Informationsintegration
Bucket Algorithmus

06.01.2020

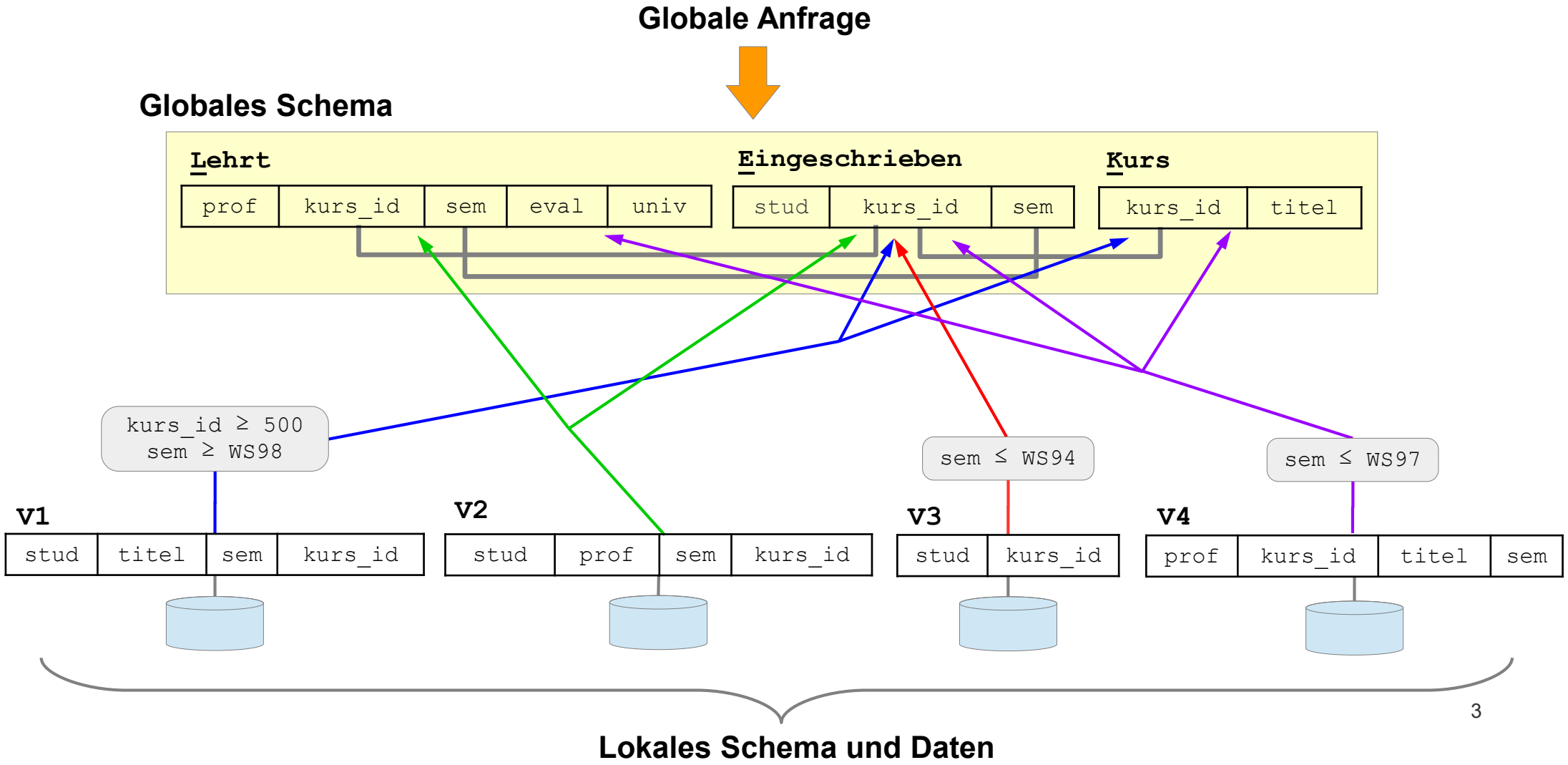
Armin Roth
Felix Naumann

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Überblick

1. Nutzbarkeit und Nützlichkeit von Views
2. Bucket Algorithmus am Beispiel
3. Bucket Algorithmus en detail

„Answering Queries using Views“



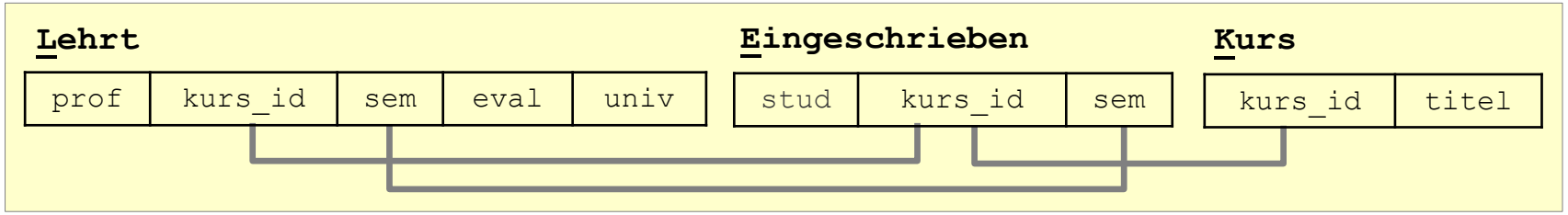
Anfrageumschreibung (WdH)

