1a) SQL Stored Procedure via IDs

Erstellen Sie analog zu Aufgabe 2d) des ersten Übungsblatts eine SQL Stored Procedure, welche den Freundschaftsgrad zweier Benutzer eines sozialen Netzwerks aktualisiert.Dazu sollen die Parameter uid1, uid2 und Fgrad übergeben werden

```
DELIMITER //

CREATE PROCEDURE updateFgrad
(IN p_uid1 INT, IN p_uid2 INT, IN p_Fgrad varchar(100))LANGUAGE SQL
BEGIN
        UPDATE Freundschaft SET Fgrad= p_Fgrad
        WHERE (uid1= p_uid1 AND uid2= p_uid2)
                OR (uid1= p_uid2 AND uid2= p_uid1);
END;
//

DELIMITER ;
```

1b) SQL Stored Procedure via Name

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.

```
DELIMITER //
CREATE PROCEDURE updateFgradViaName
(IN name1 VARCHAR(100), IN name2 VARCHAR(100), IN Fgrad
VARCHAR(100)) LANGUAGE  SQL
BEGIN
        DECLARE id1 INT;
        DECLARE id2 INT;
        SELECT uid INTO id1
        FROM Nutzer
        WHERE name= name1;
        SELECT uid INTO id2
        From Nutzer
        WHERE name = name2;
        CALL updateFgrad(id1, id2, Fgrad);
END;//
DELIMITER ;
```

1c) Stored Procedure aufrufen – JDBC

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

```
public void doIt(Connection con, String name1, String name2, String
Fgrad) {
  CallableStatement cs = con.prepareCall(
       "CALL updateFgradViaName(?,?,?)"
  );
  cs.setString(1, name1);
  cs.setString(2, name2);
  cs.setString(3, Fgrad);
  cs.execute();
  cs.close();
};
```

2a) SQL-Injection

Es sei folgendes Szenario zum Thema „Soziales Netzwerk“ gegeben.
Eine Webapplikation bietet Ihnen die Möglichkeit unter Eingabe der uid oder des Namens die Daten eines Benutzers zu ermitteln. Die notwendigen Parameter werden mittels Http-GET Request 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=$$$
```

Lösung

- Tabellenbezogene Datenidentifikation
  - `SELECT n.uid, n.name FROM Nutzer n WHERE n.uid=20 or 1=1`
- Datenbankmanipulation
  - Wenn Wirtssprache mehrere Anfragen in einem Statement erlaubt

```
SELECT n.uid, n.name FROM Nutzer n WHERE n.uid=20; DROP Table …;
DELETE FROM Nutzer where …;
```

2a-b) SQL-Injection

ii. Parameter: Name=Hans

```
SELECT n.uid, n.name  FROM Nutzer n  WHERE n.name='$$$'
```

Lösung

```
SELECT n.uid, n.name  FROM Nutzer n  WHERE n.name= ''or ''=''
```

b) Wie können Sie das Problem generell vermeiden, wenn Ihre Middleware in Java implementiert ist?

Lösung

- PreparedStatements
  - Senden des Statements an DBS→ Transformation in SQL Anfrage mit Platzhaltern und Speichern im Cache

```
PreparedStatement pstmt = con.prepareStatement(Select n.uid, n.name from
Nutzer n where n.name=?);
```

  - Binden der Parmeter an Platzhalter

```
pstmt.setInt(1,7); oder pstmt.setString (1,'hans')
```

  - Ausführung & Senden der Parameter

```
ResultSet rs = pstmt.executeQuery();
```

3a) HTML-Formular

Skizzieren Sie das Code-Skelett eines HTML-Formulares welches die Eingabe einer Nutzer-ID ermöglicht. Nach dem Abschicken des Formulars soll serverseitig das PHP-Skript listFriends.php ausgeführt werden.

