2.iterácia

Nerelačná databáza

Alexander Pajkoš

Cvičiaci: Ing. Ondrej Kachman

Študijný odbor: Informatika

Ročník: 2. Bc

Akademický rok: 2015/2016

Zadanie:

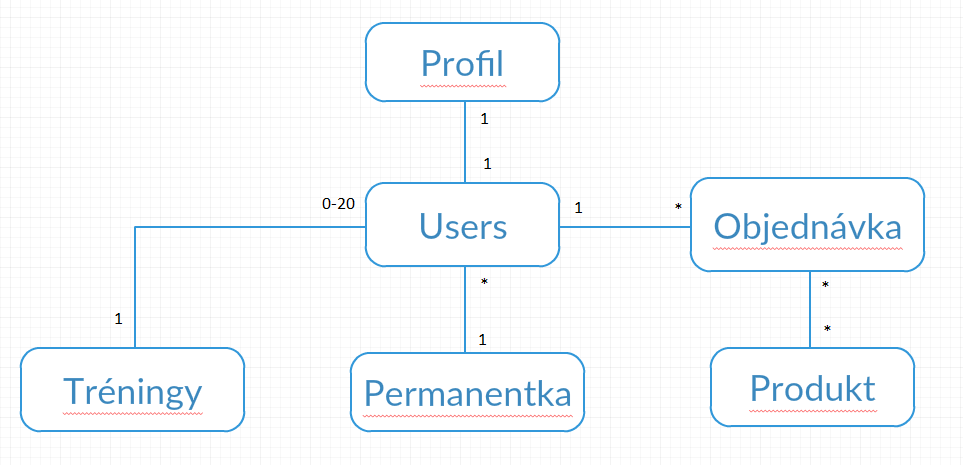
Vo vami zvolenom prostredí vytvorte databázovú aplikáciu, ktorá komplexne rieši minimálne 6 scenárov vo vami zvolenej doméne. Presný rozsah a konkretizáciu scenárov si dohodnete s Vašim cvičiacim na cvičení. Aplikáciu vytvoríte v dvoch iteráciach. V prvej iterácii, postavenej nad relačnou databázou, musí aplikácia realizovať tieto všeobecné scenáre:

* Vytvorenie nového záznamu,
* Aktualizácia existujúceho záznamu,
* Vymazanie záznamu,
* Zobrazenie prehľadu viacerých záznamov (spolu vybranou základnou štatistikou),
* Zobrazenie konkrétneho záznamu,
* Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.

