JEGYZŐKÖNYV

Modern adatbázis rendszerek MSc

2022. tavasz féléves feladat

Neo4j

Készítette: **Bolyki Balázs**
Neptun kód: **N5IF3V**

A feladat leírása:
Neo4j feladatok megoldása.

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

1. feladat:
A: Hozzunk létre User és Tweet nodeokat!

User property-k:
- username
- country

Tweet property-k:
- short
- created
- text

B: Vigyünk fel hozzá adatokat!

C: Üssünk be egy tweetet és egy :authored kapcsolatot duplikálva (mintha elrontottuk volna). Töröljük az egyik tweetet és a hozzá tartozó kapcsolatot! (id alapján)

D: Javítsuk Kittikeh hajzselés kommentjében a hajzselét úgy, hogy tartalmazzon ékezetet!

E: Hány felhasználó van az USA-ból?

F: Helyezzünk el like-okat is relationshipként!
- Pacemaker420 likeolja Hamburger minden tweetjét!
- Mindenki likeolja Kittikeh Annual exhibition-ös tweetjét!
- TRex likeolja TankYou tweetjét.

G: Adjuk vissza a felhasználók neveit és hogy ki mennyi tweetet írt!

H: Adjuk vissza azt a felhasználót, aki a legtöbbet likeolt!

I: Adjuk vissza a felhasználók neveit és azt, hogy ki hány likeot gyűjtött!

J: Kreáljunk egy olyan kapcsolatot, ami "körbelájkolást" okoz. (Két felhasználó egymás tweetjét lájkolja). Detektáljuk ezt a körbelájkolást, és rajzoljuk ki a tweetekkel együtt!

2. feladat:

A: Hozzunk létre városokat és utakat, amik közöttük vezetnek!

B: Hány olyan út vezet A városból F városba, ahol pontosan 4 várost kell érinteni?

C: Melyek azok a városok, ahova A-ból el lehet jutni?

3. feladat:
Készítsünk Java API-t a megírt Twitter adatbázisunk kezelésére!

A: Készítsünk egy programot, ami kilistázza a felhasználóneveket az adatbázisban!

B: Írjunk egy command line applikációt arra, hogy valaki tweetelhessen!

Ahol az elküldött parancsba user input kerül, ott használjunk PreparedStatement típusú parancsot!

- A program először kiírja az adatbázisban található felhasználók neveit.
- Bekéri, hogy ki vagy te.
- Bekéri a tweetet, amit készíteni akarsz.
- A beírt szöveg lesz a Tweet *text* mezője, a Tweet első 5 szava *short* mezője, a dátum pedig a mostani pillanat.

- Elkészíti a tweetet és a kapcsolatot a tweet és a User között, ahol a kapcsolat labelje :authored.
- Kiírja a sikert, és exitál.

A feladat elkészítésének lépései:
1. feladat:
create database twitter
A:
create constraint for (u:User) require u.username is unique;
create constraint for (u:User) require u.username is not null;
create constraint for (u:User) require u.country is not null;

create constraint for (t:Tweet) require t.short is not null;
create constraint for (t:Tweet) require t.created is not null;
create constraint for (t:Tweet) require t.text is not null;

B:
create (u:User {username:"TRex", country:"USA"});
create (u:User {username:"Klopacska", country:"Hungary"});
create (u:User {username:"Pacemaker420", country:"USA"});
create (u:User {username:"Hamburger", country:"USA"});
create (u:User {username:"Kittykeh", country:"Hungary"});
create (u:User {username:"TankYou", country:"Russia"});

