Konvertierung der Aufgaben zu MySQL Syntax

Autor: Mario Fentler 5CHIT
Datum: 06.01.2019

Hier werden die zuvor in PostgreSQL geschriebenen Funktionen in MySQL konvertiert.

Die Syntax in MySQL ist komplett anders als die in PostgreSQL. Beispielsweise können Stored Procedures in MySQL keine Werte zurückliefern. Somit fällt **RETURNS <TYP> AS $$ ... $$** weg. **Parameter** von solchen Procedures werden mit einem Parameternamen deklariert. Die Verwendung von $1, $2, ... ist in MySQL nicht mehr möglich. Statt **AS $$ ... $$** wird hier **BEGIN ... END;** verwendet.
Weiters kann man den **Delimiter** vom Semikolon auf etwas anderes (bsp. //) ändern, da es sonst zu Syntax Fehlern kommen kann beim Aufbau von Stored Procedures.

AU01

Erstelle eine Funktion preis99() die auf 3 verschiedene Arten die Nachkommastellen vom Preis auf 99 ändert.

Variante 1:

```
DELIMITER //
DROP PROCEDURE IF EXISTS preiserhoehung_var1 //
CREATE PROCEDURE preiserhoehung_var1()
    BEGIN
    UPDATE speise SET preis = ceil(preis) - 0.01;
    END;//
DELIMITER ;

SELECT * FROM speise;
CALL preiserhoehung_var1();
SELECT * FROM speise;
```

```
mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.00 |
|   2 | Schoko Palatschinken |  4.00 |
|   3 | Pute gebacken        |  7.00 |
|   4 | Pute natur           |  8.00 |
|   5 | Puten-Cordon         |  9.00 |
|   6 | Menue fuer 2         | 15.00 |
|   7 | Menue fuer 3         | 19.00 |
|   8 | Menue fuer 4         | 22.00 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)

mysql> CALL preiserhoehung_var1();
Query OK, 8 rows affected (0.00 sec)

mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  2.99 |
|   2 | Schoko Palatschinken |  3.99 |
|   3 | Pute gebacken        |  6.99 |
|   4 | Pute natur           |  7.99 |
|   5 | Puten-Cordon         |  8.99 |
|   6 | Menue fuer 2         | 14.99 |
|   7 | Menue fuer 3         | 18.99 |
|   8 | Menue fuer 4         | 21.99 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)
```

Ergebnis AU01 - Variante 1

Variante 2:

```
DELIMITER //
DROP PROCEDURE IF EXISTS preiserhoehung_var2 //
CREATE PROCEDURE preiserhoehung_var2()
    BEGIN
    UPDATE speise SET preis = floor(preis) + 0.99;
    END;//
DELIMITER ;

SELECT * FROM speise;
CALL preiserhoehung_var2();
SELECT * FROM speise;
```

```
mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.00 |
|   2 | Schoko Palatschinken |  4.00 |
|   3 | Pute gebacken        |  7.00 |
|   4 | Pute natur           |  8.00 |
|   5 | Puten-Cordon         |  9.00 |
|   6 | Menue fuer 2         | 15.00 |
|   7 | Menue fuer 3         | 19.00 |
|   8 | Menue fuer 4         | 22.00 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)

mysql> CALL preiserhoehung_var2();
Query OK, 8 rows affected (0.00 sec)

mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.99 |
|   2 | Schoko Palatschinken |  4.99 |
|   3 | Pute gebacken        |  7.99 |
|   4 | Pute natur           |  8.99 |
|   5 | Puten-Cordon         |  9.99 |
|   6 | Menue fuer 2         | 15.99 |
|   7 | Menue fuer 3         | 19.99 |
|   8 | Menue fuer 4         | 22.99 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)
```