- Prinzipiell:
  - Prüfe jede Kombination an Sichten auf Containment
  - Unendlich viele Kombinationen, da eine Sicht auch mehrfach in eine Umschreibung eingehen kann.
- Verbesserungen:
  - Satz: Umschreibung mit maximal so vielen Sichten wie Relationen in Anfrage (ohne range-Prädikate) [LMSS95].
  - Geschickte Vorauswahl der Sichten: Nutzbarkeit

LaV – Nutzen der Sichten

- Frage 1: Wann sind Sichten nutzbar?
  Informell:
  - Mindestens eine Relation mit Anfrage gemeinsam
  - Mindestens einige Attribute der Anfrage
  - Prädikate sind schwächer oder gleich (äquivalente Umschreibung)
  - Prädikate sind stärker / selektiver (contained Umschreibung)
- Frage 2: Wann sind Sichten nützlich?
  - Bei Optimierung mit Materialized Views: Schnellere Ausführung
  - Bei Integration mit LaV:
    - Zusätzliche Tupel
    - Zusätzliche Attribute

Beispiel – Globale Anfrage

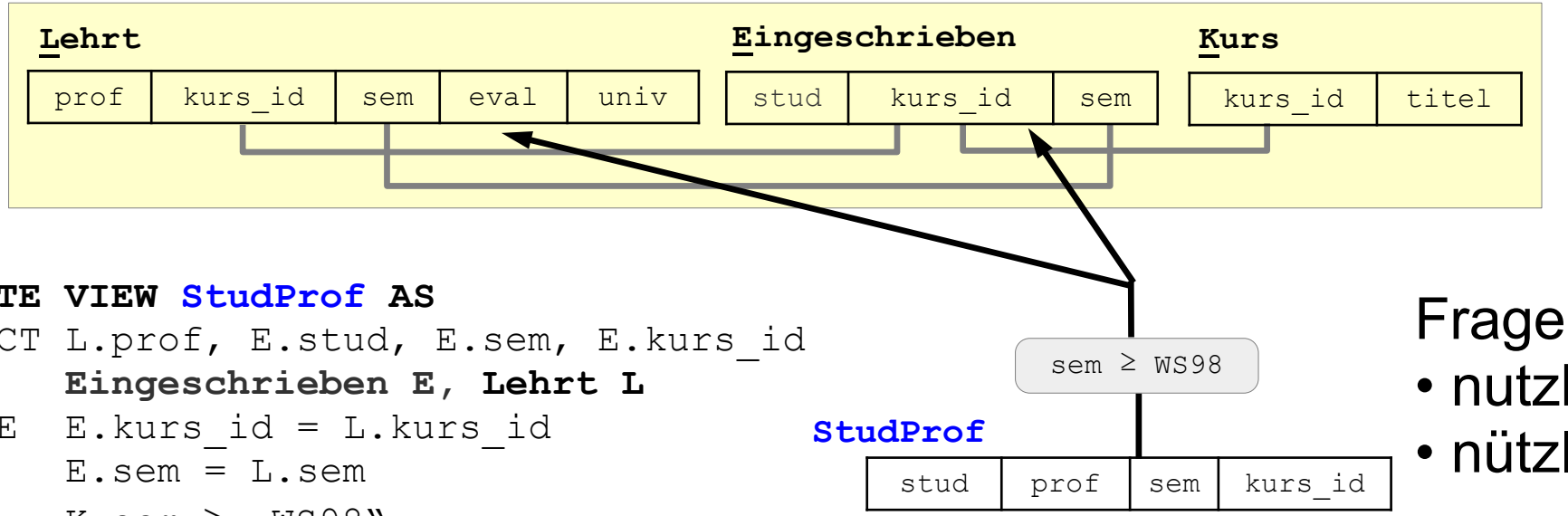


- Anfrage

```
SELECT E.stud, L.prof, E.sem
FROM   Lehrt L, Eingeschrieben E, Kurs K
WHERE  E.kurs_id = L.kurs_id
AND    E.kurs_id = K.kurs_id
AND    E.sem = L.sem
AND    K.titel LIKE „%Daten%“
AND    E.sem ≥ „WS98“
```

- In Worten?
  Finde die Teilnehmer der Kurse, deren Professoren ab dem Semester „WS98“ einen Kurs gelehrt haben, in dessen Titel das Wort „Daten“ vorkommt.
- Anmerkung: `kurs_id` ist global konsistent

LaV – Beispiel



View:

```
CREATE VIEW StudProf AS
SELECT L.prof, E.stud, E.sem, E.kurs_id
FROM   Eingeschrieben E, Lehrt L
WHERE  E.kurs_id = L.kurs_id
AND    E.sem = L.sem
AND    K.sem ≥ „WS98“
```

Globale Anfrage:

```
SELECT  E.stud, L.prof, E.sem
FROM    Lehrt L, Eingeschrieben E,
        Kurs K
WHERE   E.kurs_id = L.kurs_id
AND     E.kurs_id = K.kurs_id
AND     E.sem = L.sem
AND     K.titel LIKE „%Daten%“
AND     E.sem ≥ „WS98“
```

