Mérési jegyzőkönyv – Szoftver Laboratórium 5

3. mérés: JDBC

Név:	**Páli Márton**
Neptun kód:	**Z2PWV9**
Feladat kódja:	**28 – REND**
Mérésvezető neve:	**Kiss Máté Levente**
Mérés időpontja:	**2016-04-12 10:15**
Mérés helyszíne:	**HSZK B**
A működő alkalmazás elérhetősége:	**http://rapid.eik.bme.hu/~z2pwv9/jdbc**
Megoldott feladatok:	**1,2,3,4.1,4.2,5**
Elérhető pontszám (plusz pontok nélkül):	**50p**

Felhasználói útmutató

Az általam elkészített alkalmazás egy megadott adatbázis kezelését teszi lehetővé egy grafikus interfész segítségével. Az adatbázisomban 3 tábla van. Az első rendezvények tárolásáért felel, amelyekről tárol egy azonosítót, amellyel megkülönböztethetők, a megrendelő nevét, a rendezvény helyszínét, időpontját, napszakát, vendégszámát, azt, hogy lesz-e zene, azt, hogy mennyiben volt meghatározva a rendezvény költségvetése illetve, hogy igényeltek-e őrzés. A második megrendelhető eszközöket tárol, valamint ezeknek néhány tulajdonságát: egy azonosítót, amellyel megkülönböztethető, az eszköz nevét, márkáját, típusát, napi fenntartási költségét illetve a megvásárlásának időpontját. Az utolsó táblában megrendelések vannak tárolva, vagyis rendezvény-eszköz párosok. A párokon túl tárolva vannak még az alábbi információk: ki felelős a megrendelésért, hogyan lesz az eszköz a rendezvényre szállítva, illetve onnan vissza, valamint lehetőség van megjegyzést fűzni is.

A program a **http://rapid.eik.bme.hu/~z2pwv9/jdbc** címen érhető el. A programba a **z2pwv9** felhasználónév, **almaleves12** jelszó párossal lehet bejelentkezni, ha ez nem történik meg, a parancsok nem hajtódnak végre.

A programba való bejelentkezés után lehetősége van a felhasználónak kilistázni a rendezvényeket („Search” fül) és szűrni a megjelenő adatokat a rendezvény azonosítója alapján. Az „Edit” fül alatt új rekordokat lehet felvenni, illetve már meglévőket módosítani. A „Statistics” fül alatt egy érdekes lekérdezés eredménye lesz látható, ha a felhasználó megnyomja a „statistics” gombot. A „Log” fül alatt a felhasználó interakcióinak hatása olvasható, főként angol nyelven, valamint a program futása során itt jelennek meg a hibaüzenetek is.

Mérési feladatok megoldása

1. feladat

A megoldáshoz használt SQL utasítás

```
--1.
SELECT * FROM rendezvenyek
--2.
SELECT *
FROM rendezvenyek
WHERE rend_azon LIKE ? ESCAPE '@'
```

Magyarázat

A feladat során 3 metódust kellett implementálni.

Először a Model osztály search() metódusával kezdtem, ahol csak az SQL logikát kellett megírni. A fent látható SQL utasításokat használtam, az elsőt akkor, amikor a searchTextField üresen maradt, ekkor ugyani a tábla összes rekordját listázni kellett, a másodikat pedig akkor, amikor meg volt adva a kereséshez feltétel. A paramétert a setString(1, „%” keyword „%”); metódussal regisztráltam, hogy a csonkán megadott feltételt is tudja értelmezni a rendszer. Ahhoz, hogy a speciális karakterekre is rá tudjon keresni a felhasználó, betettem a lekérdezés végére az ESCAPE '@' részt, mivel így hasonló működést érhetünk el, mint pl.: javabab a '\'-el.

Ezután a Controller osztály search() metódusával foglalkoztam, ahol az eredménytáblát kellett egy String-tömb-listává „változtatni”. Ehhez egy while ciklust pörgettem aszerint, hogy az eredménytábla next() metódusa miszerint tér vissza, vagyis annyiszor fusson le, ahány rekord van az eredménytáblában. Ciklusonként annyi volt a dolga, hogy az eredménytábla rekordjait mezőnként egy-egy String változóba tegye, és ezeket rekordonként egy String-tömbbe, majd a ciklus végén ezt a tömböt egy erre a célra kijelölt listához adja. A metódus végén ezzel a listával tér vissza.

