PerFact
INNOVATION

…stell’ dir vor, es geht!

Assessment-Center

Aufgabenstellungen

PerFact Innovation GmbH & Co. KG

Version 1.1 vom 29.03.2016

Datei-ID:

PerFact Innovation GmbH & Co. KG – Assessment-Center

Aufgabenstellungen

Lars Bergmann

Version 1.1 vom 29.03.2016

Inhaltsverzeichnis

1 SQL **2**

1.1 Aufgabe 1 . . . 2
1.2 Aufgabe 2 . . . 2
1.3 Aufgabe 3 . . . 2
1.4 Aufgabe 4 . . . 2
1.5 Aufgabe 5 . . . 2
1.6 Aufgabe 6a . . . 3
1.7 Aufgabe 6b . . . 3

2 Lösungen **4**

2.1 4.1 . . . 4
2.2 4.2 . . . 4
2.3 4.3 . . . 4
2.4 4.4 . . . 4
2.5 4.5 . . . 5
2.6 4.6 . . . 5
2.7 4.7 . . . 6

1 SQL

Film-Datenbank als Beispiel-DB

Diese DB kann mit dem sqlitebrowser geöffnet und manipuliert werden.

1.1 Aufgabe 1

Notiere SELECT-Statements. die folgende Fragen beantworten:

- Alle Filme, die „Star Wars“ im Namen haben
- Alle Schaupieler, die „Ford“ im Namen haben
- Alle Rollen, die „Batman“ im Namen haben

1.2 Aufgabe 2

Notiere zu folgenden Aufgabenstellungen INSERT-Statements:

- Lege den Film „Antman“ aus dem Jahr 2015 an –> Daten von Wikipedia o.Ä.
- Lege die Rolle „Antman“ an und verknüpfe sie mit dem Film und dem dazugehörigen Schauspieler

1.3 Aufgabe 3

Notiere zu folgenden Aufgabenstelungen jeweils **ein** SELECT-Statement, ggf. mit JOIN:

- Liste alle Schauspieler und Rollen zum Film „Pulp Fiction“ auf
- Liste chronologisch alle Rollen und Filme auf, in denen „Michael Fassbender“ mitgespielt hat
- Liste alphabetisch alle Schauspieler mit Filmen auf, die die Rolle „Batman“ gespielt haben

1.4 Aufgabe 4

Die vorliegende Datenbank ist nicht vollständig normalisiert. Nenne mindestens eine Tabelle bzw. Spalte, die nicht in der ersten Normalform vorliegt. Beschreibe, was geändert werden muss, um die erste Normalform zu erreichen.

1.5 Aufgabe 5

Notiere eine Sequenz von SQL-Statements, mit denen die Normalisierung aus Aufgabe 4 durchgeführt werden kann.

Hinweis: Es muss keine neue Tabelle erzeugt werden. Es genügt, eine vorhandene Tabelle zu modifizieren.

1.6 Aufgabe 6a

Skizziere eine Datenstruktur, die folgende Anforderungen erfüllt:

- Eine Bücherei möchte ihre Bücher verwalten, Bücher haben Namen, Autoren und ein Erscheinungsjahr
- Für jedes Buch soll gespeichert werden, in welchem Regal es steht. In einem Regal können mehrere Bücher stehen
- In einer weiteren Tabelle sollen Leihvorgänge gespeichert werden. Aus dieser Tabelle soll ersichtlich werden, ob ein Buch vorliegt oder zur Zeit ausgeliehen ist.
- Personen, die Bücher leihen, werden mit Namen und Geburtsdatum gespeichert und mit den Leihvorgängen verknüpft
- Regale stehen auf verschiedenen Etagen. Einer Etage werden die entsprechenden Regale zugeordnet. Auf diesem Wege soll es möglich sein, den Standort eines Buches mit Etage und Regal zu bestimmen.

Hinweis: Ein skizzenhaftes ER-Diagramm in Chen- oder UML-Schreibweise ist zur Erfüllung der Aufgabe ausreichend!

1.7 Aufgabe 6b

Formuliere CREATE TABLE-Statements für die erstellte Datenstruktur!

