Übungen zu Datenbanken, Wintersemester 2022/23
Dr.-Ing. Benedikt Loepp (benedikt.loepp@uni-due.de)

Übungsblatt 8 Abgabe bis **11.01.2023, 11:59 Uhr (mittags)**

Aufgabe 18: Integritätsbedingungen

Gegeben seien die untenstehenden Relationen. Erstellen Sie **CREATE TABLE-Statements**, um die entsprechenden Tabellen zu erzeugen.

Geben Sie wie immer nur SQL-Statements an. Verwenden Sie die Syntax aus Vorlesung (SQL99) bzw. Praktikum (s. Dokumentation im Moodle-Kurs).

Berücksichtigen Sie folgende Integritätsbedingungen:

- Definieren Sie Primärschlüssel.
- Datentypen sind sinnvoll zu wählen. Es stehen `INT`, `BIGINT`, `FLOAT`, `DECIMAL`, `DATE`, `DATETIME`, `TIME`, `CHAR`, `VARCHAR`, `BLOB` und `CLOB` zur Verfügung.
- Die im Relationenschema als `TABELLE.ATTRIBUT` angegebenen Fremdschlüssel sollen durch referentielle Integritätsbedingungen ausgedrückt werden.
- Im gesamten Schema ist nur das Attribut `GebDatum` optional.
- Zwei Passagiere können nicht die gleiche E-Mail-Adresse haben.
- `Kontostand` ist eine natürliche Zahl ($\mathbb{N}_0$).
- Die `StartZeit` muss vor der `LandeZeit` liegen.
- `StartFH` und `ZielFH` müssen unterschiedlich sein (FH steht für Flughafen).
- `Sitz` ist bis zu 4 Zeichen lang, z.B. `1A` oder `104E`.
- Ein Sitz darf pro Flug höchstens einmal gebucht werden.
- `Klasse` soll nur die Werte `Economy`, `Business` und `First` annehmen.
- `Preis` ist eine Zahl zwischen 20 und 20 000 mit 5 Vor- und 2 Nachkommastellen.
- Wird ein Passagier gelöscht, sollen auch seine Buchungen gelöscht werden.

Flugzeuge: {[<u>FNummer</u>, Fluggesellschaft]}

Passagiere: {[<u>KundNr</u>, Name, Adresse, GebDatum, EMail, Kontostand]}

Fluege: {[<u>FlugNr</u>, <u>AbflugDatum</u>, Flugzeuge.FNummer, StartZeit, LandeZeit, StartFH, ZielFH]}

Buchungen: {[<u>Passagiere.KundNr</u>, <u>Fluege.FlugNr</u>, <u>Fluege.Abflugdatum</u>, <u>Sitz</u>, Klasse, Preis]}

15 Punkte

Aufgabe 19: Trigger

Gegeben sei das Schema aus der vorherigen Aufgabe.

Erstellen Sie einen Trigger mit dem Namen `DeckungPruefen`, der bei einer Buchung prüft, ob der `Kontostand` des Passagiers gedeckt ist. Der Trigger soll ausgelöst werden, bevor das Einfügen in die Tabelle `Buchungen` erfolgt. Ist der `Kontostand` des Passagiers kleiner als der `Preis` des Tickets, soll mittels `SIGNAL` eine Fehlermeldung ausgeworfen und die Buchung nicht durchgeführt werden.

Verwenden Sie die in der Vorlesung vorgestellte SQL99-Syntax für Trigger. Neben der Vorlesung eignet sich die Dokumentation aus dem Praktikum (s. Moodle-Kurs, Seite 17–24), um mehr über SQL PL und Trigger zu erfahren.

Der Einfachheit halber können Sie den folgenden Trigger vervollständigen (aber natürlich auch einen eigenen Trigger implementieren).

```
CREATE TRIGGER ________________

_________________ ON ____________

_________________________________

_____________________ MODE DB2SQL

BEGIN ATOMIC

  __________ kontostand ____________________________;

  __________ kontostand = ______________________________;

  IF ____________________________ THEN

    SIGNAL SQLSTATE '75000' ('Konto nicht gedeckt.');

  END IF;

END@
```

5 Punkte