Big Data Infrastructures

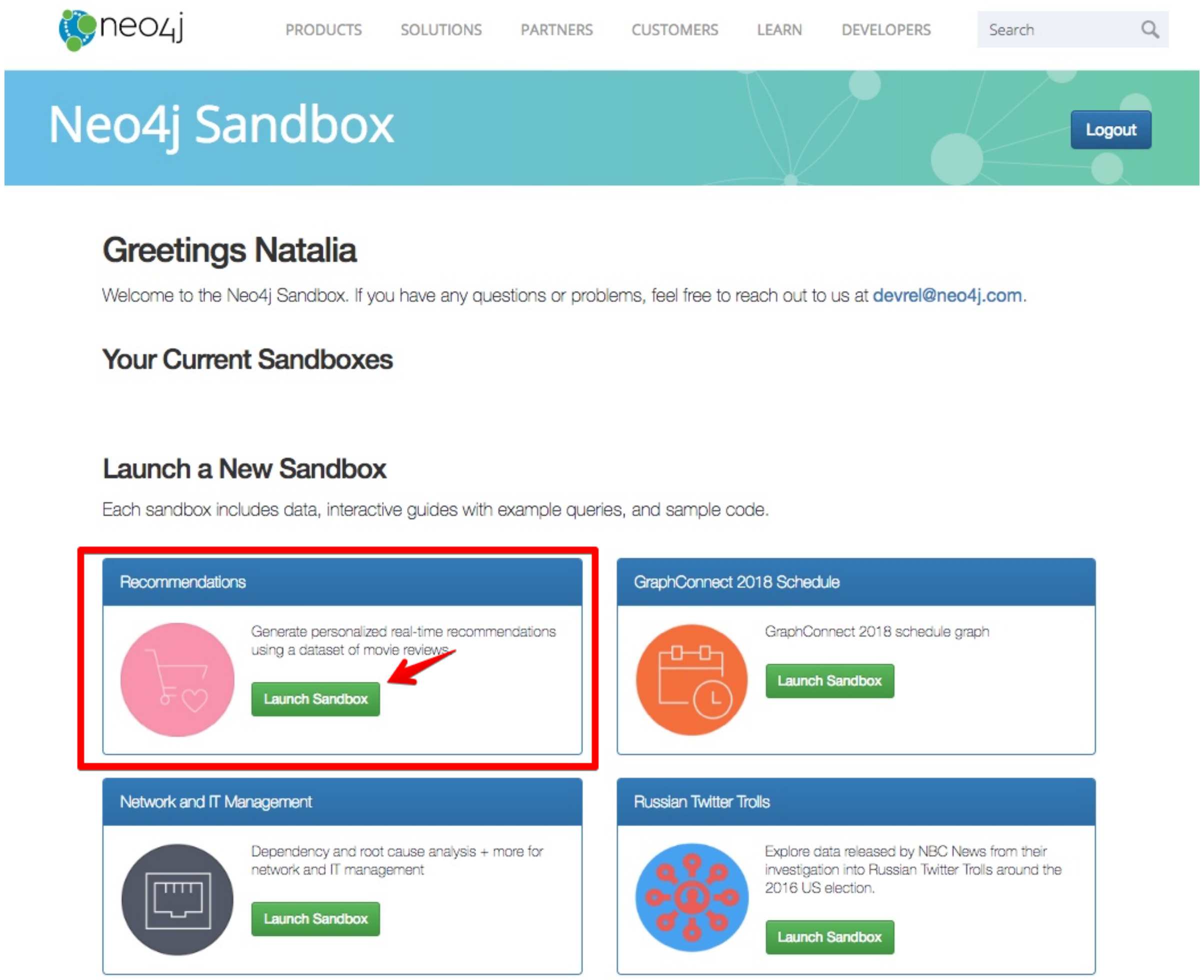
Natalia Ostapuk

Fall 2018

Lecture 6 – Graph Databases: Lab

Instructions

- Go to Neo4j Sandbox: https://neo4j.com/sandbox-v2/
- Press ”Start Now”
- Log in with the existing account (Google, GitHub, Twitter, LinkedIn) or create a new one
- Go to ”Recommendations” -> ”Launch Sandbox”
- Open Neo4j Browser



Greetings Natalia

Welcome to the Neo4j Sandbox. If you have any questions or problems, feel free to reach out to us at devrel@neo4j.com.