create (t:Tweet {short:"My hajzsele isn't working",
created:datetime("2019-06-01T18:40:32.142+0100"), text:"This is hajzsele is terrible. I am very offended. Let's hate big company!"}); // Kittykeh
create (t:Tweet {short:"Klopacska jó",
created:datetime("2020-06-09T18:40:32.142+0100"), text:"Sok barátot szereztem a Klopacskában, de a Tigrises teától a vesém külföldre menekült!"}); // Klopacska
create (t:Tweet {short:"I'm extinct.",
created:datetime("2021-04-11T11:40:32.142+0100"), text:"LOL, I saw this big meteorite. It was far away and I'm dying to see it close up.!"}); //TRex

```
create (t:Tweet {short:"The pace of this movie...",
created:datetime("2017-10-22T19:40:30.142+0100"), text:"I was watching SpiderMan and the pacing just isn't right! My pacemaker and the movie set different paces and it's so confusing!"}); //Pacemaker420
create (t:Tweet {short:"Burger king is p*ssy!",
created:datetime("2021-12-01T18:40:32.142+0100"), text:"I was eating Burger king and they gave me a hamburger so small that they should call themselves Burger Queen. I'm gonna say this on TikTok if I fit in the picture..."}); // Hamburger
create (t:Tweet {short:"McDony is making you fat",
created:datetime("2022-03-01T18:40:32.142+0100"), text:"I just realised McDonalds is putting calories into my hamburgers!!! This illuminati atrocity conspiracy thing should be punished!"}); // Hamburger
create (t:Tweet {short:"It started as a bicycle",
created:datetime("2018-12-12T18:40:32.142+0100"), text:"Tavaris! Each and every time I try to make a bicycle it turns out to be a TANK. I cannot be at peace like this!"}); //Tank you
create (t:Tweet {short:"Happy new year",
created:datetime("2020-01-01T18:40:32.142+0100"), text:"Happy New Year for everybody! I vow to make an actual selfie this year. My upper arms shouldn't be a problem!"}); //TRex
create (t:Tweet {short:"Annal exhibition",
created:datetime("2021-02-01T18:40:32.142+0100"), text:"This year as usual I make my annal exhibition. Everyone should come in! This exhibition is held every year annaly! As for those jerks who don't know 'annal' means yearly! I know how to write! Boors..."}); //Kittykeh
create (t:Tweet {short:"Pink house",
created:datetime("2020-06-13T18:40:32.142+0100"), text:"I think the color white is so plain and sterile! Accordingly the White House should be repainted as pink! Please sign the petition!"}); //Kittykeh

match (t:Tweet), (u:User {username:"Kittykeh"}) where t.short="My hajzsele isn't working" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Klopacska"}) where t.short="Klopacska jó" create (u)-[:authored]->(t);
```

match (t:Tweet), (u:User {username:"TRex"}) where t.short="I'm extinct." create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Pacemaker420"}) where t.short="The pace of this movie..." create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Hamburger"}) where t.short="Burger king is p*ssy!" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Hamburger"}) where t.short="McDony is making you fat" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"TankYou"}) where t.short="It started as a bicycle" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"TRex"}) where t.short="Happy new year" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Kittykeh"}) where t.short="Annal exhibition" create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"TRex"}) where t.short="I'm extinct." create (u)-[:authored]->(t);
match (t:Tweet), (u:User {username:"Kittykeh"}) where t.short="Pink house" create (u)-[:authored]->(t);

C:
match (t:Tweet) where id(t)=01234 detach delete t

D:
match (t:Tweet {short:"My hajzsele isn't working"}) set t.short="Hajzselé"

E:
match (u:User {country:"USA"}) return count(u)

F:
match (u1:User {username:"Pacemaker420"}), (u2:User {username:"Hamburger"})-[:authored]-(t:Tweet) create (u1)-[:likes]->(t);
match (t:Tweet {short:"Annal exhibition"}), (u:User) create (u)-[:likes]->(t)match (t:Tweet {short:"Annal exhibition"}), (u:User) where u.username!="Kittykeh" create (u)-[:likes]->(t);

match (trex:User {username:"TRex"}), (tank:User {username:"TankYou"})-[:authored]-(t:Tweet) create (trex)-[:likes]->(t);

G:
match (u:User)-[:authored]->(t:Tweet) with u.username as username, count(t) as authored return username, authored;

H:
match (u:User)-[:likes]->(t:Tweet) with u.username as username, count(t) as liked return username order by liked desc limit 1

I:
match (liker:User)-[:likes]->(t:Tweet)<-[:authored]-(author:User) with author.username as username, count(liker) as likers return username, likers

J:
match (u:User {username:"TankYou"}), (t:Tweet {short:"Happy new year"}) create (u)-[:likes]->(t); //Körbelájkolás kreálása
match (u1:User)-[:authored]->(t1:Tweet)<-[:likes]-(u2:User)-[:authored]->(t2:Tweet)<-[:likes]-(u1:User) return u1, u2, t1, t2 //Körbelájkolás detektálása