Ergebnis AU01 - Variante 2

Variante 3:

```
DELIMITER //
DROP PROCEDURE IF EXISTS preiserhoehung_var3 //
CREATE PROCEDURE preiserhoehung_var3()
    BEGIN
    UPDATE speise SET preis = TRUNCATE(preis,0) + 0.99;
    END;//
DELIMITER ;

SELECT * FROM speise;
CALL preiserhoehung_var3();
SELECT * FROM speise;
```

```
mysql> -- variante 3
mysql> DELIMITER // ;
mysql> CREATE PROCEDURE preiserhoehung_var3()
    ->     BEGIN
    ->     UPDATE speise SET preis = TRUNCATE(preis,0) + 0.99;
    ->     END;//
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql>
mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.00 |
|   2 | Schoko Palatschinken |  4.00 |
|   3 | Pute gebacken        |  7.00 |
|   4 | Pute natur           |  8.00 |
|   5 | Puten-Cordon         |  9.00 |
|   6 | Menue fuer 2         | 15.00 |
|   7 | Menue fuer 3         | 19.00 |
|   8 | Menue fuer 4         | 22.00 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)

mysql> CALL preiserhoehung_var3();
Query OK, 8 rows affected (0.01 sec)

mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.99 |
|   2 | Schoko Palatschinken |  4.99 |
|   3 | Pute gebacken        |  7.99 |
|   4 | Pute natur           |  8.99 |
|   5 | Puten-Cordon         |  9.99 |
|   6 | Menue fuer 2         | 15.99 |
|   7 | Menue fuer 3         | 19.99 |
|   8 | Menue fuer 4         | 22.99 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)
```

Ergebnis AU01 - Variante 3

AU02

Erstelle eine Funktion mit zwei Parametern: Speisen die billiger als der Durchschnittspreis aller Speisen sind, sollen um einen fixen Betrag erhöht werden. Speisen die teurer als der Durchschnittspreis sind, sollen um einen Prozentwert erhöht werden (AU02a). Achte darauf, dass sich die beiden Erhöhungen nicht gegenseitig beeinflussen (AU02b)!

Für die Aufgabe muss man Procedures mit Parametern benutzen. Da es allerdings nicht möglich ist eine Procedur innerhalb einer anderen aufzurufen, fällt hier die Hilfsmethode weg und alles wird in einer gemacht.

Parameter von Procedures werden in einer Klammer angegeben. Dort gibt man auch den Datentyp mit an. In der Procedure kann man sie dann ganz normal über den Parameternamen ansprechen.

```
DELIMITER //
DROP PROCEDURE IF EXISTS preisErhoehung //
CREATE PROCEDURE preisErhoehung (fixedValue DECIMAL(4,2),
percent DECIMAL(4,2))
BEGIN
    DECLARE avgPrice DECIMAL(20,10);
    SELECT avg(preis) INTO avgPrice FROM speise;
    UPDATE speise SET preis = preis + fixedValue
    WHERE preis <= avgPrice;
    UPDATE speise SET preis = preis + preis * percent / 100
    WHERE preis > avgPrice;
END;//
DELIMITER ;
```

In diesem Fall werden die Preise, die **kleiner** als der Durchschnittspreis sind um 1 erhöht, alle anderen, die **größer** sind um 5%.

```
SELECT * FROM speise;
CALL preisErhoehung (1,5);
SELECT * FROM speise;
```

```
mysql> -- ******
mysql> -- AU02
mysql> -- ******
mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  3.00 |
|   2 | Schoko Palatschinken |  4.00 |
|   3 | Pute gebacken        |  7.00 |
|   4 | Pute natur           |  8.00 |
|   5 | Puten-Cordon         |  9.00 |
|   6 | Menue fuer 2         | 15.00 |
|   7 | Menue fuer 3         | 19.00 |
|   8 | Menue fuer 4         | 22.00 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)

mysql> CALL preisErhoehung (1,5);
Query OK, 3 rows affected (0.00 sec)

mysql> SELECT * FROM speise;
+-----+----------------------+-------+
| snr | bezeichnung          | preis |
+-----+----------------------+-------+
|   1 | Heisse Liebe         |  4.00 |
|   2 | Schoko Palatschinken |  5.00 |
|   3 | Pute gebacken        |  8.00 |
|   4 | Pute natur           |  9.00 |
|   5 | Puten-Cordon         | 10.00 |
|   6 | Menue fuer 2         | 15.75 |
|   7 | Menue fuer 3         | 19.95 |
|   8 | Menue fuer 4         | 23.10 |
+-----+----------------------+-------+
8 rows in set (0.00 sec)
```

Ergebnis AU02 a und b

AU03

Der Tagesumsatz für einen bestimmten Kellner soll für den aktuellen Tag ermittelt werden. Verwende die Kellner-Nr als Parameter, den aktuellen Tag via CURRENT_DATE (AU03).

