Universität Leipzig, Institut für Informatik
Abteilung Datenbanken
Prof. Dr. E. Rahm,
V. Christen, M. Franke

Ausgabe: 30.04.2018
Besprechungen: 07.05.2018 (Mo)
08.05.2018 (Di)
14.05.2018 (Mo)
15.05.2018 (Di)

Datenbanksysteme II

SS 2018 – Übungsblatt 2

1. Stored Procedure

Es sei folgendes Datenbankschema zum Thema „Soziales Netzwerk“ gegeben.

```
Nutzer (
  uid          INT NOT NULL PRIMARY KEY,
  email        VARCHAR(50),
  name         VARCHAR(100)
);

Freundschaft (
  uid1         INT NOT NULL PRIMARY KEY,
  uid2         INT NOT NULL PRIMARY KEY,
  Fgrad        VARCHAR(100),
  FOREIGN KEY uid1 REFERENCES Nutzer (uid),
  FOREIGN KEY uid2 REFERENCES Nutzer (uid)
);
```

a) Erstellen Sie eine SQL Stored Procedure, welche den Freundschaftsgrad zweier Benutzer eines sozialen Netzwerks aktualisiert. Dazu sollen die Parameter *uid1, uid2* und *Fgrad* übergeben werden.

b) Erweitern Sie die obige Lösung derart, dass anstelle der Nutzer-IDs die Benutzernamen (*name1, name2*) als Eingabeparameter dienen, d.h. die Nutzer-IDs müssen auf Basis der Namen ermittelt werden. Nehmen Sie an, dass Nutzer eindeutig durch ihren Benutzernamen gekennzeichnet werden.

c) Skizzieren Sie den Aufruf dieser Stored Procedure aus einem Java-Programm mittels JDBC.

2. SQL-Injection

Es sei folgendes Szenario bzgl. des oben definierten Schemas gegeben.

Eine Webapplikation bietet Ihnen die Möglichkeit unter Eingabe der uid oder des Namens die Daten eines Benutzers zu ermitteln, dabei werden die Parameter an einen Tomcat-Server(Java) übermittelt.

a) Welche Anfragen sind bzgl. SQL- Injections kritisch zu bewerten, wenn Sie den Parameter durch das Ersetzen eines Patterns, z.B. $$$, an die Anfrage binden. Begründen Sie Ihre Antwort und definieren Sie ein mögliches Angriffsszenario.

i) **Parameter: uid=7**

„SELECT n.uid, n.name FROM Nutzer n

WHERE n.uid=$$$“

ii) **Parameter: name=Hans**

„SELECT n.uid, n.name FROM Nutzer n

WHERE n.name=$$$“

b) Wie können Sie das Problem generell vermeiden?

3. PHP (Praktische Aufgabe)

Nachfolgende Teilaufgaben können praktisch nachvollzogen werden, indem Sie die Skripte herunterladen: https://dbs.uni-leipzig.de/file/friends.zip

a) Vervollständigen Sie das Code-Skelett `index.html` eines HTML-Formulars welches die Eingabe einer Nutzer-ID ermöglicht. Nach dem Abschicken des Formulars soll serverseitig das PHP-Skript `listFriends.php` ausgeführt werden.

b) Innerhalb des Skriptes sollen mittels eines Prepared Statements (PDO) alle Freundschaftsbeziehungen des übergeben Nutzers ermittelt und tabellarisch ausgegeben werden. Skizieren sie das PHP-Skript `listFriends.php`. Verwenden Sie hierzu das Datenbankschema aus Aufgabe 2 des ersten Übungsblatts.

Hinweis:

Um ihr Skript zu testen, können Sie die VM aus der praktischen Übung nutzen. Kopieren Sie diesbezüglich den entpackten Ordner `friends` mit den enthaltenen Skripten in den Ordner `/var/www/`

mittels `sudo mv /path/to/friends /var/www/ (sudo pw: dbs2)`

Im Browser können Sie mittels `http://localhost/friends/` die Applikation aufrufen.

4. JDBC vs. PHP

a) Vergleichen sie die Verwendung von Prepared Statements in PHP und in JDBC durch die Identifikation analoger Funktionen. Verwenden sie dabei für PHP das *PDO*-Modul

b) Was ist der Vorteil von DB-Modulen wie *PDO* gegenüber der Nutzung datenbanksystemspezifischer Module wie *mysqli*?

Hinweis: Im Rahmen der praktischen Übungen werden weitere Übungsaufgaben zu den Themen PHP und JDBC zur selbstständigen Bearbeitung angeboten.