Neo4j

Yaniv Malka
Computer Science – Technion
Seminar in Databases – Prof. Oded Shmueli
The Path Forward

• NoSQL Databases
• Graph Databases
• Neo4j
  • About
  • Why
  • Some new things
• Summary
• Questions
NOSQL

• Not Only SQL

• NOSQL – (wikipedia)

• A **NoSQL** (often interpreted as **Not Only SQL**) database provides a mechanism for **storage** and **retrieval** of data that is modeled in means other than the tabular relations used in **relational databases**. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. The data structure (e.g. key-value, graph, or document) differs from the **RDBMS**, and therefore some operations are faster in NoSQL and some in RDBMS.
NOSQL Trends

• **Increasing data size (big data)**
  “Every 2 days we create as much information as we did up to 2003” – Eric Schmidt

• **Increasingly connected data (graph data)** – for example, connecting text documents to HTML.

• **Semi-structured data** – using data structures which model a complex relationships between data units.

• **Architecture** – a facade over multiple services. Going to modular, distributed applications.
NOSQL Categories

Graph DB

Key-Value Table

<table>
<thead>
<tr>
<th>RESOURCE_NO</th>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>521</td>
<td>location</td>
<td>Paris</td>
</tr>
<tr>
<td>601</td>
<td>location</td>
<td>New York</td>
</tr>
<tr>
<td>604</td>
<td>location</td>
<td>SF</td>
</tr>
<tr>
<td>521</td>
<td>order</td>
<td>3</td>
</tr>
<tr>
<td>601</td>
<td>order</td>
<td>1</td>
</tr>
<tr>
<td>604</td>
<td>order</td>
<td>2</td>
</tr>
<tr>
<td>521</td>
<td>webcategory</td>
<td>dog</td>
</tr>
<tr>
<td>601</td>
<td>webcategory</td>
<td>dog</td>
</tr>
<tr>
<td>604</td>
<td>webcategory</td>
<td>cat</td>
</tr>
</tbody>
</table>

Big Table Database

Document DB
Document DB

• Pairs of key and complex data called “Document”. A document can contain complex pairs of key-value, key-array, and even key-document.

• For example: MongoDB
Key-Value DB

• Simple and fast database which support key-value pairs only. Part of the key-value stores allows to define a type to the values- such as string, integer.

• For example: Redis, Riak, Voldemort.
Big Table DB

- Wide column store
- A column based database (instead of rows). Mainly used for big data arrays.

- For example: Cassandra, Hbase.
Graph Database

- Stores information related to relationships and social networks.
- The records in a graph database are called Nodes.
- Every node could have Properties.
- Nodes are connected through typed, directed Relationships.
- Every relationship represented by an edge, which can include more properties.

- For example – Neo4j
Graphs
Graph
Graphs Everywhere

- Biology, chemistry, physics, sociology
  - Human body, interactions, reactions, social changes.
- Internet
  - Hardware- Software interactions
- Social networks
  - Friends, family, neighbors, community.
Neo4j

“We found Neo4j to be literally thousands of times faster than our prior MySQL solution, with queries that require 10-100 times less code. Today, Neo4j provides eBay with functionality that was previously impossible.”

Volker Pacher, Senior Developer, eBay
Neo4j

• Neo4j stores data in a graph, with records called **Nodes**.
• Let’s start with the simplest graph: a single node with some named values called **Properties**.
• Properties: a pair of **name:value**.

• Nodes can be grouped together by adding a **label** to each member.
• For example: add a label to each node in our graph to represent a person.

Name: “Yaniv” From:”Israel”
More Nodes

- Like any database, storing data in neo4j can be as simple as adding more records.
Consider Relationships

- The real power of neo4j is **connected-data**. You can associate 2 nodes with a relationship which describes how the records are related.

```
“Yaniv” ----> “Yoav”

“David” ----> “Dan” ----> “Lior”
```

- Yaniv
- Yoav
- David
- Dan
- Lior
Relationships properties

• Properties stores information shared between 2 nodes.

