Informationssysteme

Sommersemester 2016

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

Formulierung und einfache Auswertung von Anfragen

Übersicht

Inhalte der nächsten Vorlesungen:

- **Konjunktive regelbasierte Anfragen**:
  $ans(x_{na}) \leftarrow Professoren(x_{pn}, x_{na}, \text{'C4'}, x_{ra})$
- **Relationenkalküle**: $\{t.name \mid t \in Professoren \wedge t.rang = \text{'C4'}\}$
- **Relationale Algebra**: $\pi_{name}(\sigma_{rang=\text{'C4'}}(Professoren))$
- **SQL:** SELECT name FROM Professoren WHERE rang='C4'

Wir betrachten die Datenbank

$$\text{CINEMA} = \{\text{Movies, Location, Pariscope}\}$$

wobei die Relationen **Movies**, **Location** und **Pariscope** die folgenden Schemata haben:

$$\begin{aligned} sch(Movies) &= \{\text{Title, Director, Actor}\} \\ sch(Location) &= \{\text{Theater, Address, Phone Number}\} \\ sch(Pariscope) &= \{\text{Theater, Title, Schedule}\} \end{aligned}$$

Beispiele und Notationen folgen weitestgehend dem Buch "Foundations of Databases" S.Abiteboul, R. Hull und V. Vianu. PDF ist verfügbar unter `http://webdam.inria.fr/Alice/`.

Movies		
Title	**Director**	**Actor**
Immer Ärger mit Harry	Hitchcock	Gwenn
Immer Ärger mit Harry	Hitchcock	Forsythe
Immer Ärger mit Harry	Hitchcock	MacLaine
Immer Ärger mit Harry	Hitchcock	Hitchcock
.....	....	....
Schreie und Flüstern	Bergman	Andersson
Schreie und Flüstern	Bergman	Sylwan
Schreie und Flüstern	Bergman	Thulin
Schreie und Flüstern	Bergman	Ullman

Location		
Theater	**Address**	**Phone Number**
Gaumont Opéra	31 bd. des Italiens	47 42 60 33
Saint André des Arts	30 rue Saint André des Arts	43 26 48 18
Le Champo	51 rue des Ecoles	43 54 51 60
....	....	...
Georges V	144 av. des Champs-Elysées	45 62 41 46
Les 7 Montparnassiens	98 bd. du Montparnassiens	43 20 32 20

Pariscope		
Theater	**Title**	**Schedule**
Gaumont Opéra	Schreie und Flüstern	20:30
Saint André des Arts	Immer Ärger mit Harry	20:15
Georges V	Schreie und Flüstern	22:15
....	....	....
Les 7 Montparnassiens	Schreie und Flüstern	20:45

Anfragen

- Wer ist der Regisseur von 'Schreie und Flüstern'?
- In welchem Theater läuft 'Schreie und Flüstern'?
- Was ist die Adresse und die Telefonnummer des Theaters 'Le Champo'?

Gib Namen und Adressen der Theater aus, die einen Bergman-Film spielen.

Zur Erinnerung:

$$
\begin{aligned}
sch(Movies) &= \{\text{Title, Director, Actor}\} \\
sch(Location) &= \{\text{Theater, Address, Phone Number}\} \\
sch(Pariscope) &= \{\text{Theater, Title, Schedule}\}
\end{aligned}
$$

Kann wie folgt berechnet werden:

Falls es jeweils Tupel r1, r2, r3 aus den Relationen
Movies, Pariscope, Location gibt, so dass
der Regisseur in r1 'Bergman' ist
und die Titel der Tupel r1 und r2 gleich sind
und die Theater in Tupel r2 und r3 gleich sind
dann möchten wir Theater und Addresse von Tupel r3.

Falls es jeweils in den Relationen Movies, Pariscope und Location Tupel $\langle x_{ti}$, 'Bergman', $x_{ac}\rangle$, $\langle x_{th}, x_{ti}, x_s\rangle$ und $\langle x_{th}, x_{ad}, x_s\rangle$ gibt dann nehme das Tupel $\langle$Theater : x_{th}, Address : $x_{ad}\rangle$ in die Antwort auf.
x_{ti}, x_{th} etc. sind Variablen.

