Übungen SQL – DML

Klaus-Georg Deck

2025-07-24

Einleitung

In diesem Skript findest du typische SQL-Übungen mit Beispielcode und den erwarteten Ergebnistabellen. Ziel ist es, die SQL-Operationen zur Datenmanipulation (DML) wie `INSERT`, `UPDATE`, und `DELETE` zu erlernen und zu vertiefen.

Voraussetzungen

Du verfügst über einen Zugang zu einem Datenbank-Server (Oracle oder Postgres) mit den Tabellen des Bike-Verleih-Szenarios und hast Grundkenntnisse in SQL, etwa indem Du die vorherigen Übungen erfolgreich absolviert hast.

Als weitere Voraussetzungen werden die Tabellen `UE_EMPLO` und `UE_CUSTOMER` benötigt. Sollten diese nicht vorhanden sein, lassen sie sich wie folgt erstellen:

```
--falls Tabellen vorhanden sind und neu erstellt werden sollen:
--DROP TABLE UE_CUSTOMER;
--DROP TABLE UE_EMPLO;

CREATE TABLE UE_EMPLO (
   ID INT PRIMARY KEY,
   NAME VARCHAR(100),
   JOB VARCHAR(100),
   SALARY INT
);

CREATE TABLE UE_CUSTOMER (
   ID INT PRIMARY KEY,
   NAME VARCHAR(100),
   IS_VIP INT DEFAULT 0
);
```

Falls die Tabellen bereits existieren, stelle sicher, dass sie leer sind. (Vgl. Aufgabe ‘Einfaches Löschen von Datensätzen’)

Aufgabe 1: `INSERT`-Anweisungen

a) Einfaches `INSERT`

Formuliere zwei SQL-Anweisungen, so dass dadurch je ein Datensatz in die Tabelle `UE_CUSTOMER` eingefügt wird. Diese Tabelle soll danach den folgenden Inhalt besitzen:

ID	NAME	IS_VIP
60000	Hannah	0
60001	Jan	1

In der Tabellendefinition ist für das Attribut `IS_VIP` als Default `0` vorgesehen. Formuliere die erste `INSERT`-Anweisung so, dass dieser Default verwendet wird und keine explizite Angabe dieses Attributwertes erfolgt.

Tipp:
Falls Du einen Fehler machst und die Tabelle leeren möchtest, verwende `DELETE FROM UE_CUSTOMER`.

b) Multi-row `INSERT`

Füge der Tabelle drei weitere Datensätze hinzu, nämlich 'Lisa', 'Lukas' und 'Max' mit beliebiger `ID`, die alle keine VIPs sind. Verwende eine einzige `INSERT`-Anweisung.

b) Fehler bei `INSERT`

Bei dieser Teilaufgabe gehen wir davon aus, dass die Tabelle `UE_CUSTOMER` mindestens den ersten Datensatz aus der Teilaufgabe a) besitzt. Formuliere eine Multi-row-`SELECT`-Anweisung, bei der der als letztes einzufügende Datensatz der Eindeutigkeit des Attributs `ID` widerspricht. Was passiert mit den vorherigen Datensätzen? Konstruiere ein geeignetes Beispiel.

c) `INSERT-SELECT-Anweisung`

Füge der Tabelle `UE_EMPLO` alle Datensätze von Mitarbeitenden aus `EMPLO` hinzu, deren Salär unter `4800` liegt. Formuliere eine SQL-Anweisung, die dies einmalig und in einer einzigen Anweisung leistet, wobei die Werte von `JOB` nicht übernommen werden sollen. Beachte, dass in `UE_EMPLO` nicht alle Attribute aus `EMPLO` vorhanden sind.
Das gewünschte Ergebnis sieht dann so aus:

ID	NAME	JOB	SALARY
16000	Mio		4700
16048	Leo		4100

Sehr fortgeschrittene Aufgabe:
Erweitere die vorherige Anweisung, so dass diese ausgeführt werden kann, unabhängig davon, ob und welche `ID`s in der Zieltabelle bereits vergeben sind. Dabei ist für jeden neuen Eintrag gegebenenfalls eine neue `ID` zu wählen, etwa so, dass man sich am Maximum der bereits vergebenen Werte orientiert.

