# Grundlagen Datenbanken: Übung 04

Tanmay Deshpande

ge94vem@mytum.de

Gruppe 20 & 21

# QR-Code für die Folien

# Wiederholung

Woche 04

# Relationenkalkül

- Relationale Algebra stärker *prozedural* orientiert (wie wird das Ergebnis einer Anfrage berechnet?)
- Relationenkalkül stärker *deklarativ* orientiert (Beschreibung der Ergebnistupel, ohne Herleitungsvorschrift)
- Aber beide gleich mächtig
- Zwei Arten von Relationenkalkül:
  - Tupelkalkül: Variablen an Tupel einer Relation gebunden
  - Domänenkalkül: Variablen an Domänen einer Relation gebunden

# Tupelkalkül

- Anfrage hat die Form {t | P(t)}, t = Ergebnistupel, P(t) = Bedingung, die beschreibt, welche Tupel aus der Relation im Ergbenis kommen

  *Professoren mit Rang ‘C4’* => {p | p ∈ Professoren ∧ p.Rang = ‘C4’}
- Es ist möglich, “nicht-existierende” Tupelformen zu generieren

  *Paare von Professorennamen und die PersNr von den ihnen zugeordneten Assistenten*

  {[p.Name, a.PersNr] | p ∈ Professoren ∧ a ∈ Assistenten ∧ p.PersNr = a.Boss}
- **Existenzquantifizierung**

  *Studenten, die mindestens eine Vorlesung bei ‘Curie’ gehört haben*

  {s | s ∈ Studenten ∧ ∃ h ∈ hören(s.MatrNr = h.MatrNr ∧

  ∃ v ∈ Vorlesungen(v.VorlNr = h.VorlNr ∧

  ∃ p ∈ Professoren(p.PersNr = v.gelesenVon ∧

  p.Name = ‘Curie’))))}

  **Allquantifizierung**

  *Studenten, die alle vierstündigen Vorlesungen hören*

  {s | s ∈ Studenten ∧ ∀ v ∈ Vorlesungen(v.SWS = 4 ⇒ ∃ h ∈ hören(h.VorlNr = v.VorlNr ∧

  h.MatrNr=s.MatrNr) )}

# Tupelkalkül – Sichere Ausdrücke

- Ein Ausdrück heißt sicher, falls das Ergebnis einer Teilmenge der Domäne der Formel ist
- Die Domäne einer Formel enthält alle Konstanten + alle Attributwerte, die in der Formel vorkommen
- Somit ist für sichere Ausdrücke garantiert, dass sie endlich sind, da wir mit endlichen Relationen arbeiten
- Beispiel einer unsicheren Anfrage: $\{n \mid \neg(n \in \text{Professoren})\}$
  Wir können unendliche Tupel finden, die nicht in der Relation Professoren liegen. Das Ergebnis ist offensichtlich keine Teilmenge der Domäne von der Formel
- Deshalb begrenzen wir uns in der Vorlesung auf sichere Ausdrücke

# Domänenkalkül

- Anfrage hat die Form $\{[v_1, v_2, \ldots, v_n] \mid P(v_1, v_2, \ldots, v_n)\}$, wobei $v_i$ $(1 \leq i \leq n)$ Domänenvariablen sind, die Attributwerte repräsentieren, und P eine Formel mit freien Variablen $v_1, v_2, \ldots, v_n$ ist
  Beispiel: Studenten => {[m, n, s] | [m,n,s] ∈ Studenten}

- Wie Tupelkalkül, kann man auch bei Domänenkalkül “nicht-existeriende” Tupel formen
  Beispiel: Nur Name und MatrNr aller Studenten
  {[m,n] | ∃s([m,n,s] ∈ Studenten)}

- Wie Tupelkalkül, gibt es auch in Domänenkalkül All- und Existenzquanitifizierung

# Domänenkalkül – Sichere Ausdrücke

- Genauso wie bei Tupelkalkül, gibt es in auch in Domänenkalkül Anfragen, die unendliche Ergebnisse haben
  Beispiel: {[p, n, rg, ra] | ¬([p, n, rg, ra] ∈ Professoren)}
- Wir definieren sichere Ausdrücke wie folgt:
  - Jedes Tupel im Ergebnis muss aus Konstanten bestehen, die zur Domäne der Formel gehören
  - Für jede existenz-quantifizierte Teilformel $\exists x(P_1)x$ muss gelten, $P_1$ ist nur für die Elemente aus der Domäne von $P_1$ erfüllbar
  - Für jede universal-quantifizierte Teilformel $\forall x(P_1)x$ muss gelten, $P_1$ ist nur dann erfüllt, wenn sie für alle Werte aus der Domäne von $P_1$ erfüllt ist

