Datenbanken und Informationssysteme
Übungen Stored Procedure

Hinweis:
Die folgenden Aufgaben basieren auf der Datenbank ‚sqlteacherdb'.

1. Aufgabe

Erstellen Sie eine Stored Procedure welcher Sie einen String übergeben können und die dann „Hello <ihr String>" in der Spalte ‚Greeting' ausgibt.

Testen Sie Ihre Kreation.

2. Aufgabe

Erstellen Sie eine Stored Procedure ‚accu' welcher Sie einen numerischen Wert übergeben können. Diesen übergebenen Wert addiert die Prozedur zu einem Session-Globalen Wert @accu hinzu und speichert das Ergebnis wiederum in @accu.

Fügen Sie einen zweiten Parameter hinzu in welchem Sie das Ergebnis der Operation speichern können.

Testen Sie Ihre Kreation indem sie ‚accu' aufrufen und den Ergebnisparameter anschliessend ausgeben.

3. Aufgabe

Erstellen Sie eine Stored Procedure welche einen neuen Mitarbeiter anlegt. Die Stored-Procedure soll dabei Mitarbeiternummer, Name, Vorname und Abteilung des Mitarbeiters als String annehmen.

Testen Sie die Prozedur für
Nummer: 42
Name: Dent
Vorname: Arthur
Abteilung: Support

Hinweise: Das Verwenden einer Sub-Query ist nicht erlaubt.
Sie dürfen annehmen, dass die übergebene Abteilung auf jeden Fall existiert.

4. Aufgabe

Lösen Sie Aufgabe 3 erneut. Diesmal allerdings mit Behandlung von Fehleingaben:

- Sollte eine noch inexistente Abteilung übergeben werden, erstellen Sie diese und fügen die Person ein.
- Sollte eine Person bereits existieren, die Abteilung jedoch abweichen, ändern Sie die Abteilung im existierenden Datensatz
- Sollte die Person mittels Name und Vorname nicht eindeutig identifizierbar sein, geben Sie eine Fehlermeldung aus und führen keine weitere Aktion durch.

5. Aufgabe	Ergänzen Sie die sqlteacherdb mit den Tabellen Rechnungen und Einzelpositionen zur Rechnung. Erstellen Sie anschliessend eine Stored-Procedure welcher Sie die Bestellnummer übergeben und die für Sie dann die benötigten Daten in die Rechnungstabelle kopiert und die zugehörigen Positionen anlegt.
6. Aufgabe	Erweitern Sie Ihre Lösung zur Aufgabe 5 so, damit eine SQLEXCEPTION ausgelöst wird, wenn eine Bestellung bereits verrechnet ist. Informieren Sie sich hier: https://dev.mysql.com/doc/refman/5.7/en/signal.html

Lösungen

1. Aufgabe

```
DELIMITER //
CREATE PROCEDURE hello
(
  IN n VARCHAR(255)
)
BEGIN
  SELECT CONCAT('Hello', n) AS 'Greeting';
END //
DELIMITER ;

CALL hello('World');
```

2. Aufgabe

```
DELIMITER //
CREATE PROCEDURE accu
(
  IN a INT,
  OUT q INT
)
BEGIN
  SELECT @accu + a INTO @accu;

  SET q = @accu;
END //
DELIMITER ;


SET @accu = 100;

SET @result = 0;
CALL accu(1, @result);
SELECT @result;

SELECT @accu;
```

3. Aufgabe

```
DELIMITER //
CREATE PROCEDURE addWorker
(
  IN nr INT,
  IN surname VARCHAR(255),
  IN name VARCHAR(255),
  IN department VARCHAR(255)
)
BEGIN
  SELECT abteilungsnr
  FROM Abteilung
  WHERE Bezeichnung=department
  INTO @abteilung;

  INSERT INTO mitarbeiter
    (mitarbeiternr, name, vorname, abteilung)
    VALUES
      (nr, surname, name, @abteilung);
END //
DELIMITER ;

CALL addWorker(42, 'Dent', 'Arthur', 'Support');
```

