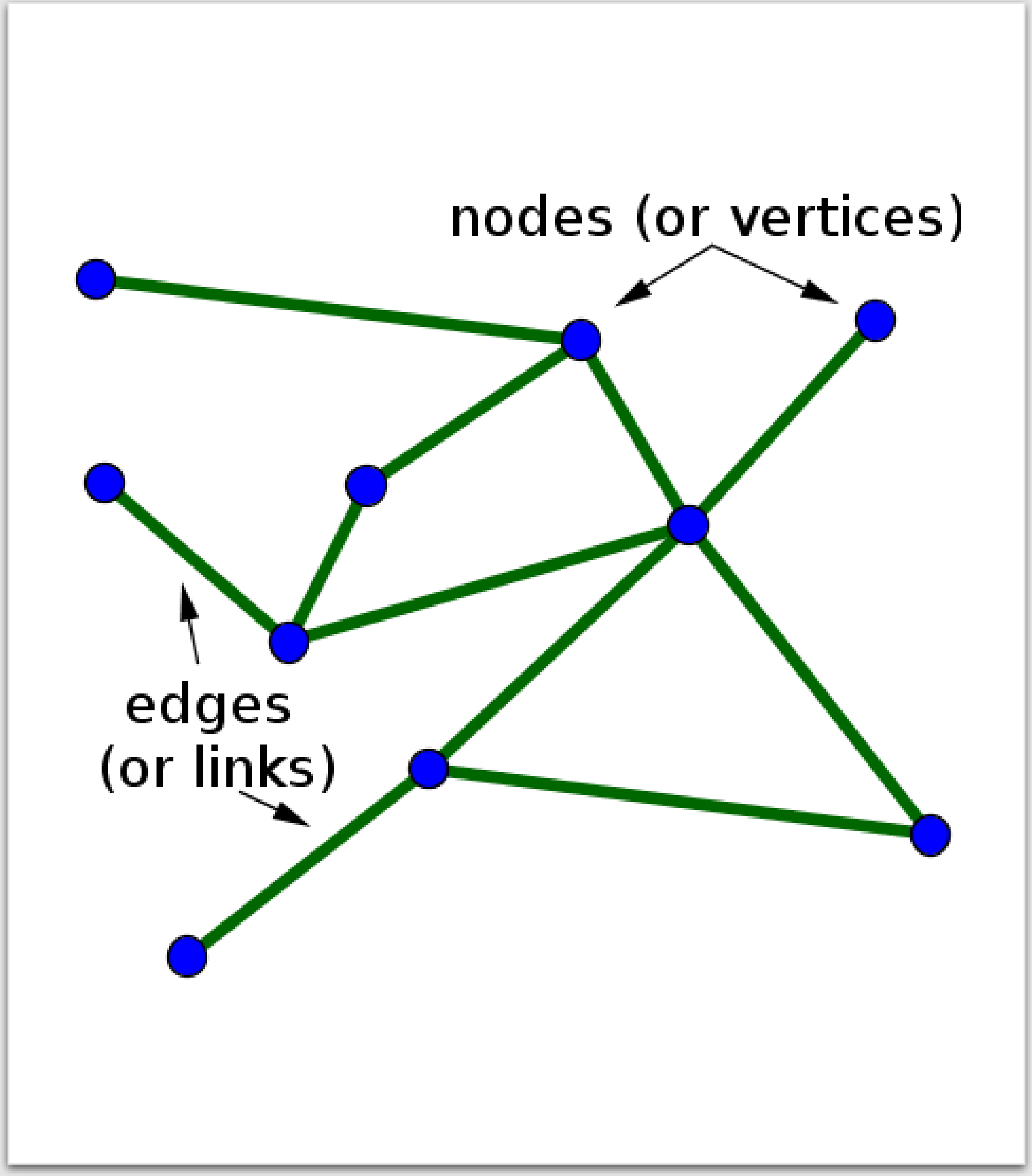
Outline

- Graph databases
- Neo4j

Graph databases

- **Graph databases** are based on **graph theory**, and use **nodes** and **edges** to represent and store data.