Wir können die Anfrage umschreiben als

$ans(x_{th}, x_{ad}) \leftarrow$ Movies(x_{ti}, 'Bergman', x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p)

wobei ans (für Answer/Antwort) eine Relation über {Theater, Address} ist

Kopf (Head) und **Rumpf (Body)**

Der Ausdruck links von $\leftarrow$ wird als Kopf bezeichnet. Der Ausdruck rechts von $\leftarrow$ heisst Rumpf.

$ans(x_{th}, x_{ad}) \leftarrow$ Movies(x_{ti}, 'Bergman', x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p)

Der obige Ausdruck kann noch vereinfacht werden als

$ans(x_{th}, x_{ad}) \leftarrow$ Movies(x_{ti}, 'Bergman', _), Pariscope(x_{th}, x_{ti}, _), Location(x_{th}, x_{ad}, _)

wobei das Zeichen _ benutzt wird, um alle Variablen zu ersetzen, die nur ein Mal auftreten.

Regelbasierte Konjunktive Anfrage

Sei $\mathbf{R}$ ein Datenbankschema. Eine **regelbasierte konjunktive Anfrage** über $\mathbf{R}$ hat die Form

$$ans(u) \leftarrow R_1(u_1), \ldots, R_n(u_n)$$

- $u, u_1, \ldots u_n$ sind Tupel
- **Notation:** falls $v =< x_1, ..., x_m >$ dann schreiben wir $R(v)$ anstelle von $R(x_1, \ldots, x_m)$
- u_i muss Stelligkeit (arity) passend zu R_i haben.
- Jede Variable aus u muss mindestens ein Mal in einem der $u_1, \ldots, u_n$ auftreten.

Semantik von Regelbasierten Konjunktiven Anfragen

Betrachtet man eine **Instanz einer Datenbank**, also Ausprägungen von Tupeln in den einzelnen Relationen, dann kann man sich eine Regel so vorstellen, dass wenn man eine gültige Belegung des Rumpfes gefunden hat, der Kopf der Regel das Ergebnis darstellt.

Betrachten wir wieder Anfrage q als

$$ans(u) \leftarrow R_1(u_1), \ldots, R_n(u_n)$$

und eine Instanz (Ausprägung) $\mathbf{I}$ der Datenbank $\mathbf{R}$. **Das Abbild von $\mathbf{I}$ unter Anfrage q ist**

$$q(\mathbf{I}) = \{\nu(u) \mid \nu \text{ ist eine Belegung der Variablen aus } q \text{ und } \nu(u_i) \in I(R_i), \forall i \in [1, n]\}$$

Belegung von Variablen: Beispiel

$ans(x_{th}, x_{ad}) \leftarrow$ Movies(x_{ti}, 'Bergman', x_{ac}), Pariscope(x_{th}, x_{ti}, x_s), Location(x_{th}, x_{ad}, x_p)

Betrachten wir folgende Belegung der Variablen:

$\nu(x_{ti}) =$ 'Schreie und Flüstern'
$\nu(x_{ac}) =$ 'Ullman'
$\nu(x_{th}) =$ 'Gaumont Opera'
$\nu(x_s) =$ '20:30'
$\nu(x_{ad}) =$ '31 bd. des Italiens'
$\nu(x_p) =$ '47 42 60 33'

Dies ist eine gültige Belegung, da Tupel $(\nu(x_{ti}), \text{“Bergman”}, \nu(x_{ac})) =$ (“Schreie und Flüstern”, “Bergman”, “Ullman”) $\in I(Movies)$, für unsere Beispiel Ausprägung I, analog für die beiden anderen Relationen bzw. Tupel.