2. feladat:
create database cities

A:
create (c:City {name:"A"});
create (c:City {name:"B"});
create (c:City {name:"C"});
create (c:City {name:"D"});
create (c:City {name:"E"});
create (c:City {name:"F"});
create (c:City {name:"G"});
create (c:City {name:"H"});

```
create (c:City {name:"I"});
create (c:City {name:"J"});
create (c:City {name:"K"});
create (c:City {name:"L"});
create (c:City {name:"M"});
create (c:City {name:"N"});
create (c:City {name:"O"});

match (c1:City {name:"A"}), (c2:City {name:"B"}) create (c1)-[:leadsTo
{distance: 12}]->(c2);
match (c1:City {name:"B"}), (c2:City {name:"C"}) create (c1)-[:leadsTo
{distance: 22}]->(c2);
match (c1:City {name:"C"}), (c2:City {name:"D"}) create (c1)-[:leadsTo
{distance: 1}]->(c2);
match (c1:City {name:"D"}), (c2:City {name:"A"}) create (c1)-[:leadsTo
{distance: 11}]->(c2);
match (c1:City {name:"C"}), (c2:City {name:"E"}) create (c1)-[:leadsTo
{distance: 34}]->(c2);
match (c1:City {name:"E"}), (c2:City {name:"F"}) create (c1)-[:leadsTo
{distance: 55}]->(c2);
match (c1:City {name:"F"}), (c2:City {name:"G"}) create (c1)-[:leadsTo
{distance: 14}]->(c2);
match (c1:City {name:"G"}), (c2:City {name:"H"}) create (c1)-[:leadsTo
{distance: 32}]->(c2);
match (c1:City {name:"H"}), (c2:City {name:"I"}) create (c1)-[:leadsTo
{distance: 37}]->(c2);
match (c1:City {name:"I"}), (c2:City {name:"E"}) create (c1)-[:leadsTo
{distance: 48}]->(c2);
match (c1:City {name:"K"}), (c2:City {name:"L"}) create (c1)-[:leadsTo
{distance: 99}]->(c2);
match (c1:City {name:"L"}), (c2:City {name:"M"}) create (c1)-[:leadsTo
{distance: 123}]->(c2);
match (c1:City {name:"M"}), (c2:City {name:"N"}) create (c1)-[:leadsTo
{distance: 3}]->(c2);
```

```
match (c1:City {name:"N"}), (c2:City {name:"O"}) create (c1)-[:leadsTo {distance: 12}]->(c2);
match (c1:City {name:"O"}), (c2:City {name:"K"}) create (c1)-[:leadsTo {distance: 34}]->(c2);
match (c1:City {name:"K"}), (c2:City {name:"M"}) create (c1)-[:leadsTo {distance: 54}]->(c2);
match (c1:City {name:"A"}), (c2:City {name:"H"}) create (c1)-[:leadsTo {distance: 54}]->(c2);
```

B:

```
match (A:City {name:"A"})-[*4]-(c:City {name:"F"}) with count(A) as rodes return rodes
```

C:

```
match (A:City {name:"A"})-[*..]-(c:City) return c
```

3. feladat:

Készítsünk Java API-t a megírt Twitter adatbázisunk kezelésére!

Előkészület: Amennyiben fut az adatbázis szerver, állítsuk azt le! Írjuk át a <neo4j-home>/conf/neo4j.conf konfigurációs fájlt, hogy a default adatbázis a twitter legyen!

```
# The name of the default database.
dbms.default_database=twitter
```

Ezek után indítsuk az adatbázis szervert a szokásos módon (Linux alatt az alábbi paranccsal).

<neo4j-home>/bin/neo4j start

Készítsünk Eclipse-ben új projektet. Legegyszerűbben úgy lehet kipróbálni a JDBC drivert, ha letöltjük a neo4j .jar driverét, és hozzáadjuk a build path-hoz Eclipse-en keresztül.

3.A.

```
package iit;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Hello {

	public static void main(String[] args) {
		try (Connection con =
DriverManager.getConnection("jdbc:neo4j:bolt://localhost:7687", "neo4j", "n5if3v");
				Statement stmt = con.createStatement()) {
			ResultSet rs = stmt.executeQuery("MATCH (n:User) with
n.username as name RETURN name");
			while (rs.next()) {
				System.out.println(rs.getString("name"));
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

3.B

```
package iit;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Tweeter {

	static Scanner sc = new Scanner(System.in);
```

```
	public static void main(String[] args) {
		try (Connection con =
DriverManager.getConnection("jdbc:neo4j:bolt://localhost:7687", "neo4j", "n5if3v")) {
			List<String> usernames = getUsernames(con);
			String username = promptUsername(usernames);
			String tweetText = promptTweet(username);
			Tweet tweet = buildTweet(tweetText);
			System.out.println("Your tweet: " + tweet);
			publishTweet(con, username, tweet);
			System.out.println("Succesfully tweeted!");
		}
		catch (MyException e) {
			System.out.println(e.getMessage());
		}
		catch (Exception e) {
			e.printStackTrace();
		}
		sc.close();
	}