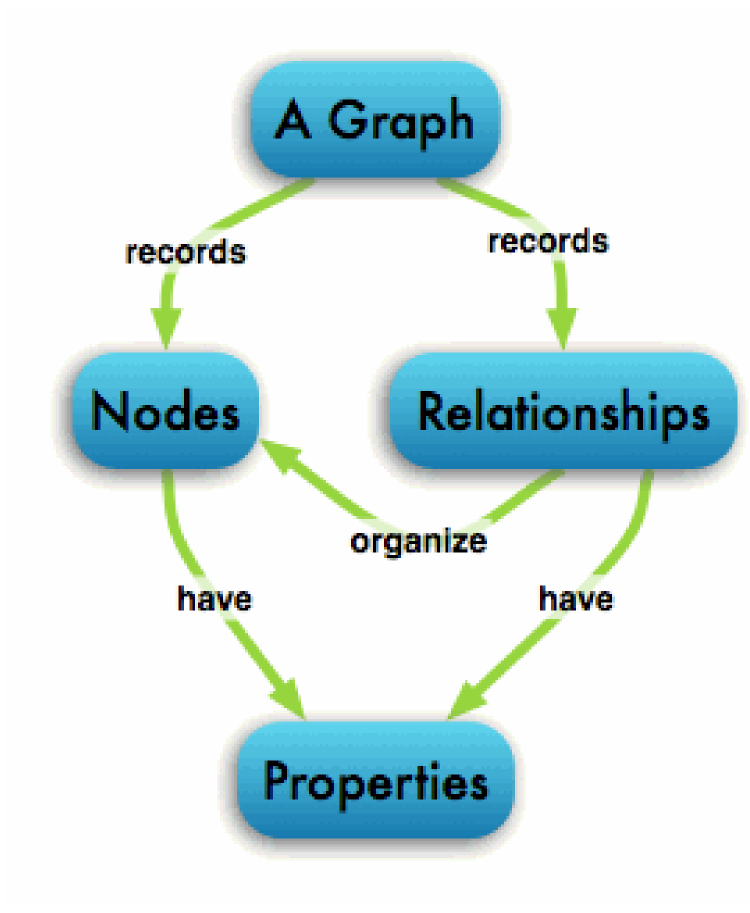
- A graph database is essentially a collection of nodes and edges. Each **node** represents a **record** in the database, and each **edge** represents a connection or **relationship** between two records.