Weitere Beispiele

Finde Name und Personalnummer aller C4 Professoren:

$$ans(x_{na}, x_{pn}) \leftarrow Professoren(x_{pn}, x_{na}, \text{'C4'}, x_{ra})$$

Welche Vorlesungen bietet Prof. Sokrates an?

$$ans(x_{ti}) \leftarrow Professoren(x_{pn}, \text{'Sokrates'}, x_{rg}, x_{ra}),$$
$$Vorlesungen(x_{vn}, x_{ti}, x_{sw}), lesen(x_{pn}, x_{vn})$$

Intensionale vs. Extensionale Relationen

Wir berechnen durch die Erstellung von Anfragen nicht nur Ergebnisse, sondern **definieren dabei implizit auch neue Relationen**, z.B. die neue Relation $C4Profs$, die alle Professoren mit Rang C4 enthält:

$$C4Profs(x_{pn},x_{na},x_{ra}) \leftarrow Professoren(x_{pn},x_{na},\text{'C4'},x_{ra})$$

Man unterscheidet zwischen Relationen, die ursprünglich in der Datenbank vorhanden sind, den sogenannten **extensionalen** Relationen, und Relationen, die durch Regeln definiert werden, den sogenannten **intensionalen** Relationen.

Auswertung (aka. Berechnung) der Anfrage

Naive brute-force Methode zur Berechnung der Anfrageergebnisse:

- Betrachtung aller möglichen Belegung anhand der in den Relationen auftretenden Werte.
- Für jede dieser Belegungen schauen ob es eine gültige Belegung ist.

Diese Vorgehensweise ist natürlich sehr teuer. Z.B. für $Movies(x_1,$'Bergman'$,x_2)$ machen nur Belegungen Sinn, die für dieses Tupel mit x_1 und x_2 aus $Movies$ den Regisseur gleich 'Bergman' haben. Idealerweise kann man mit sogenannten Indexen die Auswahl die Suche nach geeigneten Tupeln beschleunigen.

Dazu kommen wir später im **Abschnitt über Anfrageverarbeitung und Indexstrukturen**.

Monotonie

- **Monotonie:** Eine Anfrage q über R ist monoton wenn für jede Ausprägungen (Instanzen) I, J über R, gilt: Falls $I \subseteq J$ dann $q(I) \subseteq q(J)$.
- D.h. wenn neue Tupel zu den Relationen hinzukommen fallen keine Ergebnistupel weg.

Konjunktive Anfragen sind monoton

Welche Anfragen sind somit nicht möglich?

Anfragen im konjunktiven Kalkül

Gegeben eine konjunktive regelbasierte Anfrage

$$ans(e_1, \ldots, e_m) \leftarrow R_1(u_1), \ldots, R_n(u_n)$$

Die dazu äquivalente Anfrage im **konjunktiven Kalkül** ist

$$\{e_1, \ldots, e_m \mid \exists x_1, \ldots, x_k(R_1(u_1) \wedge \ldots \wedge R_n(u_n))\}$$

Die Variablen $x_1, x_2, \ldots x_k$ sind die Variablen, die im Rumpf aber nicht im Kopf der regelbasierten Anfrage auftreten.

Anfragen im konjunktiven Kalkül: Beispiel

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, \text{'Bergman'}, x_{ac}),$$
$$Pariscope(x_{th}, x_{ti}, x_s),$$
$$Location(x_{th}, x_{ad}, x_p)$$

$$ans := \{x_{th}, x_{ad} \mid \exists\, x_{ti}\ \exists\, x_{ac}\ \exists\, x_s\ \exists\, x_p\ (Movies(x_{ti}, \text{'Bergman'}, x_{ac}) \wedge$$
$$Pariscope(x_{th}, x_{ti}, x_s) \wedge$$
$$Location(x_{th}, x_{ad}, x_p))\}$$

Sei $\mathbf{R}$ ein Datenbankschema. Eine Formel über $\mathbf{R}$ im konjunktiven Kalkül ist ein Ausdruck, der eine der folgenden Formen besitzt:

(a) ein Atom über $\mathbf{R}$ der Form $R_i(u_i)$

(b) $(\phi \wedge \psi)$, mit Formeln ϕ und ψ über $\mathbf{R}$; oder

(c) $\exists x \phi$, wobei x eine Variable ist und ϕ ist eine Formel über $\mathbf{R}$.