“Yaniv” KNOWS Since: 2003 “Yoav”

“Yoav” KNOWS From: school “Lior”
Neo4j

- Sponsored by Neo Technology, 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. Neo4j is used today by hundreds of thousands of companies and organizations in almost all industries. Use cases include matchmaking, network management, software analytics, scientific research, routing, organizational and project management, recommendations, social networks, and more.
Features

• Materializing of relationships at creation time, resulting in no penalties for complex runtime queries
• **Constant** time traversals for relationships in the graph both in **depth** and in **breadth** due to efficient representation of nodes and relationships
• All relationships in Neo4j are **equally important** and fast, making it possible to materialize and use new relationships later on to “**shortcut**” and speed up the domain data when new needs arise
• 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
• Reliable and real ACID transactions
• Cluster support
Neo4j is a NoSQL Graph Database

- JVM based
- Embedded and Server
- ACID
- Clustered
- Billions of entities
- Open source
- Welcoming UI
- Easy data modeling
- Readable queries
- Active community
- High performance
- Optional schema
Ways to work with Neo4j

• Embeddable on JVM
  • Java
  • jRuby
  • Scala
  • Environment: Tomcat, Akka, Rails
Nodes and Relationships

- In neo4j queries we use () to represent a node

- \((a)\rightarrow(b)\)  some a’ related to some b’
- \((a)\rightarrow()\)  a related to some nodes that we doesn’t care about

- \((a)\rightarrow[r]\rightarrow(b)\)  you can represent the relationship with an identifier.
- \((a)\rightarrow[:KNOWS]\rightarrow(b)\)  you can use the type of the relationship
- \((a:Person)\rightarrow[:EATS]\rightarrow(b:Vegan)\)  you can add labels to nodes
- \((a \{from:"California"\})\rightarrow(b)\)  you can add properties
MATCH (n)
RETURN n;
Friend of friend

MATCH (a:Person)-->(:Person)-->(:Person)
RETURN a
Show me some code...

```java
firstNode = graphDb.createNode();
firstNode.setProperty( "message", "Hello, " );
secondNode = graphDb.createNode();
secondNode.setProperty( "message", "World!" );

relationship = firstNode.createRelationshipTo( secondNode, RelTypes.KNOWS );
relationship.setProperty( "message", "brave Neo4j " );
```
Cypher

CREATE (ee:Person { name: "Emil", from: "Sweden" })

MATCH (ee:Person) WHERE ee.name = "Emil" RETURN ee;

MATCH (ee:Person) WHERE ee.name = "Emil"
Cypher and query language

- Cypher is Neo4j’s graph query language (SQL for graphs!)
- **Declarative language** – describes what you want, not how you want.
- Based on **pattern matching**.
- Cypher is meant to be very **readable** and **expressive**

**MATCH** [Nodes and relationships]
**WHERE** [Boolean filter statement]
**RETURN** [DISTINCT] [statements [AS alias]]
**ORDER BY** [Properties] [ASC\DESC]
**SKIP** [Number] **LIMIT** [Number]
Matching nodes and relationships

Nodes:
(a), (), (:Nodetype), (a:Nodetype), (a { Property}) ,
(a:NodeType{ property})

Relationships:
(a)--(b), (a)-->(b), (a)<--(b), (a)-->((), (a)-[r]->(b),
(a)-[:relType]->(b), (a)-[:R1 | :R2]->(b),
(a)-[r:relType]->(b)

When you have 2 nodes or more:
(a)-->(b)<--(c), (a)-->(b)-->(c)
More options

Relationships
(a)-[:Rtype* ] – (b)    - any times of hops
(a)-[:Rtype*2]-(b)     - exactly 2 hops
(a)-[:Rtype*..3]-(b)     - up to 3 hops

Path’s Functions
MATCH shortestPath( (a)-[*]-(b) )
MATCH allShortestPath( (a)-[*]-(b) )
Length(path)
RETURN relationships(p)
Neo4j VS Relational DB

David LIKES Yaniv

Dan LIKES Ron

David LIKES Ron

Yaniv LIKES Dan

Yaniv LIKES Ron

Dan LIKES Yaniv

Dan LIKES Ron

Yaniv LIKES David

Ron LIKES Yaniv

Ron LIKES David

Ron LIKES Dan
Neo4j VS Relational DB

• In Relational DB the data stored using **indexing** to traverse between nodes

<table>
<thead>
<tr>
<th>Node A</th>
<th>Node B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yaniv</td>
<td>David</td>
</tr>
<tr>
<td>Yaniv</td>
<td>Dan</td>
</tr>
<tr>
<td>Yaniv</td>
<td>Ron</td>
</tr>
<tr>
<td>David</td>
<td>Yaniv</td>
</tr>
<tr>
<td>David</td>
<td>Dan</td>
</tr>
<tr>
<td>David</td>
<td>Ron</td>
</tr>
<tr>
<td>Dan</td>
<td>Yaniv</td>
</tr>
<tr>
<td>Dan</td>
<td>David</td>
</tr>
<tr>
<td>Dan</td>
<td>Ron</td>
</tr>
<tr>
<td>Ron</td>
<td>Yaniv</td>
</tr>
<tr>
<td>Ron</td>
<td>David</td>
</tr>
<tr>
<td>Ron</td>
<td>Dan</td>
</tr>
</tbody>
</table>
Neo4j VS Relational DB