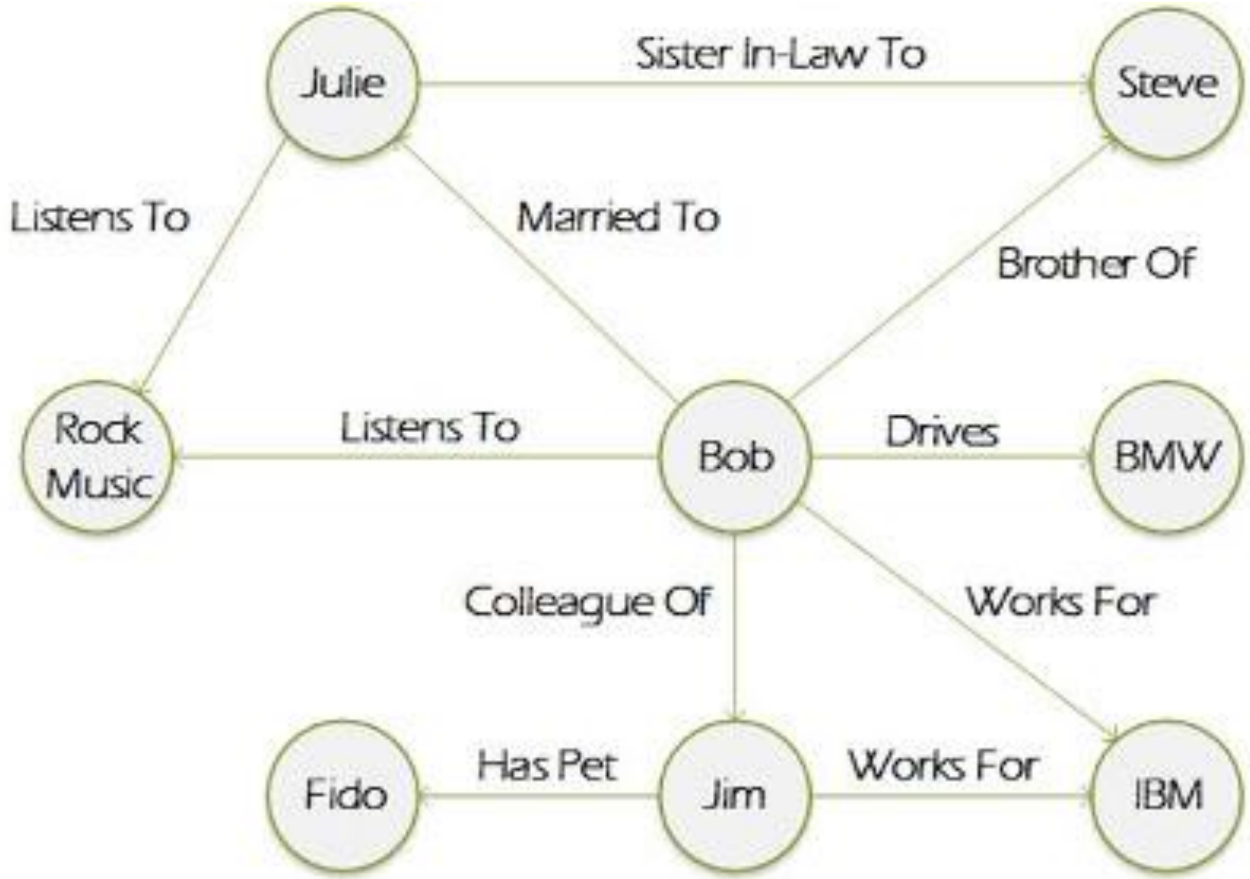
Terminology

- Graph databases store **entities** and **relationships** between entities.
- **Entities** are also known as **nodes**, which have **properties**.
- A node, or record, is an instance of an object in the application.
- **Relations** are known as **edges** that can have **properties** and **directional significance**.
- Graph databases let data to be stored once and then interpreted in different ways based on relationships.

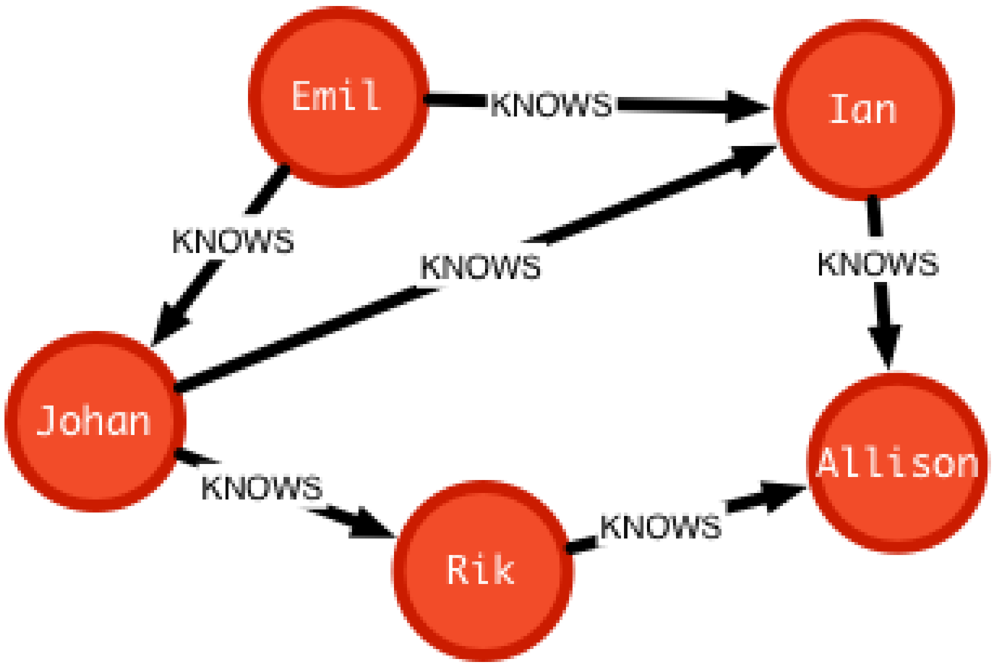
Nodes, edges, relationships



- **Nodes** are roughly the equivalent of the **record**, or **row** in a relational database, or the **document** in a document database.
- **Edges**, also known as **relationships**, are the lines that connect nodes to other nodes; they represent the relationship between them.
- **Properties** are pertinent information that relate to nodes and relationships.


Julie
Sister In-Law To
Steve
Listens To
Married To
Brother Of
Rock Music
Listens To
Bob
Drives
BMW
Colleague Of
Works For
Fido
Has Pet
Jim
Works For
IBM

