Michał Matusiak

Zadanie - grafowe bazy danych

1. **Należy wymyślić, lub znaleźć prosty graf, który ma przynajmniej 2 węzły i 3 różne krawędzie (+ kilka atrybutów).**
   Wymyśliłem graf, który posiada 5 rodzajów wierzchołków:
   - Teacher
   - Student
   - Person
   - Group
   - Course

   Ich atrybuty:

   Teacher: name,surname,age,regionOfInterest

   Student: name,surname,age

   Course: name,lengthInSemesters

   Group: groupId

   Oraz 4 rodzaje krawędzi:

   - LIKES
   - ATTENDS
   - LEADS
   - COURSE_OF_GROUP

   Informacje wypisuje następującymi zapytaniami:

```
public void databaseStatistics() {
    System.out.println(graphDatabase.runCypher("CALL db.labels()"));
    System.out.println(graphDatabase.runCypher("CALL db.relationshipTypes()"));
}
```

```
public void viewSchema() {
    System.out.println(graphDatabase.runCypher("MATCH (n)\n" +
            "OPTIONAL MATCH (n)-[r]->(x)\n" +
            "WITH DISTINCT {l1: labels(n), r: type(r), l2: labels(x)}\n" +
            "AS `first degree connection`\n" +
            "RETURN `first degree connection`;"));
}
```

Wynik:

```
+-----------+
| label     |
+-----------+
| "Student" |
| "Teacher" |
| "Course"  |
| "Group"   |
| "Person"  |
+-----------+
```

```
+--------------------+
| relationshipType   |
+--------------------+
| "LIKES"            |
| "ATTENDS"          |
| "LEADS"            |
| "COURSE_OF_GROUP"  |
+--------------------+
```

```
+-------------------------------------------------------------------------------+
| first degree connection                                                       |
+-------------------------------------------------------------------------------+
| {l1 -> ["Student","Person"], r -> "ATTENDS", l2 -> ["Group"]}                 |
| {l1 -> ["Student","Person"], r -> "LIKES", l2 -> ["Student","Person"]}        |
| {l1 -> ["Course"], r -> <null>, l2 -> <null>}                                 |
| {l1 -> ["Person","Teacher"], r -> "LEADS", l2 -> ["Group"]}                   |
| {l1 -> ["Person","Teacher"], r -> "LIKES", l2 -> ["Person","Teacher"]}        |
| {l1 -> ["Person","Teacher"], r -> "LIKES", l2 -> ["Student","Person"]}        |
| {l1 -> ["Group"], r -> "COURSE_OF_GROUP", l2 -> ["Course"]}                   |
+-------------------------------------------------------------------------------+
```

2. **Napisać funkcje do tworzenia poszczególnych obiektów i relacji w grafie.**

```
private void createLikesRelationShip(Node person1, Node person2) {
    person2.createRelationshipTo(person1, RelationshipType.withName("LIKES"));
}

private Relationship createAttendsRelationship(Node student, Node group) {
    return student.createRelationshipTo(group, RelationshipType.withName("ATTENDS"));
}

private Relationship createLeadsRelation(Node teacher, Node group) {
    return teacher.createRelationshipTo(group, RelationshipType.withName("LEADS"));
}

private Relationship createCourseOfGroupRelationship(Node course, Node group) {
    return group.createRelationshipTo(course, RelationshipType.withName("COURSE_OF_GROUP"));
}
```

```
private Node createGroup(GraphDatabaseService gdb, String groupId) {
    Node group = gdb.createNode();
    group.setProperty("id", groupId);
    group.addLabel(() -> "Group");
    return group;
}

private Node createCourse(GraphDatabaseService gdb, String name, int lengthInSemesters) {
    Node course = gdb.createNode();
    course.setProperty("name", name);
    course.setProperty("lengthInSemesters", lengthInSemesters);
    course.addLabel(() -> "Course");
    return course;
}

private Node createTeacher(GraphDatabaseService gdb, String name, String surname, int age, String regionOfInterest) {
    Node teacher = gdb.createNode();
    teacher.setProperty("name", name);
    teacher.setProperty("surname", surname);
    teacher.setProperty("age", age);
    teacher.setProperty("regionOfInterest", regionOfInterest);
    teacher.addLabel(() -> "Teacher");
    teacher.addLabel(() -> "Person");
    return teacher;
}

private Node createStudent(GraphDatabaseService gdb, String name, String surname, int age) {
    Node student = gdb.createNode();
    student.setProperty("name", name);
    student.setProperty("surname", surname);
    student.setProperty("age", age);
    student.addLabel(() -> "Student");
    student.addLabel(() -> "Person");
    return student;
}
```

