JEGYZŐKÖNYV

Modern adatbázis rendszerek MSc

LDAP és Neo4j

Készítette: **Juhász Ákos**

Neptunkód: **F58KQ8**

Dátum: 2024. május. 13.

Tartalom

1. LDAP feladat leírása: 3

2. SSH csatorna létrehozása.....3

3. JXplorer kapcsolat indítása.....3

4. LDAP java alkalmazás előkészítése.....5

5. Java kapcsolat létrehozása.....5

6. Java lekérdezések.....6

7. Neo4j feladat leírása 8

8. Node-ok létrehozása és feltöltése adatokkal 8

9. Neo4j lekérdezések és parancsok 9

9.1 Egyszerűbb select parancsok: 9

9.2 Update parancsok:.....10

9.3 Delete parancsok:.....10

9.4 Komplex lekérdezések:.....11

1. LDAP feladat leírása:

A feladat célja az egyetem LDAP adatbázisához való csatlakozás az egyetemi hálózaton kívülről, ssh csatornán keresztül. Az adatbázis elérése után új hallgatói könyvtár létrehozása, illetve oda adatok felvitele JXplorer, illetve Java alkalmazás segítségével, majd a tárolt adatok lekérése.

2. SSH csatorna létrehozása

Ahhoz, hogy akár távoli gépről is lehessen kapcsolatot létrehozni az egyetemi adatbázishoz, először létre kell hozni egy *SSH tunneling channel*-t.

A csatorna létrehozásához, szükséges a Mobaxterm alkalmazás telepítése, majd azon belül a ssh csatorna létrehozása a következő adatokkal:

- Local port forwarding
- Local Clients
    - Fordwarded port: 389
- Remote Server
    - Remote server: 193.6.5.58
    - Remote port: 389
- SSH Server
    - SSH server: jerry.iit.uni-miskolc.hu
    - SSh login: miskolci egyetemi linux felhasználónév
    - SSH port: 22

Ezt követően az így létrehozott csatornát az egyetemi linux fiók jelszavával vehetjük használatba.

3. JXplorer kapcsolat indítása

A tunneling channel elindítása utána hozzáférhetünk az egyetemi szerverhez, ezáltal el tudjuk érni kívülről JXplorer segítségével az egyetemi LDAP adatbázist is.

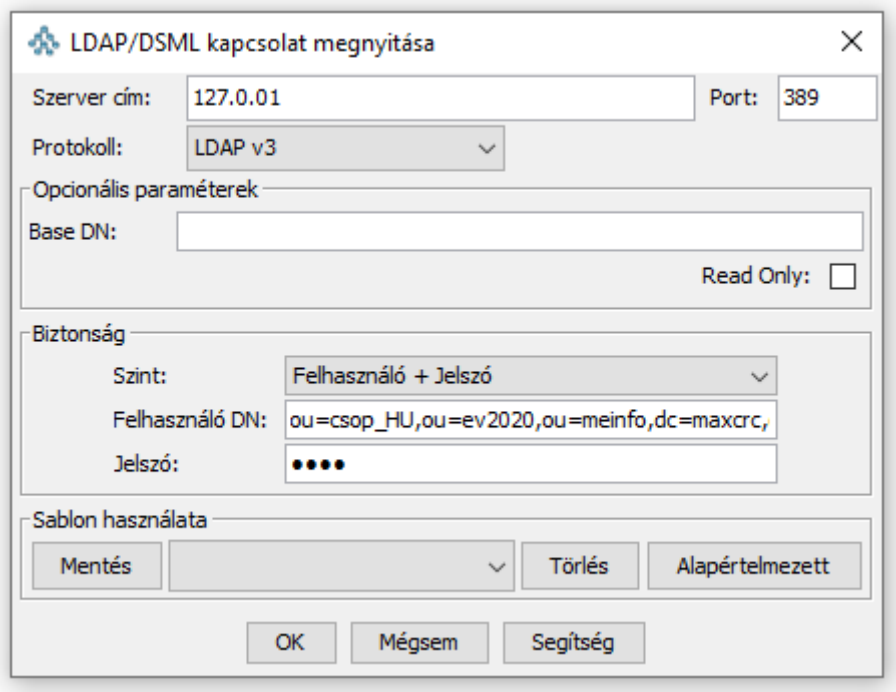
Ehhez szükséges egy kapcsolat létrehozás a JXploreren belül a következő adatokkal:

- Host: 127.0.0.1
- Level: User + Password

- User DN: ou=csop_HU,ou=ev2020,ou=meinfo,dc=maxcrc,dc=com
- Password: H578

vagy

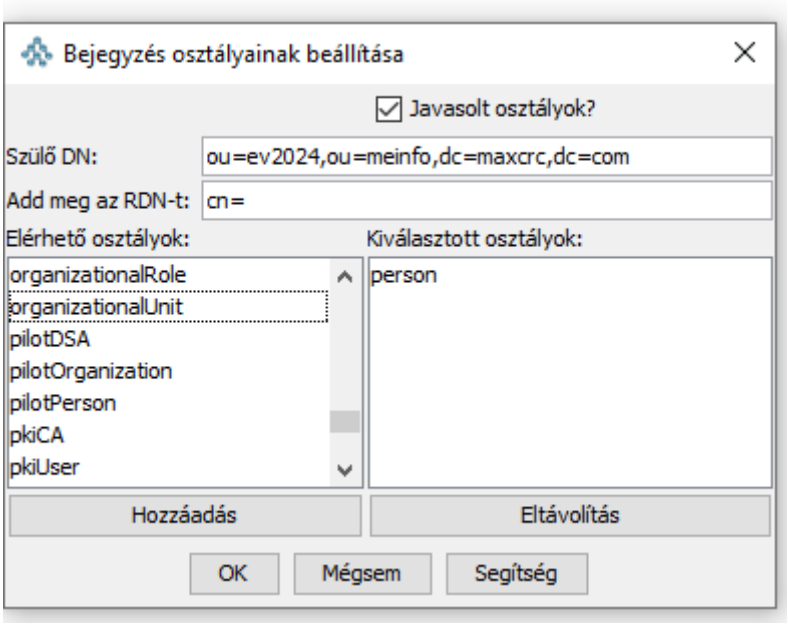
- User DN: ou=csop_SH,ou=ev2020,ou=meinfo,dc=maxcrc,dc=com
- Password: S121



JXplorer új kapcsolat felugróablak

Ha sikeres volt a csatlakozás a JXplorer-en keresztül láthatóak az előző években létrehozott kurzusokhoz tartozó adatbázis bejegyzések.

Létre kell hozni egy új bejegyzést az adott évi kurzusok mellett, ehhez meg kell adni RDN/cn azonosítót. Ez ebben az esetben a neptunkód. Majd a selected class menüpontban ki kell választani a person opciót.



JXplorerben megnyitott könyvtár beállítása

Végül pedig meg kell adni az sn attribútumot, ami a mi esetünkben a hallgató neve.

4. LDAP java alkalmazás előkészítése

Az előzőekben létrehozott könyvtárat és az azon belül lévő létrehozott adatokat a JXplorer-en kívül, java alkalmazás segítségével is elérhetjük.

Első lépésként le kell tölteni és hozzá kell adni egy külső ldap kezelő .jar könyvtárat az alkalmazásunk build pathjához. Ennek segítségével hozzáférünk az összes ldap kapcsolatot és tranzakciót kezelő metódushoz.

5. Java kapcsolat létrehozása

Az alábbi programkód segítségével érhető el az JXplorerben létrehozott könyvtár. Fontos, hogy az SSH csatornának ekkor is futnia kell, hogy elérhessük az adatbázist.

```
public static void test_connection () {
		int ldapPort = LDAPConnection.DEFAULT_PORT;
		int ldapVersion = LDAPConnection.LDAP_V3;
		String ldapHost = "127.0.0.1";
		String loginDN2 =
			"ou=csop_HU,ou=ev2020,ou=meinfo,dc=maxcrc,dc=com";
		String password2 = "H578";
		LDAPConnection lc = new LDAPConnection();
		boolean correct = true;
		try {
			lc.connect(ldapHost, ldapPort);
			System.out.println("S1");
			lc.bind(ldapVersion, loginDN2, password2.getBytes());
			System.out.println("S1");
			lc.disconnect();
		} catch (Exception ee) {
			correct = false;
		}

		System.out.println( correct ? "password is correct.":
			"The password is incorrect.\n");
}
```

Sikeres futtatás esetén, a program beolvassa az LDAP kapcsolat elérési címét és megadott jelszavát és létrehoz egy elérési utat az adatbázishoz. A továbbiakban kezelői funkciókkal bővítjük az alkalmazást.

```
<terminated> test_ldap [Java Ap
S1
S1
password is correct.
```

