Lehrstuhl für Praktische Informatik III
Prof. Dr. Guido Moerkotte
Email: moer@db.informatik.uni-mannheim.de

Datenbanksysteme 1 — 4. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Gegeben sei folgendes relationales Schema:

- Angestellte(<u>PersonalNr</u>, Name, Gehalt, Beruf, AbteilNr, ChefNr, Wohnort)
- Abteilung(<u>AbteilNr</u>, AbteilName, Ort)

Formulieren Sie folgende Anfragen in relationaler Algebra und im Tupelkalkül:

Aufgabe 1 a)

Geben Sie die Nummern und Namen aller Abteilungen aus.

Lösung

$\pi_{\text{AbteilNr, AbteilName}}(\text{Abteilung})$
$\{[a.AbteilNr, a.AbteilName] | a \in Abteilung\}$

Aufgabe 1 b)

Geben Sie die Namen und Berufe aller Angestellten aus, die in Mannheim wohnen.

Lösung

$\pi_{\text{Name, Beruf}}(\sigma_{\text{Wohnort='Mannheim'}}(\text{Angestellte}))$
$\{[a.Name, a.Beruf] | a \in \text{Angestellte} \wedge a.Wohnort =' Mannheim'\}$

Aufgabe 1 c)

Listen Sie Name, Gehalt und Abteilungsname aller Programmierer auf, die in Darmstadt beschäftigt sind.

Lösung

$\pi_{\text{Name, Gehalt, AbteilName}}(\sigma_{\text{Ort='Darmstadt} \wedge \text{ Beruf='Programmierer'}}($
Angestellte $\bowtie$ Abteilung))
$\{[a.Name, a.Gehalt, d.AbteilName] | a \in \text{Angestellte} \wedge d \in \text{Abteilung} \wedge a.AbteilNr = d.AbteilNr \wedge d.Ort =' Darmstadt' \wedge a.beruf = `Programmierer'\}$

Aufgabe 1 d)

Welche Angestellten verdienen mehr als ihre direkten Chefs?

Lösung

$\pi_{\text{A.PersonalNr, A.Name}}(\sigma_{\text{A.Gehalt} > \text{B.Gehalt}}($
$\rho_{\text{A}}(\text{Angestellte}) \bowtie_{\text{A.ChefNr} = \text{B.PersonalNr}} \rho_{\text{B}}(\text{Angestellte})))$

$\{a | a \in \text{Angestellte} \wedge \exists b \in \text{Angestellte}(a.ChefNr = b.PersonalNr \wedge a.Gehalt > b.Gehalt)\}$

Aufgabe 1 e)

Welche Abteilungen haben keine Angestellten?

Lösung

$\pi_{\text{AbteilNr}}(\text{Abteilung}) \setminus \pi_{\text{AbteilNr}}(\text{Angestellte})$

$\{d | d \in \text{Abteilung} \wedge \neg\exists a \in \text{Angestellte}(d.AbteilNr = a.AbteilNr)\}$

Aufgabe 1 f)

In welchen Abteilungen sind alle Berufe des Unternehmens vertreten?

Lösung

$(\pi_{\text{AbteilNr, AbteilName, Beruf}}(\text{Angestellte} \bowtie \text{Abteilung})) \div$
$\pi_{\text{Beruf}}(\text{Angestellte})$

$\{d | d \in \text{Abteilung} \wedge \forall x \in \text{Angestellte } (\exists a \in \text{Angestellte}(x.Beruf = a.Beruf \wedge a.AbteilNr = d.AbteilNr))\}$

Aufgabe 2

Formulieren Sie die Anfragen für folgendes relationales Schema mit Hilfe des Tupelkalkulus.

Rebsorte	{[<u>Sorte</u>, Name, Farbe]}
Wein	{[<u>ID</u>, Name, RSorte]}
Jahrgang	{[<u>WeinID, Jahr</u>, Preis, Qualität]}

Hier sind zur Wiederholung die Anfragen.

Aufgabe 2 a)

Geben die Farbe der Rebsorte Merlot an.

Lösung

$\{[r.Farbe] | r \in \text{Rebsorte} \wedge r.Name = \text{“Merlot”}\}$

Aufgabe 2 b)

Aus welcher Rebsorte ist der Wein mit dem Namen ”Bassermann-Jordan” gemacht?