Legutoljára hagytam a View osztály eseménykezelő metódusának megírást, mivel ez tűnt a legbonyolultabbnak. Itt annyi feladatom volt, hogy a String-tömb-listámat, amit a Controller visszaad, meg tudjam jeleníteni egy táblában. Ehhez két ciklust pörgettem. A külső felel azért, hogy a lista minden eleme bekerüljön a táblázatba, míg a belső azért, hogy egy-egy listaelem String-tömbjének minden eleme bekerüljön egy-egy oszlopba. A külső ciklusban először létrehoz egy változót, ami képes tárolni egy sornyi rekordot, ezt a belső ciklus fel is tölti, majd a külső ciklus végén hozzáadja a táblához.

2. feladat

A megoldáshoz használt SQL utasítás

```
--1.
SELECT eszk_azon FROM eszkozok WHERE eszk_azon = ?
--2.
INSERT INTO eszkozok VALUES (?, ?, ?, ?, ?, ?)
--3.
UPDATE eszkozok SET nev = ?,
marka = ?,
tipus = ?,
napi_ksg = ?,
vasarlas = ?
WHERE eszk_azon = ?
```

Magyarázat

A feladat megoldásához ismét 3 metódust kellett implementálni, de emellett még a View.fxml-t is módosítani kellett. Először megismerkedtem az fxml szintaktikájával, és elkészítettem a grafikus felületet az edit fülhöz. Ezek után felvettem a View osztályba az új TextField-eket és ComboBox-t. Ezek után nekiláttam az eseménykezelő függvény megírásának. A metódusban létrehozok egy Map<String, String> objektumot, és hozzáadtam a megfelelő értékeket, az útmutatóban megadott kulcsokkal. Ha kész a Map, következhet a Controller.modifyData metódus meghívása (jelenleg az AutoCommit paraméter automatikusan true értékkel adódik át). Fontos megjegyezni, hogy a szintaktikai ellenőrzést nem a View osztály intézi.

A Controller osztály modifyData függvénye annyit csinál, hogy meghívja a Model osztály modifyData metódusát, és annak a visszatérési értéke szerint hozzáadja a log-hoz a kiírandó szöveget, vagyis hogy új elem került felvételre, vagy meglévőt módosított a felhasználó. A Model osztály modifyData metódusában először megnézzük az 1. számú SQL utasítás segítségével, hogy az az azonosító, amelyet a felhasználó megadott, szerepel-e már a táblánkban. A ResultSet, ami tárolja ennek az utasításnak a hatását, vagy 1, vagy 0 sorból áll, tehát a next() metódusa vagy true (ha volt egy sor, vagyis szerepelt már az azonosító korábban) vagy false (ha nem volt sor, vagyis nem szerepelt még az azonosító). Így már könnyedén eldönthető, hogy insert, vagy update műveletet kell használni. A 2-es számú SQL utasítás valósítja meg az új rekord felvételét, a 3-as számú pedig egy már meglévő rekordot frissít. A feladatnál érdekesség még a dátum formázása, mivel a String-et, amit kapunk át kell váltanunk a megfelelő formátumú dátumra, ami csak úgy tehető meg, ha először java.util.Date –re változtatjuk, és onnan java.sql.Date-re.

Mintaadatok (E006 már szerepel; E011 még nem (kieg.: ahhoz, hogy a 4. feladattal működjön az E011-eshez még: REND_AZON: VR001):

ESZK_AZON	E006	E011
NEV	Penész	Alma
MARKA	Büdös	Apple
TIPUS	latvany	szorak
NAPI_KSG	589222	500000
VASARLAS	2009-02-02	2016-04-13

2. feladat
Magyarázat

A 3. feladatot (a dokumentum sorszámozása valamiért megviccelt, de ez a 3. feladat magyarázata) egy metódus megírásával és egy kiegészítésével oldottam meg. A Controller osztály verifyData metódusát kellett implementálni, teljes egészében. Azt kellett megvalósítani, hogy számot elfogadó mező esetén figyelmeztessen, hogyha karaktert próbált begépelni a felhasználó, a fix értékes mezőket legördülő menüvel oldjam meg, valamint hogy a dátum formátumára csak azt fogadja el, ami ÉÉÉÉ-HH-NN formátumú. Ehhez kellett használni a reguláris kifejezéseket.