Your Current Sandboxes

Launch a New Sandbox

Each sandbox includes data, interactive guides with example queries, and sample code.

Recommendations

Generate personalized real-time recommendations using a dataset of movie reviews

GraphConnect 2018 Schedule

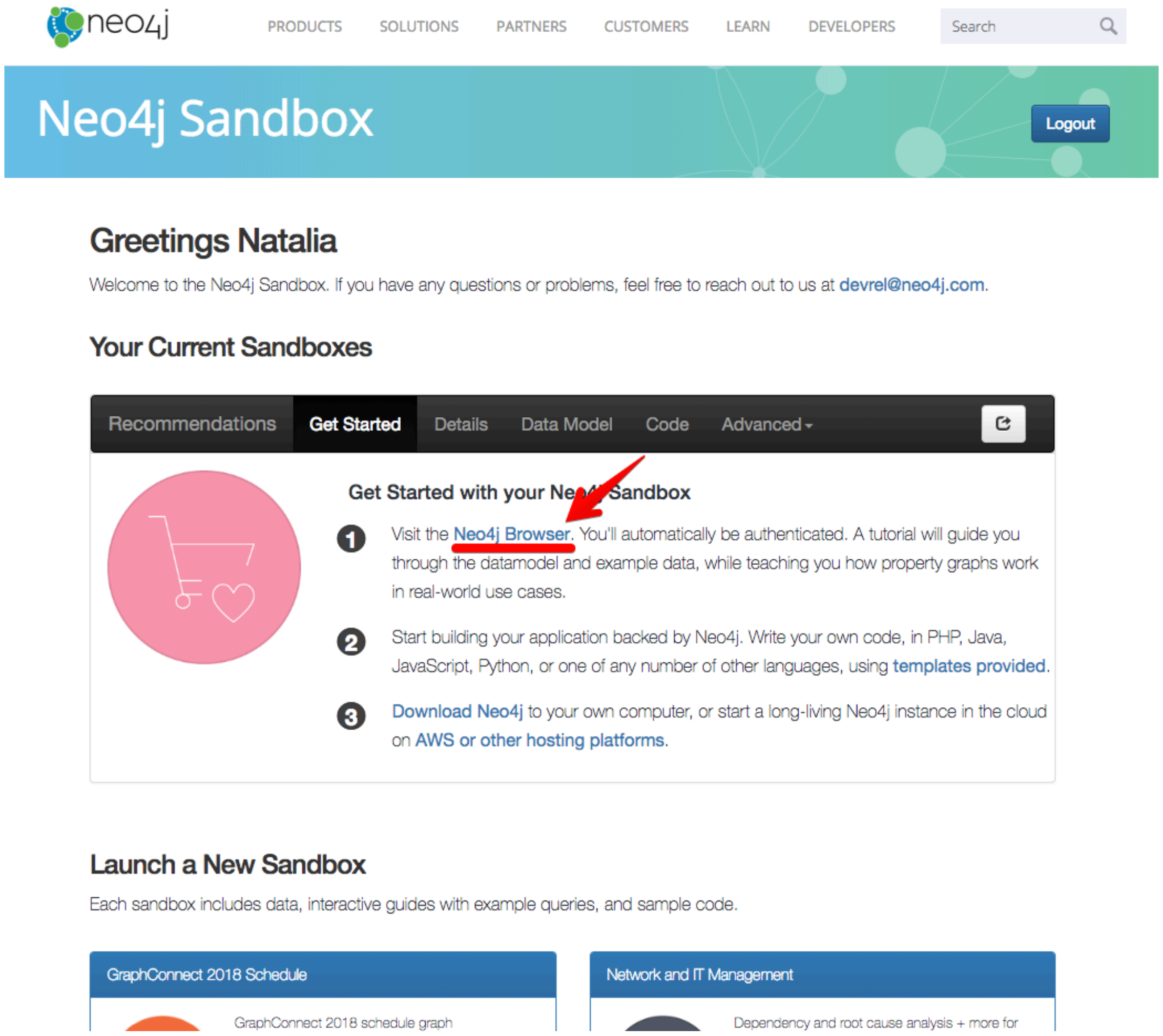
GraphConnect 2018 schedule graph

Network and IT Management

Dependency and root cause analysis + more for network and IT management

Russian Twitter Trolls

Explore data released by NBC News from their investigation into Russian Twitter Trolls around the 2016 US election.



Neo4j Sandbox

Greetings Natalia

Welcome to the Neo4j Sandbox. If you have any questions or problems, feel free to reach out to us at devrel@neo4j.com.