2 Lösungen

2.1 4.1

```
SELECT * FROM movie WHERE movie.title LIKE '%Star Wars%'
SELECT * FROM actor WHERE actor.name LIKE '%Ford%'
SELECT * FROM role WHERE role.name LIKE '%Batman%'
```

2.2 4.2

```
INSERT INTO movie (id, title, year) VALUES (<freie ID>, 'Antman', 2015)

INSERT INTO role (movie_id, actor_id, name)
VALUES ((SELECT last_insert_rowid()),
(SELECT id FROM actor WHERE actor.name = 'Rudd, Paul' order by 1 desc limit 1),
'Antman')
```

2.3 4.3

```
SELECT * FROM movie
JOIN role ON role.movie_id = movie.id
JOIN actor ON actor.id = role.actor_id
WHERE movie.title = 'Pulp Fiction'

SELECT * FROM actor
JOIN role ON role.actor_id = actor.id
JOIN movie ON movie.id = role.movie_id
WHERE actor.name = 'Fassbender, Michael'
ORDER BY movie.year DESC

SELECT * FROM role
JOIN movie ON movie.id = role.movie_id
JOIN actor ON actor.id = role.actor_id
WHERE role.name = 'Batman'
ORDER BY actor.name
```

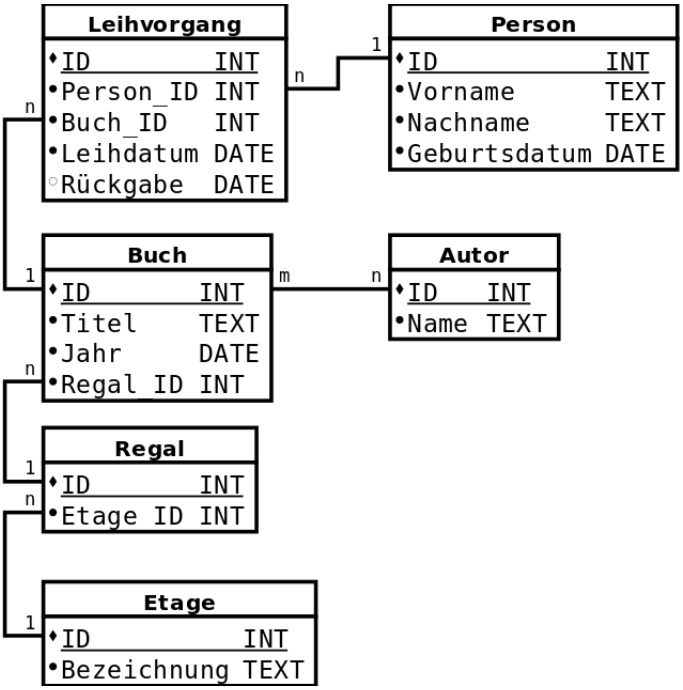
2.4 4.4

```
In der Tabelle actor werden Vor- und Nachname in der gleichen Spalte gespeichert.
Das verletzt die erste Normalform, nur atomare Daten sind zulässig, keine Listen!
```

2.5 4.5

```
ALTER TABLE actor ADD COLUMN first_name TEXT
ALTER TABLE actor ADD COLUMN last_name TEXT
UPDATE actor SET first_name = SUBSTR(name, 1, INSTR(name, ',') - 1)
UPDATE actor SET last_name = SUBSTR(name, INSTR(name, ',') + 1)
```

2.6 4.6



2.7 4.7

```
CREATE TABLE `Etage` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Bezeichnung`TEXT NOT NULL
);

CREATE TABLE `Regal` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Etage_ID`INTEGER NOT NULL,
FOREIGN KEY(`Etage_ID`) REFERENCES Etage(ID)
);

CREATE TABLE `Autor` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Name`TEXT NOT NULL
);

CREATE TABLE `Buch` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Titel`TEXT NOT NULL,
`Jahr`NUMERIC NOT NULL,
`Regal_ID`INTEGER NOT NULL,
FOREIGN KEY(`Regal_ID`) REFERENCES Regal(ID)
);

CREATE TABLE `BuchxAutor` (
`Buch_ID`INTEGER NOT NULL,
`Autor_ID`INTEGER NOT NULL,
PRIMARY KEY(Buch_ID,Autor_ID),
FOREIGN KEY(`Buch_ID`) REFERENCES Buch(ID),
FOREIGN KEY(`Autor_ID`) REFERENCES Autor(ID)
);

CREATE TABLE `Person` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Vorname`TEXT NOT NULL,
`Nachname`TEXT NOT NULL,
`Geburtsdatum`DATE NOT NULL
);

CREATE TABLE `Leihvorgang` (
`ID`INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
`Person_ID`INTEGER NOT NULL,
`Buch_ID`INTEGER NOT NULL,
`Leihdatum`DATE NOT NULL,
`Rückgabe`DATE,
FOREIGN KEY(`Person_ID`) REFERENCES Person(ID),
FOREIGN KEY(`Buch_ID`) REFERENCES Buch(ID)
);
```