Mérési jegyzőkönyv – Szoftver Laboratórium 5

3. mérés: JDBC

|  |  |
| --- | --- |
| Név: | **Szupera Zita** |
| Neptun kód: | **AWXUC6** |
| Feladat kódja: | **23-SZORAK** |
| Mérésvezető neve: | **Bacsi Tibor** |
| Mérés időpontja: | **2015-03-26 17:15** |
| Mérés helyszíne: | **HSZK** |
| A működő alkalmazás elérhetősége: | http://rapid.eik.bme.hu/~awxuc6/jdbc/ |
| Megoldott feladatok: | **1,2,3,4** |
| Elérhető pontszám (plusz pontok nélkül): | **40p** |

1. Felhasználói útmutató
2. A MySzorakApplication alkalmazás a letöltés követő indítás után egyből működés képes.
3. Az alkalmazás egy Szórakozási szokásokat tartalmazó adatbázishoz nyújt grafikus felületű hozzáférést. Az adatbázishoz használat előtt csatlakozni kell. Ehhez indítás után csak meg kell nyomni az ablak felső sávjában található *connect* gombot, a csatlakozáshoz szükséges adatokat bekérő mezők előre ki vannak töltve. Csatlakozni minden indításkor után szükséges.
4. Ezek után két fül (*Search* és *Modify*) kattinthatóvá válik. Ezeken a felületeken férhetünk hozzá az adatbázis tartalmához.
5. Az első fülön az adatbázisban található szórakozóhelyek adatait listázhatjuk ki. A lap felső részében található egy keresőmező. Ha ezt üresen hagyjuk, akkor az összes hely nevét, címét, telefonszámát kilisitázza a program. Ha a mezőt kitöltjük, akkor az alkalmazás azt egy szórakozóhely neveként értelmezi (töredékre is lehet keresni), és a beírt adatnak megfelelő helyeket jeleníti meg.
6. A második fülön az adatbázisban tárolt látogatók adatait tudjuk módosítani, új látogatókat, látogatási szokásokat felvenni. Az fülön több adatmezőt találunk, ezek némelyikét kötelező kitölteni (ezek \*-al jelöltek), a többi azonban szabadon hagyható. A bevitt adatok függvényében az alkalmazás vagy létrehoz egy új bejegyzést, vagy egy melévőt módosít (attól függően, hogy a megadott ID már szerepel-e az adatbázisban). A *Submit* gomb megnyomásakor a program adatellenőrzést végez: a telefonszám formátuma kötött, a bevételnek 15 000 és 200 0000 000 között kell lennie, az ID mezőkbe pedig csak számok kerülhetnek. Az fül tetején egy checkbox található, amely kiválasztása után a program egy szórakozóhely azonosítóját is bekéri, amelyet a felvett vagy módosított látogató egyből meg is látogat (létrejön egy látogatásí szokás bejegyzés). Ennek módja: látogató adatainak megadása, *Submit*, ha ez sikeres volt, akkor lehet megadni a szórakozóhely ID-ját, *Submit All*. Ha nem létező azonosítót adunk meg, akkor az első lépésben történt adatmódosítások elvesznek. Fontos, hogy ebben a módban a *Submit* lenyomása után a módosítások még nem véglegesek, azok csak egy helyes szórakozóhely ID megadása után mentődnek. Fontos az is, hogy a helyes működés érdekében még a *Submit* kiadása előtt be kell jelölni a checkbox-ot. Ha ezt nem tesszük meg akkor az alkalmazás úgy fog működni, mintha csak látogatói adatokat akarunk manipulálni, és a *Submit all* megnyomásakor hibát kapunk. A gombonymások után minden esetben egy felugró ablak tájékoztat az adott esemény sikerességéről, vagy sikeretelenségéről.
7. Az események sikertelenségének két oka lehet: felhasználói hiba (pl.: kötelező mező nincs kitöltve, vagy nem jó formátumban van kitöltve) vagy adatbázis hiba. Az első típusú hibát könnyen javíthatjuk a kapott hibaüzenetek alapján. Az alkalmazáshibákat azonban nagy valószínűséggel nem lehet könnyen elhárítani (esetleg újraindítás, újracsatlakozás segíthet), ez nem is a felhasználó dolga.
8. Ennek ellenére a felhasználó is tájékozódhat a program futása során keletkező eseményekről, hibákról a *Log* fülön
9. Mérési feladatok megoldása
   1. feladat

