

Aufgabe 2

Gegeben sei der folgende Ausschnitt eines Schemas für die Verwaltung von Kollektionen:

Promi : | Kleidungsstueck : | Name : VARCHAR(255), ID : INTEGER, Alter : INTEGER, Hauptbestandteil : VARCHAR(255), Wohnort : VARCHAR.(255) gehoert_zu : VARCHAR.(255)

I 1

Kollektion : | Name : VARCHAR(255), Jahr : INTEGER,

Saison : VARCHAR(255) hat_getragen : [] PromiName : VARCHAR.(355), KleidungsstueckID : INTEGER, promotet : | Datum : DATE PromiName : VARCHAR(255), 1

KollektionName : VARCHAR(255)

Die Tabelle Promi beschreibt Promis über ihren eindeutigen Namen, ihr Alter und ihren Wohnort. Kollektion enthält Informationen über Kollektionen, nämlich deren eindeutigen Namen, das Jahr und die Saison. Die Tabelle promotet verwaltet über Referenzen, welcher Promi welche Kollektion promotet. Kleidungsstück speichert die IDs von Kleidungsstücken zusammen mit dem Hauptbestandteil und einer Referenz auf die zugehörige Kollektion. Die Tabelle hat_getragen verwaltet über Referenzen, welcher Promi welches Kleidungsstück an welchem Datum getragen hat.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

- (a) Schreiben Sie SQL-Anweisungen, welche die Tabelle hat_getragen inklusive aller benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalte Datum Teil des Primärschlüssels ist.

```
1 CREATE TABLE hat_getragen (  
2   PromiName VARCHAR(255) REFERENCES Promi.Name,  
3   KleidungsstueckID INTEGER REFERENCES Kleidungsstueck.ID,  
4   Datum DATE,  
5   PRIMARY KEY (PromiName, KleidungsstueckID, Datum)  
6 )
```

Das Datenbanksystem achtet selbst darauf, dass Felder des Primärschlüssels nicht NULL sind. Deshalb muss man NOT NULL bei keinem der drei Felder angeben.

- (b) Schreiben Sie eine SQL-Anweisung, welche die Namen der Promis ausgibt, die eine Sommer-Kollektion promoten (Saison ist "Sommer").

```
1 SELECT P.PromiName FROM promotet P JOIN Kollektion K ON  
2   P.KollektionName = K.Name  
3 WHERE K.Saison = 'Sommer'  
4   SELECT P.PromiName FROM promotet P, Kollektion K  
5 WHERE K.Saison = 'Sommer' AND P.KollektionName = K.Name;
```

- (c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Promis und der Kollektionen bestimmt, welche der Promi zwar promotet, aber daraus noch kein Kleidungsstück getragen hat.

```

1  SELECT * FROM promotet P
2  WHERE P.KollektionName NOT IN (SELECT K.Name FROM Kollektion K
3  INNER JOIN Kleidungsstueck S ON S.gehoert_zu = K.Name
4  INNER JOIN hat_getragen T ON T.KleidungsstueckID = S.ID
5  WHERE T.PromiName = P.PromiName)

1  SELECT P.PromiName, P.KollektionName FROM promotet P) EXCEPT
2  (SELECT P.PromiName, P.KollektionName FROM promotet P
3  INNER JOIN Kollektion K ON P.KollektionName = K.Name
4  INNER JOIN Kleidungsstueck S ON S.gehoert_zu = K.Name
5  INNER JOIN hat_getragen T ON T.KleidungsstueckID = S.ID
6  WHERE T.PromiName = P.PromiName

```

- (d) Bestimmen Sie für die folgenden SQL-Anweisungen die minimale und maximale Anzahl an Tupeln im Ergebnis. Beziehen Sie sich dabei auf die Größe der einzelnen Tabellen.

Verwenden Sie für die Lösung folgende Notation: — Promi — beschreibt die Größe der Tabelle Promi.

(i)

```

1  SELECT k.Name
2  FROM Kollektion k, Kleidungsstueck kl
3  WHERE k.Name = kl.gehoert_zu and k.Jahr = 2018
4  GROUP BY k.Name
5  HAVING COUNT(kl.Hauptbestandteil)>10

```

– Kollektion – beschreibt die Anzahl der Tupel in der Tabelle Kollektion. Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion den Anforderungen k.Jahr = 2018 oder HAVING COUNT(...) genügt. Die maximale Anzahl an Tupeln im Ergebnis ist – Kollektion –, da nur Namen aus Kollektion ausgewählt werden.

(ii)

```

1  SELECT DISTINCT k.Jahr
2  FROM Kollektion kK
3  WHERE k.Name IN (
4  SELECT pr.KollektionName
5  FROM Promi p, promotet pr
6  WHERE p.Alter < 30 AND pr.PromiName = p.Name

```

- (e) Beschreiben Sie den Effekt der folgenden SQL-Anfrage in natürlicher Sprache

```

1  SELECT pr.KollektionName
2  FROM promotet pr, Promi p
3  WHERE pr.PromiName = p.Name
4  GROUP BY pr.KollektionName
5  HAVING COUNT (*) IN (
6
7  SELECT MAX(anzahl)
8
9  FROM (
10 SELECT k.Name, COUNT(*) AS anzahl
11 FROM Kollektion k, promotet pr
12 WHERE k.Name = pr.KollektionName
13 GROUP BY k.Name
14
15 )

```

16
17)

Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion promotet wird. Die maximale Anzahl ist das Minimum von – Kollektion – und 30, da die Promis, die Kollektionen beworben haben, die älter als 30 Jahre sind, selbst mindestens 30 Jahre alt sein müssen.

Zugrundeliegende Annahmen: Kollektionen werden nur im Erscheinungsjahr von Promis beworben; Neugeborene, die Kollektionen bewerben, werden ggf. Promis.

- (f) Formulieren Sie folgende SQL-Anfrage in relationaler Algebra. Die Lösung kann in Baum- oder in Term-Schreibweise angegeben werden, wobei eine Schreibweise genügt.

```
1  SELECT p.Wohnort
2
3  FROM Promi p, promotet pr, Kollektion k
4  WHERE p.Name = pr.PromiName
5
6  AND k.Name = pr.KollektionName
7
8  AND k.Jahr = 2018
```

- (i) Konvertieren Sie zunächst die gegebene SQL-Anfrage in die zugehörige Anfrage in relationaler Algebra nach Standard-Algorithmus.
- (ii) Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und begründen Sie dabei kurz jeden durchgeführten Schritt.