# Ausdruckskraft der Anfragesprachen

- Die relationale Algebra,
- Die relationale Tupelkalkül, eigeschränkt auf sichere Ausdrücke,
- Die relationale Domänenkalkül, eingeschränkt auf sichere Ausdrücke…

Sind gleich mächtig! D.h. Anfrage lassen sich zwischen den verschiedenen Darstellungen konvertieren

# SQL

- Deklarative Anfragesprache, die von den meisten relationalen Datenbanksystemen unterstützt wird
- Relationen als Tabellen gespeichert
- Zeile = Tupel, Spalte = Attribut

# SQL-Datentypen

- char(n) = Feste Größe n
- varchar(n) = Maximale Größe n
- numeric(p,s) = p : Gesamtzahl der gespeicherten Stellen, s: Anzahl der signifikanten Stellen
- int
- blob/raw = binäre Dateien
- xml
- date

  usw…

# Schemadefinition und Schemaveränderung in SQL

## Erstellen von Tabellen

create table Professoren
(PersNr **integer not null**,
Name **varchar(10) not null**,
Rang **character(2)**);

## Einfügen von Tupeln

- insert into hören
  select MatrNr, VorlNr from Studenten, Vorlesungen
  where Titel= `Logik‘ ;
- insert into Studenten (MatrNr, Name)
  values (28121, `Archimedes‘);

# Schemadefinition und Schemaveränderung in SQL

## Löschen von Tupeln

```
delete Studenten where Semester > 13;
```

## Verändern von Tupeln

```
update Studenten set Semester = Semester + 1;
```

# Einfache SQL-Anfragen

- Selektion
  select * from Professoren where Rang=‘C4’
- Sortierung
  select PersNr, Name, Rang from Professoren order by Rang desc, Name asc;
- Duplikateliminierung
  select distinct Rang from Professoren

# Anfragen über mehrere Relationen

Welcher Professor liest "Mäeutik"?

```
select  Name,  Titel
from Professoren , Vorlesungen
where  PersNr  =  gelesenVon and  Titel = `Mäeutik` ;
```

$$\prod_{\text{Name, Titel}} (\sigma_{\text{PersNr} = \text{gelesenVon} \wedge \text{Titel} = \text{'Mäeutik'}} (\text{Professoren} \times \text{Vorlesungen}))$$

# SQL Mengenoperationen

- Vereinigung
  select name from professoren
  union
  select name from assistenten

- Schnitt
  …intersect…
- Differenz
  …except…

  Das Schema aller Relationen muss gleich sein!

# SQL Gruppierung und Aggregation

- Aggregatsfunktionen: min, max, sum, avg, count
- Beispiel:

  ```
  select gelesenVon, Name, sum(SWS)
  from Vorlesungen, Professoren
  where gelesenVon = PersNr and Rang = ‘C4’
  group by gelesenVon, Name
  having avg(SWS) >= 3
  ```
- where → Gruppierung → having → Aggregation → Projektion
- SQL erzeugt ein Ergebnistupel pro Gruppe. Deshalb mussen im Ergebnis außer Aggregationen alle Attribute erscheinen, die in der **group by** Bedingung vorkommen

# SQL- Geschachtelte Anfragen

- Nicht-korrelierte Unteranfrage:
  select * from prüfen where Note = (select avg(Note) from prüfen)
- Korrelierte Unteranfrage:
  select PersNr, Name, (select sum(SWS) as Lehrbelastung from Vorlesungen where gelesenVon = PersNr) from Professoren
- Korrelierte Unteranfragen müssen für jedes Tupel neu berechnet werden. Deshalb sind nicht-korrelierte Unteranfragen effizienter

# SQL- exists und in

- **exists** ist erfüllt, wenn die Unteranfrage mindestens ein Ergebnistupel hat. Gibt true oder false zurück
  Bsp: select * from Professoren where exists (select * from Vorlesungen where gelesenVon = PersNr)

  **not exists** macht das Gegenteil
- **in** ist erfüllt, wenn der Wert des gewählten Attributs in der Unteranfrage vorkommt
  Bsp: SELECT * FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Orders);

  **not in** macht das Gegenteil

# SQL- all

- select Name from Studenten where Semester >= all ( select Semester from Studenten);
- Kein vollwertiger Allquantor!
- Allquantifizierung kann mit **not exists** ausgedrückt werden

# Aufgaben

Woche 04

# Aufgabe 01

- Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema im Tupel- und Domänenkalkül:
  a) Finden Sie die Vorlesungen, die keine Hörer haben.
  b) Finden Sie die Studenten, die alle Vorlesungen hören.

