SPRAWOZDANIE Z ZADANIA Z GRAFOWYCH BAZ DANYCH

Magdalena Nowak

Przygotowałam aplikację korzystając z wersji embedded - wbudowanego Neo4j. Stworzyłam ją w oparciu o projekt zamieszczony na moodle. Tworzyłam zapytania w języku Cypher.

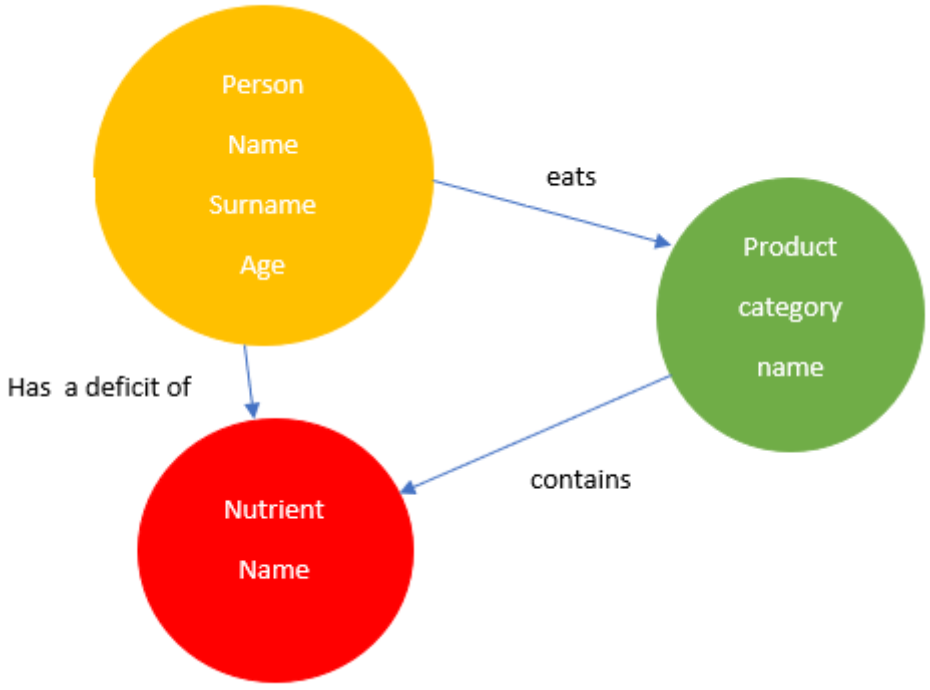
Kod dostępny tutaj ->

https://github.com/McDusia/Databases/tree/master/Neo4j

1. PONIŻEJ ZAMIESZCZONO GRAF, KTÓRY ODWZOROWANO KOLEJNO W BAZIE DANYCH.

Baza będzie przechowywała dane osób oraz produkty, które spożywają, a także składniki odżywcze, jakich mają niedobory.

Dzięki analizie danych można np. osobie zasugerować spożywanie jakiegoś produktu, którego nie spożywa, zawierający składnik, którego ma nieobór. Oczywiście jedna osoba może jeść wiele produktów i mieć niedobory wielu składników, podobnie produkty zawierają wiele składników.



2. NASTĘPNIE NAPISAŁAM FUNKCJE DO TWORZENIA OBIEKTÓW I RELACJI.

Klasy reprezentujące poszczególne węzły:

```
class Product {
    public String category;
    public String name;

    Product (String c, String n) {
        category = c;
        name = n;
    }
}

class Nutrient {
    public String name;
    Nutrient (String n) {
        name = n;
    }
}

class Person {
    public String name;
    public String surname;
    public int age;

    Person(String n, String s, int a) {
        name = n;
        surname = s;
        age = a;
    }
}
```

Metody dodające node węzły:

```
private String addProduct( String category, String productName) {

    String query = String.format("CREATE (n:Product {category: '%s', name:
'%s'}) return n",
            category, productName);
    return graphDatabase.runCypher(query);
}

private String addPerson(String name, String surname, int age) {

    String query = String.format("CREATE (n:Person { name: '%s', surname:
'%s', age: %d}) return n"
            , name, surname, age);
    return graphDatabase.runCypher(query);
}
```

```
private String addNutrient(String name) {

    String query = String.format("CREATE (n:Nutrient { name: '%s' }) return
n"
            ,name);
    return graphDatabase.runCypher(query);
}
```