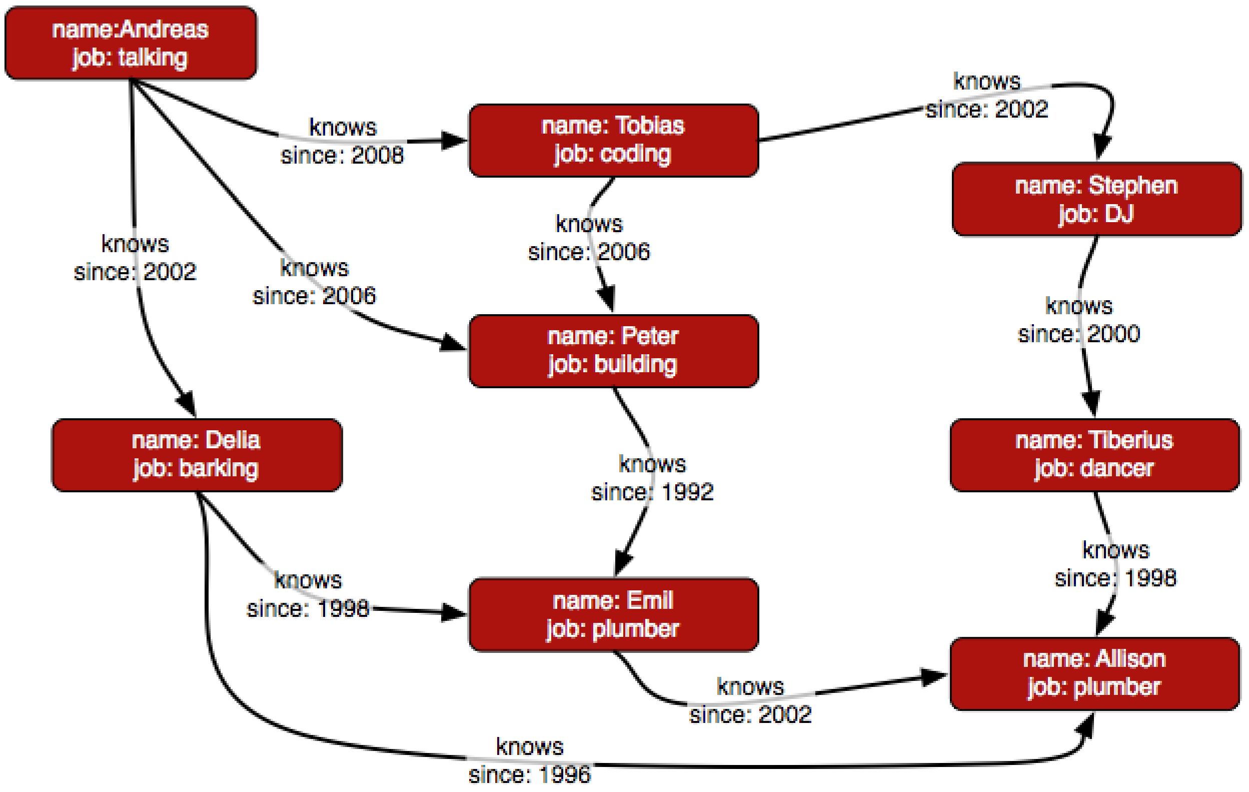
Relationships



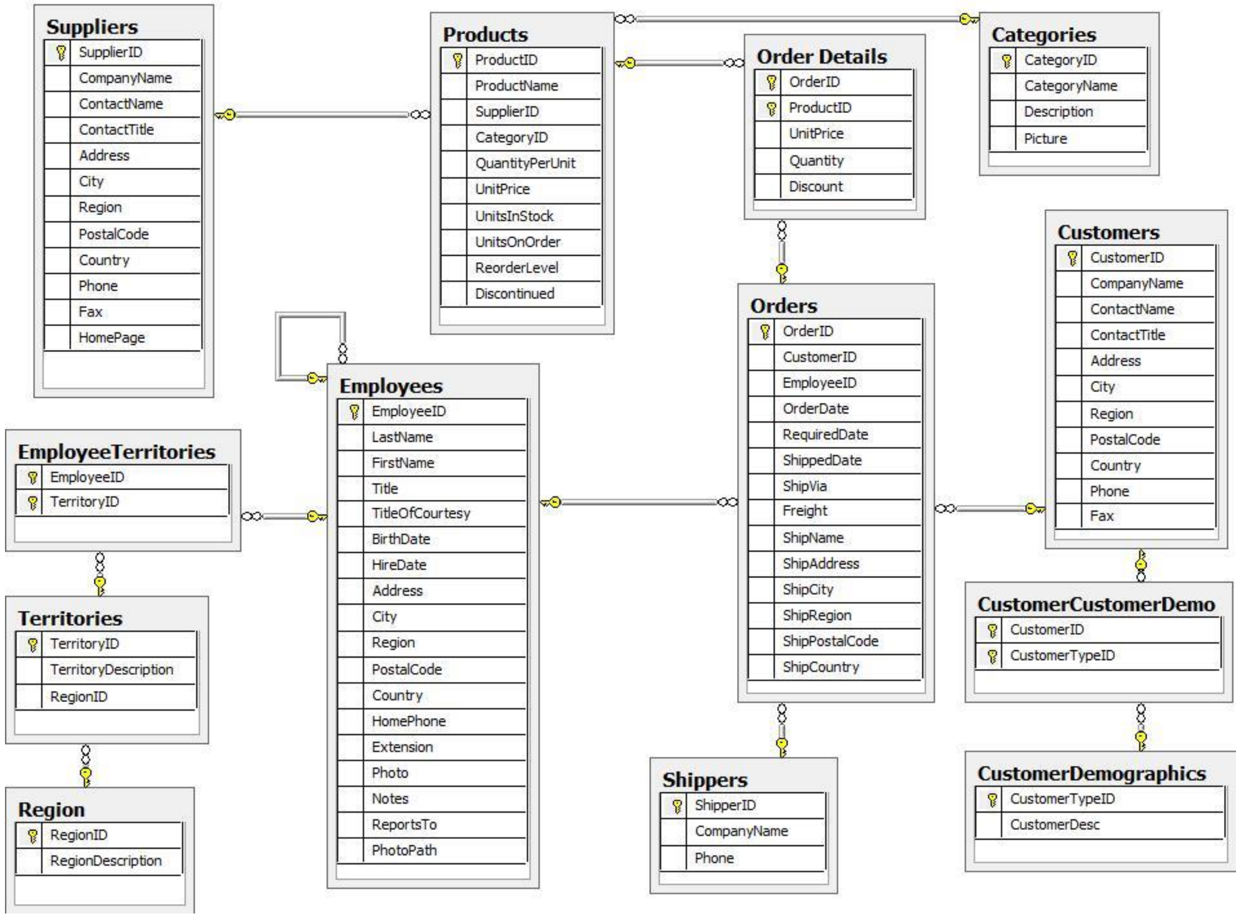
- **Relationships always have direction.**
- Relationships always have a **type**: for example, KNOWS.
- A relationship always has a **direction**, a **type**, a **start node**, and an **end node**.

