[4131658] Jan Germann
[4054962] Christian Ratz
Übungsgruppe 1

Relationale Datenbanken
19. Dezember 2012

# Aufgabenblatt 9

## 9.1 Aufgabe

Based on the given conceptual schema, please provide SQL statements to create the according tables described in the schema.

```
1  -- Employee gehort zu einer bestimmten Abteilung.
2  CREATE TABLE Employee (
3    empNr         int(11) NOT NULL AUTO_INCREMENT,
4    name          varchar(255),
5    department    int(10) NOT NULL,
6    departmentnr int(10) NOT NULL,
7    PRIMARY KEY (empNr));
8
9  -- Telefonnummer kann mehrwertig sein.
10 -- Also Tabelle fur Telefonnummer erstellen
11 CREATE TABLE PhoneNumbers (
12   employeeempnr int(11) NOT NULL,
13   phoneNr        varchar(255));
14
15 -- Die Abteilung hat einen Leiter
16 CREATE TABLE Department (
17   house    int(10) NOT NULL,
18   nr       int(10) NOT NULL,
19   manager int(11) NOT NULL,
20   budget   int(10),
21   PRIMARY KEY (house, nr));
22
23 -- Das Projekt wird uber eine Relationentabelle mit den
24 -- Mitarbeitern verknuepft.
25 CREATE TABLE Project (
26   name         varchar(255) NOT NULL,
27   description varchar(255),
28   PRIMARY KEY (name));
29
30 -- Relationentabelle um Mitarbeiter mit einem Projekt zu verknuepfen
31 CREATE TABLE Employee_Project (
32   employeeempnr int(11) NOT NULL,
33   projectname    varchar(255) NOT NULL,
34   PRIMARY KEY (employeeempnr, projectname));
35
36 -- Die offenen Tickets werden hier gespeichert.
37 CREATE TABLE Ticket (
38   nr           int(10) NOT NULL AUTO_INCREMENT,
```

```
39    title       varchar(255),
40    description varchar(255),
41    importance  int(10),
42    PRIMARY KEY (nr));
43
44  -- Relationentabelle um Tickets einem Projekt zuzuordnen
45  CREATE TABLE Project_Ticket (
46    projectname varchar(255) NOT NULL,
47    ticketnr    int(10) NOT NULL,
48    PRIMARY KEY (projectname,
49    ticketnr));
50
51  -- Relationentabelle um die Arbeit eines Mitarbeiters
52  -- an einem Ticket zu dokumentieren.
53  CREATE TABLE Employee_Ticket (
54    employeeempnr int(11) NOT NULL,
55    ticketnr      int(10) NOT NULL,
56    start         timestamp NOT NULL,
57    end           timestamp NULL,
58    PRIMARY KEY (employeeempnr,
59    ticketnr));
60
61  -- Fremdschluesselabhaengigkeiten erstellen
62  ALTER TABLE PhoneNumbers
63    ADD INDEX FKPhoneNumbe658372 (employeeempnr),
64    ADD CONSTRAINT FKPhoneNumbe658372
65    FOREIGN KEY (employeeempnr) REFERENCES Employee (empNr);
66  ALTER TABLE Employee_Project
67    ADD INDEX FKEmployee_P990046 (employeeempnr),
68    ADD CONSTRAINT FKEmployee_P990046
69    FOREIGN KEY (employeeempnr) REFERENCES Employee (empNr);
70  ALTER TABLE Employee_Project
71    ADD INDEX FKEmployee_P943600 (projectname),
72    ADD CONSTRAINT FKEmployee_P943600
73    FOREIGN KEY (projectname) REFERENCES Project (name);
74  ALTER TABLE Department
75    ADD INDEX FKDepartment689529 (manager),
76    ADD CONSTRAINT FKDepartment689529
77    FOREIGN KEY (manager) REFERENCES Employee (empNr);
78  ALTER TABLE Employee
79    ADD INDEX FKEmployee366382 (department, departmentnr),
80    ADD CONSTRAINT FKEmployee366382
81    FOREIGN KEY (department, departmentnr) REFERENCES Department (house, nr);
82  ALTER TABLE Project_Ticket
83    ADD INDEX FKProject_Ti16548 (projectname),
84    ADD CONSTRAINT FKProject_Ti16548
85  FOREIGN KEY (projectname) REFERENCES Project (name);
86  ALTER TABLE Project_Ticket
87    ADD INDEX FKProject_Ti360860 (ticketnr),
88    ADD CONSTRAINT FKProject_Ti360860
89    FOREIGN KEY (ticketnr) REFERENCES Ticket (nr);
```

```
90 ALTER TABLE Employee_Ticket
91   ADD INDEX FKEmployee_T249102 (employeeempnr),
92   ADD CONSTRAINT FKEmployee_T249102
93   FOREIGN KEY (employeeempnr) REFERENCES Employee (empNr);
94 ALTER TABLE Employee_Ticket
95   ADD INDEX FKEmployee_T659766 (ticketnr),
96   ADD CONSTRAINT FKEmployee_T659766
97   FOREIGN KEY (ticketnr) REFERENCES Ticket (nr);
```

## 9.2 Aufgabe

Based on the given statements and data, explain the consequences of the following operations:

a)
```
INSERT INTO connection VALUES(2, 5, 'sequel')
```

Der Primärschlüssel der aus `from_movie, to_movie` besteht, ist bereits vorhanden und mit dem `connection_type` 'parody' belegt. Daher wird bei diesem INSERT-Statement nichts verändert.

b)
```
DELETE FROM actor WHERE role = 'forest␣ranger'
```

Alle Zeilen in der Relation **actor** in denen ein Schauspieler die `role` 'forest ranker' besitzt, werden gelöscht. Alle anderen Relationen werden nicht verändert.

c)
```
DELETE FROM movie WHERE title = 'Adventures␣with␣RDB'
```

Der Eintrag über den Film 'Adventures with RDB' wird nicht entfernt, da die Standardeinstellung (siehe Vorlesungsfolie 20) bei Löschung keine Aktion ausgeführt werden soll, sowie bei Änderung eines Eintrag ebenfalls nichts ausgeführt werden soll.

d)
```
INSERT INTO actor VALUES(6, 85, 'important␣looking␣man')
```

Das Einfügen der Werte (6, 85, 'important looking man') in die Relation **actor** funktioniert nicht, da die `id` '6' in der Relation **person** und in der Relation **movie** die `id` '85' nicht existiert. Daher bleibt die Datenbank unverändert.

e)
```
DROP TABLE person
```

Die gesamte Tabelle **person** wird aus der Datenbank gelöscht. Alle restlichen Relationen bleibt unverändert, nur die Tabelle **person** wird aus der Datenbank entfernt.