**Experiment with Neo4j**

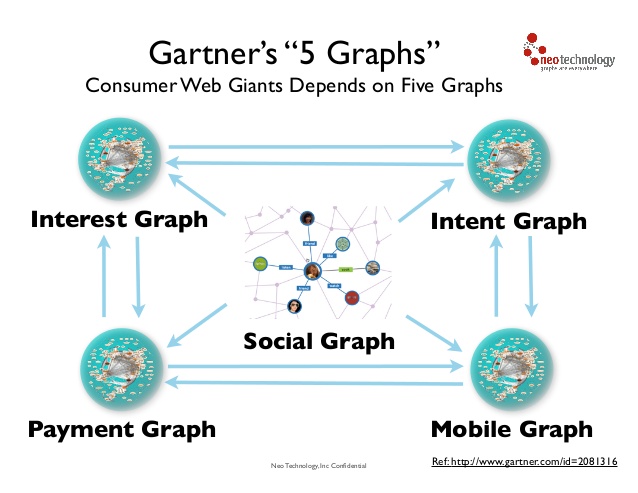
3252 Term Project

Yingjia Zhai

Dec 5, 2017



# Introduction

Graphs are extremely useful in understanding a wide diversity of datasets in fields such as science, governments, and business.Gartner’s five graphs indicate that some Consumer Web Giants, like Google andAmazon, havesignificant competitive advantagesin five areas – social, intent, consumption, interest, and mobile.

Numerous projects and products for managing, processing, and analyzing graphs have developed in recent years. The relational databases have increasing problems to cope with interdependency and complexity data. Between the beginning of 2013 and October 2016, the demand for Graph databases has increased 6 times. A survey on the worldwide adoption of graph databases by TechValidate and IBM published in February 2017 showed that 57% of company users across all industries cited speed and improved performance as the top technology benefit of using a graph database.

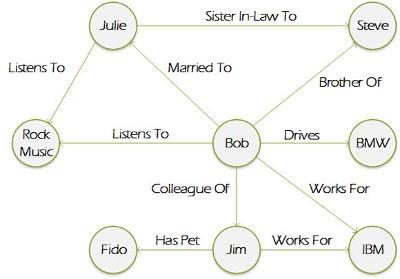
# Objectives

This report is going to discover the followings:

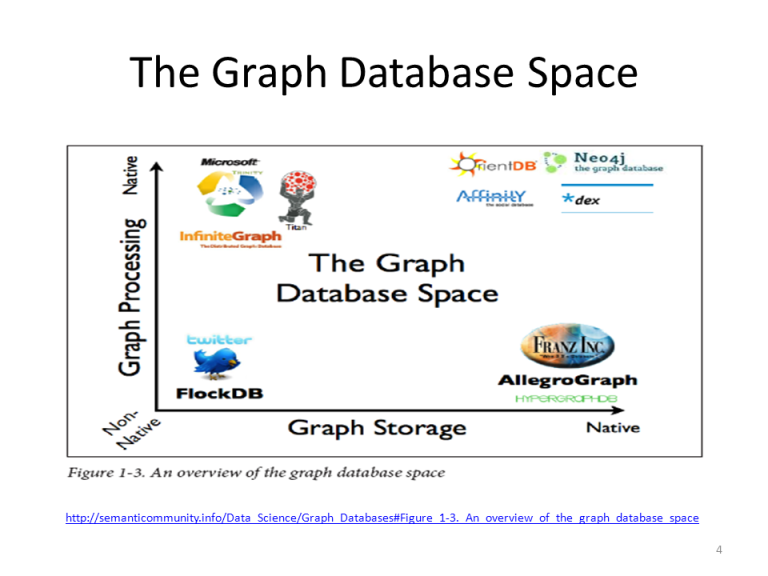
* + What is a graph database?
  + Use cases - why graphs?
  + Neo4j product overview
  + Cypher query language
  + Labeled property graph data model
  + [The Movie Database](https://neo4j.com/online_training/graphdatabases/?aliId=a2VsbHkgemhhaS9rZWxseS56aGFpd29ya0BnbWFpbC5jb20%3D#_the_movie_database)
  + RDBMS to graph
  + Neo4j in the NoSQL Ecosystem

