B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/b221/courses/b4m36ds2/

Practical Class 12

Neo4j

Yuliia Prokop
prokoyul@fel.cvut.cz

11. 12. 2022

Authors: Martin Svoboda
(martin.svoboda@matfyz.cuni.cz)
Yuliia Prokop
Czech Technical University in Prague, Faculty of Electrical Engineering

First Steps

Download and install Neo4j distribution

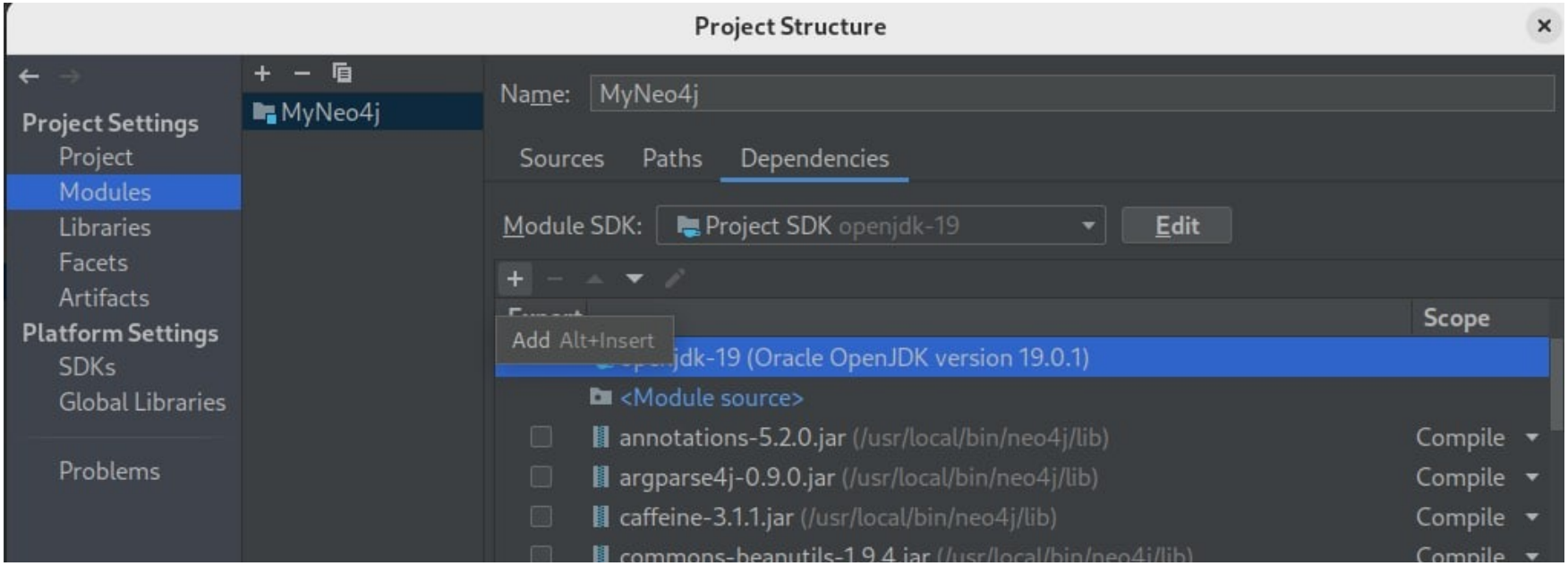
Neo4j Community Edition 5.2.0

https://neo4j.com/download-center/#community

First Steps

Create a new IntelliJ IDEA project

- Select *Java application* as a project type
- Add all the libraries from Neo4j `lib` directory
  - Use *Add JAR/Folder* in the project context menu



Database

```
import static
org.neo4j.configuration.GraphDatabaseSettings.DEFAULT_DATABASE_NAME;

import org.neo4j.dbms.api.DatabaseManagementService;
import org.neo4j.dbms.api.DatabaseManagementServiceBuilder;
import org.neo4j.graphdb.GraphDatabaseService;
import org.neo4j.io.fs.FileUtils;
```