3. **Napisać bardzo prosty populator danych. Może to być zwykły Main czy Unit-test. Może być bardzo prosty, ale tak, aby było po 5 różnych obiektów i 10 relacji każdego typu.**

```
public void fillDatabase() {
    GraphDatabaseService gdb = graphDatabase.getGraphDatabaseService();

    try (Transaction tx = gdb.beginTx()) {

        // *******NODES********
        // STUDENTS
        Node mateusz = createStudent(gdb,  name: "Mateusz",  surname: "Kowalski",  age: 21);
        Node marta = createStudent(gdb,  name: "Marta",  surname: "Kowalski",  age: 20);
        Node marcin = createStudent(gdb,  name: "Marcin",  surname: "Kmicic",  age: 20);
        Node anna = createStudent(gdb,  name: "Anna",  surname: "Barosz",  age: 23);
        Node monika = createStudent(gdb,  name: "Monika",  surname: "Warciniec",  age: 21);

        // COURSES
        Node math = createCourse(gdb,  name: "Math",  lengthInSemesters: 2);
        Node programming = createCourse(gdb,  name: "Programming",  lengthInSemesters: 1);
        Node englishLiterature = createCourse(gdb,  name: "EnglishLiterature",  lengthInSemesters: 4);
        Node physics = createCourse(gdb,  name: "Physics",  lengthInSemesters: 2);
        Node biology = createCourse(gdb,  name: "Biology",  lengthInSemesters: 1);


        // TEACHERS
        Node andrzej = createTeacher(gdb,  name: "Andrzej",  surname: "Włoch",  age: 35,  regionOfInterest: "Math");
        Node marek = createTeacher(gdb,  name: "Marek",  surname: "Janus",  age: 45,  regionOfInterest: "Programming");
        Node janusz = createTeacher(gdb,  name: "Janusz",  surname: "Nowak",  age: 32,  regionOfInterest: "Math");
        Node konstanty = createTeacher(gdb,  name: "Konstanty",  surname: "Wiśniewski",  age: 56,  regionOfInterest: "Physics");
        Node kamil = createTeacher(gdb,  name: "Kamil",  surname: "Grabowski",  age: 32,  regionOfInterest: "Biology");


        // GROUPS
        Node a5 = createGroup(gdb,  groupId: "A5");
        Node b3 = createGroup(gdb,  groupId: "B3");
        Node a1 = createGroup(gdb,  groupId: "A1");
        Node a2 = createGroup(gdb,  groupId: "A2");
        Node c5 = createGroup(gdb,  groupId: "C5");
        Node c6 = createGroup(gdb,  groupId: "C6");
        Node c7 = createGroup(gdb,  groupId: "C7");
        Node c8 = createGroup(gdb,  groupId: "C8");
        Node c9 = createGroup(gdb,  groupId: "C9");
        Node c10 = createGroup(gdb,  groupId: "C10");
```

```
        // ********RELATIONS********

        createLikesRelationShip(marta, mateusz);
        createLikesRelationShip(mateusz, marta);
        createLikesRelationShip(marta, marcin);
        createLikesRelationShip(anna, monika);
        createLikesRelationShip(monika, anna);
        createLikesRelationShip(monika, kamil);
        createLikesRelationShip(anna, kamil);
        createLikesRelationShip(mateusz, kamil);
        createLikesRelationShip(janusz, konstanty);
        createLikesRelationShip(konstanty, janusz);

        createAttendsRelationship(mateusz, a5);
        createAttendsRelationship(marcin, a5);
        createAttendsRelationship(monika, a5);
        createAttendsRelationship(anna, a5);
        createAttendsRelationship(marta, a5);
        createAttendsRelationship(mateusz, a2);
        createAttendsRelationship(anna, c5);
        createAttendsRelationship(monika, c5);
        createAttendsRelationship(mateusz, b3);
        createAttendsRelationship(mateusz, a1);

        createLeadsRelation(andrzej, a5);
        createLeadsRelation(kamil, b3);
        createLeadsRelation(marek, c5);
        createLeadsRelation(marek, c6);
        createLeadsRelation(marek, c7);
        createLeadsRelation(marek, c8);
        createLeadsRelation(marek, c9);
        createLeadsRelation(marek, c10);
        createLeadsRelation(janusz, a2);
        createLeadsRelation(konstanty, a1);

        createCourseOfGroupRelationship(math, a5);
        createCourseOfGroupRelationship(programming, a1);
        createCourseOfGroupRelationship(programming, b3);
        createCourseOfGroupRelationship(programming, a2);
        createCourseOfGroupRelationship(programming, c5);
        createCourseOfGroupRelationship(programming, c6);
        createCourseOfGroupRelationship(programming, c7);
        createCourseOfGroupRelationship(programming, c8);
        createCourseOfGroupRelationship(programming, c9);
        createCourseOfGroupRelationship(programming, c10);


        tx.success();
    }
}
```