# Lösungsvorschlag 1a

- a) Finden Sie die Vorlesungen, die keine Hörer haben.

**Formulierung im Tupelkalkül**

$$\{v \mid v \in \text{Vorlesungen} \wedge \nexists\, h \in \text{hören}(v.\text{VorlNr} = h.\text{VorlNr})\}$$

oder

$$\{v \mid v \in \text{Vorlesungen} \ \wedge \forall\, h \in \text{hören}(v.\text{VorlNr} \neq h.\text{VorlNr})\}$$

**Formulierung im Domänenkalkül**

$$\{[v,t,s,g] \mid [v,t,s,g] \in \text{Vorlesungen} \wedge \nexists\, m([m,v] \in \text{hören})\ \}$$

# Lösungsvorschlag 1b

- b) Finden Sie die Studenten, die alle Vorlesungen hören

**Formulierung im Tupelkalkül**

$$\{s \mid s \in \text{Studenten} \ \wedge \ \forall\, v \in \text{Vorlesungen}( \\ \exists\, h \in \text{hoeren}(v.\text{VorlNr} = h.\text{VorlNr} \wedge s.\text{MatrNr} = h.\text{MatrNr}))\}$$

**Formulierung im Domänenkalkül**

$$\{[m,n,s] \mid [m,n,s] \in \text{Studenten} \wedge \\ \forall\, v,t,sws,g([v,t,sws,g] \in \text{Vorlesungen} \Rightarrow [m,v] \in \text{hören})\}$$

# Aufgabe 02

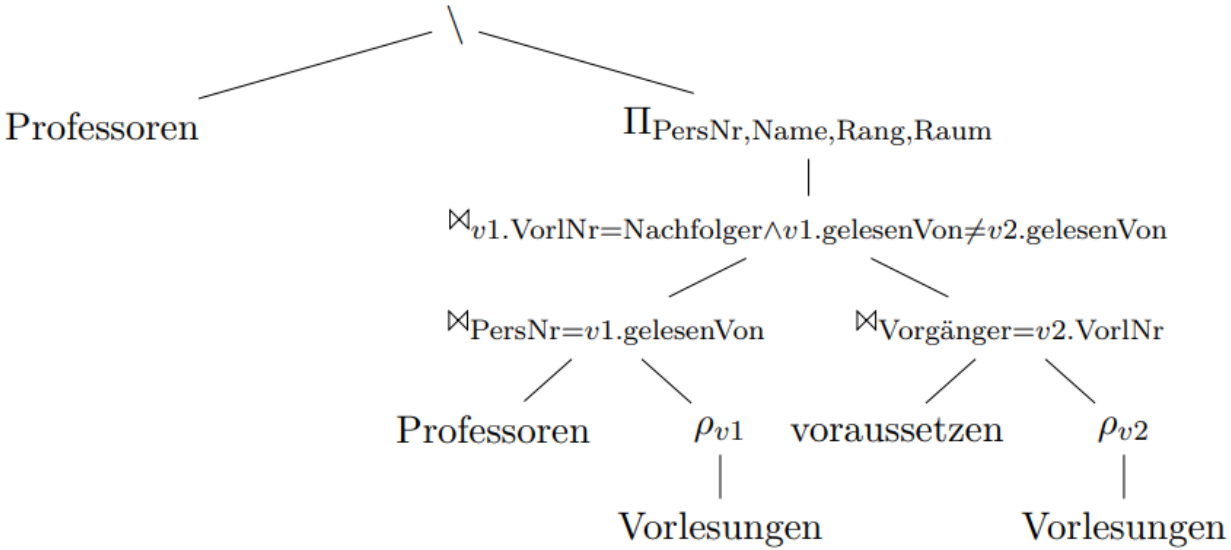
- Gegeben sei folgende Anfrage auf dem bekannten Universitätschema:

  Gesucht sind die Professoren, deren sämtliche Vorlesungen nur auf selbst gelesenen (direkten) Vorgängern aufbauen.