Lösung

$\{[r.Name] | r \in \text{Rebsorte} \wedge \exists w \in \text{Wein}(r.Sorte = w.RSorte \wedge w.Name = \text{”}BJ\text{”})\}$

Aufgabe 2 c)

Listen Sie den Namen und die Preise aller Riesling-Weine auf.

Lösung

$\{[w.Name, j.Preis] | w \in \text{Wein} \wedge j \in \text{Jahrgang} \wedge w.ID = j.WeinID \wedge \exists r \in \text{Rebsorte}(r.Sorte = w.RSorte \wedge r.Name = \text{Riesling”})\}$

Aufgabe 2 d)

Für welche Rebsorten sind keine Weine in der Datenbank? Geben Sie deren Namen an.

Lösung

$\{[r.Name] | r \in \text{Rebsorte} \wedge \neg\exists w \in \text{Wein}(r.Sorte = w.RSorte)\}$

Aufgabe 2 e)

Welche Weine haben ausschliesslich exzellente Qualität (Qualität = 1)? Listen Sie deren IDs auf.

Lösung

$\{[w.id] | w \in \text{Wein} \wedge \forall j \in \text{Jahrgang}(w.ID = j.WeinID \implies j.Qualität = 1)\}$

Aufgabe 3

Formulieren Sie die folgende Anfrage mit Hilfe des Tupelkalkulus ohne Verwendung des Allquantors $\forall$.

“Geben Sie die Namen der Studenten an, die die Vorlesung ’Bioethik’ besuchen und alle Vorlesungen besucht haben, die direkt von dieser Vorlesung vorausgesetzt werden.”

Übersetzen Sie danach den Ausdruck in SQL! Testen Sie Ihre Anfragen in der SQL-Schnittstelle auf unserer Webseite (Wählen Sie die Datenbank *unidb* aus).

Lösung

Formulierung mit Allquantor:

$$\{[s.Name] \mid s \in Studenten \wedge \exists h \in hören(s.MatrNr = h.MatrNr \wedge \exists v \in Vorlesungen(h.VorlNr = v.VorlNr \wedge v.Titel =' Bioethik' \wedge \forall vs \in voraussetzen(vs.Nachfolger = v.VorlNr \implies \exists h2 \in hören(h2.MatrNr = s.MatrNr \wedge h2.VorlNr = vs.Vorgänger))))\}$$

Verwende: $\forall t \in R(P(t)) = \neg(\exists t \in R(\neg P(t)))$ und Definition von $\implies$ ($L \implies R = \neg L \vee R$)

$$\{[s.Name] \mid s \in Studenten \wedge \exists h \in hören(s.MatrNr = h.MatrNr \wedge \exists v \in Vorlesungen(h.VorlNr = v.VorlNr \wedge v.Titel =' Bioethik' \wedge \neg(\exists vs \in voraussetzen(\neg(\neg(vs.Nachfolger = v.VorlNr) \vee \exists h2 \in hören(h2.MatrNr = s.MatrNr \wedge h2.VorlNr = vs.Vorgänger))))))\}$$

mit De Morgans Regel $\neg(L \vee R) = \neg L \wedge \neg R$:

$$\{[s.Name] \mid s \in Studenten \wedge \exists h \in hören(s.MatrNr = h.MatrNr \wedge \exists v \in Vorlesungen(h.VorlNr = v.VorlNr \wedge v.Titel =' Bioethik' \wedge \neg\exists vs \in voraussetzen(vs.Nachfolger = v.VorlNr \wedge \neg(\exists h2 \in hören(h2.MatrNr = s.MatrNr \wedge h2.VorlNr = vs.Vorgänger)))))\}$$

```
SELECT DISTINCT s.name
FROM studenten s, hoeren h, Vorlesungen v
WHERE s.matrnr = h.matrnr
 AND h.vorlnr = v.vorlnr
 AND v.titel = 'Bioethik'
 AND NOT EXISTS
    (SELECT * FROM voraussetzen vs
    WHERE vs.nachfolger = v.vorlnr
    AND NOT EXISTS
      (SELECT *
      FROM hoeren h2
     WHERE h2.matrnr = s.matrnr
      AND h2.vorlnr = vs.vorgaenger))
```