Properties

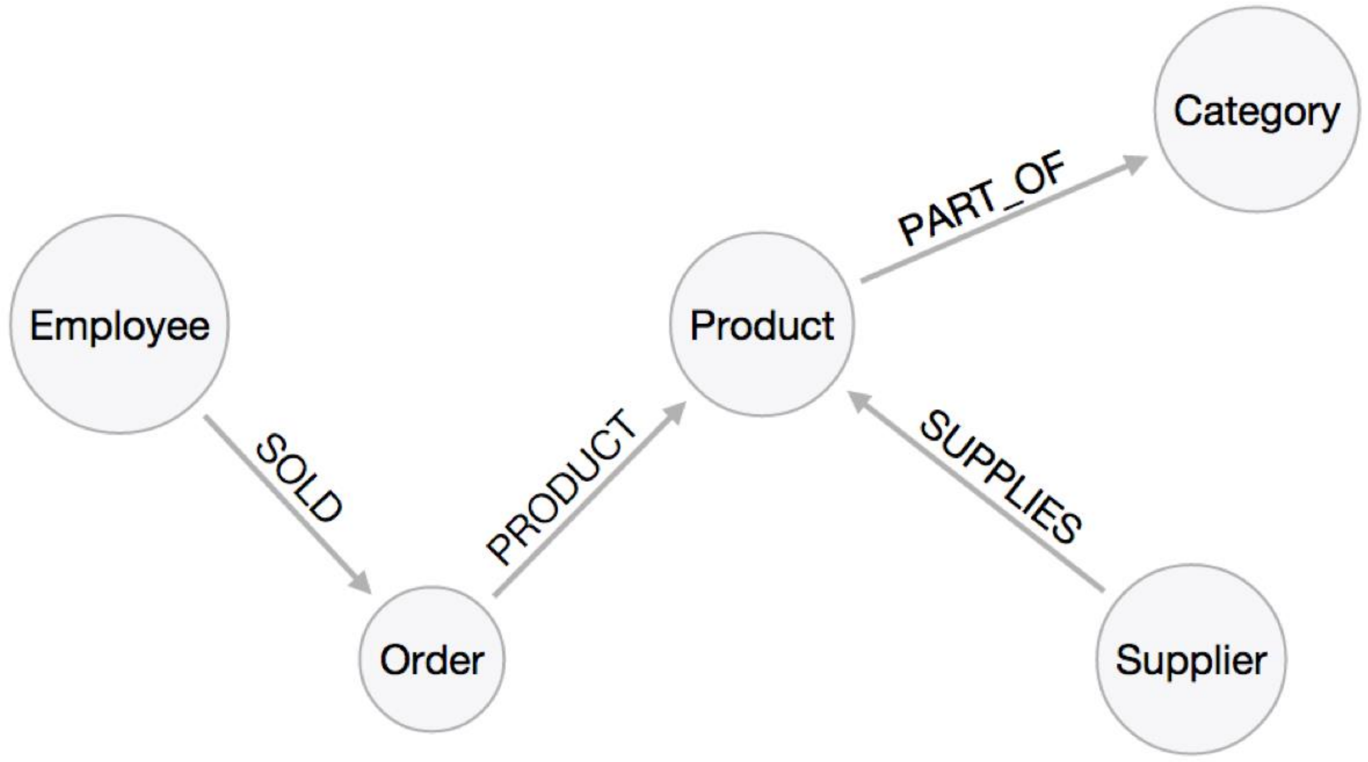
Each node or edge has a collection of key-value pairs, or **"properties"**, describing it.



