Relationale Algebra

|  |  |
| --- | --- |
| Projektion | π |
| Selektion | σ |
| Kartesisches Produkt | × |
| Verbund / Join | ⋈ |
| Division | ÷ |
|  |  |
| Durchschnitt | ∩ |
| Vereinigung | ∪ |
| AND | ˄ |
| OR | ˅ |

Aufgabe 1

Wir haben folgende Ausprägung der Relation Studenten:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SID** | **Name** | **Email** | **Age** | **Note** |
| 2833 | Jones | [jones@scs.ubbcluj.ro](mailto:jones@scs.ubbcluj.ro) | 19 | 9 |
| 2877 | Smith | [smith@scs.ubbcluj.ro](mailto:smith@scs.ubbcluj.ro) | 20 | 8 |
| 2976 | Jones | [jones@math.ubbcluj.ro](mailto:jones@math.ubbcluj.ro) | 21 | 10 |
| 2765 | Mary | [mary@math.ubbcluj.ro](mailto:mary@math.ubbcluj.ro) | 22 | 7.7 |
| 3000 | Dave | [dave@cs.ubbcluj.ro](mailto:dave@cs.ubbcluj.ro) | 18 | 5.5 |
| 3010 | Smith | [smith2@scs.ubbcluj.ro](mailto:smith2@scs.ubbcluj.ro) | 20 | 7 |
| 3020 | Sam | [sam@scs.ubbcluj.ro](mailto:sam@scs.ubbcluj.ro) | 19 | 9.5 |

1.Finde ein Attribut oder Menge von Attributen, die kein Kandidatschlüssel sein können, bzgl. dieser Ausprägung

1. Name
2. Age
3. (Name, Age)
4. (SID, Name) - nicht minimal

2.Können wir einen Kandidatschlüssel für die Relation Studenten finden, wenn wir wissen dass diese Ausprägung gültig ist?

1. SID
2. Note ? - nein

R(A, B, C, D) - nein

Aufgabe 2

Die Datenbank enthält folgende Relationen:

•**Schauspieler(ID, Name, Geburtsjahr)**

**•Filme(ID, Titel, Jahr),** wobei Jahr = wann der Film produziert wurde

**•Casting(FilmId, SchauspielerID, Charakter)** – ein Schauspieler spielt ein Charakter in einem Film

Die Primärschlüssel sind unterstrichen.

Schreibe folgenden Anfragen in SQL und/oder relationale Algebra

* Finde die Titeln der Filme produziert nach 2000 oder bevor 1950(SQL+rel.A.)

SELECT DISTINCT Titel FROM Filme WHERE Jahr > 2000 OR Jahr < 1950

Πtitel (σjahr>2000 V jahr <1950 (Filme))

2.Finde die Namen der Schauspieler die ‘Superman’ und ‘Clark Kent’ gespielt haben (SQL + rel. Alg)

**Met I**

SELECT Schauspieler.Name FROM Schauspieler

INNER JOIN Casting on Schauspieler. ID = Casting. SchauspielerID

WHERE Charakter = ‘Superman’

INTERSECT

SELECT Schauspieler.Name FROM Schauspieler

INNER JOIN Casting on Schauspieler. ID = Casting. SchauspielerID

WHERE Charakter = ‘Clark Kent’

**Met II**

SELECT Schauspieler.Name FROM Schauspieler

INNER JOIN Casting on Schauspieler. ID = Casting. SchauspielerID

WHERE

Schauspieler.ID IN (SELECT Schauspieler.ID from Casting WHERE Charakter = ‘Superman’)

AND

Schauspieler.ID IN (SELECT Schauspieler.ID from Casting WHERE Charakter = ‘Clark Kent’)

**Effizienter:**

SELECT Schauspieler.Name FROM Schauspieler

INNER JOIN Casting on Schauspieler. ID = Casting. SchauspielerID

WHERE **Charakter = ‘Superman’**

AND Schauspieler.ID IN (SELECT Schauspieler.ID from Casting WHERE Charakter = ‘Clark Kent’)

**Met IIIIII**

SELECT Schauspieler.Name FROM Schauspieler

Inner join Casting C1 on Schauspieler.ID=C1. SchauspielerID

Inner join Casting C2 on Schauspieler.ID=C2. SchauspielerID

Where C1.Character= ‘Superman’ and C2.Character=’Clark Kent’

**Ohne Join:**

From Schauspieler S, Casting C1, Casting C2

Where Schauspieler.ID=C1. SchauspielerID AND …

|Id | Name | .. | ..}SchauspielerId (C1) | Charakter (C1) | …}SchauspielerId (C2) | Charakter (C2)

ρC1 (Casting)Casting)

ρC2 (Casting)

## ΠSchauspieler.Name(σC1.Character=’Superman’ ^ C2.Character=’Clark Kent’ (Schauspieler ⋈ Schauspieler.ID = C1. SchauspielerID C1Schauspieler.ID = C1. SchauspielerID C1

⋈ Schauspieler.ID = C2. SchauspielerID C2))

## ΠSchauspieler.Name(Schauspieler ⋈ Schauspieler.ID = C1. SchauspielerID (σC1.Character=’Superman’ (C1)) (C1))

⋈ Schauspieler.ID = C2. SchauspielerID (σC2.Character= ‘Clark Kent’ C2))C2.Character= ‘Clark Kent’ C2))

3.Finde die Namen der Schauspieler die ‘Superman’ und ‘Clark Kent’ gespielt haben in derselben Produktion des Films (SQL)