Your Current Sandboxes

Get Started with your Neo4j Sandbox

1. Visit the Neo4j Browser. You'll automatically be authenticated. A tutorial will guide you through the datamodel and example data, while teaching you how property graphs work in real-world use cases.
2. Start building your application backed by Neo4j. Write your own code, in PHP, Java, JavaScript, Python, or one of any number of other languages, using templates provided.
3. Download Neo4j to your own computer, or start a long-living Neo4j instance in the cloud on AWS or other hosting platforms.

Launch a New Sandbox

Each sandbox includes data, interactive guides with example queries, and sample code.

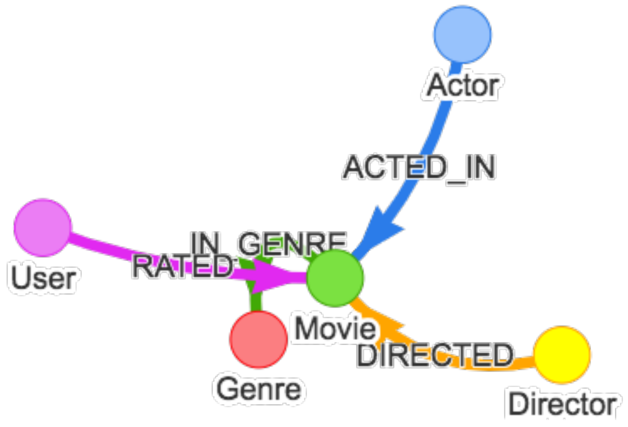
GraphConnect 2018 Schedule

GraphConnect 2018 schedule graph

Network and IT Management

Dependency and root cause analysis + more for

Data Model: Nodes and Relations



- `(:User)-[:RATED]->(:Movie)`
- `(:Actor)-[:ACTED_IN]->(:Movie)`
- `(:Director)-[:DIRECTED]->(:Movie)`
- `(:Movie)-[:IN_GENRE]->(:Genre)`

Data Model: Properties

- Nodes *Genre, Actor, Director, User*:
  - `name <string>`
- Node *Movie*:
  - `title: <String>`
  - `year: <Integer>`
  - `runtime: <Duration>`
  - `countries: <Array of strings>`
  - `languages: <Array of strings>`
  - `released: <String>`
  - `plot: <String>`
- Relationship *RATED*:
  - `rating: <Float>`
  - `timestamp: <Integer>`

Exercise 0

- Match one movie.
- Hint: limit output in RETURN clause with LIMIT keyword (RETURN … LIMIT 1)

Exercise 0: Discover Movie Structure

- Match one movie.

```
MATCH (movie:Movie)
RETURN movie LIMIT 1;
```

Exercise 0: Discover Movie Structure

- Match one movie.

```
MATCH (movie:Movie)
RETURN movie LIMIT 1;
```

movie
{ "languages": ["English"], "year": 1995, "imdbId": "0114709", "runtime": 81, "imdbRating": 8.3, "movieId": "1", "countries": ["USA"], "imdbVotes": 591836, "title": "Toy Story", "tmdbId": "862", "plot": "A cowboy doll is profoundly threatened and jealous when a new spaceman figure supplants him as top toy in a boy's room.", "poster": "http://ia.media-imdb.com/images/M/MV5BMTgwMjI4MzU5N15BMl5BanBnXkFtZTcwMTMyNTk3OA@@._V1_SX300.jpg", "released": "1995-11-22" }

Exercise 1: Simple Match

- List all genres.