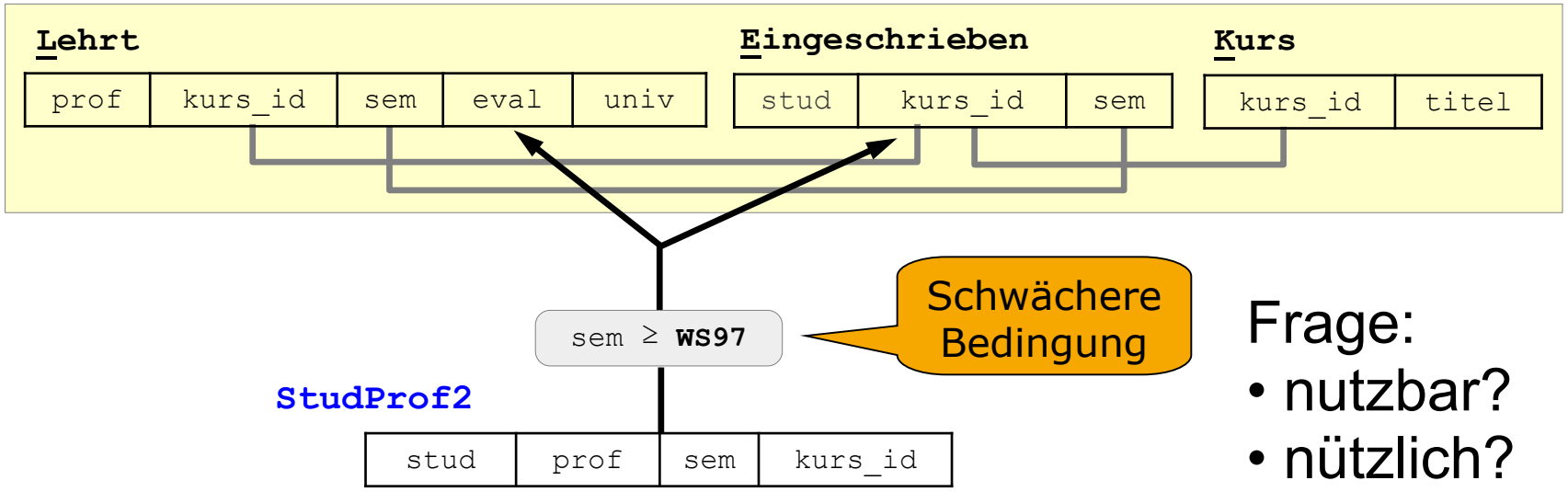
Frage:
- nutzbar?
- nützlich?

Umschreibung:

```
SELECT S.prof, S.stud, S.sem
FROM   StudProf S, Kurs K
WHERE  S.kurs_id = K.kurs_id
AND    K.titel LIKE „%Daten%“
```

Vorsicht: Umschreibung nutzt noch globale Relation **Kurs**!
Annahme dafür: Triviale Sicht

LaV – Beispiel



Frage:
- nutzbar?
- nützlich?

Globale Anfrage:

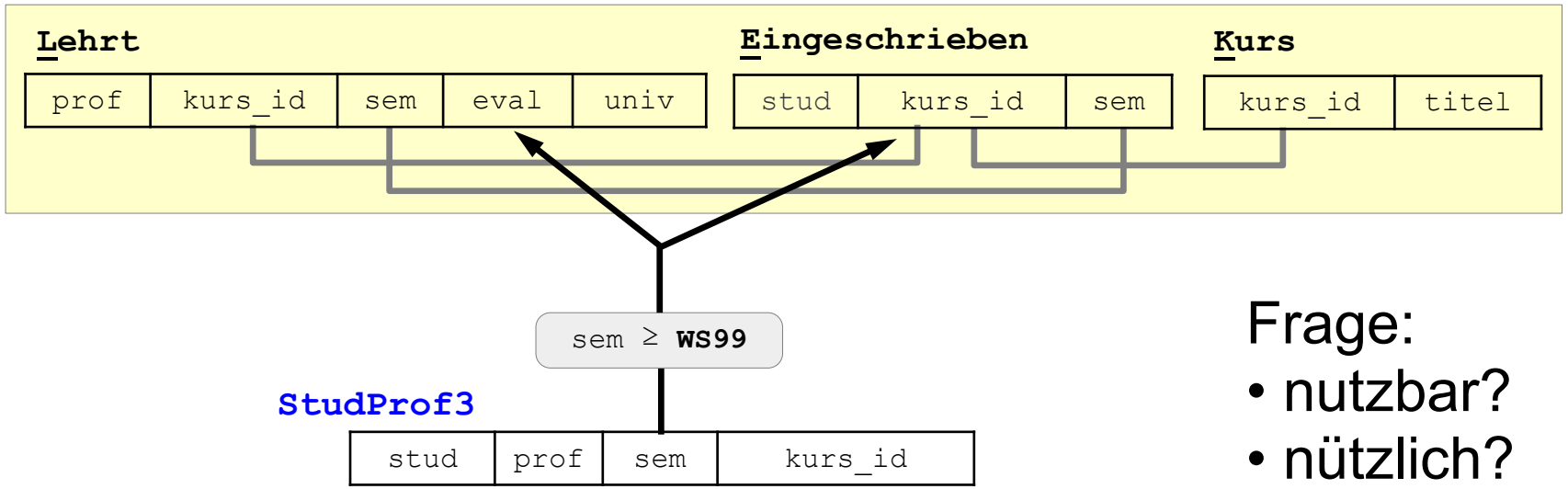
```
SELECT E.stud, L.prof, E.sem
FROM   Lehrt L, Eingeschrieben E,
       Kurs K
WHERE  E.kurs_id = L.kurs_id
AND    E.kurs_id = K.kurs_id
AND    E.sem = L.sem
AND    K.titel LIKE „%Daten%“
AND    E.sem ≥ „WS98“
```

