CS-695 NoSQL Database
Neo4J (part 1 of 2)

Dr. Chuck Cartledge

5 Nov. 2015
Table of contents I

1. Miscellanea
2. DB comparisons
3. Assgn. #6
4. Historical origins
5. Data model
6. CRUDy stuff
7. Conclusion
8. References
Corrections and additions since last lecture.

- Assignment #06 is available
- Mid-term
**How different DBs compare to a RDBMS**

We have some terms to compare now[3]

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>K/V</th>
<th>Columnar</th>
<th>Doc.</th>
<th>Graphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB. instance</td>
<td>cluster</td>
<td>cluster</td>
<td>instance</td>
<td>—</td>
</tr>
<tr>
<td>database</td>
<td>—</td>
<td>namespace</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>table</td>
<td>bucket</td>
<td>table</td>
<td>collection</td>
<td>—</td>
</tr>
<tr>
<td>row</td>
<td>key-value</td>
<td>row</td>
<td>document</td>
<td>node id</td>
</tr>
<tr>
<td>rowid</td>
<td>key</td>
<td>—</td>
<td>_id</td>
<td>—</td>
</tr>
<tr>
<td>col.</td>
<td>—</td>
<td>col. fam.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>schema</td>
<td>—</td>
<td>—</td>
<td>database</td>
<td>—</td>
</tr>
<tr>
<td>join</td>
<td>—</td>
<td>—</td>
<td>DBRef</td>
<td>—</td>
</tr>
<tr>
<td>relationship</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>id</td>
</tr>
</tbody>
</table>
In general terms:

1. Design the database, particularly the relationships between nodes
2. Parse data
3. Populate the database
4. Query the database
5. Report the path (nodes and edges) from the beginning to the end

Sarah is wistful. She is distracted, walking around in a daze. Barely talking to her long suffering Hasta. She goes out a night, dragging her favorite lawn chair behind her. She sits and look up at the stars, and mutters to herself. Hasta stays inside, warm, dry, and sketches circles and arcs. Circles and arcs are his latest adventure into the world of NoSQL databases. Sometimes when it is late, Hast will leave his sketches on the table. In the morning, Sarah will be standing over them, talking to herself. Hasta knows the signs. He knows that she sees something that he doesn’t and that he will soon create something in her image.

Sarah dreams of the Milky Way. She looks to the heavens and sees fanciful creatures, gods and demons, relationships and connections where others see only chaos. Sarah sees connections above her, below her, through her life, and by extension Hasta’s as well. She is able to see patterns in the heavens and also in the movies that she loves so much. Sarah makes the connection between her studies of the heavens, Hasta’s circles and arcs, and movies.

Sarah sits and thinks. She stares at the heavens looking for inspiration. She wanders through the Internet Movie Database looking for guidance. Her inner light starts to glow. It radiates out and drives the darkness from the corners of her life. Hasta sees the light. Hasta is drawn towards the light. And, Sarah speaks.

“Hasta dearest, I need your Neo4j skills. I know that all things are connected. I need for you to show others the way. In the darkest reaches of the 20th century, there lived a great movie director Baltazar Abadal. He cast a long shadow over the cine-graphic landscape that reached to Mitchell A. Martin. Mr. Martin will become one of the greats, Hasta. I need to know the connections between Baltazar and Mitchell. I need to know how the greats are connected. Hasta use your Neo4j skills to draw me a graph of how these truly great men are connected.”

Hasta knows what he must do. He has resources:

- actors.list.gz — a list of actors listing which movies and TV productions they appeared in. The same general rules to identify a movie line from the movies file apply here. (There are roughly 4 million actors.)
- directors.list.gz — a collection of directors listing the production (movie, TV show, video) that each was responsible for. A director may have been responsible for many productions, and many directors may have been responsible for a production.

He has places where he can check his answer: https://oracleofbacon.org/

He knows that Dijkstra’s algorithm for finding the shortest path through a graph is available as part of a library for most modern languages.

Hasta girds his loins to do battle with the data.
Königsberg and Euler [1]

- Popular Sunday puzzle was to cross all bridges exactly once
- Mayor asks Leonard Euler to solve the puzzle
- Euler declines, thought the problem was trivial
- Euler changes mind.
- Euler publishes paper in 1735 detailing impossible solution, formulates general solution

Birth place of graph theory.

Image from [1].
Types of graphs

- **Simple graph**
- **Multigraph**
- **Pseudograph**

Image from [6].

- **Nodes** $\equiv$ **Vertices**
- **Arcs** $\equiv$ **Edges**
- **Arcs** $\equiv$ **Relationships**

Terms used interchangeably.
• Released in 2010
• Java based (relative OS independence)
• Various licensing levels
  • Community — free, but only on one node
  • Enterprise — semi free; allows clustering, hot backups, and monitoring
  • Government — Federal Information Security Management Act (FISMA) and other government certification and accreditation

It is young.
Conceptually only a few things

- nodes — things that exist
- relationships — how nodes are connected
- Directionality — relationships that are INCOMING or OUTGOING
- relationships can have value (attributes)
- nodes can exist without relationships
- relationships can not exist without nodes

Image from [2].
A different view

Image from [5].
It appears that the database already exists.