# Findings

## What is a graph database?

A graph database, also called a graph-oriented database, is a type of [**NoSQL**](http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL) database that uses [graph theory](http://whatis.techtarget.com/definition/graph-theory) to store, map and query relationships. A graph database is essentially a **collection of nodes and edges**. Each node represents an entity (such as a person or business) and each edge represents a connection or relationship between two nodes. Every node in a graph database is defined by a **unique identifier**, a set of outgoing edges and/or incoming edges and a set of properties expressed as[**key/value pairs**](http://searchenterprisedesktop.techtarget.com/definition/key-value-pair).

Graph databases are well-suited for analyzing interconnections, which is why there has been a lot of interest in using graph databases to[**mine data**](http://searchsqlserver.techtarget.com/definition/data-mining)**from**[**social media**](http://whatis.techtarget.com/definition/social-media). Graph databases are also useful for working with data in business disciplines that involve complex relationships and [dynamic](http://searchnetworking.techtarget.com/definition/dynamic-and-static) [schema](http://searchsqlserver.techtarget.com/definition/schema), such as[**supply chain management**](http://searchmanufacturingerp.techtarget.com/definition/supply-chain-management). Below graph shows a pictorial overview of some of the graph databases on the market today based on their storage and processing models.



### Why are graph databases outpacing other DBMS?

• Performance:

With the data volume increasing in the future, the connections (or relationships) between data will increase dramatically. With traditional databases, relationship queries will come to a grinding halt as the number and depth of relationships increase. In contrast, graph database performance stays constant even as data grows year over year.

• Flexibility:

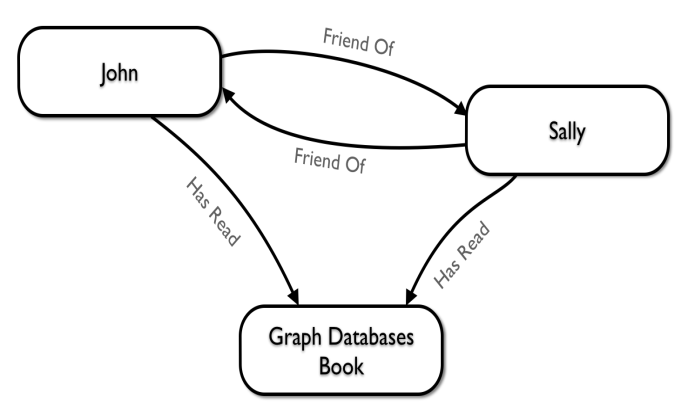
Using graph databases, IT and data architect teams can follow the speed of business because the structure and schema of a graph model flex as solutions and industry change. IT team doesn’t have to model domain ahead of time; instead, they can modify the existing structure without endangering current functionality.

• Agility:

Developing with graph databases aligns completelywith today’s agile, test-driven development practices, allowing graph-database-backed application to evolve with changing business requirements.

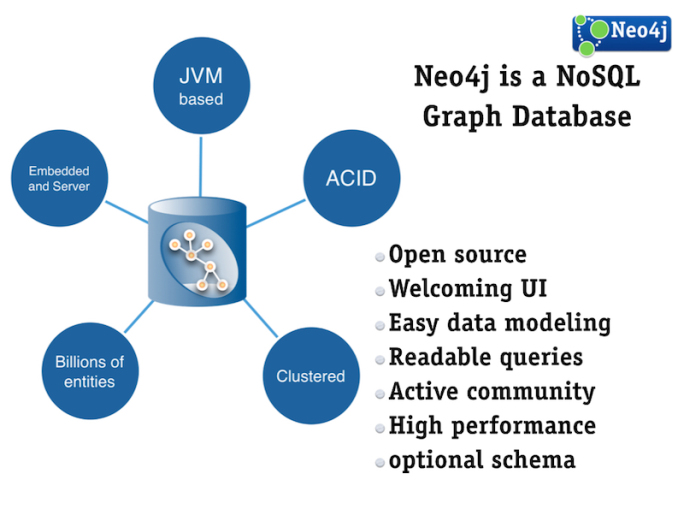
## Use cases - why graphs?

