Graph & Modern Databases

COMP1835

1

1

Graph Databases and Neo4J

https://neo4j.com/style-guide/

2

2

Objectives

- Introduce Graph theory
- Describe two different Graph data models
- Introduce Neo4J
- Discuss Neo4J data model
- Introduce Cypher language

3

3

Part 1

4

4

The problem

- Supporters of key-value stores, document databases and relational systems disagree about practically every aspect of database design, but they do agree in one aspect:
  - databases are about storing information about “things”, but sometimes (and in the last decade – very often) it is **the relationship between things**, rather then things themselves, that are the main interest of a business.
- Real-world data is increasing in volume, velocity and variety.
  - As a result, data relationships – which could be very valuable – are growing at an even faster rate.
- The problem:
  - Relational databases aren’t designed to capture rich relationship information
  - Many NoSQL solutions do not support relationships between objects (as there are no JOINS)
- So, applications (and the enterprises that create them) are missing out on critical connections essential for today’s high-stakes, data-driven decisions

5

5

Solution

- Mentioned above problems are solved by Graph databases
- Relational databases are based on a strong theoretical foundation – Relational theory, introduced by Dr Codd in 1970
- The Graph databases, (unlike other NoSQL data stores), are based on another strong theoretical foundation – Graph theory
  - a long-established branch of mathematics with many practical applications in medicine, physics, sociology, as well as in computer science

6

6

A Bit of History

- Graphs are actually quite old as a concept.
- They were invented, or at least first described, in an academic paper by the well-known Swiss mathematician Leonhard Euler in 1736.
  - He was trying to solve an old problem that we now know as the *7 bridges of Königsberg*:
  - The city (now Kaliningrad) had seven bridges that were connected to the four different parts of the city; the problem was – how to take a tour of the city, visiting every part and crossing every bridge, without having to walk a single bridge or street twice?
- Euler suggested an abstract model: consisting of the parts of the city and the bridges connecting the parts of the city – 'the first world graph'
- Using this model Euler came to the conclusion:
  - with the current state of the city, and its bridges at the time, there was no way one could take such an Eulerian Walk of the city; however adding one more bridge would immediately make it possible.

7

7

The Graph Theory

- There are many problems that can be:
  - Described using the graph metaphor of objects and pairwise relations between these objects
  - Solved by applying a mathematical algorithm to this structure
- The scientific discipline that studies these modeling and solution patterns, using graphs, is often referred to as the graph theory, and it is considered to be a part of discrete Mathematics.
- Graph theory defines the following major components of a graph:
  - Vertices, or "nodes", representing distinct objects
  - Edges, or "relationships", or "arcs" that connect these objects
  - Both vertices and edges can have properties

8

8

Benefits

- The biggest value that graphs bring to the development stack is their ability to store relationships and connections as first-class entities.
- The early adopters of graph technology reimagined their businesses around the value of data relationships.
- Now they become industry leaders:
  - LinkedIn,
  - Google,
  - Facebook
  - PayPal

A basic graph of a fraud ring sharing similar contact information.

9

9

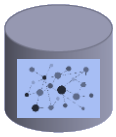
Modern Applications

- Social studies:
  - global demographics, political movements, commercial adoption of certain products by certain groups (Google, Facebook, Twitter, LinkedIn)
- Biological studies:
  - biological components (proteins, molecules, genes, and so on) and their interactions
- Computer science:
  - chip design, network management, recommendation systems, UML modeling to algorithm generation, dependency analysis and machine learning
  - Web Search
- Flow problems, route problems

10

10

Graph Database



- A graph database is an online database management system with Create, Read, Update and Delete (CRUD) operations working on a graph data model
- Unlike other databases, relationships take first priority in graph databases
- Graph databases are generally built for use with transactional (OLTP) systems
  - Accordingly, they are normally optimized for transactional performance, and engineered with transactional integrity and operational availability in mind
- Graph data stores could be divided into 2 subcategories, based on their data model:
  - RDF – based
  - Labeled Property Graph (or just Property Graph)