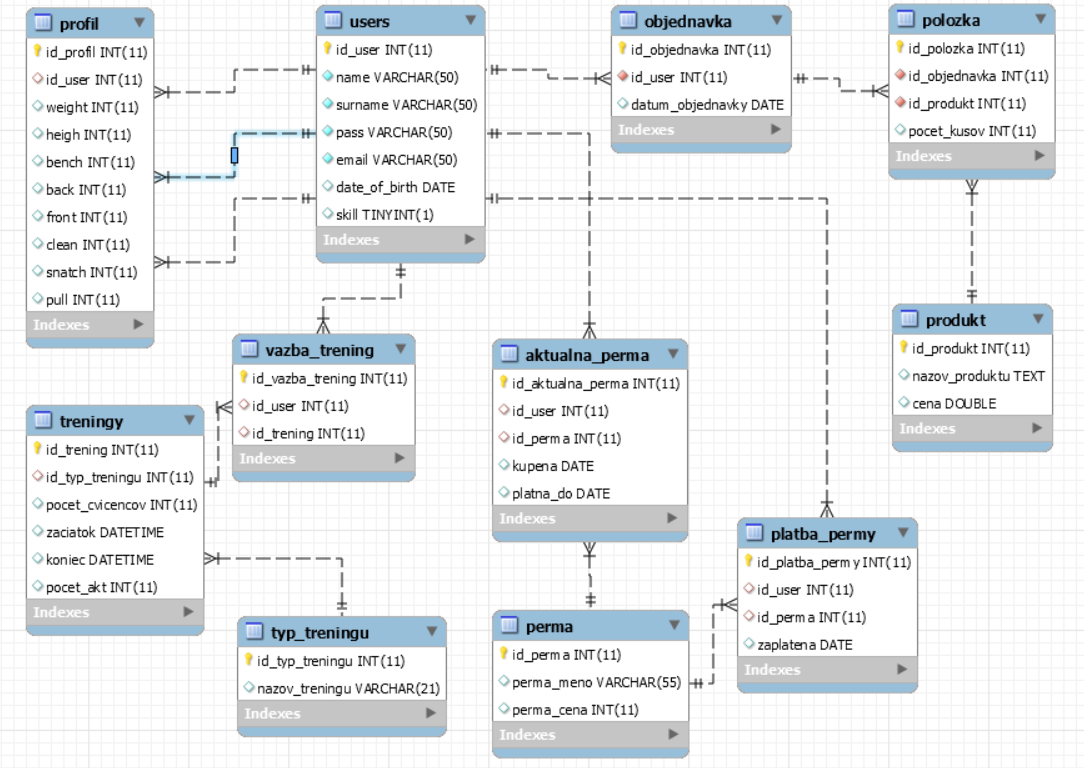
Riešenie zadania

Pre splnenie zadania bolo potrebné vymyslieť si oblasť, v ktorej budeme daný problém riešiť. Mnou zvolená oblasť je vytvorenie aplikácie slúžiacej pre členov konkrétneho fitness centra. Táto aplikácia poskytuje používateľom základne potreby ako registrácia v danom centre, objednanie a zakúpenie permanentiek (týždňová, mesačná... ), uloženie osobných údajov (ako napríklad váha, výška...), prihlásenie sa na konkrétny tréning ako aj možnosť zakúpenia produktov v danom fitness centre. Problematiku som riešil vo vývojárskom prostredí Java za podpory databázového programu MySQL. Zadávanie konkrétnych požiadaviek je riešené pomocou používateľského rozhrania nie cez konzolu. Aplikácia je navrhnutá výlučne pre používateľov, nie pre administrátorov. Riešenie druhej iterácie je implementované pomocou nerelačnej databázy REDIS. Túto databázu budem využívať pri funkcionalite nákupného košíka kde používateľ cez grafické rozhranie zadáva produkty do košíka a nasledne si ich môže zobraziť a neskôr objednať.

Návrh logického dátového modelu



Návrh fyzického dátového modelu



Jednotlivé scenáre

Po spustení aplikácie má používateľ k dispozícii tieto scenáre.

Registracia

String query = "insert into users(name, surname, pass, email) values (?, ?, ?, ?)";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

pst.setString(1, menoField.getText());

pst.setString(2, priezviskoField.getText());

pst.setString(3, passwdField.getText());

pst.setString(4, emailAdressField.getText());

pst.setString(5, ageField.getText());

pst.execute();

Keď ešte používateľ do systému nie je prihlásený tak sa má možnosť registrovať na úvodnom okne hneď napravo od prihlasovacích možností. Po vyplnení formulára a stlačení tlačidla registruj sa jeho údaje uložia do tabuľky všetkých userov.

Prihlásenie

String query = "select \* from users where email = ? and pass = ?";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

pst.setString(1, emailField.getText());

pst.setString(2, passwordField.~~getText~~());

Ak už je používateľ registrovaný tak sa automaticky prepne do okna kde mu bude mať možnosť prihlásiť sa do systému zadaním svojho emailu a prihlasovacieho hesla.

Profil

String query = "select weight, heigh, bench, back, front, clean, snatch, pull from profil where id\_user = 1 ";

Po prihlásení môže používateľ skontrolovať svoje osobné vlastnosti a schopnosti taktiež si môže pozrieť všetkých používateľ fitness centra ktorí dané centrum navštevujú. Ďalej si používateľ môže pozrieť všetkých návštevníkov ktorí majú výšku väčšiu ako je priemerná výška všetkých návštevníkov.

String query = "select name, surname, heigh from users u join profil p on u.id\_user = p.id\_user group by name having heigh > (select avg(heigh) from profil)";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

tableProfil.setModel(DbUtils.*resultSetToTableModel*(rs));

Zobrazenie aktuálnej permanentky používateľa

Používateľ si môže pozrieť akú má momentálne k dispozícii permanentku (ak nejakú má) zobrazí sa mu typ permanentky a platnosť

String query = "select name, surname, kupena, platna\_do, perma\_meno from users u join aktualna\_perma a on u.id\_user = a.id\_user join perma p on a.id\_perma = p.id\_perma where u.id\_user = '"+lblIdUser.getText()+"' ";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

tableProfil.setModel(DbUtils.*resultSetToTableModel*(rs));

Vytvorenie profilu

Ak užívateľ ešte tento profil s údajmi nemá vytvorení tak si ho môže vytvoriť v ďalšom okne poprípade si môže už vytvorené údaje aktualizovať.