A szám elfogadására az alábbi reguláris kifejezést használtam:

„[1-9]\d*” (java-ban a „\” helyett „\\” kellett hozzá) Ez megszabja, hogy egy 1-9-ig levő számjeggyel kell kezdődnie a számnak (ha 0-val kezdődne, annak sok értelme nem lenne) valamint ezután csak decimális karaktereket fogad el, viszont azokból bármennyit.

A dátum ellenőrzésére pedig ezt a reguláris kifejezést használtam:

„[1-9]\d{3}\-[0-1]\d\-[0-3]\d” Ez megszabja, hogy először 4 decimális számjegyet vár, amelyek közül az első nem nulla, majd egy kötőjelet, majd 2 decimális számjegyet, amelyek közül az első nem nagyobb mint 1, majd egy újabb kötőjelet, végül pedig ismét 2 decimális számjegyet, amelyek közül az első nem nagyobb mint 3. Ezek a megkötések azért kellenek, mivel az évszám nem kezdődhet 0-val, csak 12 hónapunk van, tehát értelmetlen lenne 20> számokat várni, illetve egy hónap maximum 31 napból állhat, ismét a korábbi érvvel tudok élni.

A legördülő menüt már a korábbi feladatban megoldottam, fix érték a tipus attribútum lehet, ezt az View.fxml-ben kellett felvennem.

Miután megvoltak a reguláris kifejezéseim, már csak össze kellett őket hasonlítani a konkrét String-ekkel, amelyeket vizsgálni akartam. Ha nem egyezik meg a formátum, akkor a felhasználót erről értesítem, és false-szal tér vissza a verifyData, ha pedig nem történt probléma, akkor true-val. A modifyData metódust ennek megfelelően kellett kiegészíteni, vagyis csak akkor forduljon a modellhez, ha már leellenőriztük a formátumot, és jót kaptunk.

3. feladat
A megoldáshoz használt SQL utasítás

```
--1.
SELECT rend_azon FROM rendezvenyek WHERE rend_azon LIKE ?
--2.
INSERT INTO foglalas VALUES (?, ?, 'Herceg', 'busz', 'busz', Null)
```

Magyarázat

A feladat megoldásához először kicsit variáltam az View.fxml-t, felvettem az Edit fülre egy új labelt, és egy új TextFieldet, valamint a Commit gombot engedélyeztettem. Ezek után a TextFieldet felvettem a View osztályba is, illetve itt még módosítottam az Edit gomb eseménykezelő függvényét azzal, hogy a Map-be felvettem még a rend_azonTextField értékét is és a Controller osztály modifyData függvényének AutoCommit paraméterét false-ra állítottam. A View osztályban még annyit módosítottam, hogy megírtam a Commit gomb eseménykezelő metódusát. Ez a függvény annyit csinál, hogy meghívja a Controller osztály commit() metódusát és annak a visszatérési értéke szerint ad hozzá új Stringet a log-hoz.

A Controller osztályban a commit() metódust kellett csak implementálni, ami annyit csinál, hogy meghívja a

Model.commit() metódust, és annak visszatérési értéke szerint cselekszik: ha true, akkor ő is visszatér true-val, viszont ha false, azaz nem sikerült a commitolás, akkor meghívja a Model.rollback() függvényt, és logolja a rollback-elést, valamint visszatér false-szal.

A Model osztályban már lényegesebb változtatásokra volt szükség. Bekerült egy új tagváltozó, amely azért felel, hogy tárolja a modifyData közben történő hibákat (ha volt hiba: false, ha nem:true), és az értéke alapján a commit() metódust segítse.

Módosítanom kellett a modifyData metódust, mivel most már insert műveletnél a frissen felvett rekordot ki is kölcsönözzük rögtön. Ahhoz hogy ez megtehető legyen, fel kell vennünk egy új foglalást, amihez 5 mező nemnull értéke kell. Ezek: rend_azon (a felhasználótól kérjük be), eszk_azon (a felhasználótól kérjük be), szallitas_oda (default értéke van), szallitas_vissza (default értéke van), felelos (legyen a friss foglalások felelőse minden esetben 'Herceg'). Ezek alapján már meg tudtam írni az új foglalás felvételét intéző SQL utasítást, ami fönt a 2. sorszámmal van jelölve. Az 1. sorszámú SQL lekérdezés ahhoz kell, hogy el tudjuk dönteni egy, a felhasználó által megadott rend_azon-ról, hogy valóban szerepel-e már. Ha ennek a lekérdezésnek a visszatérési értékének (ami egy ResultSet) a next() metódusa true-val tér vissza, akkor tényleg volt ilyen rend_azon, egyébként pedig még nem. Ahhoz hogy a felhasználó könnyedén rájöhessen a hibájára, a log-ba ilyenkor bekerül egy üzenet, ami megmondja, hogy ilyen rend_azon még nincs az adatbázisban. Ha a felhasználó egyáltalán nem adott meg rend_azon-t, akkor pedig olyan üzenet kerül a log-ba, ami erre figyelmezteti. A metódusba továbbá bekerült minden olyan helyhez, ahol hiba történt (pl.: SQLException, rend_azon mező üres stb.) a commitready tagváltozó false-ra állítása. Ha eljutottunk a függvény végéig és nem Model.ModifyResult.Error-ral térünk vissza, akkor commitready = true.

