Exercise Sheet 9: SQL II (until Thursday, 21.12.2017) (34 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!**

BITTE BEACHTEN SIE: DIESES HAUSAUFGABE MUSS PER E-MAIL ALS TXT AN IHREN HIWI GESENDET WERDEN

Aufgabe 9.1 – DDL (10 Punkte)

Aufgabe: 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:

Relationales Modell:

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

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.

Aufgabe 9.2 – DDL (12 Punkte)

Geben Sie DDL Ausdrücke an, die alle nötigen Tabellen für die Speicherung der in **Anhang A** beschriebenen Daten erstellen bzw. verändern. Beachten Sie dabei folgendes:

- Die Ausdrücke sollen **in der Reihenfolge**, in der sie aufgeschrieben wurden ausführbar sein. Es kann beispielsweise auf kein Fremdschlüssel Bezug genommen werden, der nicht zuvor eingeführt wurde.
- **Datentypen** sollen sinnvoll selbst gewählt werden
- Alle aus dem ER Diagramm und der Dokumentation hervorgehenden **Constraints** sollen so gut es geht abgebildet werden
- Alle **nicht abgebildeten Constraints** sollen annotiert werden
- Für alle Attribute, bei denen es möglich ist **`NULL`**-Werte zu verbieten, soll dies auch getan werden. Nur wenn die Datenbank durch die Einführung eines entsprechenden Constraints nicht mehr sinnvoll nutzbar wäre, soll dieser nicht eingeführt werden
- Beim Löschen bestimmter Daten sollen **Kaskaden** ausgeführt werden:
    - Beim Löschen eines Projekts sollen auch alle entsprechenden „Angestellter arbeitet an Projekt“ Tupel, sowie alle Tickets zum Projekt gelöscht werden
    - Beim Löschen eines Tickets, sollen auch alle entsprechenden „Angestellter arbeitet an Ticket“ Tupel gelöscht werden
    - Das Löschen eines Angestellten soll fehlschlagen, wenn er Leiter einer Abteilung ist. Falls er kein Leiter einer Abteilung ist, sollen alle entsprechenden „Angestellter arbeitet an Ticket“ Tupel sowie alle entsprechenden „Angestellter arbeitet an Projekt“ Tupel mit gelöscht werden
- Die Tabelle `angestellter` ist vorgegeben:

```
CREATE TABLE angestellter(
    ang_nr INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    telefonnummer VARCHAR(255) NOT NULL,
    abteilung_haus CHAR(1) NOT NULL,
    abteilung_nr INTEGER NOT NULL
)
```

Aufgabe 9.3 – Constraints (4 Punkte)

In dieser Aufgabe ist eine andere Variante der Tabelle `angestellter` vorgegeben. Hier wurde die Eigenschaft, dass ein Angestellter maximal an 3 Projekten arbeiten darf über drei Projektattribute in das Tabellenschema übernommen. Fügen Sie über ein `ALTER TABLE` Ausdruck einen Constraint hinzu, der dafür sorgt, dass die Projekte in der richtigen Reihenfolge angegeben werden müssen (also `projekt_2` erst gesetzt werden kann, wenn `projekt_1` gesetzt ist und `projekt_3` erst gesetzt werden kann, wenn `projekt_1` und `projekt_2` gesetzt sind).

```
CREATE TABLE angestellter(
  ang_nr INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  telefonnummer VARCHAR(255) NOT NULL,
  abteilung_haus CHAR(1) NOT NULL,
  abteilung_nr INTEGER NOT NULL,
  projekt_1 VARCHAR(255) DEFAULT NULL,
  projekt_2 VARCHAR(255) DEFAULT NULL,
  projekt_3 VARCHAR(255) DEFAULT NULL
)
```

Aufgabe 9.4 – DML (8 Punkte)

a) Verringern Sie das Budget von allen Abteilungen im Haus ‚C' um 20%. **(1 Punkt)**

