
Aufgaben Protokoll

A08 Portierung MySQL

Informationssysteme
5YHITM 2016/17

Maximilian Seidl

Note:
Betreuer: M.Martinides

Version 0.1
Begonnen am 4. Dezember 2016
Beendet am 4. Dezember 2016

Inhaltsverzeichnis

1	Aufgabenstellung	1
2	Lösungsideen	2
3	Probleme	2
4	Konkrete Umsetzung	2
4.1	A01	2
4.2	A02	3
4.3	A03	4
4.4	A04	4
4.5	A05	5
4.6	A06	6
4.7	A07	7
5	Beweis der Umsetzung	7

1 Aufgabenstellung

Portiere die Aufgaben AU01 bis AU07 für MySQL.

Erstelle dazu ein Dokument mit einem Kapitel je Aufgabe mit folgenden Inhalten:

- Lösungsideen
- Probleme
- konkrete Umsetzung (z.B DDL-Script, zusätzliche INSERTs, ...)
- Beweis der Umsetzung (Screenshot der Ausführung)
- ggfs. Begründung warum nicht umsetzbar

2 Lösungsideen

Beim Lösen der Aufgaben bin ich auf folgende Methoden gekommen:

- Bereits erledigte Aufgaben (A01-A07) kopiert und umgeschrieben mittels der MySQL-Doku
- Alle Aufgaben neu schreiben mittels der MySQL-Doku

Im Endeffekt haben mir beide Methoden zum Erfüllen dieser Aufgabe geholfen, wobei ich öfters auf die zweite Art zurückgegriffen habe.

Beispielsweise war es bei A03 nicht anders möglich als die Tabellen mittels einem **INNER JOIN** zusammen zu führen um so den richtigen Tagesumsatz eines Kellners herauszufinden.

3 Probleme

Während der Importierung von den PostgreSQL-Aufgaben sind keine größeren Probleme aufgetreten. Lediglich hatte ich ein Problem in meinem CREATE-Script, welches ich schnell lösen konnte. (DROP - Befehl)

4 Konkrete Umsetzung

4.1 A01

Alle Preise sollen so geändert (erhöht) werden, dass sie mit 99 Cent enden!

```
1      delimiter //
2      CREATE PROCEDURE preis99()
3      BEGIN
4          UPDATE speise SET preis=ceil(preis)-0.01;
5      END //
6      delimiter ;
7
8      SELECT * FROM speise;
9      CALL preis99();
10     SELECT * FROM speise;
```

Alle Rechnung zu denen es keine Bestellung gibt, sollen gelöscht werden.

```
2      delimiter //
      CREATE PROCEDURE loescheRechnung()
      BEGIN
4      delete from rechnung where rnr not in (select rnr from
          bestellung);
      END //
6      delimiter ;

8      INSERT INTO rechnung VALUES (10, CURDATE(), 1, 'gedruckt', 1);

10     SELECT * FROM rechnung;
      CALL loescheRechnung();
12     SELECT * FROM rechnung;
```

4.2 A02

Erstelle eine Funktion mit zwei Parametern: Alle Speisen die billiger als der Durchschnittspreis aller Speisen sind, sollen um einen fixen Betrag erhöht werden, alle Speisen die teurer sind, sollen um einen Prozentwert erhöht werden.

```
1      delimiter //
      CREATE PROCEDURE preiserhoehung(IN prozent DECIMAL(30,2),IN
          abpreis DECIMAL(30,2))
3      BEGIN
      DECLARE av DECIMAL DEFAULT (SELECT avg(preis) FROM speise);
5      UPDATE speise SET preis = preis * (100 + prozent) / 100 where
          preis > av;
      UPDATE speise SET preis = preis + abpreis where preis <= av;
7      END//
      delimiter ;

9

11     SELECT * FROM speise;
      CALL preiserhoehung(10,1.75);
      SELECT * FROM speise;
```