Umschreibung:

```
SELECT S.stud, S.prof, S.sem
FROM   StudProf2 S, Kurs K
WHERE  S.kurs_id = K.kurs_id
AND    K.titel LIKE „%Daten%“
AND    S.sem ≥ „WS98“
```

Umschreibung ist contained bzw. äquivalent.

LaV – Beispiel



Frage:
- nutzbar?
- nützlich?

Globale Anfrage:

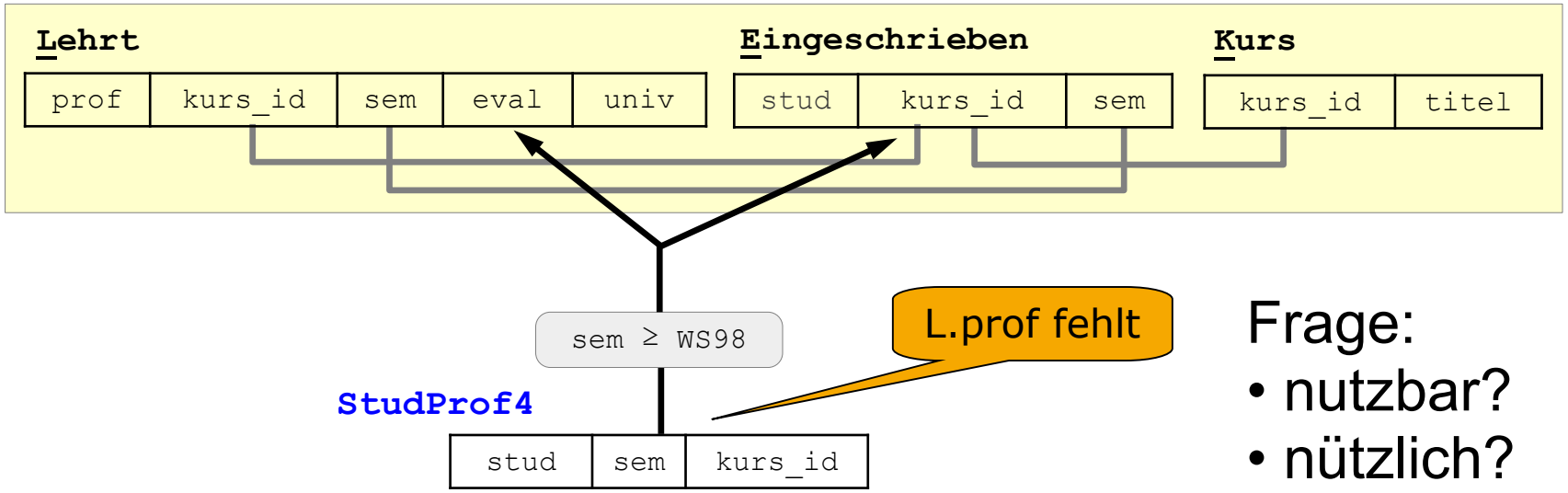
```
SELECT  E.stud, L.prof, E.sem
FROM    Lehrt L, Eingeschrieben E,
        Kurs K
WHERE   E.kurs_id = L.kurs_id
AND     E.kurs_id = K.kurs_id
AND     E.sem = L.sem
AND     K.titel LIKE „%Daten%“
AND     E.sem ≥ „WS98“
```

Umschreibung:

```
SELECT S.stud, S.prof, S.sem
FROM   StudProf3 S, Kurs K
WHERE  S.kurs_id = K.kurs_id
AND    S.kurs_id = L.kurs_id
AND    K.titel LIKE „%Daten%“
```

Umschreibung ist contained aber nicht äquivalent ⇒ OK für Daten-integration, aber ungünstig für MVs

LaV – Beispiel



Frage:
- nutzbar?
- nützlich?

Globale Anfrage:

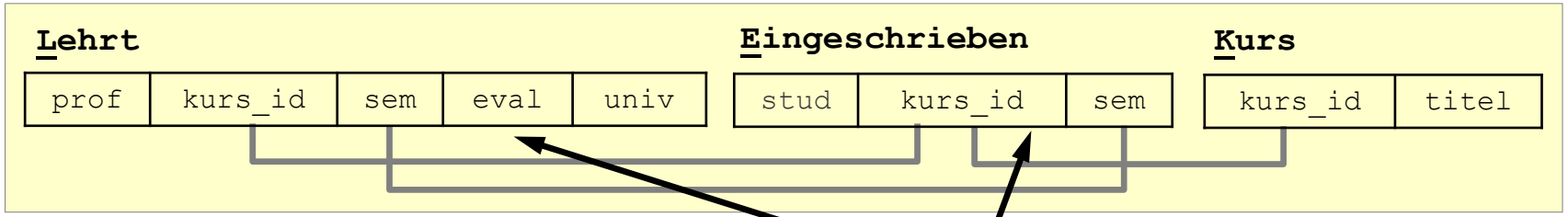
```
SELECT E.stud, L.prof, E.sem
FROM   Lehrt L, Eingeschrieben E,
Kurs K
WHERE  E.kurs_id = L.kurs_id
AND    E.kurs_id = K.kurs_id
AND    E.sem = L.sem
AND    K.titel LIKE „%Daten%“
AND    E.sem ≥ „WS98“
```