### What is a Graph?

A graph is composed of two elements: a node and a relationship.Graph is a connected data structure with nodes and relationships. Each node represents an entity (a person, place, thing, category or other piece of data), and each relationship represents how two nodes are associated. This general-purpose structure allows user to model all kinds of scenarios – from a system of roads, to a network of devices, to a population’s medical history or anything else defined by relationships.These days we are seeing many graphic projects in Finance, Social Network, HR, Manufacturing, Healthcare, Telecom, etc.

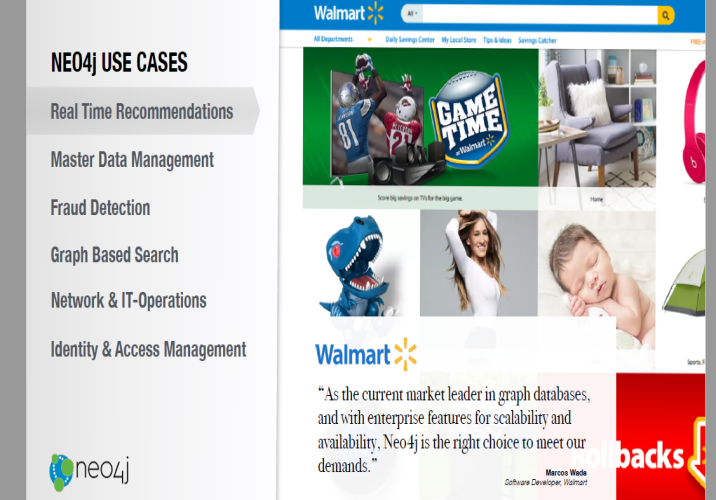
## Neo4j product overview

### What is Neo4j

* Neo4j is a **Database** - use it to reliably **store information** and **find it later**
* Neo4j’s data model is a **Graph**, in particular a **Property Graph**
* **Cypher** is Neo4j’s graph query language (**SQL for graphs!**)
* Cypher is a declarative query language: it describes **what** you are interested in, not **how** it is acquired.
* Cypher is meant to be very **readable** and **expressive**
* Neo4j provides full database characteristics including **ACID transaction compliance**, cluster support,and runtime failover, making it suitable to use graph data in **production scenarios**.
* Compact storage and memory caching for graphs, resulting in efficient scale-up and**billions of nodes** in one database on moderate hardware
* Written on top of the**JVM**

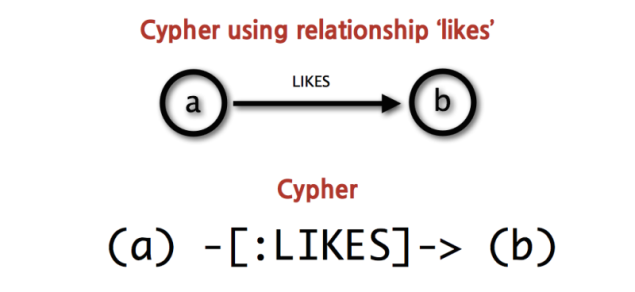
### NEO4j use cases

* Real Time Recommendations
* Master Data Management
* Fraud Detection
* Graph Based Search
* Network & IT- Operations
* Identity & Access Management.