  Formulieren Sie die Anfrage
  • in der Relationenalgebra,
  • im Tupelkalkül und
  • im Domänenkalkül.

# Lösungsvorschlag 02

**Formulierung in relationaler Algebra**



# Lösungsvorschlag 02 (contd…)

**Formulierung in Tupelkalkül**

Mit Allquantifizierung

$$
\begin{aligned}
&\{\, p \mid p \in \text{Professoren} \wedge \\
&\quad \forall v_1 \in \text{Vorlesungen}(v_1.\text{gelesenVon} = p.\text{PersNr} \Rightarrow \\
&\qquad \forall o \in \text{voraussetzen}( \\
&\qquad\quad o.\text{Nachfolger} = v_1.\text{VorlNr} \Rightarrow \\
&\qquad\quad \exists v_2 \in \text{Vorlesungen}( \\
&\qquad\qquad o.\text{Vorgaenger} = v_2.\text{VorlNr} \wedge \\
&\qquad\qquad v_2.\text{gelesenVon} = p.\text{PersNr} \\
&\qquad\quad ))) \}
\end{aligned}
$$

Ohne Allquantifizierung

$$
\begin{aligned}
&\{\text{p} \mid \text{p} \in \text{Professoren} \\
&\quad \wedge \neg\exists\ \text{v}_1 \in \text{Vorlesungen}(\text{v}_1.\text{gelesenVon}{=}\text{p.PersNr} \\
&\quad \wedge \exists\ \text{o} \in \text{voraussetzen}(\text{v}_1.\text{VorlNr}{=}\text{o.Nachfolger} \\
&\quad \wedge \exists\ \text{v}_2 \in \text{Vorlesungen}(\text{o.Vorgänger}{=}\text{v}_2.\text{VorlNr} \\
&\quad \wedge \text{v}_2.\text{gelesenVon} \neq \text{p.PersNr})))\}
\end{aligned}
$$

# Lösungsvorschlag 02 (contd…)

**Formulierung in Domänenkalkül**

$$\{\ [p, n]\ | \exists rg, ra([p, n, rg, ra] \in \text{Professoren} \wedge$$
$$\forall na, t_1, sws_1([na, t_1, sws_1, p] \in \text{Vorlesungen} \Rightarrow$$
$$\forall vo([vo, na] \in \text{voraussetzen} \Rightarrow$$
$$\exists t_2, sws_2([vo, t_2, sws_2, p] \in \text{Vorlesungen})$$
$$)))\}$$

# Aufgabe 03

- Gegeben sei die folgende Relation Zehnkampf mit Athletennamen und den von ihnen erreichten Punkten im Zehnkampf:

| Name | Punkte |
|---|---|
| Eaton | 8869 |
| Suarez | 8523 |
| Behrenbruch | 8126 |
| Hardee | 8671 |
| ... | ... |

a) Ermitteln Sie die Goldmedaillengewinner in relationaler Algebra. Eine Goldmedaille bekommen alle, für die gilt: es gibt niemand besseren (also mit mehr Punkten).

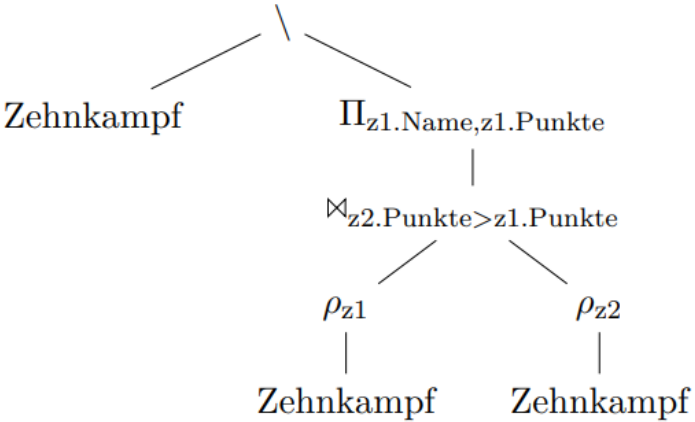
b) Ermitteln Sie die Silbermedaillengewinner im Tupelkalkül. Eine Silbermedaille bekommen alle, für die gilt: es gibt genau eine/n bessere/n.