String query = "update profil set weight = '"+textWeight.getText()+"', heigh = '"+textHeigh.getText()+"', bench = '"+textBench.getText()+"', back = '"+textBack.getText()+"', front = '"+textFront.getText()+"', clean = '"+textClean.getText()+"', snatch = '"+textSnatch.getText()+"', pull = '"+textPull.getText()+"' where id\_user = '"+lblIdUser.getText()+"'";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

História permanentiek

V tomto okne si môže používateľ pozrieť históriu kupovania permanentiek

**try**{

String query = "select name, surname, perma\_meno, zaplatena from users u join platba\_permy p on u.id\_user = p.id\_user join perma pe on p.id\_perma = pe.id\_perma where u.id\_user = '"+lblIdUser.getText()+"'";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

tableHistory\_perm.setModel(DbUtils.*resultSetToTableModel*(rs));

rs.close();

pst.close();

}**catch**(Exception ex){

ex.printStackTrace();

}

Počet celkovo doteraz zakúpených permanentiek

String query = "select name, surname, count(perma\_meno) from users u join platba\_permy p on u.id\_user = p.id\_user join perma pe on p.id\_perma = pe.id\_perma where u.id\_user = '"+lblIdUser.getText()+"'";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

tableHistory\_perm.setModel(DbUtils.*resultSetToTableModel*(rs));

rs.close();

pst.close();

Priemerná cena tovaru za rok 2015

String query = "select name, surname, avg(cena) from users u join objednavka o on u.id\_user = o.id\_user join polozka p on o.id\_objednavka = p.id\_objednavka join produkt pr on p.id\_produkt = pr.id\_produkt where extract(year from datum\_objednavky) = 2015 group by name";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

ResultSet rs = pst.executeQuery();

tableHistory\_shop.setModel(DbUtils.*resultSetToTableModel*(rs));

rs.close();

pst.close();

Funkcionalita nerelačnej databázy

Scenár v nerelačnej databáze má za úlohu používateľovi umožniť pridávanie tovaru do nákupného košíka jednoduchým klikom a následne mu je umožnené nahliadnuť do tohto košíka a zobraziť počet za produktov v danom košíku ako aj zobrazenie celkovej ceny tovaru ktorý sa chystá objednať a následne pridanie objednávky do tabuľky objednávok.

Pridanie polozky:

btnShirt1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

jedis.lpush(cacheKey, "Nike shirt sport");

jedis.lpush(cacheCena, "78");

jedis.incr(cachePocet);

}

});

Celkovy pocet produktov:

btnZobrazKS.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

textShoppingCard.setText("Pocet poloziek v kosiku je: " + jedis.get(cachePocet).toString()); /// + jedis.zcard(cacheKey).toString());

}

});

Celkova cena produktov:

btnCenaTovaru.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

jedis.sort(cacheCena);

priemerPolozky = cenaSpolu / pocet;

textShoppingCard.setText("Cena vsetkych poloziek v kosiku je: " + cenaSpolu + " Eur\nPriemerna cena polozky v nakupnom kosiku je: " + priemerPolozky + " Eur");

//System.out.println("cena je:" + cenaSpolu);

}

Objednanie:

btnOrder.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try**{

String query = "insert objednavka (id\_user, datum\_objednavky) values ('"+lblIdUser.getText()+"', '2016-05-05')";

java.sql.PreparedStatement pst = conn.prepareStatement(query);

pst.execute();

pst.close();

}**catch**(Exception ex){

ex.printStackTrace();

}

jedis.del(cacheKey);

jedis.del(cacheCena);

jedis.del(cachePocet);

}

});

Zhrnutie

Na záver by som zhrnul, že aplikácia spĺňa základné požiadavky pre používateľov. V druhej iterácii sa mi podarilo dorobiť funkcionalitu e-shopu takže zákazník teraz môže objednávať ľubovoľné množstvo tovaru a všetko bude uchovávané.