The cs695-nosql implementation appears to be a Community installation.

- There is one user: neo4j, password: neo4j
- There is a shell: neo4j-shell
- There is a ReSTful interface
- There is support in lots of different languages

Safest bet might be to have a local installation.
Web administration page is available

http://localhost:7474/browser/

Web Administration allows full access to the database.

- Create and delete nodes and relationships
- Create and edit Cypher commands

Good way to visualize your database and to test commands.
Making sure that neo4J is alive and well (1 of 2)

Checking its health:

- **Is it alive:**
  
  service neo4j-service status

- **A few details:**
  
  curl http://localhost:7474

- **Information about the current database (Cypher syntax[^1]):**
  

Making sure that neo4J is alive and well (2 of 2)

```
chuck@ubuntu:~$ service neo4j-service status
* neo4j is running
chuck@ubuntu:~$
chuck@ubuntu:~$
chuck@ubuntu:~$
curl http://localhost:7474
{
   "management": "http://localhost:7474/db/manage/",
   "data": "http://localhost:7474/db/data/"
}
chuck@ubuntu:~$
chuck@ubuntu:~$
chuck@ubuntu:~$
   "query": "MATCH (n) RETURN count(n) as nodes",
   "params": {
   }
}'}
chuck@ubuntu:~$
chuck@ubuntu:~$
```
Commands to restart/reset the system (last resort)

```
sudo service neo4j-service stop
sudo rm -rf /var/lib/neo4j/data
sudo mkdir -p /var/lib/neo4j/data/log
sudo chown -R neo4j /var/lib/neo4j/data
sudo chgrp -R adm /var/lib/neo4j/data
sudo service neo4j-service start
Then the “magic” to set password for the user neo4j.
```
Drivers and other things$^2$

Neo4J is available via commands, via ReSTful APIs, and drivers for

- Java, .NET, JavaScript,
- Python, Ruby, R
- And others.

The examples in this lecture use the Cypher language.

Too many choices. All use HTTP ReST communications backend.

$^2$http://neo4j.com/developer/language-guides/
Just some preliminaries (magic)

Changing the password (required the first time you access the database):

## CRUDy nuts and bolts

### Creating data (1 of 2)

Data in a graph database are nodes and their relationships. The general format is based on our friend JSON:

```bash
```
Creating data (2 of 2)

Data in a graph database are nodes and their relationships.

- A specific example creating a node:
  ```
  ```

- A specific example creating a relationship:
  ```
  ```
Finding data (1 of 2)

Tell neo4J to look for a node/relationship that matches a criterion/criteria:

```bash
curl -user neo4j:NEO4J -H "Accept: application/json; charset=UTF-8" -H "Content-Type: application/json" -X POST http://localhost:7474/db/data/cypher -d '{ "query" : "MATCH x{name : {startName}}-[r]->(n) RETURN type(r), n.name", "params" : { "startName" : "Foo" } }'
```
Finding data (2 of 2)

```bash
chuck@ubuntu:~$ curl --user neo4j:NEO4J -H "Accept: application/json; charset=UTF-8" -H "Content-Type: application/json" -X POST http://localhost:7474/db/data/cypher -d '{
    "query": "MATCH (x {name: {startName}})-[r]->(n) RETURN type(r), n.name",
    "params": {
        "startName": "foo"
    },
    "columns": [ "type(r)", "n.name" ],
    "data": [ [ "Comes Before", "bar" ] ]
}chuck@ubuntu:~$ 
```
CRUDy nuts and bolts

Updating data (1 of 2)

The mechanics are very similar to an SQL Update:

1. Find the node/relationship to update
2. Assign the new value

A specific example:

curl –user neo4j:NEO4J -H "Accept: application/json; charset=UTF-8" -H "Content-Type: application/json" -X POST http://localhost:7474/db/data/cypher -d '{ "query" : "MATCH (x {name: {startName}})-[r]->(n) set n.name = "Testing" RETURN type(r), n.name", "params" : { "startName" : "Foo" } }'
CRUDy nuts and bolts

```
  "query": "MATCH (x {name: {startName}})-[r]->(n) set n.name = "Testing\" REURN type(r), n.name",
  "params": {
    "startName": "foo"
  }
}'
chuck@ubuntu:~$  
```
Deleting data (1 of 2)

Deleting a node is a little more complex because relationships depend on nodes. If a node in a relationship is removed, then the relationship is “undefined.” So, the node has to be detached from all relationships prior to being deleted from the database.

A specific example:

Deleting data (2 of 2)

chuck@ubuntu:~$ curl --user neo4j:NE04J -H "Accept: application/json; charset=UTF-8" -H "Content-Type: application/json" -X POST http://localhost:7474/db/data/cypher -d '{
  "query": "MATCH (n {name: {startName}}) detach delete(n)",
  "params": {
    "startName": "bar"
  }
}


 Understand your source and destination data models

chuck@ubuntu:~$
What have we covered?

- Reviewed assignment #06
- Covered some of the Neo4J CRUDy stuff
- All Cypher commands are in the attached files

Next time: continued CRUDy and algorithmic exploration
References I