Umschreibung:

```
SELECT L.prof, S.stud, S.sem
FROM   StudProf4 S, Kurs K, Lehrt L
WHERE  S.kurs_id = K.kurs_id
AND    S.kurs_id = L.kurs_id
AND    S.sem = L.sem
AND    K.titel LIKE „%Daten%“
```

Mühselig: L muss erneut gejoint werden

LaV – Beispiel



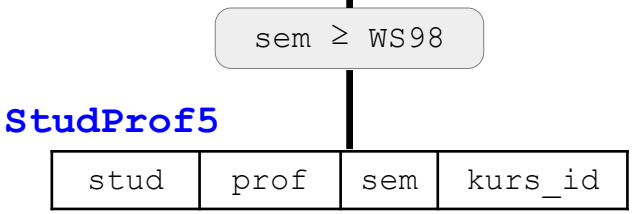
E.sem = L.sem fehlt

Quelle:

```
CREATE VIEW StudProf5 AS
SELECT E.stud, E.sem, L.prof
FROM   Eingeschrieben E,
       Lehrt L
WHERE  E.kurs_id = L.kurs_id
AND    E.sem ≥ „WS98“
```

Globale Anfrage:

```
SELECT E.stud, L.prof, E.sem
FROM   Lehrt L, Eingeschrieben E,
Kurs K
WHERE  E.kurs_id = L.kurs_id
AND    E.kurs_id = K.kurs_id
AND    E.sem = L.sem
AND    K.titel LIKE „%Daten%“
AND    E.sem ≥ „WS98“
```



Frage:
- nutzbar?
- nützlich?

Mühselig: L muß erneut gejoined werden, weil L.sem von StudProf5 nicht exportiert wird.

Umschreibung:

```
SELECT S.stud, S.sem, S.prof
FROM   StudProf5 S, Kurs K,
       Lehrt L
WHERE  S.kurs_id = K.kurs_id
AND    S.kurs_id = L.kurs_id
AND    L.sem = S.sem
AND    K.titel LIKE „%Daten%“
```

LaV – Nutzbarkeit von Sichten

- Eine Sicht V ist nutzbar für eine äquivalente Umschreibung der Anfrage Q, falls
  - Jede Relation in V muss einer Relation in Q entsprechen.
  - Bedingung in Q (Joins und Selektionen) auf den Relationen in V müssen entweder direkt in V angewendet werden

    oder

    Schwächere oder keine Bedingungen werden angewandt und die entsprechenden Attribute werden exportiert.
  - V projiziert keine Attribute heraus, die noch in Q gebraucht werden und nicht anderweitig (andere Sicht oder Basisrelation) verfügbar sind.
- Nützlichkeit hängt von DBMS und Extension ab.
  - Indizes, etc.