Freie Variablen:
Ein Auftreten einer Variablen x in einer Formel ϕ ist **frei** falls

(i) ϕ ist ein Atom; oder

(ii) $\phi = (\psi \wedge \xi)$ und das Auftreten von x ist frei in ψ oder ξ; oder

(iii) $\phi = \exists y \psi$, x und y sind verschiedene Variablen, und das Auftreten von x ist frei in ψ

Falls eine Variable nicht frei ist so nennen wir sie **gebunden**.

Anfragen im konjunktiven Kalkül

Eine Anfrage im konjunktiven Kalkül über Datenbankschema R ist ein Ausdruck der Form

$$\{e_1, ..., e_m \mid \phi\}$$

wobei ϕ eine Formel des konjunktiven Kalküls ist und die Variablen $e_1, \ldots, e_m$ und sind identisch zur Menge der freien Variablen in ϕ.

Theorem

Regelbasierte konjunktive Anfragen und Anfragen im konjunktiven Kalkül sind äquivalent.

Gleichheit bzw. Vergleichsoperatoren

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, x_d, x_{ac}), x_d = \text{'Bergman'},\ Pariscope(x_{th}, x_{ti}, x_s),\ Location(x_{th}, x_{ad}, x_p)$$

ist identisch zu

$$ans(x_{th}, x_{ad}) \leftarrow Movies(x_{ti}, \text{'Bergman'}, x_{ac}),\ Pariscope(x_{th}, x_{ti}, x_s),\ Location(x_{th}, x_{ad}, x_p)$$

Ebenfalls möglich Vergleichsoperatoren, z.B.

$$ans(x_{ma}) \leftarrow Studenten(x_{ma}, x_{na}, x_{se}), x_{se} \geq 12$$

Anfrage-Programm

Wie erwähnt, kann konzeptionell eine Anfrage so aufgefasst werden, dass sie eine neue Relation erzeugt, die dann in nachfolgenden Anfragen benutzt werden kann.

Ein **konjunktives Anfrage-Programm** ist eine Sequenz von Regeln in der Form

$$\begin{aligned} S_1(u_1) &\leftarrow body_1 \\ S_2(u_2) &\leftarrow body_2 \\ &\dots \\ S_m(u_m) &\leftarrow body_m \end{aligned}$$

Rekursion sei nicht zugelassen.

Beispiel

$$
\begin{aligned}
S_1(x,z) &\leftarrow Q(x,y), R(y,z,w) \\
S_2(x,y,z) &\leftarrow S_1(x,w), R(w,y,v), S_1(v,z) \\
S_3(x,z) &\leftarrow S_2(x,u,v), Q(v,z)
\end{aligned}
$$

Q

1	2
2	1
2	2

R

1	1	1
2	3	1
3	1	2
4	4	1

S_1

1	3
2	1
2	3

S_2

1	1	1
1	1	3
2	1	1
2	1	3

S_3

1	2
2	2

Beispiel

$$
\begin{aligned}
S_1(x,z) &\leftarrow Q(x,y), R(y,z,w) \\
S_2(x,y,z) &\leftarrow S_1(x,w), R(w,y,v), S_1(v,z) \\
S_3(x,z) &\leftarrow S_2(x,u,v), Q(v,z)
\end{aligned}
$$

Wir können für die ersten beiden Zeilen auch schreiben als

$$
\begin{aligned}
S_2(x,y,z) \leftarrow\ & Q(x_1,y_1), R(y_1,z_1,w_1), x = x_1, w = z_1, \\
& R(w,y,v), Q(x_2,y_2), R(y_2,z_2,w_2), v = x_2, z = z_2
\end{aligned}
$$

bzw. ohne Gleichheits-Operator

$$
\begin{aligned}
S_2(x,y,z) \leftarrow\ & Q(x,y_1), R(y_1,w,w_1), \\
& R(w,y,v), Q(v,y_2), R(y_2,z,w_2)
\end{aligned}
$$