Relational model



Graph model


Category
PART_OF
Employee
Product
SOLD
PRODUCT
SUPPLIES
Order
Supplier

Examples of graph databases

- Neo4j
- InfiniteGraph
- OrientDB
- FlockDB

What is Neo4j?

- Neo4j is an open-source NoSQL graph database implemented in Java and Scala.
- With development starting in 2003, it has been publicly available since 2007.
- The source code and issue tracking are available on GitHub, with support readily available on Stack Overflow and the Neo4j Google group.
- Use cases include matchmaking, network management, software analytics, scientific research, organisational and project management, recommendations, social networks...

Cypher Query Language (CQL)

- Cypher is Neo4j’s query language.

- Cypher is a **declarative**, SQL-inspired language for describing patterns in graphs.

- It allows us to state what we want to **select**, **insert**, **update** or **delete** from our graph data without requiring us to describe exactly how to do it.

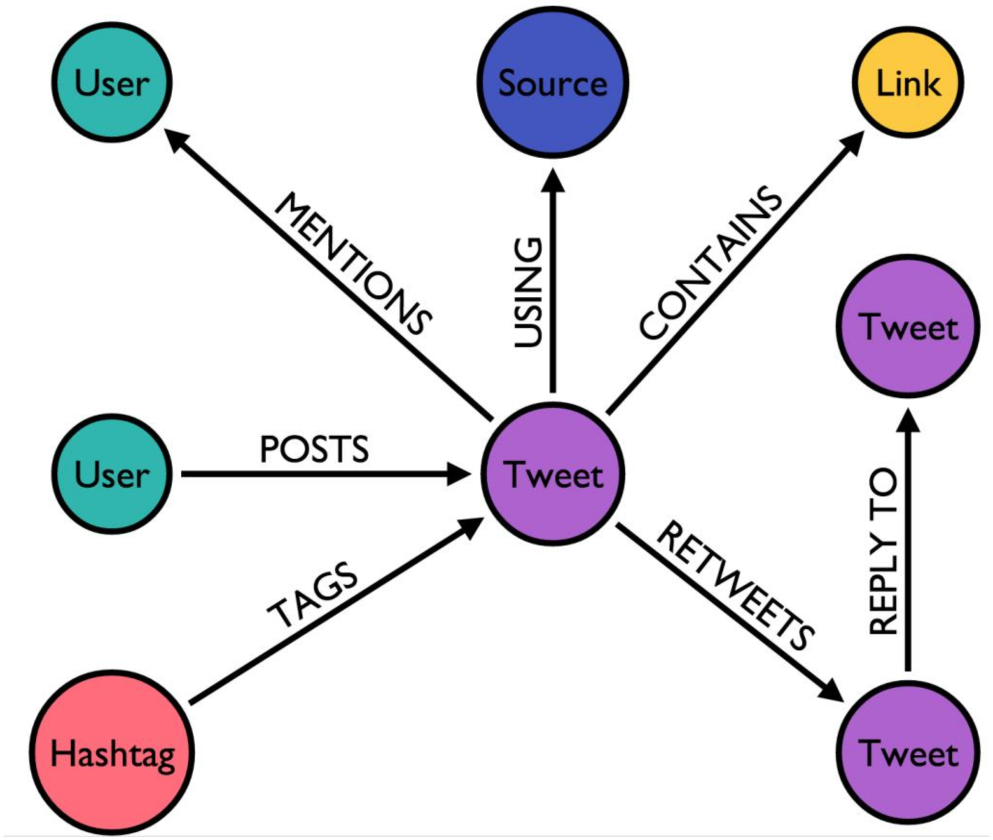
Examples

- We represent nodes (or entities) with round parentheses:
  (prod:Product)
- Relationships are drawn as arrows -->, we can add relationship-type and other information in square brackets
  -[:ORDERED_BY]->
- Bringing both together: (cust:Customer)-[:ISSUED]->(o:Order)-[:CONTAINS]->(prod:Product).
- MATCH clause for matching patterns in data.

Useful to model social networks

- **Social networks** are a natural ground to deploy and use graph databases.

- Properties can add intelligence to the relationships: since when they became friends, what is the distance between nodes, what aspects are shared between nodes...