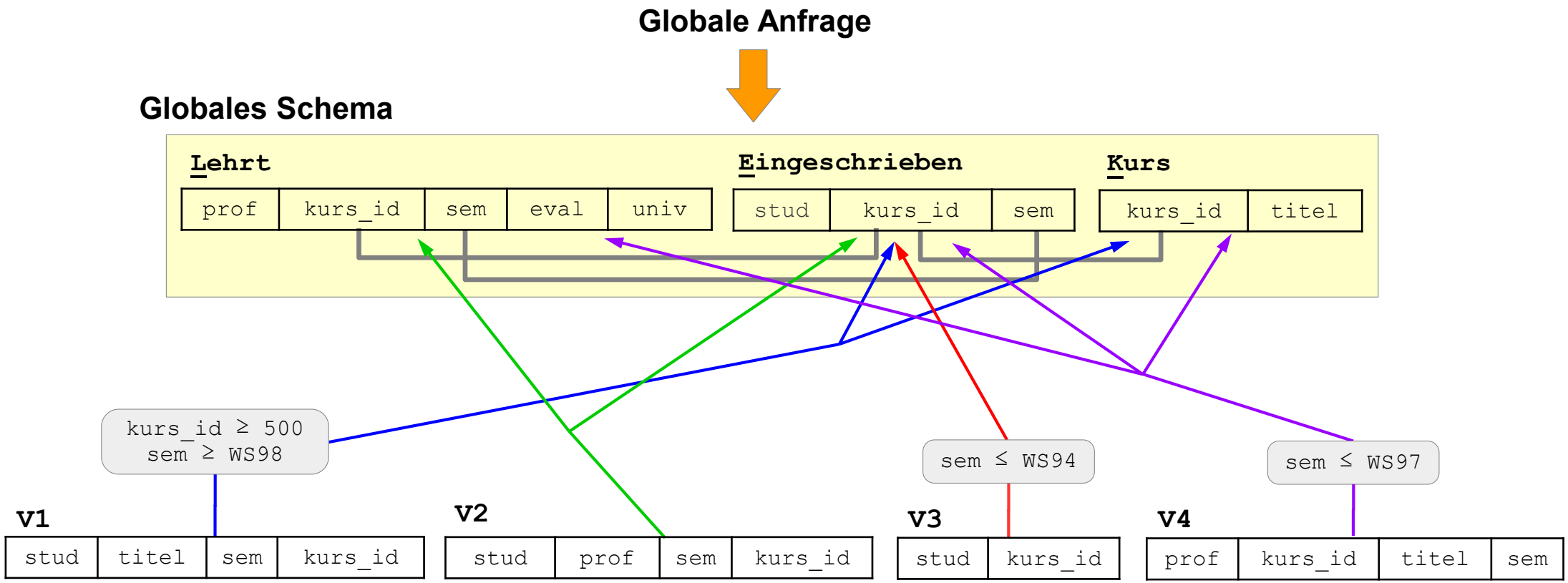
Überblick

1. Nutzbarkeit und Nützlichkeit von Views
2. Bucket Algorithmus am Beispiel
3. Bucket Algorithmus en detail

LaV – Bucket Algorithm (BA)

- Problem:
  - Man muss exponentiell viele Kombinationen auf containment prüfen.
  - Schon die Prüfung selbst ist ein NP-vollständiges Problem
    - es gibt jedoch effiziente Algorithmen
    - idR sind die Kombinationen nicht groß (Anzahl der Relationen)
- Idee zur weiteren Verbesserung:
  - Reduktion der Anzahl der Kombinationen durch geschickte Vorauswahl
  - Jede Relation der Anfrage erhält einen bucket (Korb).
  - Schritt 1: Füge in jeden bucket alle Sichten, die für die Relation nutzbar sind.
  - Schritt 2: Prüfe alle Anfrageumschreibungs-Kombinationen, die aus jedem bucket genau eine Sicht enthalten.

LaV – BA – Beispiel



V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98

V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)

V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94

V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

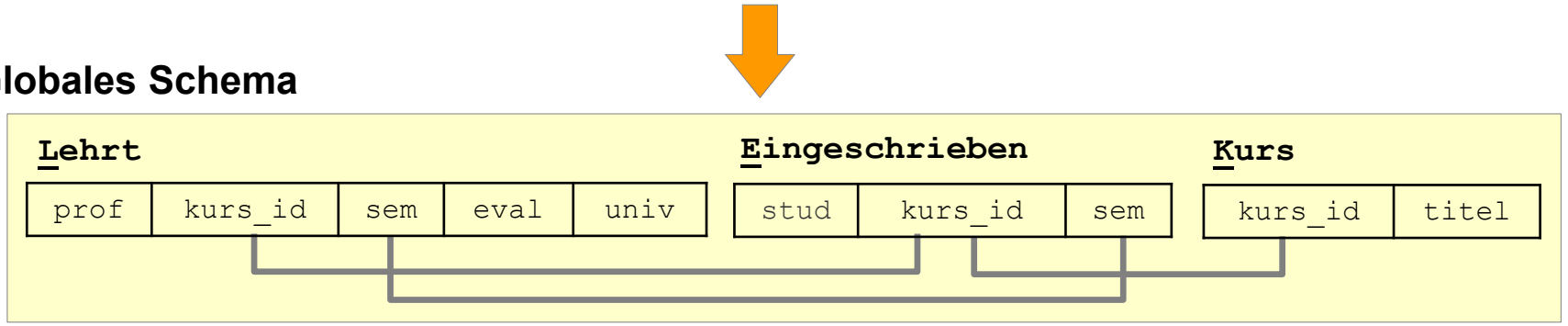
LaV – BA – Beispiel

Globale Anfrage Q:

```
SELECT E.stud, K.kurs_id, L.prof
FROM   Lehrt L, Eingeschrieben E, Kurs K
WHERE  E.kurs_id = L.kurs_id
AND    E.sem = L.sem
AND    K.kurs_id = E.kurs_id
AND    E.sem ≥ WS95 AND kurs_id ≥ 300
```

```
Q(stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem),
                          K(kurs_id, titel), sem ≥ WS95, kurs_id ≥ 300
```

Globales Schema



LaV – BA – Beispiel

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98

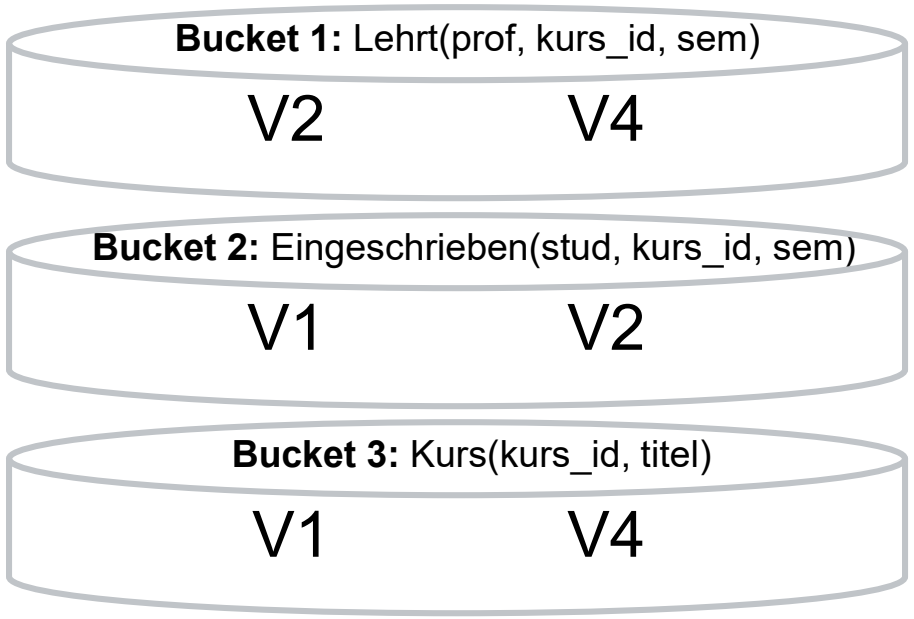
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)