Metody dodające relacje:

```
private String linkProductAndNutrient(String category, String name, String
nutrientName) {

    String query = String.format("MATCH (p:Product),(n:Nutrient)"+
            "WHERE p.category = '%s' AND p.name = '%s' AND n.name = '%s' "+
            "CREATE (p)-[r:CONTAINS]->(n) RETURN r", category, name,
nutrientName);
    return graphDatabase.runCypher(query);
}

private String linkPersonAndProduct(String name, String personName,
                                    String personSurname) {

    String query = String.format("MATCH (p:Person),(pr:Product)"+
            "WHERE p.name = '%s' AND p.surname = '%s' " +
            "AND pr.name = '%s' "+
            "CREATE (p)-[r:EATS]->(pr) RETURN r", personName,
personSurname, name);

    return graphDatabase.runCypher(query);
}

private String linkPersonAndDeficit(String name, String surname, String
nutrientName) {

    String query = String.format("MATCH (p:Person),(n:Nutrient)"+
                    "WHERE p.name = '%s' AND p.surname = '%s' AND "+
                    "n.name = '%s' CREATE (p)-[r:HAS_A_DEFICIT_OF]->(n)
RETURN r",
            name, surname, nutrientName);

    return graphDatabase.runCypher(query);
}
```

3. POPULATOR DANYCH:

Funkcja dodająca węzły i relacje:

```
public void addData() {

    List<Nutrient> nutrients = new ArrayList<>();
    nutrients.add(new Nutrient("witamina A"));
    nutrients.add(new Nutrient("witamina C"));
    nutrients.add(new Nutrient("witamina D"));
```

```
nutrients.add(new Nutrient("potas"));
nutrients.add(new Nutrient("magnez"));

List<Product> products = new ArrayList<>();
products.add(new Product("owoce","banan"));
products.add(new Product("warzywa","pomidor"));
products.add(new Product("owoce","truskawka"));
products.add(new Product("owoce","cytryna"));
products.add(new Product("nabiał","śmietana"));
products.add(new Product("nabiał","jogurt"));
products.add(new Product("nabiał","masło"));
products.add(new Product("nabiał","ser żółty"));
products.add(new Product("warzywa","marchewka"));

List<Person> people = new ArrayList<>();
people.add(new Person("Jan","Kowalski", 50));
people.add(new Person("Helena","Kowalska", 42));
people.add(new Person("Jacek","Nowak", 13));
people.add(new Person("Michał","Naj", 24));
people.add(new Person("Zofia","Lis", 40));

for(Nutrient nutrient : nutrients) {
    addNutrient(nutrient.name);
}

for(Product product : products) {
    addProduct(product.category, product.name);
}

for(Person person : people) {
    addPerson(person.name, person.surname, person.age);
}

//add dependency between products and nutrients
linkProductAndNutrient("owoce","banan", "potas");
linkProductAndNutrient("owoce","banan", "magnez");
linkProductAndNutrient("warzywa","pomidor", "potas");
linkProductAndNutrient("warzywa","pomidor", "witamina C");
linkProductAndNutrient("owoce","truskawka","witamina C");
linkProductAndNutrient("owoce","cytryna","witamina C");
linkProductAndNutrient("nabiał","śmietana","witamina D");
linkProductAndNutrient("nabiał","jogurt","witamina D");
linkProductAndNutrient("nabiał","masło","witamina A");
linkProductAndNutrient("warzywa","marchewka","witamina A");
linkProductAndNutrient("nabiał","ser żółty","witamina A");

//add dependency between products and people
linkPersonAndProduct("banan", "Jan","Kowalski");
linkPersonAndProduct("truskawka", "Jan","Kowalski");
linkPersonAndProduct("cytryna", "Jan","Kowalski");
linkPersonAndProduct("cytryna","Jacek","Nowak");
linkPersonAndProduct("śmietana", "Jacek","Nowak");
linkPersonAndProduct("jogurt", "Jacek","Nowak");
linkPersonAndProduct("banan", "Jacek","Nowak");
linkPersonAndProduct("marchewka", "Zofia","Lis");
linkPersonAndProduct("banan", "Zofia","Lis");
linkPersonAndProduct("masło", "Jacek","Nowak");

//add dependency between people and deficit of nutrient
linkPersonAndDeficit("Zofia","Lis", "witamina D");
linkPersonAndDeficit("Zofia","Lis", "witamina C");
```

```
    linkPersonAndDeficit("Jacek","Nowak", "potas");
    linkPersonAndDeficit("Jacek","Nowak", "magnez");
    linkPersonAndDeficit("Jan","Kowalski", "witamina D");
    linkPersonAndDeficit("Jan","Kowalski", "witamina A");
    linkPersonAndDeficit("Jan","Kowalski", "potas");
    linkPersonAndDeficit("Michał","Naj", "potas");
    linkPersonAndDeficit("Michał","Naj", "magnez");
    linkPersonAndDeficit("Michał","Naj", "witamina A");

}
```