## Cypher query language

### What is Cypher

Cypher is Neo4j’s open graph query language. Cypher’s syntax provides a familiar way to match patterns of nodes and relationships in the graph. Nodes are surrounded with parentheses which look like circles, e.g. **(node)**. It can be given variable like **(p)** for a person or **(t)** for a thing, and make variable names like **(person)** or **(thing)**. Relationships are basically an arrow **-->** between two nodes. Relationship information can be placed in square brackets inside of the arrow. Nodes and relationship expressions can be used to build more complex patterns. For example: friend-of-a-friend **(user)-[:KNOWS]-(friend)-[:KNOWS]-(foaf).**

### Examples for Cypher: Friendship Table

### Create a Record for Yourself

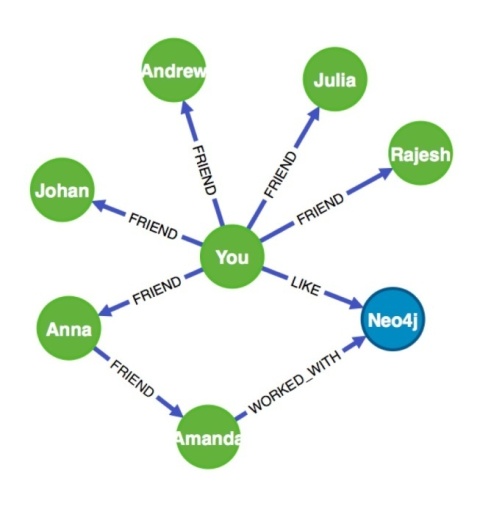
**CREATE (you:Person {name:"You"})**

**RETURN you**

**Find Your Friends**

**MATCH (you {name:"You"})-[:FRIEND]->(yourFriends)**

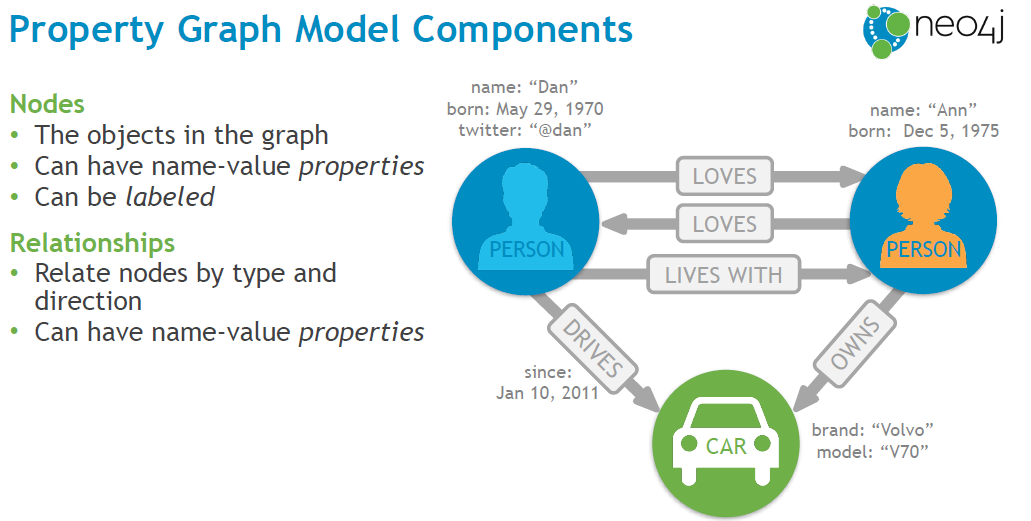
**RETURN you, yourFriends**



## Labeled property graph data model

The property graph contains connected entities (the *nodes*) which can hold any number of attributes (key-value-pairs). Nodes can be tagged with labels representing their different roles in your domain.

*Relationships* provide directed, named semantically relevant connections between two node-entities. A relationship always has a direction, a type, a ***start node***, and an ***end node***. Like nodes, relationships can have any properties. In most cases, relationships have quantitative properties, such as weights, costs, distances, ratings, time intervals, or strengths. Below is an example for Property Graph Model.

