Course NDBI040: **Big Data Management and NoSQL Databases**

Practice 05:

Graph Databases: Neo4j

Martin Svoboda

5. 1. 2016

Faculty of Mathematics and Physics, Charles University in Prague

Outline

- Graph databases
  - **Neo4j** tutorial
- Assignment 5

Neo4j

Graph Database

Installation

- **Download**
  - http://neo4j.com/download/other-releases/
  - Neo4j 2.3.1
    - Community edition
    - Windows 32 bit
    - ZIP distribution

Embedded Neo4j

- **NetBeans project**
  - Create a new project
    - Java application
  - Add all Neo4j libraries
    - …\neo4j-community-2.3.1\lib***.jar**

Connection

- Create an embedded database
  - `org.neo4j.graphdb.factory.GraphDatabaseFactory`
  - `org.neo4j.graphdb.GraphDatabaseService`
  - ```
    GraphDatabaseService db =
      new GraphDatabaseFactory().
      newEmbeddedDatabase("db-dir-name");
    ```
- Close connection
  - `db.shutdown();`
    ```

Transactions

- Prepare and use transactions
  - `org.neo4j.graphdb.Transaction`
  - `Transaction tx = db.beginTx();`
  - `try {`

```
    …
    tx.success();
  } catch (Exception e) {
    …
    tx.failure();
  } finally {
    tx.close();
  }
```

Nodes

- Create new nodes and add properties
  - `org.neo4j.graphdb.Node`
  - ```
    Node n1 = db.createNode();
    n1.setProperty("name", "Irena");
    ```
  - ```
    Node n2 = db.createNode();
    n2.setProperty("name", "Martin");
    ```
- Access existing nodes
  - `db.getNodeById(1);`
  - `n1.getProperty("name");`

Relationships

- Define relationship types
  - `org.neo4j.graphdb.RelationshipType`
  - ```
    enum RelTypes implements RelationshipType {
      KNOWS, …
    }
    ```
- Create a new relationship
  - `org.neo4j.graphdb.Relationship`
  - ```
    Relationship r1 =
      n1.createRelationshipTo(n2, RelTypes.KNOWS);
    r1.setProperty("from", 2008);
    ```
- Access relationships
  - `db.getRelationshipById(1);`

Traversals

- Traversal framework
  - **Description**
    - **Expanders** – relationship types and directions to follow
    - **Order** – depth-first / breadth-first traversal algorithm
    - **Uniqueness** – visit nodes / relationships / paths only once
    - **Evaluator** – traversal continuation and result inclusion
  - **Starting nodes**
  - **Traverser**

Traversals

- Find all friends of Irena
  - `org.neo4j.graphdb.traversal.TraversalDescription`
  - …
  - `TraversalDescription td = db.traversalDescription();`
  - `td.breadthFirst();`
  - `td.evaluator(…);`
  - `td.relationships(…);`
  - `td.uniqueness(…);`
  - `Traverser t = td.traverse(n1);`
  - `for (Path p : t) { ... }`

Cypher

- Cypher query clauses
  - **START** – starting points
  - **MATCH** – graph pattern to match
  - **WHERE** – filtering criteria
  - **RETURN** – result construction
  - **CREATE** – insert new data
  - **DELETE** – remove existing data
  - **SET** – update existing data

Queries

- Find all persons and their names
  - `org.neo4j.graphdb.Result`
  - ```
    Result result = db.execute(
      "MATCH (n) RETURN n, n.name"
    );
    ```
  - ```
    while (result.hasNext()) {
      Map<String, Object> row = result.next();
      for (Entry<String, Object> column :
           row.entrySet()) {
        …
      }
    }
    ```

References

- Neo4j
  - http://neo4j.com/
- **Cypher Query Language**
  - http://neo4j.com/docs/stable/cypher-query-lang.html
- **Java and embedded Neo4j**
  - http://neo4j.com/docs/stable/tutorials-java-embedded.html

Assignment 5

Graph Databases: Neo4j

Assignment 5

- **Create a non-trivial real-world application** that will work with Neo4j embedded in Java
  - Choose any database domain
    - E.g. social network, geographical maps, …
  - Create a meaningful application
  - Comment your source code and queries
- **Submit your solution** by e-mail
  - svoboda@ksi.mff.cuni.cz
  - Deadline: 19. 1. 2016

Assignment 5

- **Requirements**
  - Create a database with at least 2 kinds of nodes and 2 types of relationships
  - Insert about 10 nodes and 10 relationships, both with a few meaningful properties
  - Define and process results of at least 2 traversals
    - Use various expanders, evaluators, …
  - Execute at least 5 Cypher queries
    - Use all discussed clauses