Sichten (Englisch: Views)

$$\begin{aligned} Marilyn(x_t) &\leftarrow Movies(x_t, x_d, \text{'Monroe'}) \\ ChampoInfo(x_t, x_s, x_p) &\leftarrow Pariscope(\text{'Le Champo'}, x_t, x_s), \\ &\quad\ Location(\text{'Le Champo'}, x_a, x_p) \end{aligned}$$

Versuchen wir damit die folgende Anfrage auszudrücken: **"Welche Titel in $Marilyn$ werden im Le Champo um 21:00 Uhr gespielt?"**

$$ans(x_t) \leftarrow Marilyn(x_t), ChampoInfo(x_t, \text{'21:00'}, x_p)$$

Falls diese Sichten nur virtuell sind (also nicht wirklich materialisiert/ausgeprägt) so wird die Anfrage doch ausgeführt via

$$\begin{aligned} ans(x_t) &\leftarrow Movies(x_t, x_d, \text{'Monroe'}), \\ &\quad\ Pariscope(\text{'Le Champo'}, x_t, \text{'21:00'}), \\ &\quad\ Location(\text{'Le Champo'}, x_a, x_p) \end{aligned}$$

Views (2)

Eine alternative Formulierung:

$$
\begin{aligned}
Marilyn &:= \{x_t \mid \exists\, x_d(Movies(x_t, x_d, \text{'Monroe'}))\};\\
ChampoInfo &:= \{x_t, x_s, x_p \mid \exists x_a(Pariscope(\text{'Le Champo'}, x_t, x_s)\\
&\quad \wedge Location(\text{'Le Champo'}, x_a, x_p))\};\\
ans &:= \{x_t \mid Marilyn(x_t)\\
&\quad \wedge\, \exists\, x_p(ChampoInfo(x_t, \text{'21:00'}, x_p))\}
\end{aligned}
$$

Was ist mit Negation?

Lassen wir nun Literale L_i der Form $R(v)$ oder $\neg R(v)$ zu, d.h. wir haben Regeln

$$q : S(u) \leftarrow L_1, \ldots, L_n$$

wobei R der Name einer Relation ist, v ist ein Tupel der passenden Stelligkeit und S tritt nicht im Rumpf der Regel auf (also: nicht rekursiv).

"In welchen Hitchcock Filmen hat Hitchcock nicht mitgespielt?"

$$ans \leftarrow Movies(x, \text{'Hitchcock'}, z),$$
$$\neg Movies(x, \text{'Hitchcock'}, \text{'Hitchcock'})$$

Was muss in der Datenbank gelten, damit diese Anfrage korrekt ist?

„Gib alle Filme aus, bei denen nur Schauspieler mitgewirkt haben, die mal unter Hitchcock Regie gespielt haben.“

$$
\begin{aligned}
Hitch\text{-}actor(z) &\leftarrow Movies(x, \text{'Hitchcock'}, z) \\
not\text{-}ans(x) &\leftarrow Movies(x,y,z), \neg Hitch\text{-}actor(z) \\
ans(x) &\leftarrow Movies(x,y,z), \neg not\text{-}ans(x)
\end{aligned}
$$

Relationenkalküle: Das Domänenkalkül

Wenn wir zum konjunktiven Kalkül die **Negation** hinzunehmen, erhalten wir ein **Relationenkalkül**, nämlich das **Domänenkalkül**. Es gibt auch noch ein weiteres Relationenkalkül, das **Tupelkalkül**, welches wir uns später ansehen.

Wir erlauben nun auch gleich noch weitere **Vergleichsoperatoren**, haben also Basis-Formeln (Atome) der Form $R(u)$ mit Relation R und Tupel u über Domänenvariablen und Konstanten, sowie Atome der Form $e = e'$, $e \neq e'$, $e \geq e'$ etc. für Domänenvariablen oder Konstanten e und e'.