11

11

Types of Graph Databases –RDF

- RDF stands for Resource Description Framework
  - It's a W3C standard for data exchange in the Web
  - It has been developed in the 1990s for the modelling of the web resources and relationships between them
  - It became very popular at the beginning of the century with the article the Semantic Web published in *Scientific American* by Tim Berners-Lee, Jim Hendler and Ora Lassila
  - It is one of the earliest standards for representing and processing graph data
- RDF model presents the world in terms of connected entities
- Information in RDF is expressed in triples:
  - subject-predicate-object data structure
- Native RDF databases first were known as triplestores, next they were called quad stores, then RDF stores, and most recently they call themselves "semantic graph database":
  - Examples: AllegroGraph, AWS Neptune, StarDog, Oracle Spatial…

12

12

Types of Graph Databases –LPG

- The Labeled Property Graph (LPG), was developed more or less at the same time by a group of Swedish engineers
  - They were developing a ECM system in which they decided to model and store data as a graph
  - The motivation was not so much about exchanging or publishing data; they were more interested in efficient storage that would allow for fast querying and fast traversals across connected data
- If the RDF model is more about data exchange, the Labeled Property Graph is purely about storage and querying
- The Property Graph model provides a richer model for representing complex data by associating both nodes and relationship with attributes:
  - Examples: Neo4j , TigerGraph (formerly named GraphSQL), DataStax ...

13

13

Data model – LP Graph

- In graph theory graph is formed of two components: vertices and the edges that connect them
- In a LPG:
  - Vertices are called nodes,
    - Nodes have a uniquely identifiable ID and a set of key-value pairs, or properties, that characterize them
  - Edges, or connections between nodes called relationships
    - Relationships have an ID and a type and a set of key value of pairs (properties) that characterize the connections
  - Vertices and Edges have internal structure

14

14

Data model – RDF Graph

- In RDF the core component is a triple:
  - subject-predicate-object
- Vertices, called nodes, or resources
  - they can be either subject or object
  - they are identified by a URI, which is a unique identifier
- Edges, called relationships,
  - are identified by a URI
- Both vertices and Edges are identified by a URI, which is a unique identifier
- Neither nodes nor edges have an internal structure; they are purely a unique label

15

15

Quiz

- **Question 1:**Graph theory is a very recent field in modern Mathematics, invented in the late 20th century by Leonard Euler:
  A. True
  B. False
- **Question 2:** Name one field that graphs are NOT used for in today's science/application fields:
  A. Route problems
  B. Social studies
  C. Accounting systems
  D. Biological studies
- **Question 3:** Graphs are a very niche phenomenon that can only be applied to a very limited set of applications/research fields:
  A. True
  B. False

16

16

Part 2

Neo4J

https://neo4j.com/style-guide/

17

17

Neo4J

- Official Online Resources - http://neo4j.org
- History — Created at Neo Technologies in 2003. (Yes, this database has been around before the term NoSQL was known popularly)
- Technologies and Language — Implemented in Java
- Access Methods
  - A command-line access to the store is provided
  - Neo4J Browser, Neo4J Desktop and Neo4J Bloom
  - REST interface also available
  - Client libraries for Java, Python, Ruby, Clojure, Scala, and PHP exist
- Query Language
  - Cypher Query Language
  - Supports SPARQL protocol and RDF Query Language
- Open-Source License – community edition
- Who Uses It
  - More than 75% of the Fortune 100: Walmart, eBay, Adobe, Verizon, Orange, Airbus, Boeing, Volvo, JPMorgan Chase, UBS, Marriott, …