A megoldáshoz használt SQL utasítás

Paraméteres keresés:

**select name, address, phone from places where name like ? ESCAPE'\\'**

Paraméter nélküli keresés:

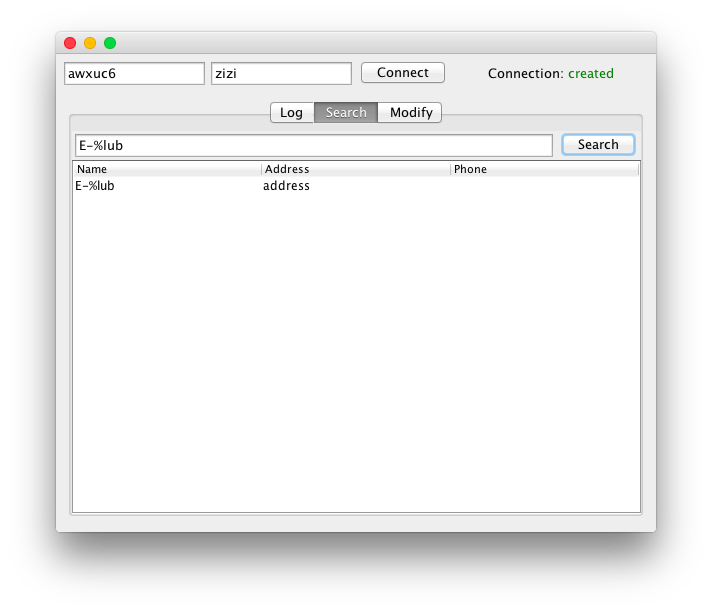
**select name, address, phone from places**

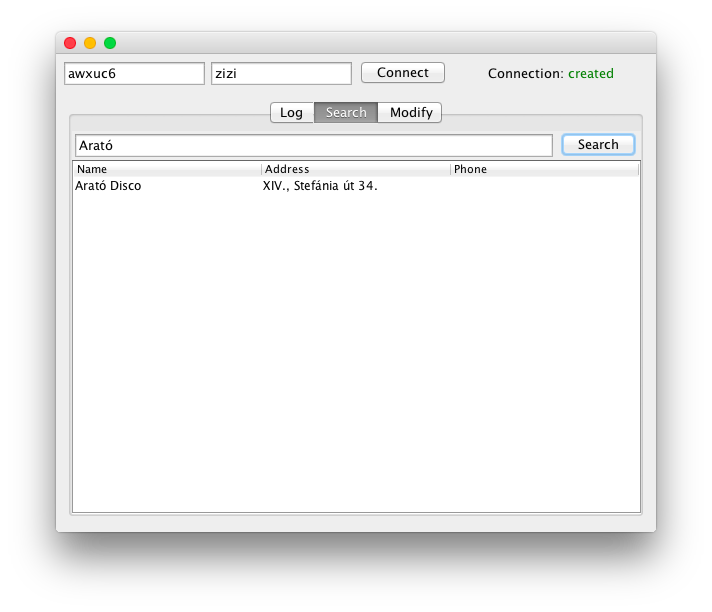
Magyarázat

A Search gomb lenyomásakor ellenőrzöm a keresőmező tartalmát:

* Ha ez üres, akkor egy paraméter nélküli keresést futtatok le, amely az összes szórakozóhelyet kilistázza
* Ha a mező nem üres, akkor a kapott sztringet beállítom a paraméteres keresés paraméterének.
  + Mivel töredékre is tudni kell keresni, az SQL utasításban a *like* kulcsszót használom és a paramétert %-ek között adom át, ami az SQL szintaktika szerint azt jelenti, hogy a kapott szerint előtt és után is állhat még szöveg.
  + Mivel az utasításban *like* kulcsszót használok, külön meg kell oldani a speciális karakterek kereshetőségét is (Oracle-ben használt speciális karakterek *like* kulcsszó után: \, \_, %, {, }). Ennek módja a speciális karakter elé illesztett \ karakter. A megoldásomban készítettem egy tömböt, amibe beletettem az előbb felsorolt karaktereket. A keresőmezőből kapott sztringen egyesével megnézem, hogy szerepl-e az adott karakter, ha igen, akkor a karakte elé ’\’ jelet teszek, ehhez a java *replaceAll* függvényét használom. A helyes működéshez még annyit kell tudni, hogy a \ karakter a javaban (is) escape karakterként működik, tehát a *„\”* sztringben a \ semlegesíti a „ karaktert, ennek az elekerülése végett *„\”* helyett *„\\”*-t kell írni, emellett az SQL parancsban jelölni kell, hogy a \ jel után álló karaktert ne kezelje speciális karakterként.