4. Aufgabe

```
DELIMITER //
CREATE PROCEDURE addWorker2
(
  IN nr INT,
  IN surname VARCHAR(255),
  IN name VARCHAR(255),
  IN department VARCHAR(255)
)
proc: BEGIN
  DECLARE depId INT DEFAULT NULL;

  -- First we try to find out if the apartment is already
  -- existing.
  SELECT ABTEILUNGSNR
  FROM abteilung
  WHERE abteilung.BEZEICHNUNG=department
  INTO depId;

  -- In case the department does not yet exist...
  IF depId IS NULL THEN
    -- ... we have to create it. So we have to
    -- calculate a new ID first as the department
    -- id is not AUTO_INCREMENT.
    SELECT MAX(abteilungsnr)+1
    FROM abteilung
    INTO depId;

    -- Now we insert the department.
    INSERT INTO abteilung
      (abteilungsnr, bezeichnung)
      VALUES
        (depId, department);
  END IF;

  -- Check if there is already a user existing with the
  -- given surename/name combination.
  SELECT COUNT(mitarbeiternr), abteilung, mitarbeiternr
```

```
  FROM mitarbeiter
  WHERE MITARBEITER.NAME=surname
    AND MITARBEITER.VORNAME=name
  INTO @usercount, @actual_dep, @userid;

  -- In case the user is already existing...
  IF NOT @usercount = 0 THEN
    -- ...we check if the department differs...
    IF NOT depId = @actualdep THEN
      -- ... and update it ...
      UPDATE mitarbeiter
        SET abteilung=depId
        WHERE MITARBEITERNR=@userid;
      LEAVE proc;
    ELSE
      -- ... otherwise we output a message ...
      SELECT 'Doppelter Eintrag';
      LEAVE proc;
    END IF;
  END IF;

  -- Reaching this point depId contains a valid
  -- department id and the user does not yet exist.
  -- Hence we can safely insert the given user.
  INSERT INTO mitarbeiter
    (mitarbeiternr, name, vorname, abteilung)
    VALUES
      (nr, surname, name, depId);
END //
DELIMITER ;

CALL addWorker2(42, 'Dent', 'Arthur', 'Support');
```

5. Aufgabe

```
-- Create needed tables
DROP TABLE IF EXISTS rechnung_posten;
DROP TABLE IF EXISTS rechnung;
DROP PROCEDURE IF EXISTS createInvoice;

CREATE TABLE rechnung
(
RECHNUNGNR INT PRIMARY KEY AUTO_INCREMENT,

BESTELLNR INT,
RECHNUNGSDATUM DATETIME NOT NULL,

FOREIGN KEY (BESTELLNR) REFERENCES bestellung(BESTELLNR)
);


CREATE TABLE rechnung_posten
(
Id INT PRIMARY KEY AUTO_INCREMENT,

RECHNUNGNR INT,
ARTIKELNR INT,
LIEFERMENGE INT,

FOREIGN KEY (RECHNUNGNR) REFERENCES rechnung(RECHNUNGNR),
FOREIGN KEY (ARTIKELNR) REFERENCES artikel(ARTIKELNR)
);

DELIMITER //
CREATE PROCEDURE createInvoice
(
IN _bestellnr INT
)
BEGIN

INSERT INTO rechnung (bestellnr, rechnungsdatum)
SELECT bestellnr, current_date FROM bestellung
   WHERE bestellnr = _bestellnr;

INSERT INTO rechnung_posten (rechnungnr, artikelnr, liefermenge)
SELECT last_insert_id(), artikelnr, liefermenge FROM posten
   WHERE bestellnr = _bestellnr;
END //
DELIMITER ;

CALL createInvoice(1);
select * from rechnung;
select * from rechnung_posten;
```

6. Aufgabe

```
DROP PROCEDURE IF EXISTS createInvoice;

DELIMITER //
CREATE PROCEDURE createInvoice
(
IN _bestellnr INT
)
BEGIN

IF exists(select * from bestellung where bestellnr = _bestellnr) THEN
    -- SQLSTATES die mit 01 beginnen, ergeben eine Warnung
    SIGNAL SQLSTATE '01000'
      SET MESSAGE_TEXT = 'Warnung: Bestellung nicht vorhanden';
END IF;

IF exists(select * from rechnung where bestellnr = _bestellnr) THEN
    -- SQLSTATES die nicht mit 00, 01 oder 01 beginnen, ergeben einen Fehler
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Fehler: Bestellung bereits verrechnet';
END IF;

INSERT INTO rechnung (bestellnr, rechnungsdatum)
SELECT bestellnr, current_date FROM bestellung
   WHERE bestellnr = _bestellnr;

INSERT INTO rechnung_posten (rechnungnr, artikelnr, liefermenge)
SELECT last_insert_id(), artikelnr, liefermenge FROM posten
   WHERE bestellnr = _bestellnr;
END //
DELIMITER ;

CALL createInvoice(1);   -- ergibt einen Fehler
CALL createInvoice(10);  -- ergibt eine Warnung
```