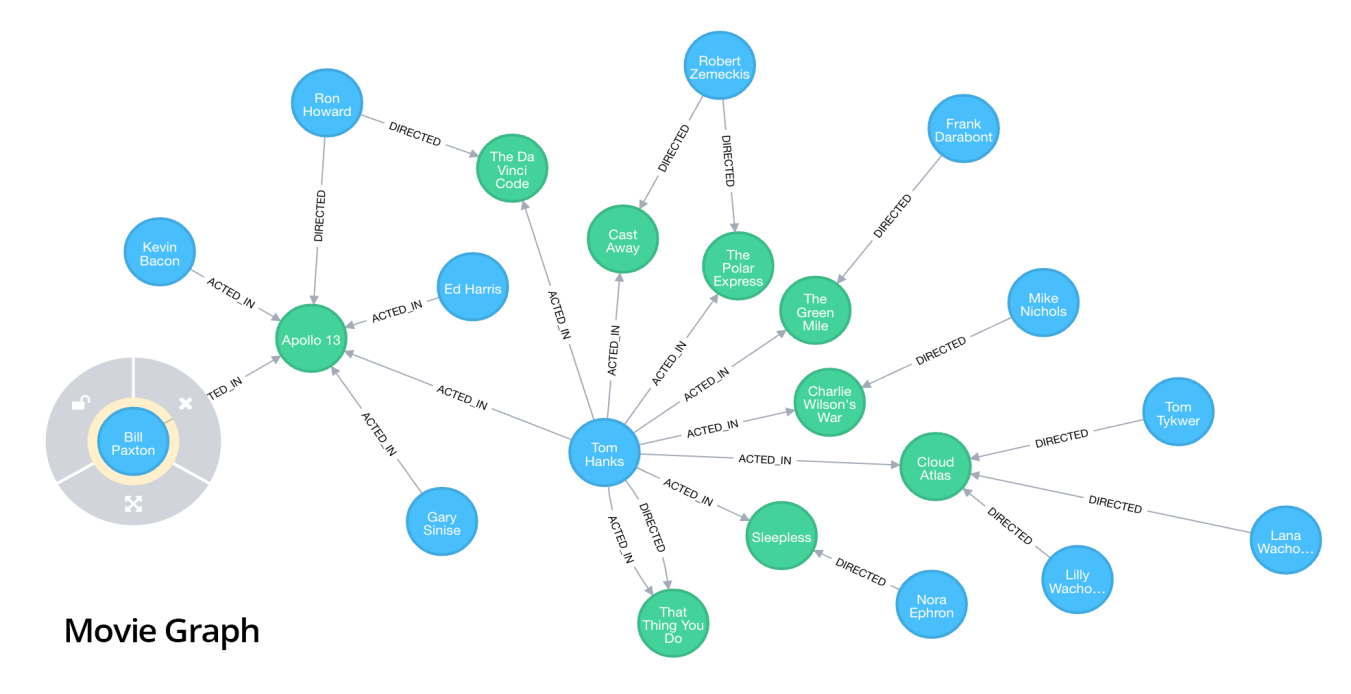


## [The Movie Database](https://neo4j.com/online_training/graphdatabases/?aliId=a2VsbHkgemhhaS9rZWxseS56aGFpd29ya0BnbWFpbC5jb20%3D#_the_movie_database)and Graph

Here we will give an example: the Movie Dataset to learn how to create model, load data and query data in Neo4j.

The data model in this Movie Dataset includes nodes with three different labels (each with their own properties), and six different types of relationships (one of which has its own property). In brief, the graph is made up of **Person**, **Movie**, and **Genre** nodes that are related to each other in various ways.

The Movie Graph is a mini graph application containing actors and directors that are related through the movies they've collaborated on.

Examples Code:

* **The node’s properties** (**title**, **released**, et cetera) are represented as a list of key/value pairs, enclosed within a pair of braces.

**{title:"A Title", released:2000, …​}**

* **Find information by using Properties.**  **MATCH** and **RETURN** nodes whose **title** is "The Matrix".