Po wywołaniu:

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

```
+------------+
| label      |
+------------+
| "Nutrient" |
| "Product"  |
| "Person"   |
+------------+
3 rows

+--------------------+
| relationshipType   |
+--------------------+
| "CONTAINS"         |
| "EATS"             |
| "HAS_A_DEFICIT_OF" |
+--------------------+
3 rows
```

Metody dostępowe:

```
private String getPeople() {

    String query = String.format("MATCH (p:Person) RETURN p");

    return graphDatabase.runCypher(query);
}

private String getProducts() {

    String query = String.format("MATCH (p:Product) RETURN p");
    return graphDatabase.runCypher(query);
}

private String getNutrients() {

    String query = String.format("MATCH (n:Nutrient) RETURN n");
    return graphDatabase.runCypher(query);
}
```

```
+-------------------------------------------------+
| p                                               |
+-------------------------------------------------+
| Node[9]{name:"Jan",surname:"Kowalski",age:50}   |
| Node[25]{age:42,name:"Helena",surname:"Kowalska"} |
| Node[26]{age:13,name:"Jacek",surname:"Nowak"}   |
| Node[27]{name:"Michał",surname:"Naj",age:24}    |
| Node[28]{name:"Zofia",surname:"Lis",age:40}     |
+-------------------------------------------------+
5 rows
```

```
+-----------------------------------------------+
| p                                             |
+-----------------------------------------------+
| Node[1]{name:"pomidor",category:"warzywa"}    |
| Node[2]{name:"truskawka",category:"owoce"}    |
| Node[3]{name:"cytryna",category:"owoce"}      |
| Node[4]{name:"śmietana",category:"nabiał"}    |
| Node[5]{name:"jogurt",category:"nabiał"}      |
| Node[6]{name:"masło",category:"nabiał"}       |
| Node[7]{category:"nabiał",name:"ser żółty"}   |
| Node[8]{name:"marchewka",category:"warzywa"}  |
| Node[24]{name:"banan",category:"owoce"}       |
+-----------------------------------------------+
9 rows
```

```
+-----------------------------+
| n                           |
+-----------------------------+
| Node[0]{name:"witamina A"}  |
| Node[20]{name:"witamina C"} |
| Node[21]{name:"witamina D"} |
| Node[22]{name:"potas"}      |
| Node[23]{name:"magnez"}     |
+-----------------------------+
5 rows
```

```
private String findPerson(String name, String surname) {

    String query = String.format("MATCH (p:Person)"+
                    "WHERE p.name = '%s' AND p.surname = '%s' RETURN p",
            name, surname);

    return graphDatabase.runCypher(query);
}
```

```
+-----------------------------------------------+
| p                                             |
+-----------------------------------------------+
| Node[9]{name:"Jan",surname:"Kowalski",age:50} |
+-----------------------------------------------+
1 row
```