SELECT Schauspieler.Name FROM Schauspieler

Inner join Casting C1 on Schauspieler.ID=C1. SchauspielerID

Inner join Casting C2 on Schauspieler.ID=C2. SchauspielerID

Where C1.Character= ‘Superman’ and C2.Character=’Clark Kent’

AND C1.FilmID = C2.FilmID

4.Finde die Namen der Schauspieler, die zusammen mit Victoria Abril in einem Film gespielt haben und die geboren wurden, nachdem Victoria in dem ersten Film gespielt hat (SQL)

SELECT s.Name FROM Schauspieler s

INNER JOIN Casting c ON s.ID = c.SchauspielerID

WHERE c.FilmID IN (SELECT ca.FilmID FROM Schauspieler sc

INNER JOIN Casting ca ON sc.ID = ca.SchauspielerID

WHERE sc.Name = ‘Victoria Abril’)

AND

s.Geburtsjahr > (SELECT TOP 1 Jahr from Filme F

Inner join Casting C2 on

C2.FIlmdID = F.ID

Inner join Schauspieler S2 on

S2.ID = C2. SchauspielerID

where S2.Name = ‘Victoria Abril’

Order by F.Jahr asc)

SELECT MIN(Jahr) from Filme F

Inner join Casting C2 on

C2.FIlmdID = F.ID

Inner join Schauspieler S2 on

S2.ID = C2. SchauspielerID

where S2.Name = ‘Victoria Abril’

1. Finde die Namen der Schauspieler die ‘Superman’ oder ‘Clark Kent’ gespielt haben.

Select distinct S.Name

From Schauspieler S, Casting C

Where C.SchaupielerId = S.Id

And (C.Charakter = ‘Superman’

OR C.Charakter = ‘Clark Kent’ )

Aufgabe 3

Die Datenbank enthält folgende Relationen:

•Lieferanten (LID, Lname, Adresse)

•Produkte (PID, Pname, Farbe)

•Katalog (LID, PID, Preis)

Die Primärschlüssel sind unterstrichen.

Katalog enthält die Preise für Produkte von unterschiedlichen Lieferanten.

Geben sie an, was die folgenden Anfragen ausgeben.

πLName(πLID(**(σFarbe = ‘rot’Produkte) ⋈ (σPreis < 100 Katalog)**) ⋈ Lieferanten)

* Lieferantennamen, die ein rotes Produkt liefern, das weniger als 100 kostet.

πLName(πLID(**(σFarbe = ‘rot’Produkte) ⋈ (σPreis < 100 Katalog) ⋈ Lieferanten)**)

πLName((σFarbe = ‘rot’ Produkte) ⋈ (σPreis < 100 Katalog) ⋈ Lieferanten) ∩

πLName((σFarbe = ‘grün’ Produkte) ⋈ (σPreis < 100 Katalog) ⋈ Lieferanten)

* Lieferantennamen, die ein rotes Produkt liefern, das weniger als 100 kostet und ein grünes Produkt, das weniger 100 kostet

(πLID, PID Katalog ) ÷ πPID Produkte

* LID des Lieferanten die jeder Produkt liefert

ρR1(Katalog) , ρR2(Katalog)

πR1.PID(σR1.PID = R2.PID ˄ R1.LID <> R2.LID (R1 × R2))

* PIDs der Produkte, die von zwei unterschiedlichen Lieferanten geliefert werden

Aufgabe 4

Um die Bücher in der Bibliothek zu speichern brauchen wir:

BuchID, ISBN, Kategorie, SerialNr, KopieNr, Titel, Author

Jedes Buch (jede Kopie) hat eine eindeutige Nummer (BuchID) den wir benutzen können um die Anleihe des Buches zu speichern. Diese Nummer ist einem Buch zugeordnet als ein Barcode, hat aber keine besondere Bedeutung.

ISBN ist eindeutig für eine Edition eines Buches.

Jedes Buch gehört zu einer Kategorie. In einer Kategorie haben die Bücher unterschiedliche SerialNr. **Wenn wir mehrere Kopien von demselben Buch haben, dann werden diese durch den KopieNr unterscheidet.**

Welche der folgenden fkt. Abh. gelten? (Nicht aus dem Text, sonder logisch in dem beschriebenen Kontext) Gebe eine kurze Erklärung dafür.

BuchID → ISBN, Kategorie, SerialNr, KopieNr - Ja

ISBN → BuchID - Nein

ISBN → Kategorie, SerialNr - Ja

ISBN → KopieNr -Nein

ISBN → Titel, Author - Ja

KopieNr → ISBN, BuchId - Nein

Aufgabe 5

Die Ausprägung einer Relation S(A, B, C) enthält folgende Tupeln:

(1,2,3)

(4,2,3)

(5,3,3)

Welche der folgenden fkt. Abh. können für die Relation S nicht gelten?

a) A → B

b) BC → A - gilt nicht 2,3 - 1 und 2,3 - 4

c) B → C

Könnt ihr fkt. Abh. identifizieren, die für die Relation S gelten?

Nein!!

**Hausaufgabe**

Aufgabe 2

1. Finde **die Namen** der Schauspieler, die “Fletcher Christian” spielen in einer Produktion des Films “Mutiny on the Bounty” und **das Produktionsjahr** des Films (SQL + rel.A.)
2. Finde die Namen der Schauspieler die zwei unterschiedliche Charaktern in derselben Filmproduktion gespielt haben (SQL)
3. Finde die Paare von unterschiedlichen Schauspielernamen, die denselben Charakter in unterschiedlichen Filmproduktionen derselben Films gespielt haben (SQL)