Um das zu überprüfen muss die Tabelle Rechnung noch überarbeitet werden, da dort die Rechnungen alle auf ein fixes Datum gesetzt wurden. Dazu wird dort einfach ein Datum auf CURRENT_DATE gesetzt und dann beim selecten nach dem dazugehörigen Kellner selected.

```
INSERT INTO rechnung VALUES (7, CURRENT_DATE, 1, 'bezahlt', 1);
```

```
DELIMITER //
DROP PROCEDURE IF EXISTS umsatz_au03 //
CREATE PROCEDURE umsatz_au03(kellnerNr INTEGER(255))
    BEGIN
        SELECT SUM(speise.preis)
        FROM speise,rechnung,bestellung
        WHERE speise.snr = bestellung.snr
        AND rechnung.rnr = bestellung.rnr
        AND rechnung.status = 'bezahlt'
        AND rechnung.knr = kellnerNr
        AND rechnung.datum = CURRENT_DATE;
    END;//
DELIMITER ;

CALL umsatz_au03(1);
```

```
mysql> CALL umsatz_au03(1);
+--------------------+
| SUM(speise.preis)  |
+--------------------+
|               3.00 |
+--------------------+
1 row in set (0.00 sec)
```

Ergebnis AU03

AU04

Erstelle eine weitere Funktion zur Berechnung der MWSt. Zeige von allen Speisen den Brutto-Preis als Brutto und die darin enthaltene MWSt. als Spalte MWSt an (AU04a). Die Ausgabe Brutto/MWSt soll auf zwei Nachkommastellen beschränkt werden (AU04b).

a)

Für die erste Aufgabe wird der Preis aus der Tabelle Speise ausgelesen und so modifiziert, dass die Spalten Brutto und Mehrwert ausgegeben werden können.

```
DELIMITER //
DROP PROCEDURE IF EXISTS au04a //
CREATE PROCEDURE au04a()
    BEGIN
        SELECT bezeichnung,
        preis AS "Netto",
        preis * 1.2 AS "Brutto",
        preis * 0.2 AS "Mehrwertsteuer"
        FROM speise;
    END;//
```

```
DELIMITER ;
CALL au04a();
```

```
mysql> CALL au04a();
+----------------------+-------+--------+----------------+
| bezeichnung          | Netto | Brutto | Mehrwertsteuer |
+----------------------+-------+--------+----------------+
| Heisse Liebe         |  3.00 |  3.600 |          0.600 |
| Schoko Palatschinken |  4.00 |  4.800 |          0.800 |
| Pute gebacken        |  7.00 |  8.400 |          1.400 |
| Pute natur           |  8.00 |  9.600 |          1.600 |
| Puten-Cordon         |  9.00 | 10.800 |          1.800 |
| Menue fuer 2         | 15.00 | 18.000 |          3.000 |
| Menue fuer 3         | 19.00 | 22.800 |          3.800 |
| Menue fuer 4         | 22.00 | 26.400 |          4.400 |
+----------------------+-------+--------+----------------+
8 rows in set (0.00 sec)
```

Ergebnis AU04a

b)

Um zahlen auf Nachkommastellen zu beschränken wird die **CAST** Funktion verwendet.

```
DELIMITER //
DROP PROCEDURE IF EXISTS au04b //
CREATE PROCEDURE au04b()
    BEGIN
        SELECT bezeichnung,
        preis AS "Netto",
        CAST(preis * 1.2 AS DECIMAL (6,2)) AS "Brutto",
        CAST(preis * 0.2 AS DECIMAL (6,2)) AS "Mehrwertsteuer"
        FROM speise;
    END;//

DELIMITER ;
CALL au04b();
```

```
mysql> CALL au04b();
+----------------------+-------+--------+----------------+
| bezeichnung          | Netto | Brutto | Mehrwertsteuer |
+----------------------+-------+--------+----------------+
| Heisse Liebe         |  3.00 |   3.60 |           0.60 |
| Schoko Palatschinken |  4.00 |   4.80 |           0.80 |
| Pute gebacken        |  7.00 |   8.40 |           1.40 |
| Pute natur           |  8.00 |   9.60 |           1.60 |
| Puten-Cordon         |  9.00 |  10.80 |           1.80 |
| Menue fuer 2         | 15.00 |  18.00 |           3.00 |
| Menue fuer 3         | 19.00 |  22.80 |           3.80 |
| Menue fuer 4         | 22.00 |  26.40 |           4.40 |
+----------------------+-------+--------+----------------+
8 rows in set (0.00 sec)
```

Ergebnis AU04b

AU05

Es soll eine Funktion zur Anzeige der Bezeichnungen der noch nie bestellten Speisen (AU05a) erstellt werden. Erweitere die aufrufende SELECT-Anweisung so, dass das Ergebnis als Tabelle mit den Spaltenüberschriften "Bezeichnung" und "Nettopreis" angezeigt wird (AU05b).

a)

