Vysoké učení technické v Brně

BPC2T Počítače a programování

Práce s databází v Javě pomocí JDBC

Jiří Přinosil a spol.

12. října 2017

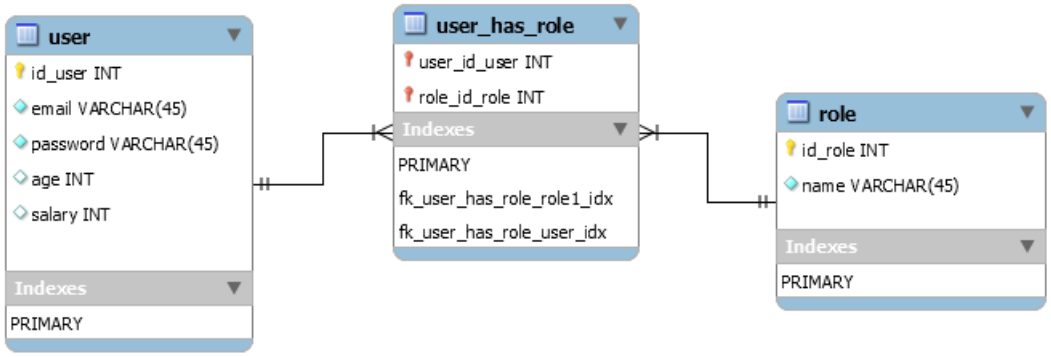
Úkoly:

Toto cvičení je určené na procvičení práce s databází v jazyku Java pomocí JDBC (Java Database Connectivity). Úkoly spočívají v doplňování základních příkazů CRUD (Create, Read, Update, Delete) pro manipulaci s daty v databázi a implementaci komplexní metody pro vrácení požadovaných dat z databáze.

Připojení k databázi je v projektu implementováno ve třídě `cz.vutbr.feec.dbconnection.dbconn.DBConnection` pomocí návrhového vzoru Singleton, který zabezpečuje vytvoření maximálně 1 instance dané třídy (není vhodné vytvářet velké množství připojení k databázi), lepším řešením než návrhový vzor Singleton pro připojení do databáze by bylo využití tzv. Dependency Injection, to je ovšem problematika mimo rozsah tohoto kurzu.

Na obrázku 1 je vidět návrh databáze na základě, kterého budete implementovat Vaše SQL (Structured Query Language) dotazy nad touto databází.

Návrh databáze:



Obrázek 1: Návrh databáze uživatelů a jejich rolí.

Přejděte do Vašeho vývojového prostředí a jděte do třídy `RunApp`. V main metodě je představeno 8 možností pomocí kterých budete volat patřičné metody, kde některé bude třeba implementovat.

1. Podívejte se na volbu 1 a prozkoumejte formát příkazu `INSERT`.
2. Implementujte metodu volby 2 s předpisem `insertNewUser(String email, String name, String surname)`, tak abyste využili

`PreparedStatement`[1]. Stačí tedy vhodně doplnit proměnnou `insertUser` o SQL příkaz `INSERT`.

3. Implementujte metodu `getAllUserEmailAndNameAndSurname()` ve volbě 3, tak že vhodně doplníte proměnnou `selectAlluserEmailNameAndSurname` o příkaz SQL `SELECT`, kterým vrátíte z databáze údaje o uživateli jako: email, name, surname.

4. Projděte si metodu `getAllUsersWithRoleUser` a podívejte se do přednášek, jak funguje SQL JOIN a rozmyslete si, jak se liší oproti kartézskému součinu.

5. Doplňte v metodě `increase20PercentOfSalary(String email)`, proměnnou `update20PercentOfSalary` o SQL příkaz UPDATE, tak aby se salary (mzda) uživatele zvýšila o 20%

6. Implementujte metodu `deleteUserByEmail(String email)` doplněním proměnné `userToDelete`, kde napíšete SQL příkaz `DELETE`, tak že vymažete uživatele dle zadaného emailu.

7. Implementujte všechny kroky v metodě `printAllRolesInDB()` nezbytné pro manipulaci s daty v databázi:

   - Získejte connection k databázi,
   - vytvořte SQL příkaz k získání všech rolí v DB,
   - vložte tento příkaz jako prepareStatement k získané connection k databázi,
   - vyvolejte tento `PreparedStatement`,
   - uložte výsledek do `ResultSetu`,
   - přes cyklus projděte výsledek `ResultSetu` a vypište role v systému (pro získání správné kolonky se podívejte na návrh databáze, abyste znali název sloupce (popřípadě lze ještě získat data ze sloupce pořadovým číslem (začínajíc od 1)))
   - tyto operace obalte v `try-catch` bloku popřípadě v `try-with-resources` bloku.

8. Ukončete běh aplikace volbou 8.

[1]Pomocí PreparedStatements předkompilováváme SQL příkazy, čímž obvykle zvýšíme rychlost vícenásobného dotazování a rovněž zabezpečíme SQL příkazy proti útokům typu `SQL INJECTION`. Pro více informací o PreparedStatements si pročtěte následující stránku: https://en.wikipedia.org/wiki/Prepared_statement