Exercise Sheet 12: Application Programming II (until Thursday, 25.01.2018) (42 Points)

Please note: you need **50%** of all exercise points to receive the *Studienleistung* for this lecture. In order to pass the RDB I Module, you need both the Studienleistung **and** you need to pass the exam. Exercises have to be turned in until **Thursday before the lecture** either in the lecture hall or into our mailbox at the IFIS floor (Mühlenpfordtstraße 23, 2nd floor). Please do not forget your **Matrikelnummer** and your **tutorial group number** on your solutions. **If you forget** to write your Matrikelnummer and/or your tutorial group number, you get **automatically 0 points**. Your solutions may be in German or English. Unless otherwise specified: **Always use your own words!**

Exercise 12.1 (22 Punkte)

Anwendungsszenario

Sie sind kürzlich von einer lokalen Firma eingestellt worden, um dabei zu helfen, die Datenhaltung von manuell geführten Tabellen in einer Tabellenkalkulation (z.B. MS Excel) in eine relationale Datenbank zu überführen. Diese Excel-Dateien werden unter anderem dafür verwendet, jedem Mitarbeiter eine Liste von Aufgaben zuzuteilen. Dafür existiert eine Datei „aufgaben.xls“, in der zu jeder Aufgabe eine kleine Beschreibung hinterlegt ist und eine Datei „mitarbeiter.xls“, in der jedem Mitarbeiter eine Liste von Aufgaben zugewiesen wird. Die Dateien sehen in etwa wie folgt aus:

`aufgaben.xls`

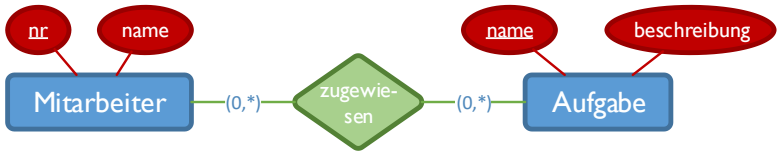
Kaffee kochen	Jeden Morgen bis 9 Uhr eine Kanne Kaffee kochen	…
Druckerpapier auffüllen	Einmal täglich Papierstand prüfen und bei Bedarf auffüllen	…
Einladung verschicken	Vor jedem Meeting Mails an die Mitarbeiterliste versenden	…
…	…	

`mitarbeiter.xls`

17	Herr Maus	Kaffee kochen, Einladungen verschicken	…
18	Frau Schmidt	Druckerpapier auffüllen, Tafel wischen, Weihnachtsfeier planen	…
…	…	…	

a) **(2P)** In welcher Normalform befinden sich die oben beschriebenen Excel-Tabellen? Begründen Sie ihre Antwort!

b) **(5P)** Sie haben vor die Daten aus den Excel-Tabellen gemäß des folgenden Schemas in eine Datenbank zu integrieren:



Dafür haben Sie ein kleines Skript erstellt, das aus den obigen Excel-Tabellen Komma-getrennte Textdateien (CSV-Dateien) generiert. Diese sehen in etwa wie folgt aus:

`mitarbeiter.csv`:

```
17, "Herr Maus"
18, "Frau Schmidt"
…
```

`aufgaben.csv`:

```
"Kaffee kochen", "Jeden Morgen bis 9 Uhr eine Kanne […]"
"Druckerpapier auffüllen", "Einmal täglich Papierstand […]"
"Einladung verschicken", "Vor jedem Meeting Mails an […]"
…
```

`mitarbeiter_aufgaben.csv`:

```
17, "Kaffee kochen"
17, "Einladungen verschicken"
18, "Druckerpapier auffüllen"
18, "Tafel wischen"
18, "Weihnachtsfeier planen"
…
```

Ihre Datenbank bietet eine Funktion an, CSV-Textdateien (wie die oben gezeigten Dateien) in bereits erstellte leere Tabellen zu importieren (vorausgesetzt die Anzahl der Spalten und deren Datentypen stimmen überein).

Aufgabe: Geben Sie **`CREATE TABLE`**-Statements an, mit denen die zu den obigen CSV-Dateien passenden Tabellen angelegt werden.
Beachten Sie dabei:

- Vergeben Sie geeignete Primärschlüssel sowie Fremdschlüssel!
- Geben Sie jedem Fremdschlüssel-Constraint einen expliziten Namen!

c) **(1P)** Ihre Datenbank kann die CSV-Textdateien nun einzeln importieren. In welcher Reihenfolge müssen Sie die Dateien nun importieren?

d) **(3P)** Sie importieren die Dateien in der korrekten Reihenfolge. Beim Einlesen der mitarbeiter_aufgaben.csv erhalten sie jedoch die Meldung, dass die entsprechenden Fremdschlüssel-Constraints in der mitarbeiter_aufgaben-Tabelle nicht erfüllt sind. Irgendetwas stimmt nicht! Um dem Problem auf den Grund zu gehen, entschließen Sie sich, den Import erneut ohne Fremdschlüssel-Constraints zu wiederholen.

Aufgabe: Geben Sie einen **`ALTER TABLE`**-Ausdruck an, der die Fremdschlüssel-Constraints aus der mitarbeiter_aufgaben-Tabelle entfernt (falls Ihre Constraints aus Aufgabe 6b keine Namen haben sollten, verwenden sie **`fk_mitarbeiter`** und **`fk_aufgabe`** stattdessen).

e) **(3P)** Sie haben die Fremdschlüsselbeziehungen erfolgreich entfernt und konnten die CSV-Dateien nun importieren. Nach dem Import fällt Ihnen ein Eintrag in der mitarbeiter_aufgaben-Tabelle auf, der „Herrn Biber" die Aufgabe „Kafeee kochn" zuteilt. Offenbar wurde die ursprüngliche Excel-Datei per Hand geführt und es haben sich Rechtschreibfehler eingeschlichen!