Um die Speisen zu erhalten, die noch nie bestellt wurden muss man schon bestellte Speisen von der Gesamtmenge an Speisen abziehen und von diesen dann den Namen ausgeben.

```
DELIMITER //
DROP PROCEDURE IF EXISTS au05//
CREATE PROCEDURE au05()
BEGIN
    SELECT speise.bezeichnung AS "Bezeichnung" FROM speise
    WHERE speise.snr NOT IN (SELECT snr FROM bestellung);
END;//
DELIMITER ;
CALL au05();
```

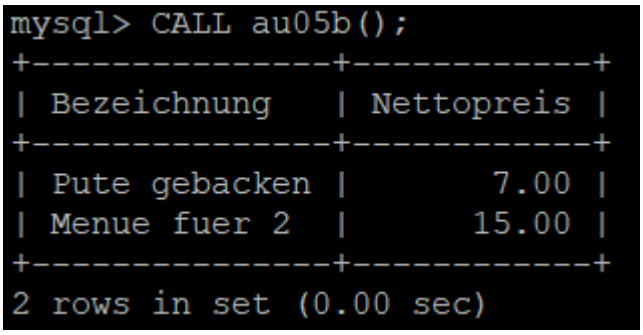
```
mysql> CALL au05();
+---------------+
| Bezeichnung   |
+---------------+
| Pute gebacken |
| Menue fuer 2  |
+---------------+
2 rows in set (0.00 sec)
```

Ergebnis AU05a

b)

Das geht in SQL ganz einfach, indem man **SELECT xyz AS <Spaltenname>** dazuschreibt.

```
DELIMITER //
DROP PROCEDURE IF EXISTS au05b//
CREATE PROCEDURE au05b()
BEGIN
    SELECT speise.bezeichnung AS "Bezeichnung",
    speise.preis AS "Nettopreis" FROM speise
    WHERE speise.snr NOT IN (SELECT snr FROM bestellung);
END;//
DELIMITER ;
CALL au05b();
```



Ergebnis AU05b

AU06

Erstelle zwei Funktionen die in der SELECT-Klausel eingebettet werden, um damit zusätzliche Spalten je Kellner anzeigen zu können. Die aufrufende SELECT-Anweisung soll folgende Ausgabe produzieren:
Kellnername, Anzahl der Rechnungen, Status der spätesten Rechnung

Diese Aufgabenstellung ist nicht in MySQL lösbar. Der Grund dafür ist, dass man sich die Werte anzahlRechnung und statusRechnung nicht in Variablen zwischenspeichern kann. In diesen Variablen kann man sie nicht speichern weil sie mehr als eine Row als Result haben. Desweiteren kann man keine Procedures in einer Procedure aufrufen.

AU07

Es soll eine Liste der Kellner und deren jeweiliger Tagesumsatz ausgegeben werden.

Diese Abfrage ist schon etwas schwieriger. Um die Aufgabe zu lösen wird auch eine temporäre Tabelle benötigt (helperTable). Diese hat die Spalten Kellnername und Tagesumsatz.

Um die Daten der Abfrage dort hinein zu bekommen muss in der Stored Procedure **INSERT INTO <tablename>** ausgeführt werden.

```
DROP TABLE IF EXISTS helperTable;
CREATE TABLE helperTable(
    Kellnername VARCHAR(255),
    Tagesumsatz NUMERIC
);

DELIMITER //
DROP PROCEDURE IF EXISTS au07 //
CREATE PROCEDURE au07()
BEGIN
    INSERT INTO helperTable
    SELECT kellner.name AS "kellnername",
    SUM(speise.preis) AS "tagesumsatz"
    FROM kellner
    INNER JOIN rechnung ON (rechnung.knr = kellner.knr)
    INNER JOIN bestellung ON (bestellung.rnr = rechnung.rnr)
    INNER JOIN speise ON (speise.snr = bestellung.snr)
    WHERE rechnung.datum = CURRENT_DATE AND
    rechnung.status = 'bezahlt'
    GROUP BY kellner.name
    UNION SELECT kellner.name, 0 FROM kellner;
END;//
DELIMITER ;
CALL au07();
SELECT * FROM helperTable;
```

```
mysql> CALL au07();
Query OK, 4 rows affected (0.00 sec)

mysql> SELECT * FROM helperTable;
+-------------+-------------+
| Kellnername | Tagesumsatz |
+-------------+-------------+
| Kellner1    |           3 |
| Kellner1    |           0 |
| Kellner2    |           0 |
| Kellner3    |           0 |
+-------------+-------------+
4 rows in set (0.00 sec)
```

Ergebnis AU07