18

18

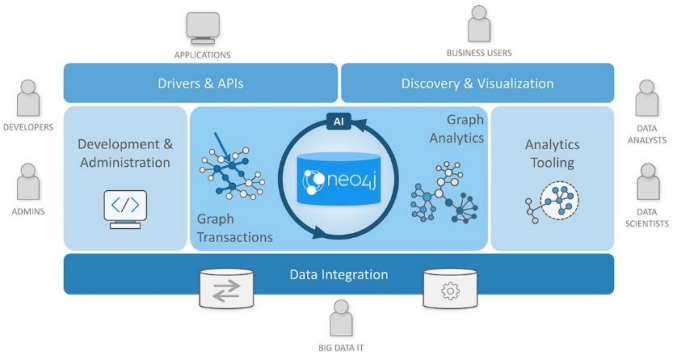
Advantages of Neo4j

- **Flexible data model**
  - Neo4j provides a flexible simple and yet powerful data model, which can be easily changed according to the applications and industries
- **High availability**
  - Neo4j is highly available for large enterprise real-time applications with transactional guarantees
  - Neo4j supports full ACID (Atomicity, Consistency, Isolation, and Durability) rules
- **Connected and semi structures data**
  - Using Neo4j, you can easily represent connected and semi-structured data.
- **Easy retrieval**
  - Using Neo4j, you can not only represent but also easily retrieve (traverse/navigate) connected data faster when compared to other databases.
  - Neo4j provides a built-in **Neo4j Browser** web application to create and query your graph data
- **Cypher query language**
  - Neo4j provides a declarative query language to represent the graph visually, using an ascii-art syntax. The commands of this language are in human readable format and very easy to learn
- **No joins**
  - Using Neo4j, it does NOT require complex joins to retrieve connected/related data as it is very easy to retrieve its adjacent node or relationship details without joins or indexes

19

19

Neo4j Graph Platform



- Components:
  - Neo4j Desktop – application to manage local instances of Neo4j
  - Neo4j Browser – online browser interface to query and view the data in the database (basic visualization capabilities using Cypher query language)
  - Neo4j Bloom – visualization tool for business users that does not require any code or programming skills to view and analyse data

Image source: https://neo4j.com/developer/graph-platform/ 20

20

Data Modeling in Neo4J

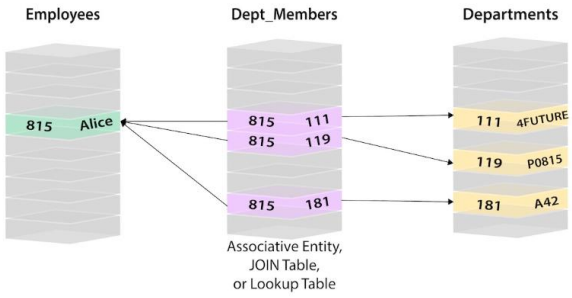
- Neo4J is LP Graph database
- Its graph data model consists of four fundamental building blocks:
  - **Nodes:** These are typically used to store entity information
  - **Relationships:** These are used to connect nodes to one another explicitly and therefore provide a means of structuring your entities
    - Relationships should be directional
  - **Properties:** Both nodes and relationships are containers for properties, which are effectively name/value pairs
  - **Labels:** This was a fundamental data model construct that was added to Neo4j with Version 2.0 at the end of 2013
    - Labels are a means to quickly and efficiently create subgraphs
    - By assigning labels to nodes, Neo4j makes the data model simpler, avoiding need to work with a type property on the nodes, or a need to connect nodes to definition nodes that provide meta-information about the graph