Implementálnom kellett továbbá a commit metódusát az osztálynak. Ennek a feladata, hogy meghívja a connection.commit()-ot. Ez viszont csak akkor történhet meg, hogyha a commitready tagváltozó true értékű, mert különben olyan módosítást commitálnánk, ami hibás. Így ennek ellenőrzése elengedhetetlen. A commit metódus igazzal tér vissza, hogy sikeres volt a connection.commit() és commitready == true. Az osztály rollback() függvényét is implementáltam, ez viszonylag egyszerű volt, csak meg kellett hívni a connection.rollback() metódust.

Mintaadatok (E012 sikeres; E013 sikertelen (VR100 jelű rendezvény nem létezik)):

ESZK_AZON	E012	E013
NEV	MP3 lejátszó	Flaska
MARKA	Sony	Sprite
TIPUS	hang	szorak
NAPI_KSG	100000	90000
VASARLAS	2010-12-12	2016-03-12
REND_AZON	VR005	VR100

4.2: Ha a Commit gomb lenyomása nélkül, vagy kapcsolatmegszakadás miatt nem sikerül commitolni, és bezárul a program, akkor azok az adatok, amelyek már módosítva voltak, és commitra vártak, elvesznek. Ez igen nagy kényelmetlenséget okozhat. Emiatt érdemes az ilyen lehetőségekre felkészülni azáltal, hogy a program bezárásakor automatikusan commitolunk. Ehhez a View osztály initData metódusát kell módosítani, azon belül is a handle függvényt. Ennek annyit kell csinálnia, hogy bezárásnál meghívja a controller.commit() metódust. Ezáltal annyi kiegészítést kell tennem, hogyha valaki úgy próbálja meg bezárni az alkalmazást, hogy nem connect-elt, akkor Exception-t fog dobni a program.

2. feladat

A megoldáshoz használt SQL utasítás

```
SELECT DISTINCT a.megrendelo, b. megrendelo
FROM rendezvenyek a, rendezvenyek b, eszkozok, foglalas c, foglalas d
WHERE a.rend_azon = c.hova AND c.mit = d.mit
AND c.mit = eszkozok.ESZK_AZON  AND eszkozok.nev = 'Peacekép'
AND c.hova != d.hova AND c.hova = a.rend_azon AND d.hova = b.rend_azon
```

Magyarázat

A feladat megoldásához 3 metódust kellett módosítani, először a View-ban módosítottam a Statistics fül oszlopait, hogy csak egy legyen, és annak megadtam a fejlécét. Ezek után a gomb eseménykezelő metódusát kellett implementálni, ami nagyban hasonlított az első feladatéhoz, ugyanúgy is valósítottam meg, annyi különbséggel, hogy a statisticsTable-t és a hozzá tartozó tagváltozókat használtam, illetve a Controller osztály getStatistics metódusát hívtam meg.

A Controller osztlyban is nagyon hasonlóra sikerült a getStatistics metódus, mint az első feladat ehhez kapcsolódó metódusa. Itt, mivel csak 1 oszlopot fogunk kilistázni a String[ ] egy elemű lesz, a ResultSet meghatározásához pedig a Model.getStatistics metódust kellett meghívni.

A Model feladata volt az SQL lekérdezés meghívása, és az eredménytábla meghatározása. Ehhez a fenti lekérdezést használtam. Ahhoz, hogy mindez hatékonyan működjön, nem használtam halmazműveleteket, csak illesztéseket, valamint nem használtam a megoldáshoz beágyazott lekérdezést. Azt, hogy minden pár csak egyszer jelenjen meg, nem sikerült megoldnaom.

Vélemény(ek) a mérésről

Vélemény, építő jellegű kritika.