Database

Create a new embedded database

```
import static
org.neo4j.configuration.GraphDatabaseSettings.DEFAULT_DATABASE_NAME;
import org.neo4j.dbms.api.DatabaseManagementService;
import org.neo4j.dbms.api.DatabaseManagementServiceBuilder;
import org.neo4j.graphdb.GraphDatabaseService;
import org.neo4j.io.fs.FileUtils;

public class Neo4jApp {
   private static final java.nio.file.Path dbDirectory =
                java.nio.file.Path.of("MyNeo4jDB");

   public static void main(String[] args) throws IOException{
   FileUtils.deleteDirectory(dbDirectory);
   DatabaseManagementService managementService = new
                DatabaseManagementServiceBuilder(dbDirectory).build();
   GraphDatabaseService database =
                managementService.database(DEFAULT_DATABASE_NAME);
...
   managementService.shutdown();
}
}
```

Start the database server

Shut down the database

Transactions

Start a new database transaction

All operations have to be performed in a transaction.

```
import org.neo4j.graphdb.Transaction;


try ( Transaction tx = database.beginTx() )
{
        //Database operations go here
        tx.commit();
}
```

Nodes

Create graph nodes for a few actors

- Create nodes, add ACTOR labels, add properties
  - trojan, Ivan Trojan, 1964
  - machacek, Jiří Macháček, 1966
  - schneiderova, Jitka Schneiderová, 1973
  - sverak, Zdeněk Svěrák, 1936
- Remember node references

```
import org.neo4j.graphdb.Node;
import org.neo4j.graphdb.Label;
```

```
Node actor1 = tx.createNode();
actor1.setProperty("id", "trojan");
actor1.setProperty("name", "Ivan Trojan");
actor1.setProperty("year", 1964);
actor1.addLabel(Label.label("ACTOR"));
```

Nodes

```
Node actor1, actor2, actor3, actor4;
try ( Transaction tx = database.beginTx() ) {
    actor1 = tx.createNode();
    actor1.setProperty("id", "trojan");
    actor1.setProperty("name", "Ivan Trojan");
    actor1.setProperty("year", 1964);
    actor1.addLabel(Label.label("ACTOR"));
    actor2 = tx.createNode();
    actor2.setProperty("id", "machacek");
    actor2.setProperty("name", "Jiří Macháček");
    actor2.setProperty("year", 1966);
    actor2.addLabel(Label.label("ACTOR"));
    actor3 = tx.createNode();
    actor3.setProperty("id", "schneiderova");
    actor3.setProperty("name", "Jitka Schneiderová");
    actor3.setProperty("year", 1973);
    actor3.addLabel(Label.label("ACTOR"));
    actor4 = tx.createNode();
    actor4.setProperty("id", "sverak");
    actor4.setProperty("name", "Zdeněk Svěrák");
    actor4.setProperty("year", 1936);
    actor4.addLabel(Label.label("ACTOR"));
    tx.commit();
}
```

Relationships

Define relationship types for our graph

```
import org.neo4j.graphdb.RelationshipType;
```

```
private enum RelTypes implements RelationshipType {
    KNOWS
}
```

Relationships

Create relationships between our actors

- Create relationships of KNOWS type
  - trojan → machacek
  - trojan → schneiderova
  - machacek → trojan
  - machacek → schneiderova
  - sverak → machacek
- Consider these relationships as symmetric

```
import org.neo4j.graphdb.Relationship;

Relationship relationship;
relationship = actor1.createRelationshipTo( actor2, RelTypes.KNOWS );
```

Relationships

```
actor1.createRelationshipTo( actor2, RelTypes.KNOWS );
actor1.createRelationshipTo( actor3, RelTypes.KNOWS );
actor2.createRelationshipTo( actor1, RelTypes.KNOWS );
actor2.createRelationshipTo( actor3, RelTypes.KNOWS );
actor4.createRelationshipTo( actor2, RelTypes.KNOWS );
```

Traversal Framework

Traversal framework

- Allows us to express and execute graph traversal queries
- Based on callbacks, executed lazily