21

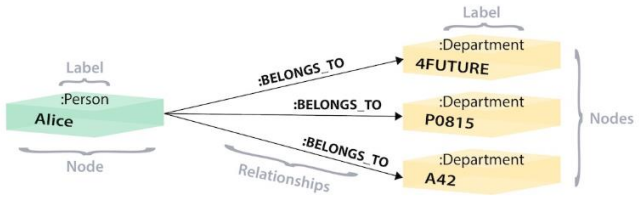
21

From Relational to Graph

- Relational model



- Graph model

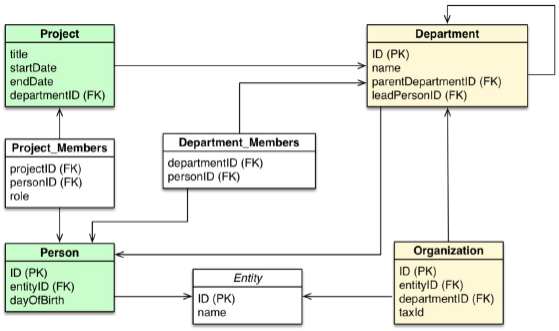


22

22

Example 2– Relational Model

- Let's use a relational database model of a domain with Persons and Projects within an Organization with several Departments:

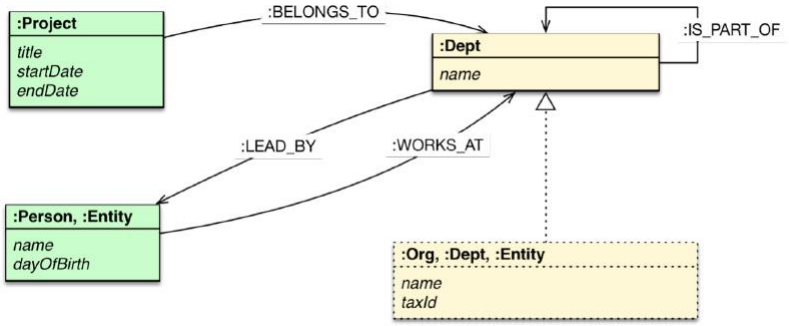


23

23

Example 2 – Graph Model

- A graph data model of the same domain with Persons and Projects within an Organization with several Departments will look like this:



- With the graph model, all of the initial JOIN tables have now become data relationships

24

24

Quiz

- **Question 1:** Neo4j is written in:
  A. Java Script
  B. Python
  C. Java
- **Question 2:** Neo4j is an ACID database
  A. True
  B. False
- **Question 3:** The four fundamental data constructs of Neo4j are:
  A. Table, record, field, and constraint
  B. Node, relationship, property, and schema
  C. Node, relationship, property, and label
  D. Document, relationship, property, and collection

25

25

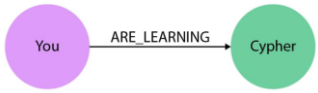
Part 3
Intro to Cypher

https://neo4j.com/style-guide/

26

26

Query Language



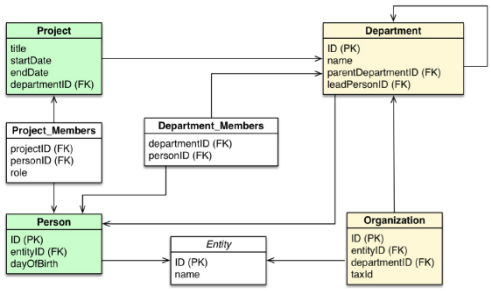
- Neo4J is using a Cypher - an open, multi-vendor query language for graph technologies
- Cypher
  - is a declarative query language
  - is built on the basic concepts and clauses of SQL, but with added graph-specific functionality, making it simple to work with a rich graph model without being overly verbose
  - was originally intended to be used with the graph database Neo4j, but was opened up through the openCypher project in October 2015
  - is now used by over 10 products and tens of thousands of developers

27

27

SQL vs. Cypher Query Example

- ERD:



- SQL Query: list names of the employees in the “IT Department”

