Modern adatbázis rendszerek MSc – 11. Gyakorlat

Téma: **Gráf adatbázis modell programozása. Neo4J kezelése, API programozása**

GitHub repository: **NEPTUNKODMDB**

Mappa neve: **NEPTUNKOD_0506**

Fejlesztő-környezet: Neo4J

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

Az elkészült feladatokat töltse fel a GitHub mappába!

1. feladat:

a) Node-ok létrehozása és feltöltése adatokkal:

meal node:

create (:meal {id:”m1”, name:”Piedone”, price:1710, category: „Pizza”})

create (:meal {id:”m2”, name:” Magyaros”, price:1850, category: „Pizza”})

create (:meal {id:”m3”, name:” BBQ_csirkemell_box”, price:2120, category: „Húsétel”})

customer node:

create (:customer {id:”c1”, address:”Kazincy_Ferenc_utca-1”, phonenumber: „0620122222”})

create (:customer {id:”c2”, address:”Bartók_Béla_utca-4”, phonenumber: „123”})

create (:customer {id:”c3”, address:”Kuruc_utca-22”, phonenumber: „06203125664”})

b) Node-ok közötti kapcsolatok létrehozása:

match (m:meal {id:”m1”}), (c:customer {id:”c1”}) create (m) –[:ordered_by]->(c)

match (m:meal {id:”m2”}), (c:customer {id:”c1”}) create (m) –[:ordered_by]->(c)

match (m:meal {id:”m1”}), (c:customer {id:”c2”}) create (m) –[:ordered_by]->(c)

match (m:meal {id:”m3”}), (c:customer {id:”c3”}) create (m) –[:ordered_by]->(c)

c) Egyszerűbb select parancsok:

Pizza típusú ételek lekérdezése

match (m:meal {category:”Pizza”}) return m.name

1800-nál drágább ételek lekérezése

match (m:meal) where m.price > 1800 return m.name, m.price

Pizzát rendelő megrendelők lekérdezése

match (m:meal {category:”Pizza”}) –[]-> (c.customer) return c.id, c.address

d) Update parancsok:

Új „mennyiség” mező hozzáadása a kapcsolatokhoz, egy ember egy ételből akár többet is rendelhetett (alapérteke 1).

match () –[l:ordered_by]->() set l.amount = 1

Az új mennyiség mező először háromra állítása, majd bizonyos megrendelések esetében 1-el valónövelése.

match (m:meal {id:”m1”}) –[l:ordered_by]->() set l.amount = 3

match (m:meal {id:”m2”}) –[l:ordered_by]->() set l.amount = l.amount+1

e) Delete parancsok:

C3-as megrendelő által rendelt ételek törlése.

match (m:meal)–[l:ordered_by]->(c:customer {id:”c3”}) detach delete m

Az m1-es étel és c1-es megrendelő kapcsolatának törlése.

match (m:meal {id:”m1”)–[l]->(c:customer {id:”c1”}) delete l

f) Felvitt mező típus törlésével:

A hozzáadott mennyiség mező törlése.

match (l:ordered_by) remove l.amount

g) Komplex lekérdezések:

A legdrágább étel lekérdezése.

match (m:meal) return m.name, m.price order by m.price desc limit 1

Lekérdezni milyen ételkategóriák vannak tárolva.

match (m:meal) return distinct m.category order by m.category

Átlagos ételár lekérdezése.

match (m:meal) return avg(m.price)

Ételkategóriánként az átlagos ételár lekérdezése.

match (m:meal) return m.category, avg(m.price)

Megrendelőnként a rendelt ételek lekérdezése.

match (c:customer) <-[]-(m:meal) return c.adress, collect(m.name) as order

2. feladat:

Java kliens alkalmazás írása az adatok kezelésére.

Használt driver:

https://neo4j.com/blog/neo4j-jdbc-driver-3-0-3-and-3-1-0-released/

a) Node-ok létrehozása, adatok felvitele:

```
public static void add_meals() {
		try {
			Class.forName("org.neo4j.jdbc.Driver");
			Properties props = new Properties();
			props.setProperty("user", "neo4j");
			props.setProperty("password", "neo4j");
			Connection con = DriverManager.getConnection(
			"jdbc:neo4j:http://localhost:8389",props);
			String query;

			mealT[] meals = new mealT[3];
			meals[0] = new mealT("m1", "Piedone", 1710, "Pizza");
			meals[0] = new mealT("m2", "Magyaros", 1850, "Pizza");
			meals[0] = new mealT("m3", "BBQ_csirkemell_box", 2120,
			"Húsétel");

			query = "ceate (a:meal {_id:{1}, name:{2}, price:{3},
			category:{4}})";
			PreparedStatement stmt = con.prepareStatement(query);
			for (int i=0; i<5; i++) {
				stmt.setString(1,  meals[i].id);
				stmt.setString(2,  meals[i].name);
				stmt.setInt(3,  meals[i].price);
				stmt.setString(4,  meals[i].category);
				stmt.executeUpdate();
			}

			con.close();
		}	catch(Exception ee) {
			System.out.println(ee.getMessage());
		}
	}
```

b) Query kezelő metódus írása az adott árnál drágább ételek lekérdezésére.

```
public static void read_meals(int limprice) {
		try {
			Class.forName("org.neo4j.jdbc.Driver");
			Properties props = new Properties();
			props.setProperty("user", "neo4j");
			props.setProperty("password", "neo4j");
			Connection con = DriverManager.getConnection(
			"jdbc:neo4j:http://localhost:8389",props);

			String query = "MATCH (a:meal) WHERE a.price > {1} RETURN
			 a.name";
			PreparedStatement stmt = con.prepareStatement(query);
			stmt.setInt(1,  limprice);
			ResultSet rs = stmt.executeQuery();
			while(rs.next()) {
				System.out.println(rs.getString("a.name"));
			}

			con.close();
		}	catch(Exception ee) {
			System.out.println(ee.getMessage());
		}
	}
```