Contents

• What is a graph database?
• RDBMS vs graph databases
• Introduction to Neo4j
• Data Model
• Architecture
• Cypher Query
• Integration
What is a Graph Database?

Use of graph structure to store, map and query data/relations

Node: data item (a person, a business, an account)
Edges: connection or a relationship between two nodes
Example
Some of the most successful companies using graphs
How is Data Stored?

person  
person_friend  
friend
Why Graph Databases?

Real-world data exists as objects and relationship between objects
  • data is increasing in volume
  • and getting more connected

Graph databases are primarily designed to handle such kind and scale of data
Challenges in RDBMS

- Complex to model and store relationships
- Performance degrades with increase in number and depth of JOINs
- Cannot process high volumes at near real time.
- Expensive to scale
Why Graph Databases? continued..

Benefits:

- Schema-free
- No redundant data
- Low query latency
- Scalable
Neo4j

- Open Source
- Implemented in Java and Scala
- Cypher: mature and rivals SQL
Neo4j features

• Capacity:
  • Nodes – 35 billion
  • Relationships – 35 billion
  • Properties/Labels – 275 billion
• High data integrity
• Native graph processing
• Integration
• High scalability
• Data browser
Data Modeling
Property Graph Data Model

• Four building blocks
  • Nodes
  • Relationships
  • Properties
  • Labels
Nodes

- **Person**
  - name: John
  - age: 27

- **Person**
  - name: Sally
  - age: 32

- **Book**
  - title: Graph Databases
  - authors: Ian Robinson, Jim Webber

- **FRIEND OF**
  - since: 01/09/2013

- **HAS READ**
  - on: 02/09/2013
  - rating: 5

- **HAS READ**
  - on: 02/09/2013
  - rating: 4
Relationships

- **Person**
  - **name**: John
  - **age**: 27
  - **FRIEND OF**
    - **since**: 01/09/2013

- **Person**
  - **name**: Sally
  - **age**: 32
  - **FRIEND OF**
    - **since**: 01/09/2013

- **Book**
  - **title**: Graph Databases
  - **authors**: Ian Robinson, Jim Webber
  - **HAS READ**
    - **on**: 02/09/2013
    - **rating**: 4
Properties

:Person

name: John
age: 27

FRIEND OF
since: 01/09/2013

:Person

name: Sally
age: 32

FRIEND OF
since: 01/09/2013

:Book

title: Graph Databases
authors: Ian Robinson, Jim Webber

HAS READ
on: 02/09/2013
rating: 4

HAS READ
on: 02/03/2013
rating: 5
Think in Patterns

• Drop the WHERE clause
• Adopt the MATCH clause
Architecture
Native Graph Processing

- Index-free adjacency
- Each node maintains direct references to its adjacent nodes
- Efficient query time
Native Graph Storage

- Traversal API
- Core API
- Cypher
- Lock Manager
- Transaction Management
- Page Graph
- Transaction Log
- Record Files
- Disks
Native Graph Storage

- Traversal API
- Core API
- Cypher
- Lock Manager
- Page Graph
- Transaction Management
- Record Files
- Transaction Log
- Disks
Native Graph Storage

- Traversal API
- Core API
- Cypher
- Lock Manager
- Transaction Management
- Page Graph
- Transaction Log
- Record Files
- Transaction Log
- Disks
Cypher Query Language

MATCH (n)-->({})
RETURN n
What is Cypher?

• Neo4j’s open graph query language
• Uses patterns to describe graph data
• Familiar SQL-like clauses
• Describe what to find, not how to find it
Lets Learn to Query!

Let us create a node “you”

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

RETURN you
Let’s find ourselves and add a new relationship to a new node.

MATCH (you:Person {name:"You"})

CREATE (you)-[:LIKE]->(neo:Database {name:"Neo4j"})

RETURN you, like, neo
Let's Learn to Query!