# Lösungsvorschlag 03 a

Goldmedaillengewinner in relationaler Algebra

$$(\text{Zehnkampf} \setminus (\Pi_{\text{Name,Punkte}}(\rho_{\text{z1}}\text{Zehnkampf} \bowtie_{\text{z2.Punkte}>\text{Punkte}} \rho_{\text{z2}}\text{Zehnkampf})))$$

In Operatorbaumdarstellung:



# Lösungsvorschlag 03 b

Silbermedaillengewinner in Tupelkalkül

$$
\begin{aligned}
\{k| \quad & k \in \text{Zehnkampf} \wedge \\
& \exists k_{gold} \in \text{Zehnkampf}( \\
& k_{gold}.\text{Punkte} > k.\text{Punkte} \wedge \forall k_{andere} \in \text{Zehnkampf}( \\
& k_{andere}.\text{Punkte} \geq k_{gold}.\text{Punkte} \Rightarrow k_{andere}.\text{Name} = k_{gold}.\text{Name}) \wedge \\
& \neg \exists k_{zwischen} \in \text{Zehnkampf}(k_{zwischen}.\text{Punkte} > k.\text{Punkte} \wedge \\
& k_{zwischen}.\text{Punkte} < k_{gold}.\text{Punkte}))\}
\end{aligned}
$$

# Aufgabe 04

- Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL:

  (a) Finden Sie die Studenten, die Sokrates aus Vorlesung(en) kennen.
  (b) Finden Sie die Studenten, die Vorlesungen hören, die auch Fichte hört.
  (c) Finden Sie die Assistenten von Professoren, die den Studenten Fichte unterrichtet haben – z.B. als potentielle Betreuer seiner Diplomarbeit.
  (d) Geben Sie die Namen der Professoren an, die Xenokrates aus Vorlesungen kennt.
  (e) Welche Vorlesungen werden von Studenten im Grundstudium (1.-4. Semester) gehört? Geben Sie die Titel dieser Vorlesungen an.

# Losüngsvorschlag 4a

- Die Studenten, die Sokrates aus Vorlesungen kennen

- ```
  select s.Name, s.MatrNr
  from Studenten s, hoeren h, Vorlesungen v, Professoren p
  where s.MatrNr = h.MatrNr and
        h.VorlNr = v.VorlNr and
        v.gelesenVon = p.PersNr and
        p.Name ='Sokrates';
  ```
  ```

# Losüngsvorschlag 4b

- die Studenten, die Vorlesungen hören, die auch Fichte hört.

- ```
  select distinct s1.Name, s1.MatrNr
  from Studenten s1, Studenten s2, hoeren h1, hoeren h2
  where s1.MatrNr = h1.MatrNr
  and s1.MatrNr != s2.MatrNr
  and s2.MatrNr = h2.MatrNr
  and h1.VorlNr = h2.VorlNr
  and s2.Name ='Fichte';
  ```
  ```

# Losüngsvorschlag 4c

- die Assistenten von Professoren, die den Studenten Fichte unterrichtet haben – z.B. als potentielle Betreuer seiner Diplomarbeit

- ```
  select a.Name, a.PersNr
  from Assistenten a, Professoren p, Vorlesungen v, hoeren h, Studenten s
  where a.Boss = p.PersNr
  and p.PersNr = v.gelesenVon
  and v.VorlNr = h.VorlNr
  and h.MatrNr = s.MatrNr
  and s.Name ='Fichte';
  ```
  ```

# Losüngsvorschlag 4d

- die Namen der Professoren an, die Xenokrates aus Vorlesungen kennt.

- ```
  select p.PersNr, p.Name
  from Professoren p, hoeren h, Vorlesungen v, Studenten s
  where p.PersNr = v.gelesenVon
  and v.VorlNr = h.VorlNr
  and h.MatrNr = s.MatrNr
  and s.Name ='Xenokrates';
  ```
  ```

# Losüngsvorschlag 4e

- Welche Vorlesungen werden von Studenten im Grundstudium (1.-4. Semester) gehört? Geben Sie die Titel dieser Vorlesungen an

- ```
  select v.Titel
  from Vorlesungen v, hoeren h, Studenten s
  where v.VorlNr = h.VorlNr
  and h.MatrNr = s.MatrNr
  and s.Semester between 1 and 4;
  ```
  ```