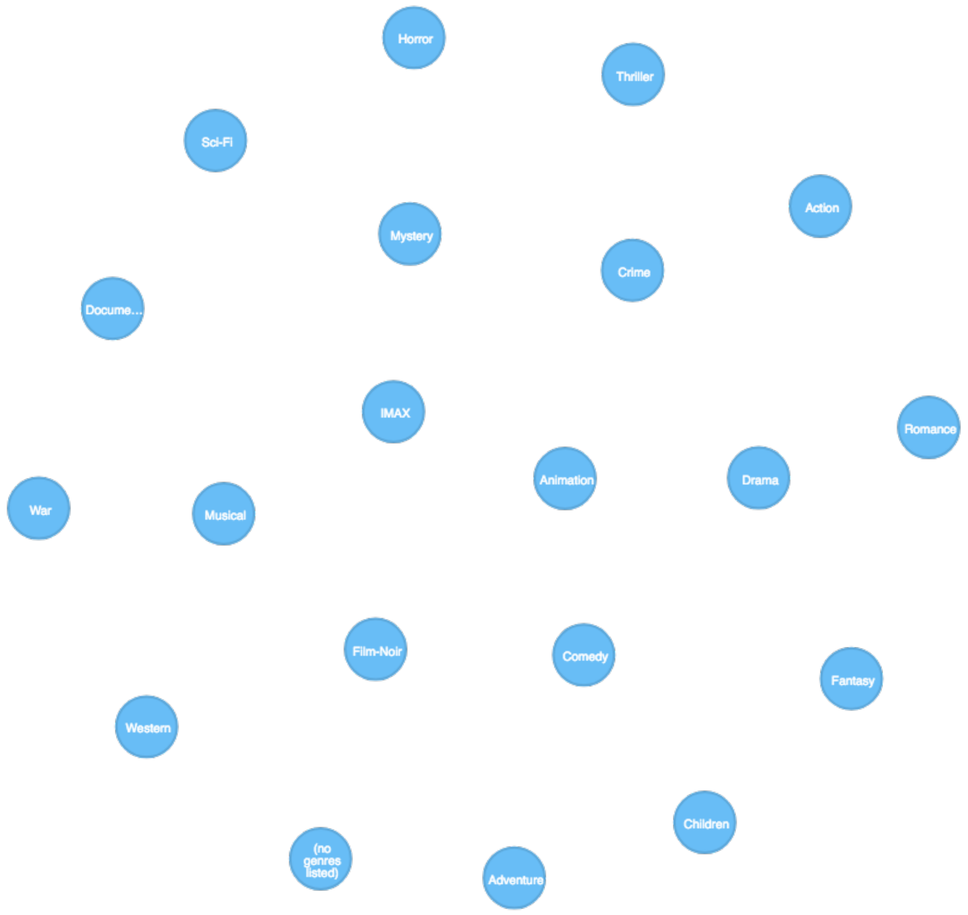
Exercise 1: Simple Match

- List all genres.

```
MATCH (genre:Genre)
RETURN genre;
```

Exercise 1: Simple Match

Graph view:



Exercise 1: Simple Match

Table view:

genre
{ "name": "Adventure" }
{ "name": "Animation" }
{ "name": "Children" }
{ "name": "Comedy" }
{ "name": "Fantasy" }
{ "name": "Romance" }
{ "name": "Drama" }
{ "name": "Action" }
{ "name": "Crime" }
{ "name": "Thriller" }
{ "name": "Horror" }
{ "name": "Mystery" }
{ "name": "Sci-Fi" }
{ "name": "Documentary" }
{ "name": "IMAX" }
{ "name": "War" }
{ "name": "Musical" }
{ "name": "Western" }
{ "name": "Film-Noir" }
{ "name": "(no genres listed)" }

Exercise 1: Simple Match

- List all genres.

```
MATCH (genre:Genre)
RETURN genre.name AS genre;
```

Exercise 1: Simple Match

genre
"Adventure"
"Animation"
"Children"
"Comedy"
"Fantasy"
"Romance"
"Drama"
"Action"
"Crime"
"Thriller"
"Horror"
"Mystery"
"Sci-Fi"
"Documentary"
"IMAX"
"War"
"Musical"
"Western"
"Film-Noir"
"(no genres listed)"

Exercise 2: Match Path

- List 5 movies in genre Action.

- Hint: we can specify a property value either in the node itself (key-value pair in curly brackets) or in WHERE clause

Exercise 2: Match Path

- List 5 movies in genre Action.

```
MATCH (movie:Movie)-[:IN_GENRE]->(genre:Genre
{name:"Action"})
RETURN movie.title LIMIT 5;

MATCH (movie:Movie)-[:IN_GENRE]-
>(genre:Genre)
WHERE genre.name = "Action"
RETURN movie.title LIMIT 5;
```

Exercise 2: Match Path

movie.title
"Dracula Untold"
"Stretch"
"Predestination"
"American Sniper"
"Big Hero 6"

Exercise 3: Order Results

- List 3 movies in genre Comedy with the highest rating.