4. **Napisać funkcję do pobrania wszystkich relacji dla danego węzła**

```
public String getRelationshipsById(String nodeId) {
    return graphDatabase.runCypher(String.format("MATCH (x)-[r]-(y) " +
            "WHERE id(x) = %s " +
            "RETURN x,r,y", nodeId));
}

public List<String> getNodesRelationships(String label, String key, String value) {
    return graphDatabase.findNodesRelationships(label, key, value);
}
```

```
public List<String> findNodesRelationships(final String label, final String key, final Object value) {
    try (Transaction transaction = graphDatabaseService.beginTx()) {
        final ResourceIterator<Node> result = graphDatabaseService.findNodes(Label.label(label), key, value);
        transaction.success();
        return result
                .stream()
                .map(node -> node.getRelationships().spliterator())
                .flatMap(split -> StreamSupport.stream(split, parallel: false))
                .map(Object::toString)
                .collect(toList());
    }
}
```

Przykładowe wywołanie :

```
System.out.println(solution.getRelationshipsById( nodeId: "39"));
solution.getNodesRelationships( label: "Course", key: "name", value: "Programming").forEach(System.out::println);
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------+
| x                                                                           | r              | y                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------+
| Node[39]{age:32,regionOfInterest:"Biology",name:"Kamil",surname:"Grabowski"} | :LEADS[61]{} | Node[41]{id:"B3"}                                      |
| Node[39]{age:32,regionOfInterest:"Biology",name:"Kamil",surname:"Grabowski"} | :LIKES[9]{}  | Node[0]{age:21,name:"Mateusz",surname:"Kowalski"}      |
| Node[39]{age:32,regionOfInterest:"Biology",name:"Kamil",surname:"Grabowski"} | :LIKES[8]{}  | Node[3]{name:"Anna",surname:"Barosz",age:23}           |
| Node[39]{age:32,regionOfInterest:"Biology",name:"Kamil",surname:"Grabowski"} | :LIKES[7]{}  | Node[8]{age:21,name:"Monika",surname:"Warciniec"}      |
+-----------------------------------------------------------------------------------------------------------------------------------------------------+
4 rows

(44)-[COURSE_OF_GROUP,74]->(10)
(41)-[COURSE_OF_GROUP,72]->(10)
(49)-[COURSE_OF_GROUP,79]->(10)
(46)-[COURSE_OF_GROUP,76]->(10)
(42)-[COURSE_OF_GROUP,71]->(10)
```

5. **Napisać funkcje do znalezienia ścieżki dla danych dwóch węzłów**

```
public String shortestPathBetweenNodesById(String id1, String id2, String maxDepth) {
    return graphDatabase.runCypher(String.format("MATCH (x),(y), " +
            "p = shortestPath((x)-[*..%s]-(y)) " +
            "WHERE id(x) = %s AND id(y) = %s " +
            "RETURN p", maxDepth, id1, id2));
}
```

Wywołanie:

```
System.out.println(solution.shortestPathBetweenNodesById( id1: "1",  id2: "2",  maxDepth: "15"));
```

Wynik:

```
+-------------------------------------------------------------------------------------------------------------------+
| p                                                                                                                 |
+-------------------------------------------------------------------------------------------------------------------+
| [Node[1]{age:20,name:"Marta",surname:"Kowalski"},:LIKES[4]{},Node[2]{age:20,name:"Marcin",surname:"Kmicic"}] |
+-------------------------------------------------------------------------------------------------------------------+
```