Traversal description

- **Defines rules and other characteristics of a traversal**

Traverser

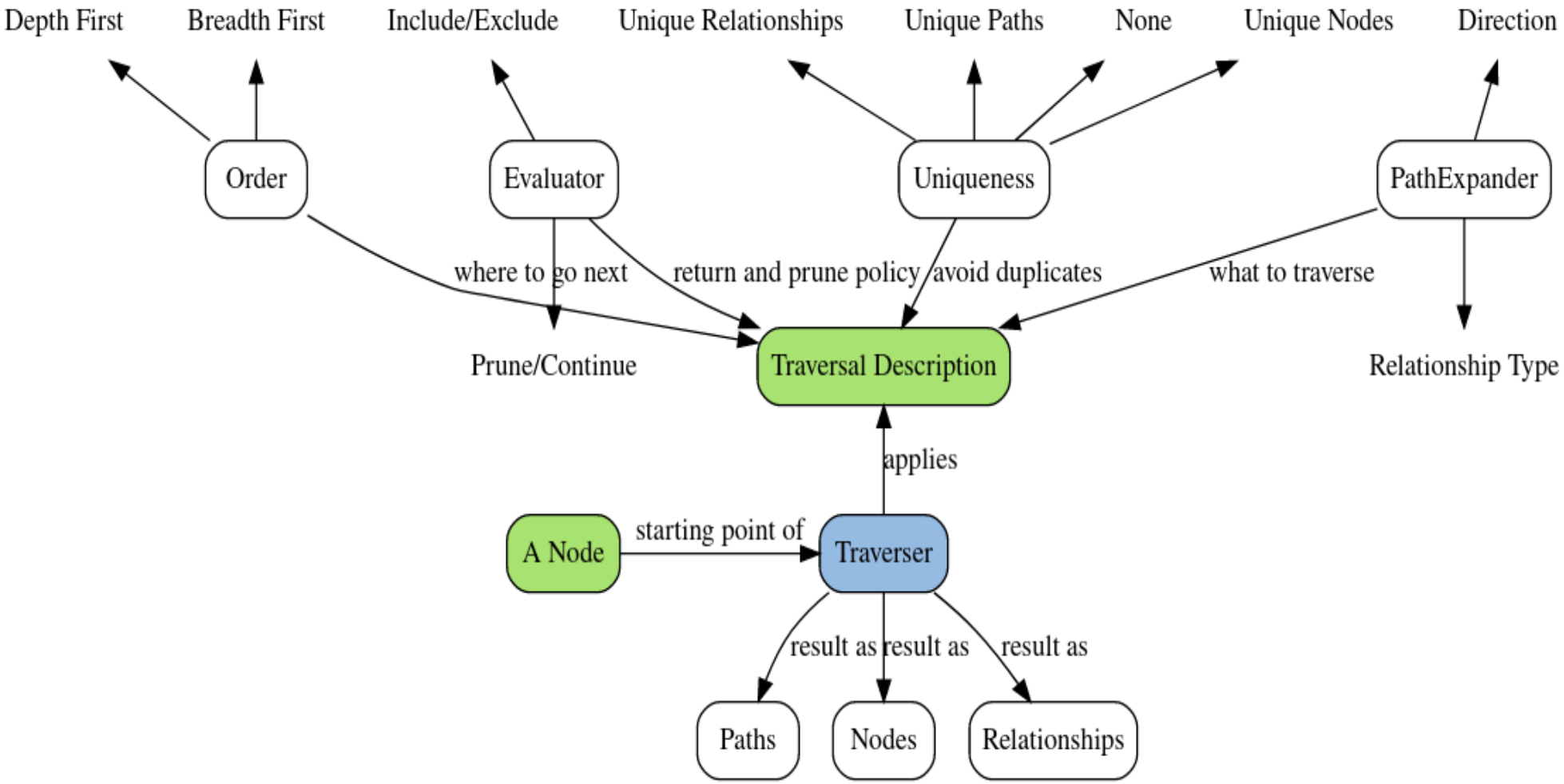
- Initiates and **manages a particular graph traversal** according to…
  - the provided traversal description, and
  - graph node / set of nodes where the traversal starts
- Allows for the **iteration over the matching paths**, one by one