- Hints:
  - You need property *imdbRating*
  - Results can be ordered in RETURN clause same way as in SQL (ORDER BY … DESC)

Exercise 3: Order Results

- List 3 movies in genre Comedy with the highest rating.

```
MATCH (m:Movie)-[:IN_GENRE]->(genre:Genre)
WHERE genre.name = "Comedy"
RETURN m.title AS movie, m.imdbRating AS
rating
     ORDER BY m.imdbRating DESC
     LIMIT 3;
```

Exercise 3: Order Results

movie	rating
"Ice Age: The Great Egg-Scapade"	null
"Neighbors 2: Sorority Rising"	null
"Keanu"	null

Exercise 3: Order Results

- List 3 movies in genre Comedy with the highest rating (property *imdbRating*).

- We are not interested in movies without rating:

```
MATCH (m:Movie)-[:IN_GENRE]->(genre:Genre)
WHERE genre.name = "Comedy"
   AND m.imdbRating IS NOT NULL
RETURN m.title AS movie, m.imdbRating AS
rating
   ORDER BY rating DESC
   LIMIT 3;
```

Exercise 3: Order Results

movie	rating
"Bill Hicks: Revelations"	8.9
"Pulp Fiction"	8.9
"George Carlin: Jammin' in New York"	8.9

Exercise 4: WITH Clause

- How many reviews does each Lord of the Rings movie have? Order output by the number of reviews.
- Hint: here we need first to apply aggregation function (COUNT), and then order results by aggregated values. We can do this in WITH clause.

Exercise 4: WITH Clause

- How many reviews does each Lord of the Rings movie have?

```
MATCH (m:Movie)<-[:RATED]-(u:User)
WHERE m.title CONTAINS "Lord of the Rings"
WITH m.title AS movie, COUNT(*) AS reviews
RETURN movie, reviews
     ORDER BY reviews DESC
     LIMIT 5;
```

Exercise 4: WITH Clause

movie	reviews
"Lord of the Rings: The Fellowship of the Ring, The"	200
"Lord of the Rings: The Two Towers, The"	188
"Lord of the Rings: The Return of the King, The"	176
"Lord of the Rings, The"	19

Exercise 5: Leverage Graph Structure

- How many users rated the movie “Godfather, The” not lower than 4.0?
- Hint: we can filter on relationship properties, too.

Exercise 5: Leverage Graph Structure

- How many users rated the movie “Godfather, The” not lower than 4.0?

```
MATCH (m:Movie)<-[r:RATED]-(u:User)
WHERE m.title = "Godfather, The"
     AND r.rating >= 4.0
RETURN COUNT(u);
```

Result:

COUNT(u)
178

Exercise 6: Add Node

- Create a user node for yourself.

Exercise 6: Add Node

- Create a user node for yourself.

```
CREATE (:User {name: "Natalia"});
```

Exercise 7: Set Properties

- Set additional properties for your user: age, sex, native language… Go creative!
- Hint: to set a property value, you need first to match the node.

Exercise 7: Set Properties

- Set additional properties for your user: age, sex, native language… Go creative!

```
MATCH (u:User {name: "Natalia"})
SET u.age = 28,
     u.sex = "female",
     u.native_language = "Russian";
```

Exercise 8: Create Relationship

- Rate your favorite movie.

- Hints:
  - Check whether your favorite movie is in the dataset.
  - If not – you can create one! (or change your preferences)
  - To create a relationship, you need first to match both nodes.

Exercise 8: Create Relationship

- Rate your favorite movie.

```
MATCH (m:Movie), (u:User)
WHERE m.title = "American Beauty"
     AND u.name = "Natalia"
CREATE (u)-[:RATED {rating: 4.8}]->(m);
```

Exercise 9: Second-Order Relationships

- Which users rated your favorite movie? Output user name and rating, sort in descending order.

Exercise 9: Second-order Relationships

- Which users rated your favorite movie? Output user name and rating, sort in descending order.

```
MATCH (u:User)-[r:RATED]->(m:Movie)<-
[:RATED]-(me:User {name: "Natalia"})
RETURN u.name, r.rating
     ORDER BY r.rating DESC;
```

Exercise 10: Third-order relationships

- Which movies rated users, who rated high (≥ 4.5) your favorite movie?

Exercise 10: Third-order relationships

- Which movies rated users, who rated high (≥ 4.5) your favorite movie?

```
MATCH (m:Movie)<-[rated:RATED]-(u:User)-
[rated_my:RATED]->(my_movie:Movie)<-
[:RATED]-(me:User)
WHERE me.name = "Natalia" AND
     rated_my.rating >= 4.5
RETURN m.title;
```