	static List<String> getUsernames(Connection con) throws MyException {
		List<String> names = new ArrayList<>();
		try(Statement stmt = con.createStatement()) {
			ResultSet rs = stmt.executeQuery("MATCH (n:User) with n.username as
name RETURN name");
			while (rs.next()) {
				names.add(rs.getString("name"));
			}
			if (names.isEmpty()) {
				throw new MyException("No users.");
			}
		}
		catch (SQLException e) {
			throw new MyException("Could not retrieve users.");
		}
		return names;
	}

	static String promptUsername(List<String> usernames) {
		String username = "";
		System.out.println("Available usernames:");
		printList(usernames);
		do {
			System.out.println("Who are you?");
			username = sc.nextLine();
		} while (!usernames.contains(username));
		return username;
	}

	static String promptTweet(String username) {
		System.out.println("Please, dear " + username + " write a tweet!");
```

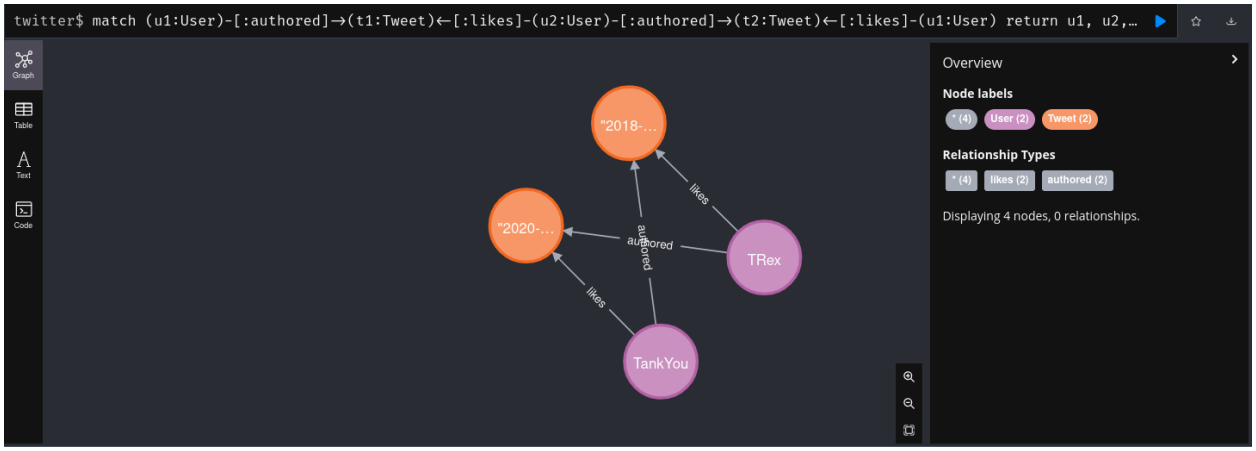
```
                String tweet = sc.nextLine();
                return tweet;
        }

        static Tweet buildTweet(String tweetText) {
                String[] words = tweetText.split(" ");
                int limit = Math.min(5, words.length);
                StringBuilder shortText = new StringBuilder();
                for (int i = 0; i < limit; i++) {
                        shortText.append(words[i] + " ");
                }
                return new Tweet(shortText.toString().trim(), tweetText, LocalDateTime.now());
        }

        static void publishTweet(Connection con, String username, Tweet tweet) throws
MyException{
                try(PreparedStatement stmtCreateTweet =
                                con.prepareStatement("CREATE (t:Tweet {short:?,
created:datetime(?), text:?})");
                        PreparedStatement stmtConnect =
                                con.prepareStatement("match (t:Tweet), (u:User {username:?})
where t.short=? create (u)-[:authored]->(t)")) {
                        stmtCreateTweet.setString(1, tweet.getShortText());
                        stmtCreateTweet.setString(2, tweet.getDate());
                        stmtCreateTweet.setString(3, tweet.getText());
                        stmtCreateTweet.executeUpdate();
                        stmtConnect.setString(1, username);
                        stmtConnect.setString(2, tweet.getShortText());
                        stmtConnect.executeUpdate();
                }
                catch (SQLException e) {
                        throw new MyException("Could not make tweet correctly.");
                }
        }

        static void printList(List<String> list) {
                for (String item : list) {
                        System.out.println(item);
                }
        }
}

class Tweet {

        // "2020-06-13T18:40:32.142+0100"
        static DateTimeFormatter dtf = DateTimeFormatter.ISO_LOCAL_DATE_TIME;