4.3 A03

Der Tagesumsatz für einen bestimmten Kellner soll für den aktuellen Tag ermittelt werden.

```
2      CREATE FUNCTION tagesumsatz(kel INT)
      RETURNS DECIMAL(30,2) DETERMINISTIC
      RETURN
4      (select sum(preis*anzahl) from speise
      inner join bestellung on speise.snr = bestellung.snr
6      inner join rechnung on rechnung.rnr = bestellung.rnr
      where knr=kel AND rechnung.status like 'bezahlt%' AND rechnung.
          datum = CURDATE());
8
10     SELECT tagesumsatz(1);
      SELECT tagesumsatz(2);
      select tagesumsatz(3);
```

4.4 A04

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. Zusatz: Die Anzeige bei Brutto/MWSt soll auf zwei Nachkommastellen beschränkt werden.

```
2      CREATE FUNCTION bruttoPreis(preis DECIMAL(30,2))
      RETURNS DECIMAL(30,2) DETERMINISTIC
      RETURN
4      (SELECT round(preis*1.2, 2));
      delimiter //
6
      CREATE FUNCTION mwstPreis(preis DECIMAL(30,2))
      RETURNS DECIMAL(30,2) DETERMINISTIC
      RETURN
10     (SELECT round(preis*0.2, 2));
      delimiter //
12     CREATE PROCEDURE ausgabe1()
      BEGIN
14     Select bezeichnung, bruttoPreis(speise.preis) as "Brutto", mwst(
      speise.preis*1.2) as "MWST" from speise;
16     END //
      delimiter ;
18
      call ausgabe1();
```

4.5 A05

Erstelle eine Funktion zur Anzeige der Bezeichnungen der noch nie bestellten Speisen.

```
2      delimiter //
      CREATE PROCEDURE nichtbestellt()
      BEGIN
4      Select bezeichnung from speise where snr not in (Select snr from
          bestellung);
      END //
6      delimiter ;

8      call nichtbestellt();
```

Zusatz: Erweitere die aufrufende SELECT-Anweisung so, dass das Ergebnis als Tabelle mit den Spaltenüberschriften "Bezeichnung" und "Nettopreis" angezeigt wird. als Select nicht möglich, da call procedure nicht in einer select Anweisung aufgerufen werden kann. mögliche Variante- neue Procedure

```
1      delimiter //
      CREATE PROCEDURE ausgabe2()
3      BEGIN
      Select bezeichnung, preis as "Nettopreis" from speise where
          bezeichnung in (Select bezeichnung from speise where snr not
              in (Select snr from bestellung));
5      END //
      delimiter ;

7      call ausgabe2();
```

4.6 A06

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

```
1      CREATE FUNCTION anzahlrechnungen(kel INTEGER)
      RETURNS BIGINT DETERMINISTIC
3      RETURN
      (SELECT count(rechnung.knr)
5      FROM rechnung
      WHERE rechnung.knr = kel);

7      CREATE FUNCTION letzteRechnung(kel INTEGER)
      RETURNS CHAR(8) DETERMINISTIC
9      RETURN
11     (select status from rechnung where datum = (select max(datum)
      from rechnung where rechnung.knr = kel) and rechnung.knr =
      kel limit 1);

13     delimiter //
      CREATE PROCEDURE ausgabe3()
15     BEGIN
      Select distinct kellner.name, anzahlrechnungen(rechnung.knr) as
      'Anzahl Rechnungen', letzterechnung(rechnung.knr) as 'Letzte
      Rechnung' from rechnung inner join kellner on kellner.knr =
      rechnung.knr order by kellner.name asc;
17     END //
      delimiter ;

19     call ausgabe3();
```