Node traversal

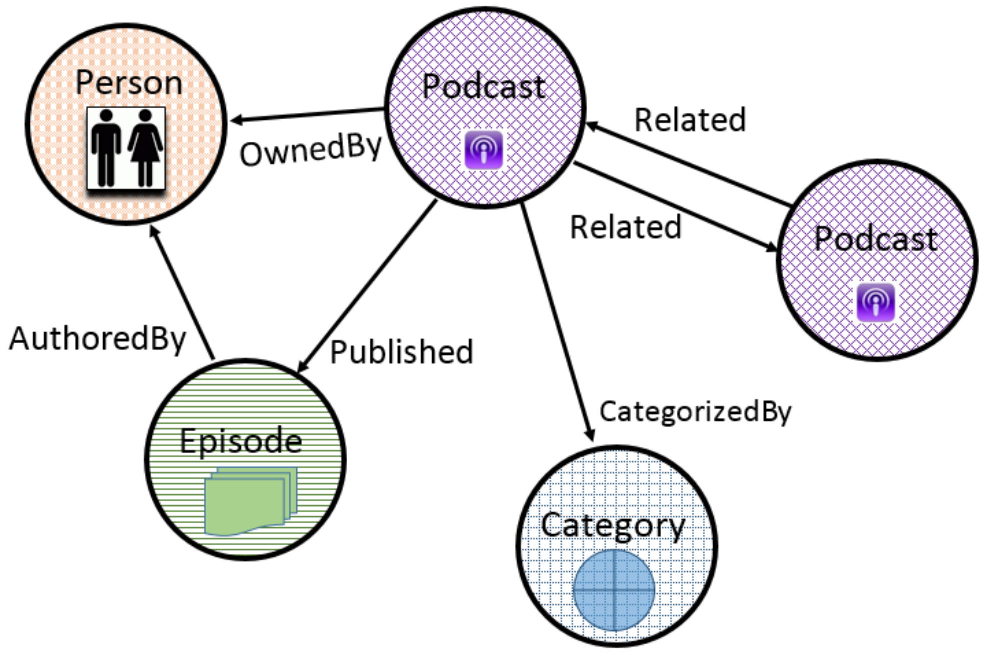
- Graph databases are really powerful when you want to traverse graphs at any depth and specify a starting node for the traversal.

```
Node barbara = nodeIndex.get(“name”, “Barbara”);
```

```
Traverser friendTraverser = Barbara.traverse(Order.
BREADTH_FIRST, StopEvaluator.END_OF_GRAPH,
EdgeType.FRIEND, Direction.OUTGOING);
```

The feature is used in social networks to show relationships between any two nodes, the distance between nodes (i.e., the number of “hops” on the graph needed to reach the destination from the start).

Recommendation engines



- As nodes and relationships are created, they can be used to make **recommendations** like "your friends also bought this product", or "when invoicing this item, these other items are usually invoiced."

Example – Wal-Mart

- Wal-Mart (the multinational retail corporation) uses Neo4j to store lists of products that people buy, determine how those people and products relate to other people and products, and make recommendations based on those observations.

- Because of the way graph databases maintain relationships among the data stored in the graph, it can very quickly determine how things relate to one another.

Not to use when...

- If you want to regularly update all, or a subset of nodes—for example, in an analytics solution where all entities need to be updated with a newly added property—graph databases may not be optimal, since changing a property in all nodes is not straightforward.