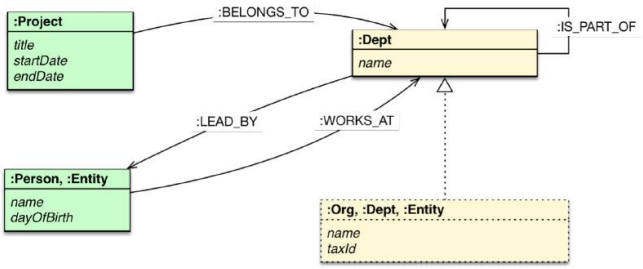
```
SELECT name FROM Entity
LEFT JOIN Person
ON Entity.ID=Person.entityID
LEFT JOIN Department_Members
ON Person.ID = Department_Members.PersonId
LEFT JOIN Department
ON Department.Id = Department_Members.DepartmentId
WHERE Department.name = "IT Department";
```

28

28

SQL vs. Cypher Query Example

- Graph data model:



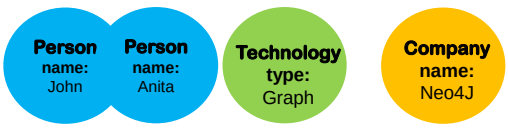
- Cypher query: list names of the employees in the "IT Department"

```
MATCH (p:Person)-[:WORKS_AT]->(d:Dept)
WHERE d.name = "IT Department"
RETURN p.name
```

29

29

Node



- To depict nodes in Cypher, you use parentheses, e.g. (node)
  - If the node is not relevant to your return results, you can specify an anonymous node using empty parentheses (). This means that you will not be able to return this node later in the query
- Node Variables
  - If later you want to refer to the node, you can give it a variable name, for example (p) for person. In real-world queries, it is advisable to use longer, more expressive variable names
- Node Labels
  - You can also group similar nodes together by assigning a node label, for example Person, Technology, Company, etc.

```
()                  //anonymous node (no label or variable)
                      can refer to any node in the database
(p:Person)          //using variable p and label Person
(:Technology)       //no variable, label Technology
(work:Company)      //using variable work and label Company
```

30

30

Create a Node

- Creating a Single node
  - You can create a node in Neo4j by simply specifying the name of the node that is to be created along with the CREATE clause
  - Syntax: `CREATE (node_name);` Example: `CREATE (n)`
- Create a node with a label: `CREATE (p:Person)`
- Create a node with multiple labels: `CREATE (p:Person:Swedish)`
- Create node and add labels and properties
  - properties are name-value pairs that provide additional details to your nodes and relationships:

```
CREATE (p:Person { name: 'Andy', title: 'Developer' })
```

- Return created node:

```
CREATE (u:USER)
RETURN u
```

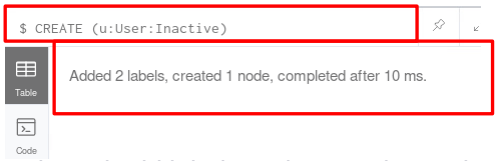
31

31

Create a Node in Neo4J Browser

- Once you install Neo4j, you can access Neo4j Browser using the following URL http://localhost:7474/browser/
- Create a node with a label:

- Create a node with multiple labels:



- Create node and add labels and properties and return created node

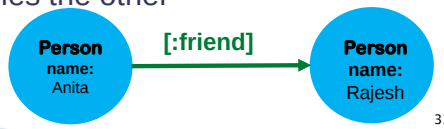
```
$ CREATE (u:User {name:"John",surname:"Doe"})
  RETURN u
```

32

32

Relationships

- Relationships are represented in Cypher using an arrow **-->** or **<--** between two nodes (directional!)
- Additional information, such as how nodes are connected (relationship type) and any properties pertaining to the relationship, can be placed in square brackets inside of the arrow
- Relationships in Neo4j can be traversed in both directions with the same speed
  - Also direction can be completely ignored
  - There is no need to create two different relationships between nodes, if one implies the other



33

33

Creating Relationship

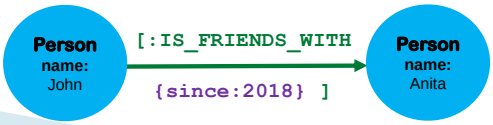
- To create relationships along with nodes, use the relationship pattern: **`-[:relationship_name]->`**

```
CREATE (p:Person)-[:LIKES]->(t:Technology)
```


```
CREATE (:User {name: "Jack", surname: "Smith"})-[:Sibling]->
(:User {name: "Mary", surname: "Smith"})
```