4. POBIERANIE WSZYSTKICH RELACJI DLA DANEGO WĘZŁA

Pobieranie po jednym typie relacji dla węzła (format: węzeł – relacja - węzeł):

```
private String findPersonDeficits(String name, String surname) {

    String query = String.format("MATCH deficits = (p:Person)-[r
:HAS_A_DEFICIT_OF]->(n:Nutrient) " +
            "WHERE p.name = '%s' AND p.surname='%s' RETURN deficits", name,
surname);
    return graphDatabase.runCypher(query);
}


System.out.println(findPersonDeficits("Jan", "Kowalski"));
```

```
+-----------------------------------------------------------------------------------------------------------+
| deficits                                                                                                  |
+-----------------------------------------------------------------------------------------------------------+
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:HAS_A_DEFICIT_OF[36]{},Node[22]{name:"potas"}]            |
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:HAS_A_DEFICIT_OF[35]{},Node[0]{name:"witamina A"}]        |
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:HAS_A_DEFICIT_OF[34]{},Node[21]{name:"witamina D"}]       |
+-----------------------------------------------------------------------------------------------------------+
3 rows
```

```
private String findProductsEatenByPerson(String name, String surname) {

    String query = String.format("MATCH products = (p:Person)-[r :EATS]-
>(pr:Product) " +
            "WHERE p.name = '%s' AND p.surname='%s' RETURN products", name,
surname);
    return graphDatabase.runCypher(query);
}


System.out.println(findProductsEatenByPerson("Jan", "Kowalski"));
```

```
+-----------------------------------------------------------------------------------------------------------+
| products                                                                                                  |
+-----------------------------------------------------------------------------------------------------------+
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:EATS[2]{},Node[3]{name:"cytryna",category:"owoce"}]       |
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:EATS[1]{},Node[2]{name:"truskawka",category:"owoce"}]     |
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},:EATS[30]{},Node[24]{name:"banan",category:"owoce"}]       |
+-----------------------------------------------------------------------------------------------------------+
3 rows
```

```
private String findNutrientsInProduct(String name) {

    String query = String.format("MATCH nutrients = (p:Product)-[r
:CONTAINS]->(n:Nutrient) " +
            "WHERE p.name = '%s' RETURN nutrients", name);
    return graphDatabase.runCypher(query);
}
System.out.println(findNutrientsInProduct("masło"));
```

```
+-------------------------------------------------------------------------------------+
| nutrients                                                                           |
+-------------------------------------------------------------------------------------+
| [Node[6]{name:"masło",category:"nabiał"},:CONTAINS[27]{},Node[0]{name:"witamina A"}] |
+-------------------------------------------------------------------------------------+
1 row
```

Pobieranie wszystkich relacji dla danego typu węzła (format: sama relacja):

```
private String getRelationshipsForProduct(String name) {

    String query = String.format("MATCH (p:Product)-[r]-(n) " +
            "WHERE p.name = '%s' RETURN r", name);
    return graphDatabase.runCypher(query);
}

private String getRelationshipsForPerson(String name, String surname) {

    String query = String.format("MATCH (p:Person)-[r]-(n) " +
            "WHERE p.name = '%s' AND p.surname = '%s' RETURN r", name,
surname);
    return graphDatabase.runCypher(query);
}
private String getRelationshipsForNutrient(String name) {

    String query = String.format("MATCH (n:Nutrient)-[r]-(a) " +
            "WHERE n.name = '%s' RETURN r", name);
    return graphDatabase.runCypher(query);
}


System.out.println(getRelationshipsForPerson("Jan", "Kowalski"));
+-------------------------+
| r                       |
+-------------------------+
| :HAS_A_DEFICIT_OF[36]{} |
| :HAS_A_DEFICIT_OF[35]{} |
| :HAS_A_DEFICIT_OF[34]{} |
| :EATS[2]{}              |
| :EATS[1]{}              |
| :EATS[30]{}             |
+-------------------------+
6 rows


System.out.println(getRelationshipsForProduct("banan"));
+-----------------+
| r               |
+-----------------+
| :EATS[8]{}      |
| :EATS[6]{}      |
| :EATS[30]{}     |
| :CONTAINS[20]{} |
| :CONTAINS[0]{}  |
+-----------------+
5 rows
```

```
System.out.println(getRelationshipsForNutrient("witamina A"));
+-------------------------+
| r                       |
+-------------------------+
| :HAS_A_DEFICIT_OF[39]{} |
| :HAS_A_DEFICIT_OF[35]{} |
| :CONTAINS[29]{}         |
| :CONTAINS[28]{}         |
| :CONTAINS[27]{}         |
+-------------------------+
5 rows
```

Pobieranie relacji jednego typu dla węzła:

```
private String getRelationshipsToNutrientsForProduct(String name) {

    String query = String.format("MATCH (p:Product)-[r]->(n) " +
            "WHERE p.name = '%s' RETURN r", name);
    return graphDatabase.runCypher(query);
}

private String getRelationshipsToProductsForPerson(String name, String
surname) {

    String query = String.format("MATCH (p:Person)-[r]->(n: Product) " +
            "WHERE p.name = '%s' AND p.surname = '%s' RETURN r", name,
surname);
    return graphDatabase.runCypher(query);
}

private String getRelationshipsToNutrientsForPerson(String name, String
surname) {

    String query = String.format("MATCH (p:Person)-[r]->(n: Nutrient) " +
            "WHERE p.name = '%s' AND p.surname = '%s' RETURN r", name,
surname);
    return graphDatabase.runCypher(query);
}

System.out.println(getRelationshipsToNutrientsForPerson("Jan",
"Kowalski"));
```

```
+-------------------------+
| r                       |
+-------------------------+
| :HAS_A_DEFICIT_OF[36]{} |
| :HAS_A_DEFICIT_OF[35]{} |
| :HAS_A_DEFICIT_OF[34]{} |
+-------------------------+
3 rows
```

```
System.out.println(getRelationshipsToProductsForPerson("Jan", "Kowalski"));
```

```
+-------------+
| r           |
+-------------+
| :EATS[2]{}  |
| :EATS[1]{}  |
| :EATS[30]{} |
+-------------+
3 rows
```

```
System.out.println(getRelationshipsToNutrientsForProduct("banan"));
```

```
+-----------------+
| r               |
+-----------------+
| :CONTAINS[20]{} |
| :CONTAINS[0]{}  |
+-----------------+
2 rows
```