Tesztelés:

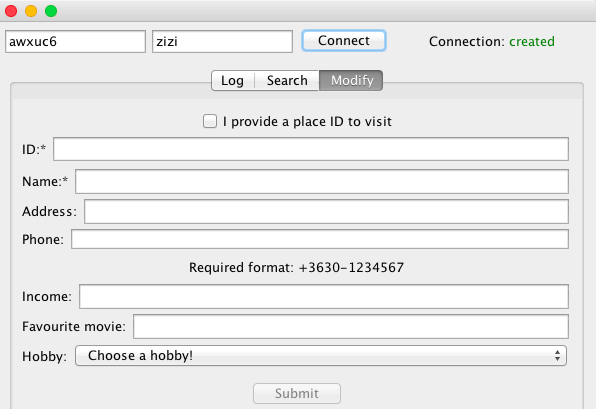
speciális karakterek kereshetősége

töredéknévre keresés

* 1. Feladat

1. A megoldáshoz használt SQL utasítás

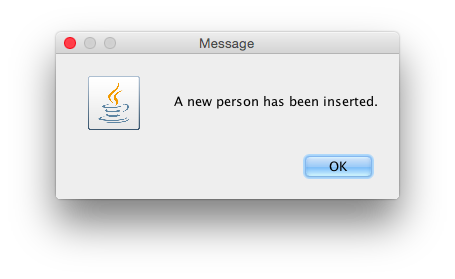
A kapott adatokat először megpróbálom beszúrni:

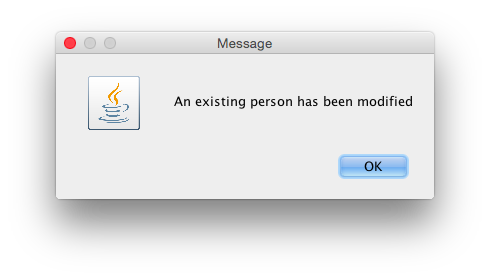
1. **insert into persons values (?, ?, ?, ?, ?, ?, ?)**
   * 1. Ha nem sikerül (SQLIntegrityConstraintViolationExceptiont kapok), akkor frissítem a megfelelő rekordot:
2. **update persons set name = ?, address = ?, phone = ?, income = ?, hobby = ?, favourite\_movie = ? where person\_id = ?**
3. 
   * 1. Magyarázat
     2. Új beviteli mezők

Az adatok módosításakor két kötelező mező van: Név, ID, amíg ezek nincsenek kitöltve, a Submit gombot nem lehet lenyomni.

A Submit gomb megnyomásakor először megpróbálom beszúrni új látogatóként a kapott adatokat, ha a látogató már létezik, akkor egy SQLIntegrityConstraintViolationExceptiont fogok kapni (az ID mező egyediségére vonatkozó kéknyszer sérült) és beszúrás helyett módosítani fogom az adatbázist.

Az elvégzett műveletről felugró ablakban értesítem a felhasználót:

* Új felhasználó bevitele:
  + ID: 13 (még nincs ilyen ID, javításkor használható tesztelésre)
  + Név: Zizi
* Felhasználó módosítása:
  + ID 10111
  + Név: Példa



Az adatbázisban keletkező hibákat felugró ablakkal jelzem, illetve a *Log* panelre kiírom a hibaüzeneteket.

3. feladat

* + 1. Magyarázat

A *Submit* gomb lenyomásakor ellenőrzöm a kapott adatokat:

* ID mező csak számot tartalmazhat
* Telefonszám formátuma
* Bevétel formátuma és értéke

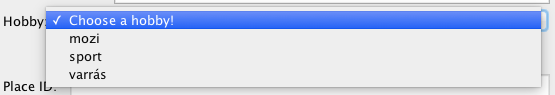
Az ellenőrzést a modell osztály végzi, hogy az alkalmazás osztályban minél kevesebb modellspecifikus adat legyen, ha az ellenőrzés sikerteln, a modell kivételt dob, amely a hiba típusát jelzi, és amelyet az alkalmazás osztály kap el, és megjeleníti egy felugró ablakban a felhasználó számára érthető formában.