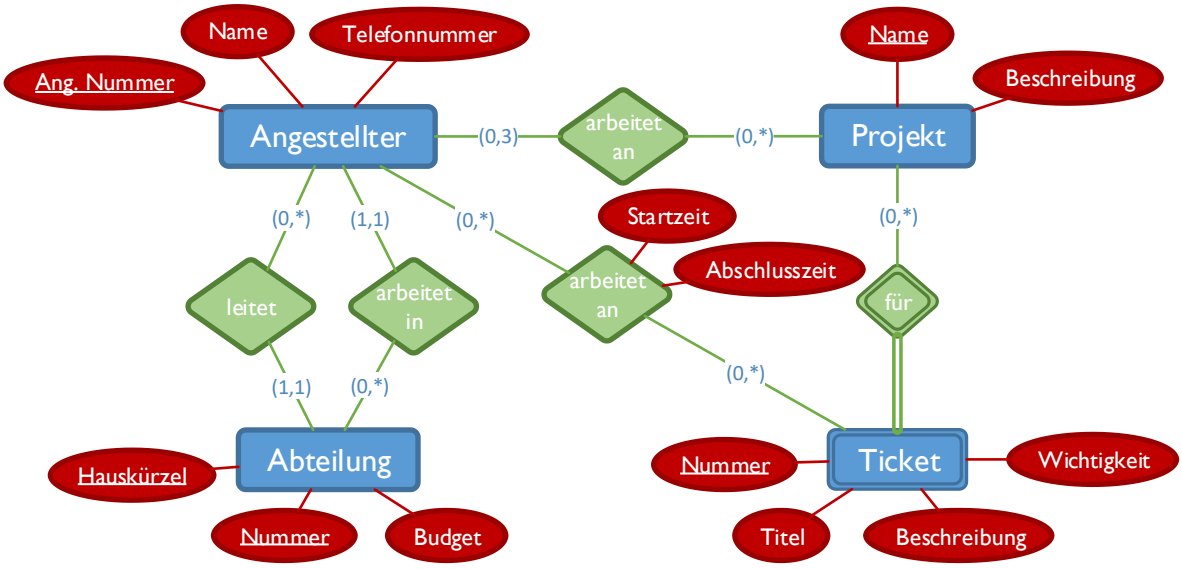
b) Löschen Sie alle Angestellten, die gerade an keinem Projekt arbeiten **(2 Punkte)**

c) Stellen Sie eine Anfrage in SQL, die zu jedem Ticket (Projektname, Nummer) einen Status angibt. Der Status soll ‚Unbearbeitet', ‚Wird bearbeitet' oder ‚Abgeschlossen' sein. Ein Ticket wird bearbeitet, wenn ein Tupel in der ‚Angestellter arbeitet an Ticket' Relation vorhanden ist und keine Abschlusszeit existiert. Existiert eine Abschlusszeit, ist das Ticket abgeschlossen. **(5 Punkte)**

Hinweis: Benutzen Sie für diese Anfrage die `CASE`-Klausel

Anhang A

Das folgende Schema beschreibt eine einfache Firmendatenbank



Dokumentation

- Die Firma ist in mehrere *Abteilungen* unterteilt. Die *Abteilungen* werden aus einer Kombination aus *Hauskürzel* und *Nummer* identifiziert
  - z.B. Abteilung ‚A-7' für ‚Haus A Nummer 7'
- Das Budget einer Abteilung wird in Eurocent angegeben und ist standardmäßig 0,00€
- Jede *Abteilung* wird von einem *Angestellten geleitet* und jeder *Angestellter* arbeitet in genau einer *Abteilung*
  - *Angestellte* arbeiten außerdem an *Projekten* (maximal 3 verschiedene)
- Zu einem *Projekt* können *Tickets* angelegt werden
  - *Ticketnummern* sind nur eindeutig innerhalb des *Projekts*
  - Die *Wichtigkeit* kann *‚niedrig', ‚normal'* oder *‚dringend'* sein
  - Die Zeit zu der ein *Angestellter* anfängt ein *Ticket* zu bearbeiten anfängt, wird als *Startzeit* eingetragen
  - Sobald der *Angestellte* fertig ist, wird eine *Endzeit* eingetragen
  - Sobald eine *Endzeit* eingetragen ist, gilt das *Ticket* als *‚abgeschlossen'*