V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94

V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage:

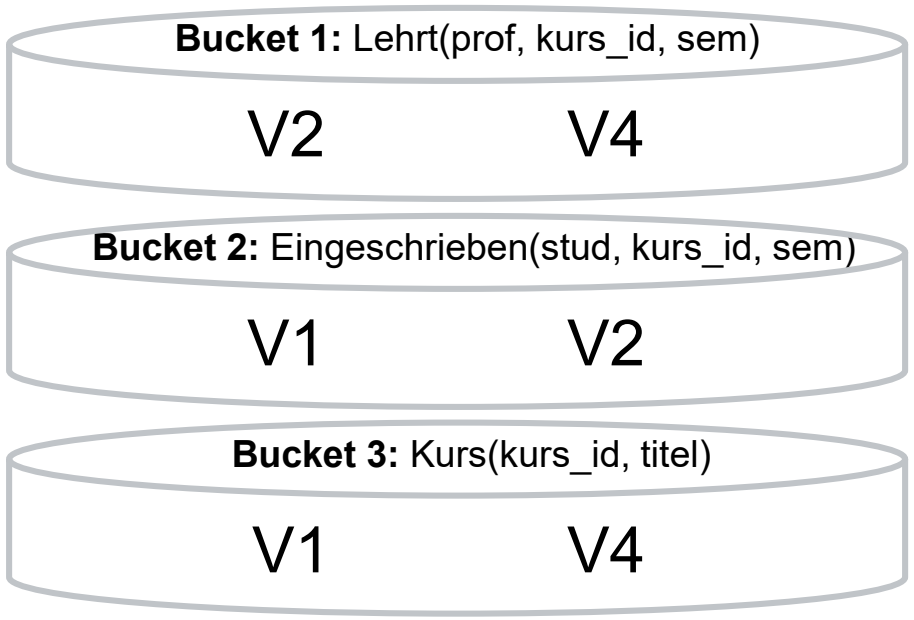
Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300



LaV – BA – Beispiel

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)
V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94
V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage: Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel),
sem ≥ WS95, kurs_id ≥ 300



Kombinationen

V2, V1, V1 → V1,V2
V4, V1, V4 → Ø
V2, V2, V4 → V2,V4
V2, V2, V1 → V1,V2
V2, V1, V4 → Ø
V4, V1, V1 → Ø
V4, V2, V4 → V2,V4
V4, V2, V1 → Ø

LaV – BA – Beispiel

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)
V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94
V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage: Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel),
sem ≥ WS95, kurs_id ≥ 300

Kombinationen

V2, V1, V1 → V1,V2
V4, V1, V4 → Ø
V2, V2, V4 → V2,V4
V2, V2, V1 → V1,V2
V2, V1, V4 → Ø
V4, V1, V1 → Ø
V4, V2, V4 → V2,V4
V4, V2, V1 → Ø

Ergebnis des Algorithmus:
V1,V2 ∪ V2,V4

LaV – BA – Beispiel

Quelle 1:

```
CREATE VIEW V1 AS
SELECT E.stud, K.titel, K.sem, K.kurs_id
FROM   Eingeschrieben E, Kurs K
WHERE E.kurs_id = K.kurs_id
AND      K.kurs_id LIKE „%VL_“
AND      E.sem ≥ WS98
```

Quelle 2:

```
CREATE VIEW V2 AS
SELECT E.stud, L.prof, E.sem, L.kurs_id
FROM   Eingeschrieben E, Lehrt L
WHERE E.kurs_id = L.kurs_id
AND      E.sem = L.sem
```

Quelle 4:

```
CREATE VIEW V4 AS
SELECT L.prof, K.kurs_id, K.titel, E.sem
FROM   Eingeschrieben E, Kurs K, Lehrt L
WHERE E.kurs_id = K.kurs_id
AND      E.kurs_id = L.kurs_id
AND      L.sem = E.sem
AND      E.sem ≤ WS97
```

Anfrage Q:

```
SELECT E.stud, K.kurs_id, L.prof
FROM   Lehrt L, Eingeschrieben E, Kurs K
WHERE E.kurs_id = L.kurs_id
AND      E.sem = L.sem
AND      K.kurs_id = E.kurs_id
AND      E.sem ≥ WS95
```