Create Your Friends

MATCH (you:Person {name:"You"})

FOREACH (name in ["Johan","Rajesh","Anna","Julia","Andrew"] |

CREATE (you)-[:FRIEND]->(:Person {name:name})))
Let's Learn to Query!

Find Your Friends

MATCH (you {name: "You"}) [:FRIEND]->(yourFriends)
RETURN you, yourFriends
Let's Learn to Query!

Create Second Degree Friends and Expertise

MATCH (neo:Database {name:"Neo4j"})

MATCH (anna:Person {name:"Anna"})

CREATE (anna)-[:FRIEND]->(:Person:Expert {name:"Amanda"})-[:WORKED_WITH]->(neo)
Comparing RDBMS to Graph database
Comparing the Joins and Cypher Query

We want to see who bought *Chocolade*. Let’s join the four tables together in Relational Model

```
SELECT DISTINCT c.CompanyName 
FROM customers AS c 
JOIN orders AS o ON (c.CustomerID = o.CustomerID) 
JOIN order_details AS od ON (o.OrderID = od.OrderID) 
JOIN products AS p ON (od.ProductID = p.ProductID) 
WHERE p.ProductName = 'Chocolade';
```

The graph model is much simpler, as we don’t need join tables, and expressing connections as graph patterns, is easier to read too.

```
MATCH (p:Product {productName:"Chocolade"})<-[:PRODUCT]-(:Order)<-[:PURCHASED]-(c:Customer) 
RETURN distinct c.companyName;
```
Use Cases

From the name graph database it might come to our mind that it is suitable for social networking domain, but Neo4j has a strong presence in so many other areas

• Real time recommendation
• Master data management
• Fraud detection
• Graph based search
• IT operations and network management
Integration
Neo4j Driver API

- Bolt protocol
- Currently supports .NET, Java, JavaScript and Python
- Uniformity across languages

Acquire

C#

PM> Install-Package Neo4j.Driver -Version 1.0.2

Javascript

npm install neo4j-driver@1.0.4
How to use Neo4j driver API?

• Database object -> Driver
• Driver -> Session
• Run....
• Security
Example - Java

```java
import org.neo4j.driver.v1.*;

Driver driver = GraphDatabase.driver("bolt://localhost", AuthTokens.basic("neo4j", "neo4j") );
Session session = driver.session();

session.run("CREATE (a:Person {name:'Arthur', title:'King'})");

StatementResult result = session.run("MATCH (a:Person) WHERE a.name = 'Arthur' RETURN a.name AS name, a.title AS title" );
while ( result.hasNext() )
{
    Record record = result.next();
    System.out.println( record.get("title").asString() + " " + record.get("name").asString() );
}

session.close();
driver.close();
```
HTTP-API

• POST one or more cypher statements
• Keep transactions open
• Result formats

POST /db/data/transaction/commit {
  "statements": [
    {"statement": "CREATE (p:Person {firstName:{name}}) RETURN p",
     "parameters": {"name": "Daniel"}}
  ]}
REST-API

- Discoverability
- Get Service root

POST [http://localhost:7474/db/data/cypher](http://localhost:7474/db/data/cypher)

Accept: application/json; charset=UTF-8
Content-Type: application/json

```json
{
  "query": "MATCH (x {name: {startName}})-[r]-(friend) WHERE friend.name = {name} RETURN TYPE(r)",
  "params": {
    "startName": "I",
    "name": "you"
  }
}
```
Neo4j Browser

- Developer focused
- Export results
- Visualization
Drawbacks

• Scalability
• Complex Domains
• Complex types
• Deleted Records
What Companies Say?

- Ebay
- Walmart
- Telenor
- Glassdoor
- SNAP
- Most popular graph database
- Growth -250%
References

- https://neo4j.com/
- http://www.slideshare.net/thobe/an-overview-of-neo4j-internals
- “Graph Databases”, Second Edition, O’Reilly Media
Questions?
Thank You :)