Aufgabe: Geben Sie ein SQL-Ausdruck an, um alle Zeilen in der mitarbeiter_aufgaben**-**Tabelle zu finden, die einen „Tippfehler" enthalten (also, einen Ausdruck der jene Zeilen mit Aufgaben findet, die nicht in der aufgaben –Tabelle existieren)

f) **(3P)** Sie haben alle fehlerhaften Aufgaben von Hand überprüft und korrigiert, und danach die Fremdschlüssel-Constraints wieder eingeführt. Was eine Arbeit!
Zuletzt soll nun eine Sicht (View) erstellt werden, in welcher für **jeden** Mitarbeiter die Anzahl der Aufgaben welche ihm zugewiesen wurden angezeigt werden sollen.

Aufgabe: Geben Sie ein SQL-Statement an, welches eine solche View erzeugt.

g) **(3P)** Leider stellte sich nach einiger Zeit heraus, dass Herr Biber wirklich keinen Kaffee kochen kann – sein Gebräu schmeckt stets scheußlich! Man beauftragt Sie dauerhaft sicherzustellen, dass Herr Biber niemals wieder zum Kaffee kochen eingeteilt werden kann.

Aufgabe: Schreiben Sie einen Trigger der verhindert das Herr Biber zum Kaffee kochen eingeteilt wird (falls noch ein weiterer Mitarbeiter Herr Biber heißen sollte, so hat dieser Glück gehabt: auch er muss dann nie mehr wieder Kaffee kochen).

h) **(2P)** Sie haben es geschafft! Die Excel-Tabellen sind nun feinsäuberlich in der Datenbank! Glückwunsch!
Nun soll eine Java-Applikation mit einem Benutzerinterface für die von Ihnen erstellte Datenbank geschrieben werden. Ihre Kollegen streiten ob man den Zugriff auf die Datenbank direkt mit JDBC oder doch besser mit Hilfe eines Persistenz-Frameworks realisieren soll.

Aufgabe: Diskutieren Sie kurz die Vor- und Nachteile der Verwendung eines Persistenz-Frameworks.

Exercise 12.2 DDL und Triggers (20 Punkte)

1) **(10P)** Geben Sie `CREATE TABLE` Ausdrücke in SQL an, um die Tabellen zu folgendem Relationalen Modell in der Datenbank zu erstellen. Stellen Sie dabei sicher, dass die unten aufgeführten Bedingungen erfüllt sind:

Antwort (original →Tweet, antwort →Tweet)
Tweet(id, text, datum, uhrzeit, benutzer →Benutzer)
Benutzer(id, name, alter, land)
Hashtag(id, name)
Hashtag_genutzt(tweet → Tweet, hashtag →Hashtag, istPositiv)
Erklärungen:

- Datum hat das Format JJJJMMDD. Zum Beispiel 20170317 für das heutige Datum.
- Uhrzeit hat das Format SSMM. Zum Beispiel 1230 für 12:30 Uhr.
- istPostitiv gibt an, ob ein Hashtag entweder positiv oder negativ genutzt wird. Er hat den Wert -1 wenn er negativ ist und 1 wenn er positiv ist.

Bedingungen:

- Das Alter von Personen liegt zwischen 18 und 120 Jahren.
- Benutzer müssen bei der Registrierung ihr Alter angeben.
- Usernamen beginnen immer mit einem „@“.
- Wenn ein Benutzer seinen Account löscht, dann werden automatisch alle seine Tweets gelöscht. Antworten auf seine Tweets, die nicht von ihm stammen werden davon nicht beeinflusst.
- istPositiv kann nur die Werte -1 oder 1 annehmen.
- Der Name eines Hashtags beginnt immer mit einem „#“.
- Der Text eines Tweets hat eine maximale Länge von 140 Zeichen.

In dieser Aufgabe sollen Sie Trigger verwenden, um weitere Integritätsbedingungen für das gegebene Datenbankschema sicherzustellen.

1. **(5P)** Wir möchten sicherstellen, dass nachträgliche Änderungen der Tweets für alle sichtbar sind, indem wir einen einfachen Verlauf dokumentieren. Erstellen Sie einen Trigger der beim Update eines Tweet Eintrags überprüft ob sich der Text des Tweets geändert hat. Wenn ja, soll das Textfeld des Tweets wie folgt um den neuen Text ergänzt werden:
   `UPDATE: <update Text> ORIGINAL: <alter Text>`
   Beispiel: Der original Tweet-Text ist „Bad Hombres“, das Update hat als Text „Bad Guys“, dann soll der neue Text sein: „UPDATE: Bad Guys ORIGINAL: Bad Hombres“

2. **(5P)** Erstellen Sie einen Trigger, der bei jedem Eintrag eines neuen Tweets in die Datenbank überprüft, ob dies der erste Tweet dieses Nutzers ist. Falls ja, so soll in der Tabelle Hashtag_genutzt ein Eintrag zwischen dem aktuellen Tweet und dem Hashtag #FirstTweet erstellt werden. Gehen sie davon aus, dass der Hashtag #FirstTweet bereits existiert, die Id 1 hat und natürlich positiv genutzt wird.