Aufgabe 2: `UPDATE`-Anweisungen

Stelle für diese Aufgabe sicher, dass die Tabelle `UE_EMPLO` jeweils die entsprechenden Datensätze aus `EMPLO` enthält. Dies erreicht man etwa folgendermassen:

```
DELETE FROM UE_EMPLO;

INSERT INTO UE_EMPLO( ID, NAME, JOB, SALARY )
SELECT ID, NAME, JOB, SALARY FROM EMPLO;
```

a) Einfaches UPDATE

Gib eine SQL-Anweisung an, die in UE_EMPLO für den Datensatz mit der ID 16038 den Namen von 'David' auf 'David E.' ändert und das Salär um 10 erhöht.

Gib eine SQL-Anweisung an, die in UE_EMPLO in allen Datensätzen, die keinen Job angegeben haben (NULL), den Namen löscht, d.h. auf NULL setzt.

a) Mehrfaches UPDATE

Gib SQL-Anweisungen an, die folgende Gehaltserhöhung für Mitarbeitende umsetzen:

Bedingung an Salär	Änderung des Salärs
unter 5000	+ 5%
zwischen 5000 und 8000	+ 4%
alle anderen	+ 3%

Im Idealfall verwendet man eine einzige Anweisung.

Aufgabe 3: UPDATE-Anweisungen mit Subselect

Stelle für diese Aufgabe wieder sicher, dass die Tabelle UE_EMPLO genau die entsprechenden Datensätze aus EMPLO enthält. Dies erreicht man etwa folgendermassen:

```
DELETE FROM UE_EMPLO;

INSERT INTO UE_EMPLO( ID, NAME, JOB, SALARY )
SELECT ID, NAME, JOB, SALARY FROM EMPLO;
```

Formuliere eine SQL-Anweisung, die bei allen Mitarbeitenden, deren Salär niedriger als das Durchschnittssalär ist, das Salär erhöht und zwar um 5%.

Stelle anschliessend wieder den ursprünglichen Zustand der Tabelle UE_EMPLO her und formuliere eine SQL-Anweisung, mit der folgendes erreicht wird: Alle Mitarbeitenden, deren Salär höher als das Durchschnittssalär der Mitarbeitenden des gleichen Jobs (JOB) ist, sollen eine Erhöhung um den Betrag 111 erhalten.

Aufgabe 4: DELETE-Anweisungen

a) Mehrfaches Löschen

Lösche in der Tabelle UE_EMPLO den Datensatz mit ID 16000. Was passiert, wenn Du versuchst, diesen Datensatz nochmal zu löschen?

b) DELETE ohne WHERE-Klausel

In der WHERE-Klausel werden bei DELETE-Anweisungen die Datensätze spezifiziert, die gelöscht werden sollen. Was passiert, wenn keine WHERE-Klausel vorhanden ist? Erkläre das Verhalten.

b) DELETE mit komplexeren Bedingungen

Stelle zunächst sicher, dass die Tabelle UE_EMPLO die Datensätze aus EMPLO enthält:

```
DELETE FROM UE_EMPLO;

INSERT INTO UE_EMPLO( ID, NAME, JOB, SALARY )
SELECT ID, NAME, JOB, SALARY FROM EMPLO;
```

Formuliere eine SQL-Anweisung, die alle Datensätze von Mitarbeitenden in `UE_EMPLO` löscht, deren Name (`NAME`) mehr als einmal vorkommt.

Stelle anschliessend den ursprünglichen Zustand der Tabelle her. Formuliere eine SQL-Anweisung, mit der in `UE_EMPLO` pro Datensatz-Gruppe mit gleichem Namen alle Datensätze bis auf den mit der grössten `ID` gelöscht werden. Diese Anweisung löscht also nicht alle Datensätze von mehrfachvorkommenden Namen, sondern belässt den mit der jeweils grössten `ID`.