**MATCH (movie:Movie {title:"The Matrix"})**

**RETURN movie**

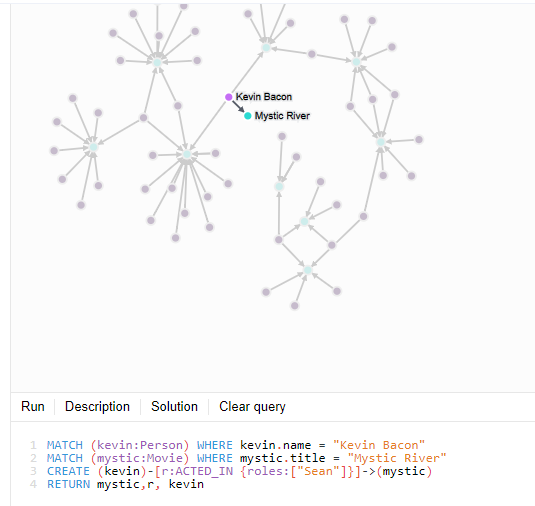
* **Relationships:** (actor:Person)-[:ACTED\_IN]->(movie:Movie) **d**escribe the pattern to retrieve only nodes that had a relationship type ACTED\_IN with other nodes.
* **Patterns：**Combining the syntax for nodes and relationships, we can express patterns.

**MATCH (matrix:Movie {title:"The Matrix"} )<-[role:ACTED\_IN {roles:["Neo"]}]-(keanu:Person {name:"Keanu Reeves"})**

**RETURN matrix, role, keanu**



* **Find** the actor Kevin Bacon and the movie Mystic River and add the relationship between the movie and the actor to the dataset.



* **CREATE** the Movie nodes by using the data from movies.csv as properties.

**LOAD CSV WITH HEADERS**

**FROM** [**http://neo4j.com/docs/stable/csv/intro/movies.csv**](http://neo4j.com/docs/stable/csv/intro/movies.csv)

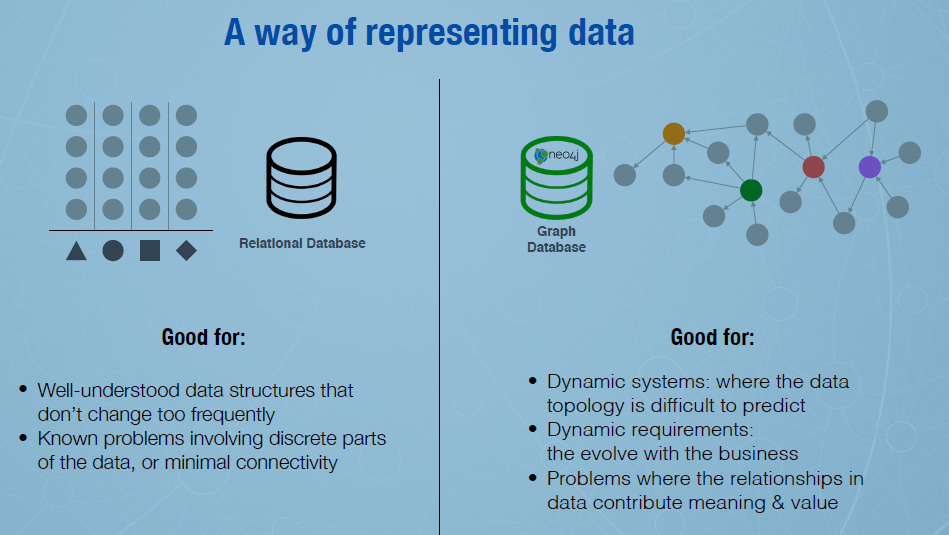
**AS line**

**CREATE (movie:Movie { id:line.id, title:line.title, released:toInt(line.year) });**

# Appendices

## RDBMS to graph

Below graph indicates that Graph Database is good for dynamic systems where the relationship between data is continually growing with business.



### Neo4j in the NoSQL Ecosystem

With the advent of the NoSQL movement, businesses of all sizes have a variety of modern options from which to build solutions relevant to their use cases.

* Calculating average income? Ask a relational database.
* Building a shopping cart? Use a key-value Store.
* Storing structured product information? Store as a document.
* Describing how a user got from point A to point B? Follow a graph.