4.7 A07

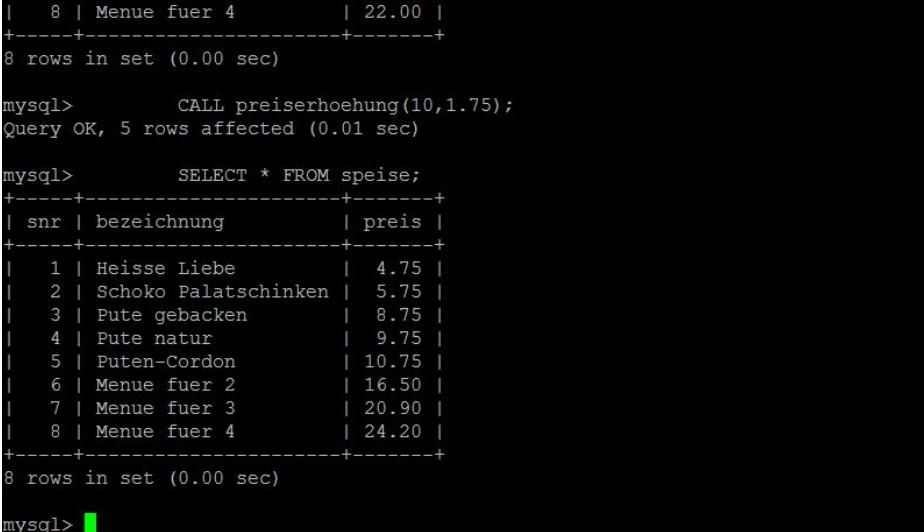
Es soll eine Liste der Kellner und deren jeweiliger Tagesumsatz ausgegeben werden. Korrekte Lösung mit der Original-DB: nur eine Zeile mit Kellner1 und 71.00

```
2      delimiter //
3      CREATE PROCEDURE umsatzkellner()
4      BEGIN
5          select kellner.name, sum(preis*anzahl) from kellner
6          inner join rechnung on kellner.knr = rechnung.knr
7          inner join bestellung on rechnung.rnr = bestellung.rnr
8          inner join speise on speise.snr = bestellung.snr
9          where rechnung.status = 'bezahlt' AND rechnung.datum =
10             current_date
11          group by kellner.knr;
12      END //
13      delimiter ;

call umsatzkellner();
```

5 Beweis der Umsetzung

Umsetzung folgender Aufgabe: A02



```
| 8 | Menue fuer 4 | 22.00 |
+-----+-----+
8 rows in set (0.00 sec)

mysql> CALL preiserhoehung(10,1.75);
Query OK, 5 rows affected (0.01 sec)

mysql> SELECT * FROM speise;
+-----+-----+-----+
| snr | bezeichnung | preis |
+-----+-----+-----+
| 1 | Heisse Liebe | 4.75 |
| 2 | Schoko Palatschinken | 5.75 |
| 3 | Pute gebacken | 8.75 |
| 4 | Pute natur | 9.75 |
| 5 | Puten-Cordon | 10.75 |
| 6 | Menue fuer 2 | 16.50 |
| 7 | Menue fuer 3 | 20.90 |
| 8 | Menue fuer 4 | 24.20 |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

Abbildung 1: Screenshot der Konsole beim Durchführen von A02

Umsetzung folgender Aufgabe: **A06**

```
mysql> delimiter //
mysql> CREATE PROCEDURE ausgabe3()
  -> BEGIN
  -> Select distinct kellner.name, anzahlrechnungen(rechnung.knr) as 'Anzahl R
rechnungen', letzterechnung(rechnung.knr) as 'Letzte Rechnung' from rechnung inne
r join kellner on kellner.knr = rechnung.knr order by kellner.name asc;
  -> END //
ERROR 1304 (42000): PROCEDURE ausgabe3 already exists
mysql> delimiter ;
mysql>
mysql> call ausgabe3();
+-----+-----+-----+
| name      | Anzahl Rechnungen | Letzte Rechnung |
+-----+-----+-----+
| Kellner1  | 3 | bezahlt    |
| Kellner2  | 2 | offen      |
| Kellner3  | 1 | gedruckt   |
+-----+-----+-----+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

Abbildung 2: Screenshot der Konsole beim Durchführen von A06

Umsetzung folgender Aufgabe: **A05**

```
mysql> delimiter //
mysql> CREATE PROCEDURE nichtbestellt()
  -> BEGIN
  -> Select bezeichnung from speise where snr not in (Select snr from bestel
lung);
  -> END //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql>
mysql> call nichtbestellt();
+-----+
| bezeichnung |
+-----+
| Pute gebacken |
| Menue fuer 2 |
+-----+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

Abbildung 3: Screenshot der Konsole beim Durchführen von A05

Abbildungsverzeichnis

1	Screenshot der Konsole beim Durchführen von A02	7
2	Screenshot der Konsole beim Durchführen von A06	8
3	Screenshot der Konsole beim Durchführen von A05	8