Futtatási eredmény

6. Java lekérdezések

Az alábbi programkód futtatásával az adatbázis elérése után az általunk megadott könyvtárat nyitjuk meg, és az előzőekben kitöltött ’sn’ attribútumját olvassuk le, majd íratjuk ki a konzolra.

```
public static void name_query () {
            int ldapPort = LDAPConnection.DEFAULT_PORT;
            int ldapVersion = LDAPConnection.LDAP_V3;
            String ldapHost = "127.0.0.1";
            String loginDN2 = "ou=csop_HU,ou=ev2020,ou=meinfo,dc=maxcrc,dc=com";
            String password2 = "H578";
            String searchBase =
            "cn=NEPTUNKÓD,ou=ev2024,ou=meinfo,dc=maxcrc,dc=com";
            String searchFilter = "";
            int searchScope = LDAPConnection.SCOPE_BASE;
            LDAPConnection lc = new LDAPConnection();

            try {
                  lc.connect(ldapHost, ldapPort);
                  System.out.println("S1");
                  lc.bind(ldapVersion, loginDN2, password2.getBytes());
                  System.out.println("S2");
                  LDAPSearchResults searchResults =
                  lc.search(searchBase, searchScope, searchFilter, null, false);

                  while (searchResults.hasMore()) {
                  LDAPEntry nextEntry = null;

                  try {
                  nextEntry = searchResults.next();
                  System.out.println("\n" + nextEntry.getDN());
                  com.novell.ldap.LDAPAttributeSet attributeSet =
                  nextEntry.getAttributeSet();
                  Iterator allAttributes = attributeSet.iterator();

                  while(allAttributes.hasNext()) {
                  LDAPAttribute attribute = (LDAPAttribute)allAttributes.next();
                  String attributeName = attribute.getName();

                  if (attributeName.compareTo("sn")==0) {
                        //where we ask for the "sn" attribute
                        Enumeration<String> allValues =
                        attribute.getStringValues();
                        String attributevalue = (String)
                        allValues.nextElement();
                        System.out.println("    " + attributeName + " :: " +
                        attributevalue);
                  }
                  }
                  } catch (LDAPException e) {
                        System.out.println("Error: " + e.toString());
                  }
                  }

                  lc.disconnect();
            } catch (Exception ee) {}
}
```

Sikeres futtatás esetén a kapcsolat létrejön, és a konzolon megjelenik a keresett könyvtár elérési útja, illetve a benne tárolt sn attribútum tartalma, ami ebben az esetben a felhasználó neve.

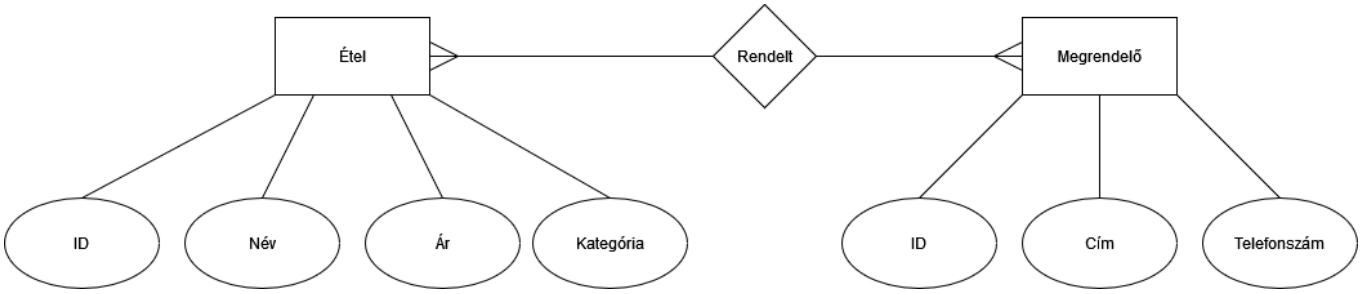
```
<terminated> test_ldap [Java Application] C:\Users\admin\.p2\
S1
S2

cn=F58KQ8,ou=ev2024,ou=meinfo,dc=maxcrc,dc=com
   sn :: Juhász Ákos
```

Futtatási eredmény

7. Neo4j feladat leírása

A következő feladat célja a neo4j gráf alapú adatbáziskezelő rendszerben, node-ok létrehozása és azokon különböző parancsok végrehajtása. A parancsok között lekérdező, adatmódosító, struktúramódosító és törlő funkciókat tesztelünk.

A feladatok kivitelezésénél az alábbi ER modell alapján fogunk dolgozni, ami egy étterem adatbázis modelljének egy részlete. A következő lekérdezésekhez ez alapján töltjük fel adatokkal az adatbázisunkat, bár a mezők elnevezése angol nyelven fog történni.



ER modell ábra

8. Node-ok létrehozása és feltöltése adatokkal

Az étterem által árult ételek „meal” node-jai:

```
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”})
```

Az étteremnél rendelő vendégek adatait rögzítő „customer” node-ok létrehozása:

```
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”})
```

Ahhoz, hogy nyilván tudjuk tartani, hogy ki milyen ételre adott le rendelést, létre kell hoznunk a két node között egy élt. Ez az él az ételektől mutat a megrendelők felé „ordered_by” vagy „általa rendelve” névvel ellátva.

```
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)
```

9. Neo4j lekérdezések és parancsok

9.1 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
```

9.2 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 való növelése.

```
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
```

9.3 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
```

A hozzáadott mennyiség mező törlése.

```
match (l:ordered_by) remove l.amount
```

9.4 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
```