```
<html>
<head>
        <title>HTML Example</title>
</head>
<body>
        <form method="post" action="listFriends.php">
                <input type="text" name="uid" maxlength="30"/>
                <input type="submit" name="Mache Los!"/>
        </form>
</body>
</html>
```

3b) Freundschaftsbeziehungen auflisten

```
<html>
<head>
   <title>PHP Example</title>
</head>
<body>
<?php
   $userName = "root";
   $pw = "****";
   $host = "localhost";
   $dbName = "stud_employee";

   // Datenbankverbindung herstellen
   $con = new PDO("mysql:host=$host;dbname=".$dbName, $userName, $pw)
        or DIE ("<p><Verbindungsfehler!</p></body></html>");

   // Fehlerbehandlung setzen (bei Datenbankfehlern abbrechen)
   $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

   //SQL Statement
   $sql= “SELECT n1.name AS name1, n2.name AS name2, f.Fgrad “;
   $sql.=“FROM Freundschaft AS f, Nutzer AS n1, Nutzer AS n2 “;
   $sql.=“WHERE (f.uid1=? OR f.uid2=?) AND n1.uid= f.uid1 AND n2.uid= f.uid2“;
?>
```

3b) Freundschaftsbeziehungen auflisten (Forts.)

```
<table>
   <tr><th>Nutzer 1</th><th>Nutzer 2</th><th>Freundschaftsgrad</th></tr>
<?php
   $stmt = $con->prepare($sql);
   $stmt->execute(array($_POST[“uid“], $_POST[“uid“]));
   $result= $stmt->fetchAll(PDO::FETCH_ASSOC);

   if(!empty($result))
   {
      foreach($result as $row)
      {
         echo “<tr>“;
         echo “<td>“.$row[“name1“].“</td>“;
         echo “<td>“.$row[“name2“].“</td>“;
         echo “<td>“.$row[“Fgrad“].“</td>“;
         echo “</tr>“;
      }
    }
?>
</table>
</body>
</html>
```

4a) Prepared Statements in JDBC / PDO

Update:

```
Connection con = DriverManager.getConnection(url, user, pw);

String sql= "UPDATE Buch SET verlagsid = ? WHERE verlagsid = ?;
PreparedStatement ps = con.prepareStatement(sql);

ps.setInt(1, verlagsIdNeu);
ps.setInt(2, verlagsIdAlt);

ps.executeUpdate();
con.commit();
```

```
$con = new PDO($url, $userName, $pw)
$sql = "UPDATE Buch SET verlagsid = ? WHERE verlagsid = ?";

$stmt = $con->prepare($sql);

$stmt->execute(array($verlagsIdNeu, $verlagsIdAlt));
$stmt->commit();
```

4a) Prepared Statements in JDBC / PDO

Select:

```
Connection con = DriverManager.getConnection(url, user, pw);

String sql= "SELECT title FROM Buch WHERE verlagsid = ?";
PreparedStatement ps = con.prepareStatement(sql);

ps.setInt(1, verlagsId);

ResultSet rs= ps.executeQuery();
while(rs.next()) {
   System.out.println("Name: "+rs.getString(1));
}
```

```
$con = new PDO($url, $userName, $pw)
$sql = "SELECT title FROM Buch WHERE verlagsid = ?";

$stmt = $con->prepare($sql);
$stmt->execute(array($verlagId));

$result= $stmt->fetchAll();
foreach($result as $row) {
   echo("Name: ".$row["title"]."\n");
}
```

4b) Mysqli vs. PDO

- DB-spezifische Module
  - MySQL, Postgres, DB2, …
  - Funktionen / API abgestimmt auf Funktionalität d. DBS (z.B. Autoincrement vs. Sequences)

- Bibliotheken für DBS-unabhängigen Zugriff
  - Beispiele: PDO, PEAR::DB, ...
  - Einheitliche Schnittstelle für unterschiedliche DBS
  - Vorteil: DBS kann ausgetauscht werden, ohne Code anzupassen