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), KleidungsstuecklD : 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 Hauptbe- standteil 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 Du- plikate enthalten dürfen. Sie dürfen geeignete Views definieren.

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

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

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

(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)
```sql
SELECT k.Name
FROM Kollektion k, Kleidungsstueck kl
WHERE k.Name = kl.gehoert_zu and k.Jahr = 2018
GROUP BY k.Name
HAVING COUNT(k1.Hauptbestandteil)>10
```

(ii)
```sql
SELECT DISTINCT k.Jahr
FROM Kollektion kK
WHERE k.Name IN (
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.KollektionNane
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  )
```

(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 re- lationaler Algebra nach Standard-Algorithmus.

(ii) Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und be- gründen Sie dabei kurz jeden durchgeführten Schritt.