Traversal Description

Components of a **traversal description**

- **Order**
  - Which graph traversal algorithm should be used
- **Expanders**
  - What relationships should be considered
- **Uniqueness**
  - Whether nodes / relationships can be visited repeatedly
- **Evaluators**
  - When the traversal should be terminated
  - What should be included in the query result

Traversal Framework



Source: neo4j.com

Graph Traversals

Find all friends (even indirect) of actor *Ivan Trojan*

- Print full actor names

```
import org.neo4j.graphdb.traversal.TraversalDescription;
import org.neo4j.graphdb.traversal.Evaluators;
import org.neo4j.graphdb.traversal.Uniqueness;
import org.neo4j.graphdb.traversal.Traverser;
import org.neo4j.graphdb.Direction;
import org.neo4j.graphdb.Path;
```

```
TraversalDescription td;
td = tx.traversalDescription()
     .breadthFirst()
     .relationships( RelTypes.KNOWS, Direction.BOTH )
     .evaluator( Evaluators.excludeStartPosition())
     .uniqueness( Uniqueness.NODE_GLOBAL );

Traverser traverser = td.traverse( actor1 );

for ( Path p : traverser ) {
  System.out.println(p.endNode().getProperty("name"));
}
```

Traversal #1
Jiří Macháček
Jitka Schneiderová
Zdeněk Svěrák

Nodes and Relationships

Add nodes for movies into our graph

- Create nodes, add MOVIE labels, add properties
  - samotari, Samotáři, 2000
  - medvidek, Medvídek, 2007
  - vratnelahve, Vratné lahve, 2006
- Remember node references

Create relationships between movies and actors

- Create relationships of PLAY type
  - samotari → trojan
  - samotari → machacek
  - samotari → schneiderova
  - medvidek → trojan
  - vratnelahve → sverak

Graph Traversals

Find all actors who played in *Medvídek* movie **together with all their friends** and friends of friends as well

- Use a single graph traversal, **implement a custom evaluator**
- Print full actor names

```
import org.neo4j.graphdb.traversal.Evaluator;
import org.neo4j.graphdb.traversal.Evaluation;
```

```
public static class MyEvaluator implements Evaluator {
  @Override
  public Evaluation evaluate(Path path) {
    return ...;
  }
}
```

```
td.evaluator(new MyEvaluator());
```

Graph Traversals

```
public static class MyEvaluator2 implements Evaluator {
    @Override
    public Evaluation evaluate(Path path) {
        if (path.endNode().hasLabel(Label.label("MOVIE"))) {
            return Evaluation.EXCLUDE_AND_CONTINUE;
        } else {
            return Evaluation.INCLUDE_AND_CONTINUE;
        }
    }
}
```

Graph Traversals

```
TraversalDescription td2;
td2 = tx.traversalDescription()
        .breadthFirst()
        .relationships(RelTypes.PLAY, Direction.OUTGOING)
        .relationships(RelTypes.KNOWS, Direction.BOTH)
        .evaluator(new MyEvaluator2())
        .uniqueness(Uniqueness.NODE_GLOBAL);

Traverser traverser2 = td2.traverse( movie2 );

for ( Path p : traverser2 ) {
   System.out.println(p.endNode().getProperty("name"));
}
```

Traversal #2
Ivan Trojan
Jiří Macháček
Jitka Schneiderová
Zdeněk Svěrák

Cypher Queries

Find all movies

- Express and execute a Cypher query
- Return movie nodes, print movie titles

```
import org.neo4j.graphdb.Result;
import java.util.Map;
```

```
Result result = tx.execute("MATCH (n:MOVIE) RETURN n");
while (result.hasNext()) {
  Map<String, Object> row = result.next();
  Node n = (Node)row.get("n");
  System.out.println(n.getProperty("title"));
}
```

Traversal #3
Samotáři
Medvídek
Vratné lahve

References

Embedded database and traversal framework

- https://neo4j.com/docs/java-reference/3.0/

JavaDoc

- https://neo4j.com/docs/java-reference/3.0/javadocs/

Cypher query language

- https://neo4j.com/docs/developer-manual/3.0/cypher/

Cypher reference card

- https://neo4j.com/docs/cypher-refcard/3.0/