- Index-free adjacency
Stored Data

- Nodestore.db
- Relationshipstore.db
- RelType.db
- Propertystore.db
Stored Data

**Node (15 bytes)**
- inUse
- nextRelId
- nextPropId
- labels
- extra

1  5  9  14

**Relationship (34 bytes)**
- inUse
- firstPrevRelId
- secondPrevRelId
- nextPropId
- firstNode
- secondNode
- relationshipType
- firstNextRelId
- secondNextRelId
- firstInChainMarker

1  5  9  13  17  21  25  29  33  34
Name: David
Age: 32

Name: Dan
Age: 32

Name: Ron
Studies: CS
From: Israel

Name: Yaniv
Transactions

In order to fully maintain data integrity and ensure good transactional behavior, Neo4j supports the ACID properties:

- **atomicity**: If any part of a transaction fails, the database state is left unchanged.
- **consistency**: Any transaction will leave the database in a consistent state.
- **isolation**: During a transaction, modified data cannot be accessed by other operations.
- **durability**: The DBMS can always recover the results of a committed transaction.
Embedded Neo4j

- In **embedded** mode, Neo4j runs in the same process as our application.
- Embedded Neo4j is ideal for hardware devices, desktop applications, and for incorporating in our own application servers.
- Some of the advantages of embedded mode include: **Low latency** Because our application speaks directly to the database, there’s **no network overhead**.

**Choice of APIs** We have access to the full range of APIs for creating and querying data: the Core API, Traversal Framework, and the Cypher query language.
Why Neo4j

• **World’s best and first graph database** – Neo4j is used by thousands of organizations, including 50+ of the Global 2000.

• **Biggest and most active Graph community on the planet** – has the largest and most vibrant community of graph db:
  • 1 million+ downloads, adding 50k downloads per month
  • 20k+ graph education registrants
  • 500+ neo4j events per year
  • 100+ technology and service partners
  • 200 enterprise subscription customers, including 50+ of the global 2000
Why Neo4j

• Highly performance read and write scalability, without compromise.
• High performance thanks to native graph storage & processing
• Concurrency – supports multiple users at the same time.
• Easy to learn
  • Mature UI with intuitive interaction and built in learning
  • Time tested training ecosystem to meet your needs
  • A wealth of training materials bringing years of deployment experience to your desktop
  • Expert authored books for in-depth learning
Why Neo4j

- **Superb Value for Enterprise and Startup Projects**
  Most Neo4j customers find their **total cost of ownership decreases**, because they’ve optimized their production environment and increased efficiency. With Neo4j, you can choose the [license](#) and bundle that you need, and add clustering and data replication capabilities that make sense for your deployment and your organization.

- Staggering loading speed of huge data sizes, with very low memory footprint
- Choose how much and which data to import, without worrying about volume
- Very easy to model the reality on neo4j
Summary

• Neo4j is a great tool to represent database in graphs.

• Neo4j has been used by the biggest companies in the industry, and has a lot of benefits: high performance, wide community and support, easy learning.

• Graph is very comfortable way to model the reality. In graph database you can model routes, social networks, big organizations and hierarchy, etc.
Resources and References

- Neo4j online course [http://neo4j.com/graphacademy/online-course/](http://neo4j.com/graphacademy/online-course/)
- Neo4j website [http://neo4j.org](http://neo4j.org)
- Neo4j learning resources [http://neo4j.org/resources](http://neo4j.org/resources)
- Graph Databases 2nd Edition: [http://neo4j.com/books/graph-databases/?utm_source=GPPC&gclid=CjwKEAiAgvyxBRDmuviAj67g-XQSJABTLMcHWgtjYMACN7uoVktUi7SAqVp2v6T8JpTP1Ea6d3UiZBoCODnw_wcB](http://neo4j.com/books/graph-databases/?utm_source=GPPC&gclid=CjwKEAiAgvyxBRDmuviAj67g-XQSJABTLMcHWgtjYMACN7uoVktUi7SAqVp2v6T8JpTP1Ea6d3UiZBoCODnw_wcB)