Alle Atome sind Formeln. Darauf basierend haben wir **Formeln der Form**

- $(\phi \wedge \psi)$ für Formeln ψ und ϕ
- $(\phi \vee \psi)$ für Formeln ψ und ϕ
- $\neg\phi$ für Formeln ϕ
- $\exists x \phi$ für Variable x und Formel ϕ
- $\forall x \phi$ für Variable x und Formel ϕ

"In welchen Hitchcock Filmen hat Hitchcock nicht mitgespielt?"

$$\{x_t \quad | \quad \exists x_a Movies(x_t, \text{'Hitchcock'}, x_a) \wedge \neg Movies(x_t, \text{'Hitchcock'}, \text{'Hitchcock'})\}$$

"Gib alle Filme aus, bei denen nur Schauspieler mitgewirkt haben, die mal unter Hitchcock Regie gespielt haben."

$$\{x_t \quad | \quad \exists x_d, x_a Movies(x_t, x_d, x_a) \wedge \forall y_a(\exists y_d Movies(x_t, y_d, y_a) \Rightarrow \exists z_t Movies(z_t, \text{'Hitchock'}, y_a))\}$$

Das Relationale Tupelkalkül

Das bislang betrachtetete Relationen Kalkül wird auch als **relationales Domänenkalkül** bezeichnet, da dort Domänenvariablen an die Domänen der einzelnen Attribute (Spalten) der Relationen gebunden werden.

Eine Anfrage im **relationalen Tupelkalkül** hat die Form

$$\{t \mid P(t)\}$$

wobei t eine **Tupelvariable** ist und $P(t)$ ein Prädikat (eine Formel). $P(t)$ muss erfüllt sein damit t Teil des Ergebnis ist.

$$\{t.title \mid \exists\ s \in Movies\ (t.title = s.title \wedge s.director = \text{'Hitchcock'}$$

$$\wedge \neg \exists u \in Movies\ (u.title = s.title \wedge u.actor = \text{'Hitchcock'}))\}$$

Das Relationale Tupelkalkül: Definition

Formeln des **Tupelkalküls** bestehen aus Atomen der Form $t \in R$, mit Tupelvariabel t und Relation R, der Form $t.x = t.y$, $t.x > t.y$, etc. für Tupelvariablen t und s und Attributnamen x und y, sowie der Form $t.x = c$ und $t.x > c$ etc. für Konstanten c.

Alle Atome sind Formeln und dann haben wir noch **Formeln der Form**

- $(\phi \wedge \psi)$ für Formeln ψ und ϕ
- $(\phi \vee \psi)$ für Formeln ψ und ϕ
- $\neg\phi$ für Formeln ϕ
- $\exists x\phi$ für Variable x und Formel ϕ
- $\forall x\phi$ für Variable x und Formel ϕ

Beispiele

- **C4-Professoren:**

$$\{p \mid p \in Professoren \wedge p.Rang = C4\}$$

- **Paare von Professoren (Name) und Assistenten (PersNr):**

$$\{p.Name, a.PersNr \mid p \in Professoren \wedge a \in Assistenten$$
$$\wedge p.PersNr = a.Boss\}$$

- **Studenten mit mindestens einer Vorlesung von Prof. Curie:**

$$\{s|s \in Studenten$$
$$\wedge \exists\, h \in h\ddot{o}ren\ (s.MatrNr = h.MatrNr$$
$$\wedge \exists\, v \in Vorlesungen\ (h.VorlNr = v.VorlNr$$
$$\wedge \exists\, p \in Professoren\ (p.PersNr = v.gelesenVon$$
$$\wedge\ p.Name = \text{`}Curie\text{`}))\}$$

.... in SQL ... ist es sehr ähnlich:

Studenten mit mindestens einer Vorlesung von Prof. Curie:

```
SELECT s.*
FROM Studenten s
WHERE EXISTS (
    SELECT h.*
    FROM hören h
    WHERE h.MatrNr=s.MatrNr AND EXISTS (
        SELECT *
        FROM Vorlesungen v
        WHERE v.VorlNr=h.VorlNr AND EXISTS (
            SELECT *
            FROM Professoren p
            WHERE p.Name='Curie' AND p.PersNr=v.gelesenVon)))
```

Sicherheit von Anfragen

- **Einschränkung auf Anfragen mit endlichem Ergebnis.**
- Zum Beispiel ist die Anfrage

$$\{n|\neg(n \in Professoren)\}$$

nicht sicher.