A Risk Discovery Information System

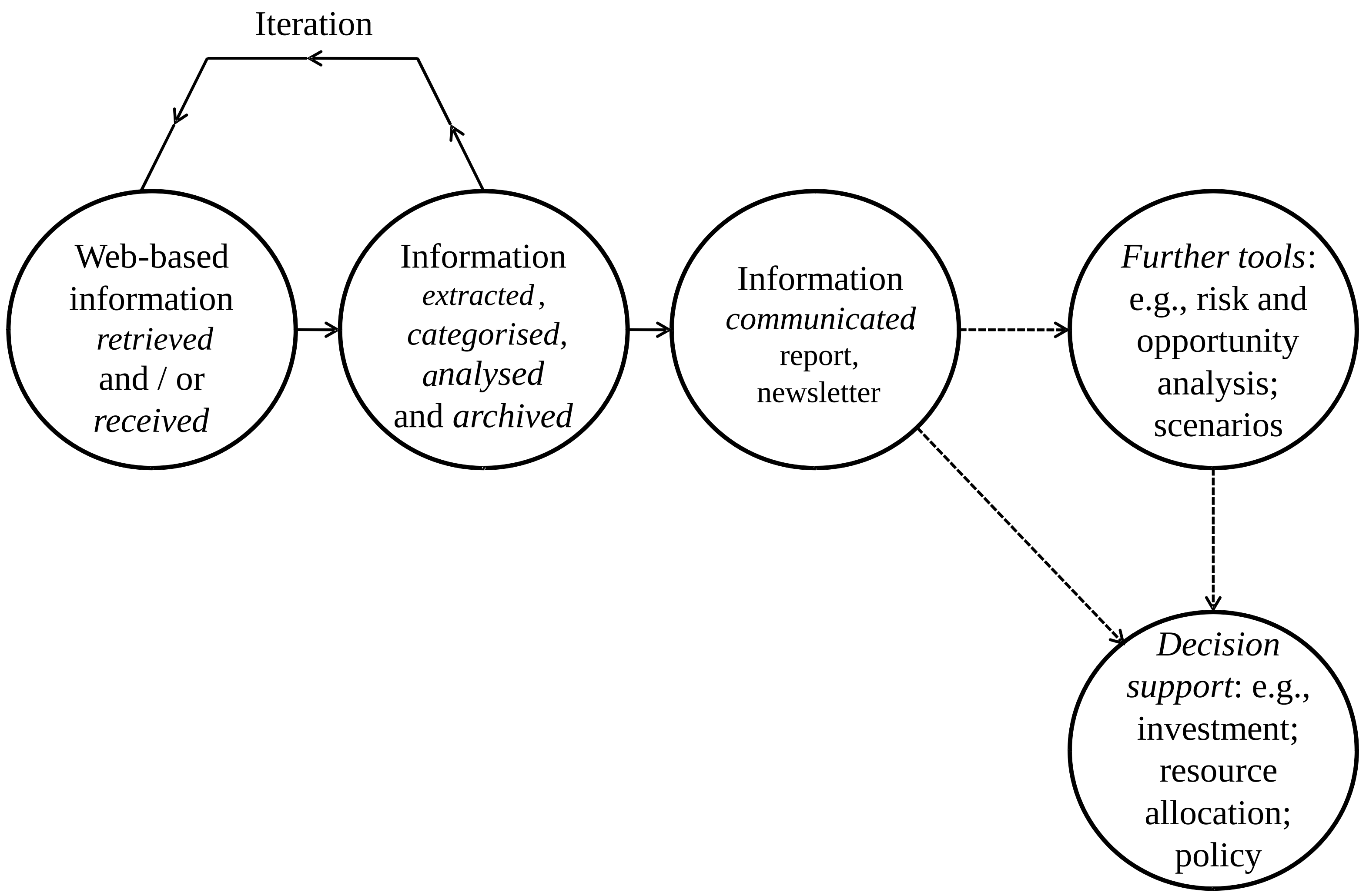
- Lloyd’s defines an **emerging risk** as an issue that is perceived as **potentially significant** but may **not be fully understood** yet, in terms of insurance, pricing, reserving or capital setting.

- Lloyd’s requires an information system to **identify emerging risks**, accumulate reliable data and knowledge about them, and thus **support decision making**.

Specification

- **Who will use the system:** Lloyd’s Emerging Risks Team.
- **Where and when it will be used:** Lloyd’s of London in September 2014.
- **What the system will do:** Continuously and automatically search the Web for information related to specific topics chosen by Lloyd’s Emerging Risks Team.

System specification


Iteration
Web-based information *retrieved* and / or *received*
Information *extracted*, *categorised*, *analysed* and *archived*
Information *communicated* report, newsletter
Further tools: e.g., risk and opportunity analysis; scenarios
Decision support: e.g., investment; resource allocation; policy

Implementation

- We developed a prototype to continuously search the Web for new and emerging information.
- Our prototype organised the collected information and displayed it on a Web browser.
- Marco A. Palomino, Alexandra Vincenti, Richard Owen, (2013) **"Optimising Web-based information retrieval methods for horizon scanning"**, foresight, 15(3), pp.159-176.

Comparison

	Emerging Risks Group	
	Very relevant	Relevant
Week 1	3	1
Week 2	2	1
Week 3	0	1
Week 4	3	1

Comparison

	Emerging Risks Group		Web-based Approach	
	Very relevant	Relevant	Very relevant	Relevant
Week 1	3	1	29	66
Week 2	2	1	19	64
Week 3	0	1	11	74
Week 4	3	1	5	74

Comparison

	Emerging Risks Group Current Practice		Web-based Horizon Scanning	
	Very relevant	Relevant	Very relevant	Relevant
Week 1	3	1	29	66
Week 2	2	1	19	64
Week 3	0	1	11	74
Week 4	3	1	5	74

M. A. Palomino, A. Vincenti, and R. Owen, "Optimising web-based information retrieval methods for horizon scanning", *foresight*, vol. 15, no. 3, pp. 159 - 176, 2013, ISSN 1463-6689.

References

- Neo4j: https://neo4j.com/product/
- Neo4j Tutorial: http://www.tutorialspoint.com/neo4j/
- Cypher Query Language: https://neo4j.com/developer/cypher/