        String shortText;
        String text;
        LocalDateTime date;
```

```
        public Tweet(String shortText, String text, LocalDateTime date) {
                this.shortText = shortText;
                this.text = text;
                this.date = date;
        }

        public String getShortText() {
                return shortText;
        }

        public String getText() {
                return text;
        }

        public String getDate() {
                return date.format(dtf);
        }

        @Override
        public String toString() {
                return shortText + "\n" + date.format(dtf) + "\n" + text;
        }
}
```

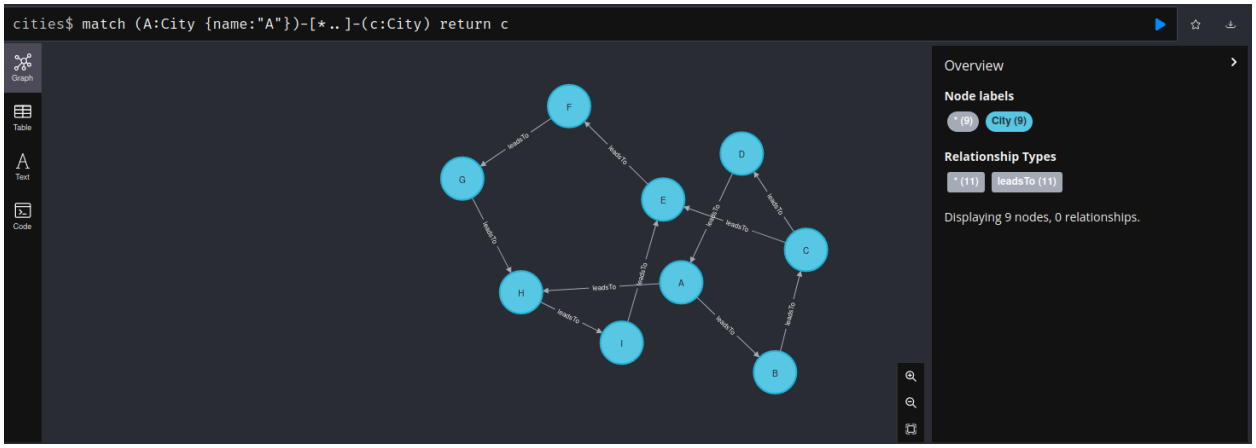
A futtatás eredménye:

Az eredmények részletezése feladatonként változik. A Neo4j Browser verziója többféle ábrázolásmódra is alkalmas: táblázatos, szöveges, nóduszokkal rajzolt. Alább található néhány feladat ábrázolása. Minden feladathoz csatolni az eredményt fölöslegesen nagy terjedelmű lenne.

1.J



2.C



3. feladat:

Java API Használat példák.

3.A.

Hello.java × Tweeter.java

```
1  package iit;
2
3  import java.sql.Connection;
7
8  public class Hello {
9
10     public static void main(String[] args) {
11         try (Connection con = DriverManager.getConnection("jdbc:neo4j:bolt://localhost:7687", "neo4j", "n5if3v");
12                 Statement stmt = con.createStatement()) {
13             ResultSet rs = stmt.executeQuery("MATCH (n:User) with n.username as name RETURN name");
14             while (rs.next()) {
15                 System.out.println(rs.getString("name"));
16             }
17         } catch (Exception e) {
18             e.printStackTrace();
19         }
20     }
21 }
```

Problems Javadoc Declaration Console ×

```
<terminated> Hello (1) [Java Application] /home/bbalage/Other/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (May 11, 2022, 10:11:04 AM
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Hamburger
Kittykeh
Klopacska
Pacemaker420
TRex
TankYou
```

3.B.

Hello.java | Tweeter.java

```
13 import java.util.Scanner;
14
15 public class Tweeter {
16
17     static Scanner sc = new Scanner(System.in);
18
19     public static void main(String[] args) {
20         try (Connection con = DriverManager.getConnection("jdbc:neo4j:bolt://localhost:7687", "neo4j", "n5if3v"))
21             List<String> usernames = getUsernames(con);
22             String username = promptUsername(usernames);
23             String tweetText = promptTweet(username);
24             Tweet tweet = buildTweet(tweetText);
25             System.out.println("Your tweet: " + tweet);
26             publishTweet(con, username, tweet);
27             System.out.println("Succesfully tweeted!");
28         }
```

Problems | Javadoc | Declaration | Console

```
<terminated> Tweeter [Java Application] /home/bbalage/Other/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (May 11, 2022, 10:11:53 AM
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Available usernames:
Hamburger
Kittykeh
Klopacska
Pacemaker420
TRex
TankYou
Who are you?
Klopacska
Please, dear Klopacska write a tweet!
I think coffee is lame. Everyone should drink tiger tea!
Your tweet: I think coffee is lame.
2022-05-11T10:12:38.143463567
I think coffee is lame. Everyone should drink tiger tea!
Succesfully tweeted!
```