- Relationships also can have properties

```
-[rel:IS_FRIENDS_WITH {since: 2018}]->
```



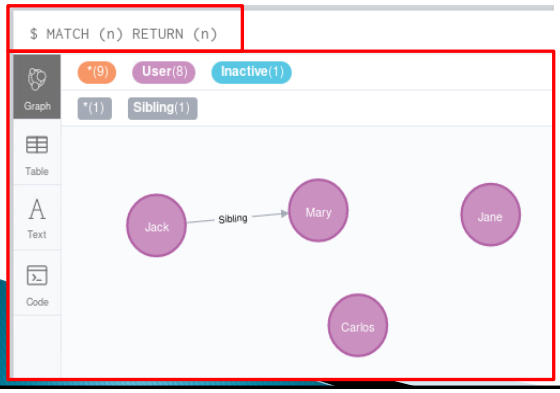
34

34

Query

- In order to search for an existing node, relationship, label, property, or pattern in the database use **MATCH** and **RETURN**
  - It works similar to SELECT in SQL
  - To see all nodes use:

```
MATCH (p:Person)
RETURN p
```



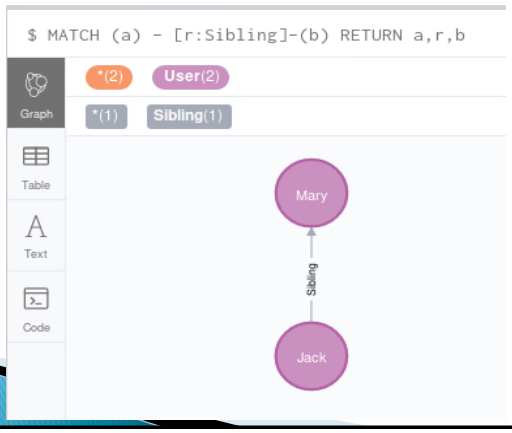
35

35

Return Nodes and Relationships

- You can return nodes along with relationship

```
MATCH (a) -[r:Sibling]-(b)
RETURN a,r,b
```



36

36

Filtering Queries

- To specify additional criteria for the search, use curly braces { }

```
MATCH (a:Person {name: 'Anita'} )
RETURN a
```

- You can search by relationship as well:

```
MATCH (:Person {name: 'Anita'})-[:WORKS_FOR]->(comp:Company)
RETURN comp
```

- You can also display only a particular property of the output node:

```
MATCH (:Person {name: 'Anita'})-[:WORKS_FOR]->(comp:Company)
RETURN comp.name
```

- Just like with SQL, you can rename return results by using the **AS** keyword and aliasing the property with a cleaner name.

```
MATCH (anna:Customer {name:'Anna'})-[rel:PURCHASED]-(order:Order)
RETURN order.orderId AS OrderID, order.orderDate AS 'Purchase Date',
anna.customerIdNo AS CustomerID, order.orderNumOfLineItems AS
'Number Of Items'
```

37

37

Quiz

- **Question 1: Which statement is correct?**
  - **A.** Relationships must be created along with the nodes
  - **B.** Nodes must be created first and then relationships between nodes
  - **C.** Nodes can be created without relationships
  - **D.** Relationships can be created without nodes

- **Question 2: In cypher which command is alternative to SQL SELECT command?**
  - **A.** SELECT
  - **B.** FIND
  - **C.** MATCH
  - **D.** QUERY

- **Question 3: What are the properties of the relationships and why they are useful?**

38

38

Essentials

- ✓ Introduced Graph theory
- ✓ Described two different Graph data models
- ✓ Introduced Neo4J
- ✓ Discussed Neo4J data model
- ✓ Introduced Cypher language