Nachtrag: Die Anfrage in natürlicher Sprache:
Erste Subquery in natürlicher Sprache:

```
SELECT DISTINCT s.name
```

```
FROM studenten s, hoeren h, Vorlesungen v
WHERE s.matrnr = h.matrnr
 AND h.vorlnr = v.vorlnr
 AND v.titel = 'Bioethik'
 AND NOT EXISTS
    (SELECT * FROM voraussetzen vs
    WHERE vs.nachfolger = v.vorlnr
    AND NOT EXISTS
      (vorausgesetzten Vorlesungen, die der Student gehört hat))
```

Zweite Subquery in natürlicher Sprache:

```
SELECT DISTINCT s.name
FROM studenten s, hoeren h, Vorlesungen v
WHERE s.matrnr = h.matrnr
 AND h.vorlnr = v.vorlnr
 AND v.titel = 'Bioethik'
 AND NOT EXISTS
    (vorausgesetzte Vorlesungen für 'Bioehtik'
    AND NOT EXISTS
      (vorausgesetzten Vorlesungen, die der Student gehört hat))
```

Zusammen:

```
SELECT DISTINCT s.name
FROM studenten s, hoeren h, Vorlesungen v
WHERE s.matrnr = h.matrnr
 AND h.vorlnr = v.vorlnr
 AND v.titel = 'Bioethik'
 AND es gibt keine für Bioethik vorausgesetzte Vorlesung, die der Student
nicht gehört hat
```

Wieder mit Allquantor:

```
SELECT DISTINCT s.name
FROM studenten s, hoeren h, Vorlesungen v
WHERE s.matrnr = h.matrnr
 AND h.vorlnr = v.vorlnr
 AND v.titel = 'Bioethik'
 AND der Student hat alle für Bioethik vorausgesetzten Vorlesungen gehört
```

Aufgabe 4

Gegeben sei das Universitätsschema aus der Vorlesung und algebraische Ausdrücke. Formulieren Sie die algebraischen Ausdrücke als Anfragen in SQL. Testen sie Ihre Anfragen in der SQL-Schnittstelle auf unserer Webseite (Wählen sie die Datenbank *unidb* aus).

Aufgabe 4 a)

$\pi_{Name}(\text{Studenten} \bowtie \text{hören} \bowtie \sigma_{\text{Titel}='\text{Logik}'}(\text{Vorlesungen}))$

Lösung

In Worten: Finden Sie die Namen aller Studenten, die die Vorlesung 'Logik' hören.

```
SELECT distinct s.Name
FROM Studenten s, hoeren h, Vorlesungen v
WHERE v.Titel = 'Logik'
 AND s.MatrNr = h.MatrNr
 AND h.VorlNr = v.VorlNr
```

Aufgabe 4 b)

$\pi_{Name}(\text{Professoren} \ltimes \text{prüfen})$

Lösung

In Worten: Finden Sie die Namen aller Professoren, die schon mind. eine Prüfung abgehalten haben.

```
SELECT distinct Name
FROM Professoren
WHERE PersNr IN (SELECT PersNr FROM pruefen)
```

Alternative Lösung: mit `exists(...)`

Aufgabe 4 c)

$\pi_{MatrNr}(\text{Studenten}) \setminus \pi_{MatrNr}(\text{prüfen})$

Lösung

In Worten: Finden Sie die Matrikelnummern aller Studenten, die noch keine Prüfung abgelegt haben.

```
SELECT distinct MatrNr
FROM Studenten
WHERE MatrNr NOT IN (SELECT MatrNr FROM pruefen)
```

Alternative Lösung: mit `except`

Aufgabe 4 d)

$\pi_{VorlNr}(\text{Vorlesungen}) \setminus \pi_{v1.VorlNr}(\rho_{v1}(\text{Vorlesungen}) \bowtie_{v1.SWS<v2.SWS} \rho_{v2}(\text{Vorlesungen}))$

Lösung

In Worten: Finden Sie die Vorlesungsnummern der Vorlesungen mit den meisten Semesterwochenstunden.

```
SELECT distinct VorlNr
FROM Vorlesungen
WHERE SWS = (SELECT MAX(SWS) FROM Vorlesungen)
```

Alternative Lösung: mit `not exists(...)`