Ha a kapott adatok megfelelőek, akkor a megfelelő művelet végrehajtódik, ennek típusáról egy felugró ablakban értesítem a felhasználót.

Ellenőrzéshez használt reguláris kifejezések:

* egy adott kifejezésben csak szám szerepelhet:
  + az alábbi reguláris kifejezést használtam: [0-9]\*
* telefonszám formátumának ellenőrzése:
  + a telefonszám elvárt formátuma: +3630-1234567
  + ellenőrzés: a Magyarországon használatos 3 féle telefonszám valamelyikére kell illeszkednie a kapott adatnak
  + ellenőrzéshez használt reguláris kifejezés: +36(([0-9]{2}-[0-9]{7})|([1-9][0-9]-[0-9]{6})|([0-9]-[0-9]{7}))
  + kifejezés magyarázata:
    - a kifejezésnek kötlezően +36-al kell kezdődnie
    - ezután háromféle ’törzs’ következhet
      * mobilszám formátum: XX-XXXXXXX
      * fővárosi vezetékes formátum: X-XXXXXXX
      * egyéb vezetékes formátum: XX-XXXXXX

A hobby mezőhöz egy JcomboBox-ot hoztam létre, amelyben négy mező található.

Ellenőrzéskor megnézem, hogy melyik elem van kiválasztva, ha az első, akkor úgy veszem, hogy nem adtak meg hobbit, ellenkező esetben pedig a megfeleő sztringet szúrom be az adatbázisba.

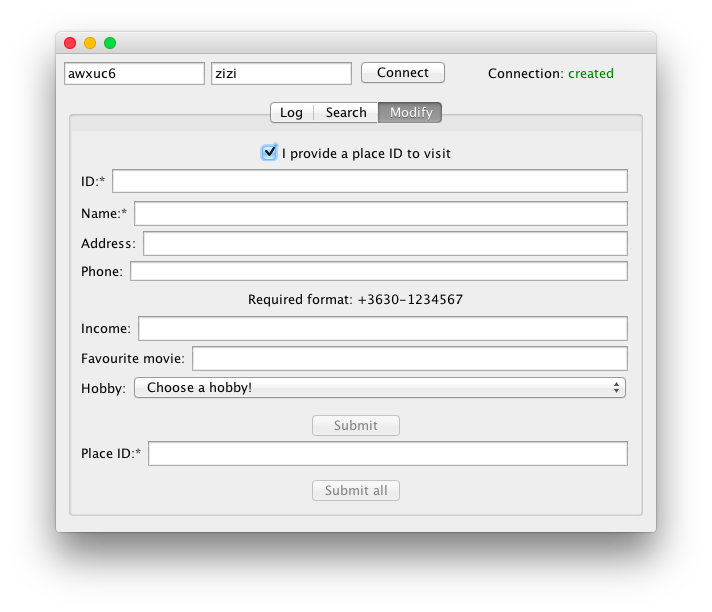
***4. Feladat***

***SQL utasítás:***

**insert into visits (place\_id, person\_id) values (?, ?)**

***Magyarázat***

Ezt a feladatot ugyanabban a fülben oldottam meg, mint az előző kettőt, bővítettem azonban egy checkbox-al, amely ha ki van választva, akkor a felhasználónak előbb be kel írnia a látogató adatait, *Submit*olnia kell, majd meg kell adni a szórakozóhely azonosítóját és a *Submit allr*a kell nyomnia.

A megvalósítást úgy oldottam meg, hogy a a Submit hatására a modellben meghívott függvény paraméterben kap egy bool változót, és ennek értéke alapján kikapcsolhatja az autocommitot. Ezek után a Submit all gomb hatására meghívódó függvény megpróbál létrehozni egy látogatási szokás bejegyzést, ha ez sikerül, komittol, ha nem, rollbacket hív. A függvény végén mindenképp visszaállítom az autoCommitot igazra.

Az ID mező itt sem tartalmazhat számokon kívül más karaktert.

Tesztelés:

- Sikeres beszúrás:

* + ID: 10110
  + Name: Name
  + Place ID: 10011
* Sikertelen beszúrás:
  + ID: 10110
  + Name: Name
  + Place ID: 1