5. FUNKCJE ZNAJDUJĄCE ŚCIEŻKI DLA DANYCH DWÓCH WĘZŁÓW

```
//--------------   Find paths  ----------------
private String findPathsPersonToProducts_Nodes(String name, String
surname, String prName) {

    String query = String.format("MATCH path = (p:Person)--
(pr:Product) " +
            "WHERE p.name = '%s' AND p.surname='%s' AND pr.name='%s'
RETURN nodes(path)", name,surname,prName);
    return graphDatabase.runCypher(query);
}

private String findPathsPersonToProducts_Relationships(String name,
String surname, String prName) {

    String query = String.format("MATCH path = (p:Person)--
(pr:Product) " +
            "WHERE p.name = '%s' AND p.surname='%s' AND pr.name='%s'
RETURN relationships(path)", name,surname,prName);
    return graphDatabase.runCypher(query);
}

private String findPathsPersonToNutrient_Nodes(String name, String
surname, String nName) {

    String query = String.format("MATCH path = (p:Person)--
(n:Nutrient) " +
            "WHERE p.name = '%s' AND p.surname='%s' AND n.name='%s'
RETURN nodes(path)", name,surname,nName);
    return graphDatabase.runCypher(query);
}

private String findPathsPersonToNutrient_Relationships(String name,
String surname, String nName) {
```

```
        String query = String.format("MATCH path = (p:Person)--
    (n:Nutrient) " +
                        "WHERE p.name = '%s' AND p.surname='%s' AND
    n.name='%s' RETURN relationships(path)",
                name,surname,nName);
        return graphDatabase.runCypher(query);
    }

    private String findPathsProductToNutrient_Nodes(String pName, String
    nName) {

        String query = String.format("MATCH path = (p:Product)--
    (n:Nutrient) " +
                "WHERE p.name = '%s' AND n.name='%s' RETURN nodes(path)",
    pName,nName);
        return graphDatabase.runCypher(query);
    }

    private String findPathsProductToNutrient_Relationships(String pName,
    String nName) {

        String query = String.format("MATCH path = (p:Product)--
    (n:Nutrient) " +
                        "WHERE p.name = '%s' AND n.name='%s' RETURN
    relationships(path)",
                pName,nName);
        return graphDatabase.runCypher(query);
    }
```

Przykłady użycia:

```
System.out.println(findPathsPersonToProducts_Nodes("Jan",
"Kowalski","banan"));
```

```
+-----------------------------------------------------------------------------------------+
| nodes(path)                                                                             |
+-----------------------------------------------------------------------------------------+
| [Node[9]{name:"Jan",surname:"Kowalski",age:50},Node[24]{name:"banan",category:"owoce"}] |
+-----------------------------------------------------------------------------------------+
1 row
```

```
System.out.println(findPathsPersonToProducts_Relationships("Jan",
"Kowalski","banan"));
```

```
+----------------------+
| relationships(path)  |
+----------------------+
| [:EATS[30]{}]        |
+----------------------+
1 row
```

```
System.out.println(findPathsProductToNutrient_Nodes("cytryna", "witamina
C"));
```

```
+-------------------------------------------------------------------------+
| nodes(path)                                                             |
+-------------------------------------------------------------------------+
| [Node[3]{name:"cytryna",category:"owoce"},Node[20]{name:"witamina C"}] |
+-------------------------------------------------------------------------+
1 row
```

```
System.out.println(findPathsProductToNutrient_Relationships("cytryna",
"witamina C"));
```

```
+---------------------+
| relationships(path) |
+---------------------+
| [:CONTAINS[24]{}]   |
+---------------------+
1 row
```