- **Das Ergebnis ist unendlich.**
- **Bedingung: Ergebnis des Ausdrucks muss Teilmenge der Domäne der Formel sein.**
- Die **Domäne** einer Formel enthält
  - alle in der Formel vorkommenden Konstanten
  - alle Attributwerte von Relationen, die in der Formel referenziert werden

Unsichere Anfragen im Domänenkalkül

$$\{x \mid \neg Movies(\text{'Schreie und Flüstern'}, \text{'Bergman'}, x)\}$$

$$\{x_{pn}, x_{na}, x_{rg}, x_{ra} \mid \neg Professoren(x_{pn}, x_{na}, x_{rg}, x_{ra})\}$$

$$\{x, y \quad \mid Movies(\text{'Schreie und Flüstern'}, \text{'Bergman'}, x) \\ \vee Movies(y, \text{'Bergman'}, \text{'Ullman'})\}$$

Wieso sind diese Anfragen unsicher?

Sicherheit des Domänenkalküls

- Sicherheit ist analog zum Tupelkalkül, aber etwas komplizierter, da Variablen nicht an Tupel einer Relation sondern an einzelne Domänen gebunden sind.
- Zum Beispiel ist

$$\{x_{pn}, x_{na}, x_{rg}, x_{ra} \mid \neg(Professoren(x_{pn}, x_{na}, x_{rg}, x_{ra}))\}$$

  nicht sicher.
- Ein Ausdruck

$$\{x_1, x_2, ..., x_n \mid P(x_1, x_2, ..., x_3)\}$$

  ist **sicher**, falls folgende drei Bedingungen gelten:

$\{x_1, x_2, ..., x_n | P(x_1, x_2, ..., x_3)\}$ sicher falls ....

1. Falls **Tupel** $(c_1, c_2, ..., c_n)$ mit Konstanten c_i im Ergebnis enthalten ist, so muss jedes c_i $(1 \leq i \leq n)$ in der Domäne von P enthalten sein.
2. **Für jede existenz-quantifizierte Teilformel** $\exists x(P_1(x))$ muss gelten, dass P_1 nur für Elemente aus der Domäne erfüllbar sein kann – oder evtl. für gar keine.
   Das heisst: Wenn für eine Konstante c das Prädikat $P_1(c)$ erfüllt ist, so muss c in der Domäne von P_1 enthalten sein.
3. **Für jede universal-quantifizierte Teilformel** $\forall x(P_1(x))$ muss gelten, dass sie dann und nur dann erfüllt ist, wenn $P_1(x)$ für alle Werte der Domäne von P_1 erfüllt ist.
   Das heisst: $P_1(d)$ muss für alle d, die nicht in der Domäne von P_1 enthalten sind, auf jeden Fall erfüllt sein.

Zusammenfassung

- **Konjunktive regelbasierte Anfragen und Kalküle**, als regel- bzw. logikbasierte Möglichkeit Anfragen zu erstellen
- Dabei geben diese Anfragen vor **wie Ergebnistupel aussehen sollen bzw. wie sie zusammengehören, aber nicht wie das Ergebnis tatsächlich berechnet werden soll.**
- Sogenannte **deklarative Anfragen**.
- **Domänenkalkül und Tupelkalkül gleich ausdrucksstark (mächtig).**
- Wie bereits erwähnt, **ist auch SQL deklarativ und durch das relationale Tupelkalkül inspiriert.**
- Wir betrachten nun die relationale Algebra.
- Und danach SQL.

Übersicht

- Konjunktive regelbasierte Anfragen:
  $ans(x_{na}) \leftarrow Professoren(x_{pn}, x_{na}, \text{'C4'}, x_{ra})$
- Relationenkalküle: $\{t.name \mid t \in Professoren \wedge t.rang = \text{'C4'}\}$
- **Relationale Algebra**: $\pi_{name}(\sigma_{rang=\text{'C4'}}(Professoren))$
- **SQL:** SELECT name FROM Professoren WHERE rang='C4'