Umgeschriebene Anfrage Q‘

V1,V2 ∪ V2,V4**:**

```
SELECT V1.stud, V1.kurs_id, V2.prof
FROM   V1, V2
WHERE V1.sem = V2.sem
AND      V1.kurs_id = V2.kurs_id

UNION

SELECT V2.stud, V2.kurs_id, V2.prof
FROM   V2, V4
WHERE V2.sem = V4.sem
AND      V2.kurs_id = V4.kurs_id
AND      V2.prof = V4.prof
AND      V2.sem ≥ WS95
```

Überblick

1. Nutzbarkeit und Nützlichkeit von Views
2. Bucket Algorithmus am Beispiel
3. Bucket Algorithmus en detail

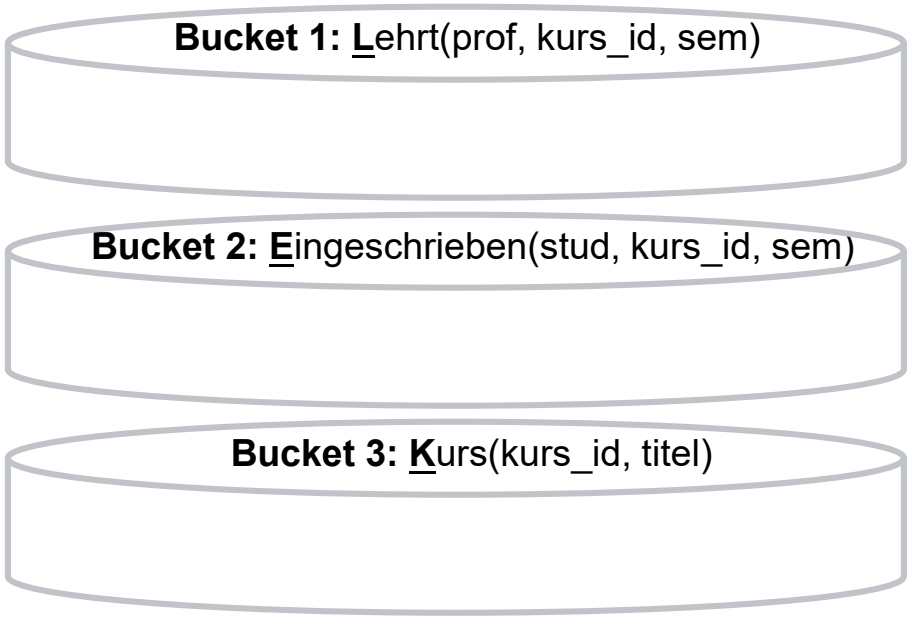
BA – en detail

Anfrage:
Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel), sem ≥ WS95, kurs_id ≥ 300

Teilziele (*subgoals*) von Q

Erzeuge für jedes Teilziel einen *bucket*.

Fülle Sichten in die *buckets*.



Eine Sicht wird in ein bucket gestellt, wenn mindestens ein Teilziel der Sicht „geeignet“ ist. Also:

Für jeden bucket

  Für jede Sicht

    Für jedes Teilziel der Sicht

      Prüfung auf „Eignung“:

      1. Unifier: Mapping von allen Attributen im Teilziel von Q auf Attribute im Teilziel von V, d.h.: Alle Anfrageattribute müssen vorkommen.

      2. Kompatibilität: Prädikate (von Q und V) sind passend (also widerspruchsfrei).

BA – en detail

procedure generate-relevant-sources($\mathcal{I}$,Q)
/* $\mathcal{I}$ is a set of information sources, and Q is a conjunctive query of the form $q(\bar{X}) : (\exists \bar{X}) g_1(\bar{X}_1) \wedge \ldots g_m(\bar{X}_m) \wedge C_q$, where C_q is the conjunction of order atoms in Q. */

for every subgoal $g_i(\bar{X}_i)$, $1 \leq i \leq m$ **do:**
  $relevantSources_i = \emptyset$
  for every non-order conjunct $u(\bar{Y})$ in a formula
    r_v in the pair (v, r_v) in a description of a source in
    $\mathcal{I}$ **do:**
    if $g_i = u$ **or** g_i and u are non-disjoint classes **then:**
      Let ψ be the mapping defined on the variables of
      r_v as follows:
        if Y is the j'th variable in $\bar{Y}$ and is not
          existentially quantified in r_v
          then $\psi(Y) = X_j$, where X_j is the j'th
          variable in $\bar{X}_i$.
          else $\psi(Y)$ is a new variable that does not
          appear in Q or r_v.
      Let $C(Q)$ and $C(v)$ be the conjunction of
        constraint subgoals in Q and $\psi(r_v)$, respectively.
      if $C(Q) \wedge C(v)$ is satisfiable, then add $\psi(r_v)$
        to $relevantSources_i$.
return $\{relevantSources_1, \ldots, relevantSources_m\}$.
end generate-relevant-sources.

Für jedes Teilziel in Q

erzeuge einen bucket

Für jedes Teilziel in allen V

Finde Unifier (mapping)

Prüfe Kompatibilität der Prädikate

Füge V in bucket ein

[LRO96a]

BA – en detail

Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V1 (stud, titel, sem, kurs_id) :-
E(stud,kurs_id,sem),
K(kurs_id,titel),
kurs_id ≥ 500,
sem ≥ WS98

prof wird nicht exportiert!

BA – en detail

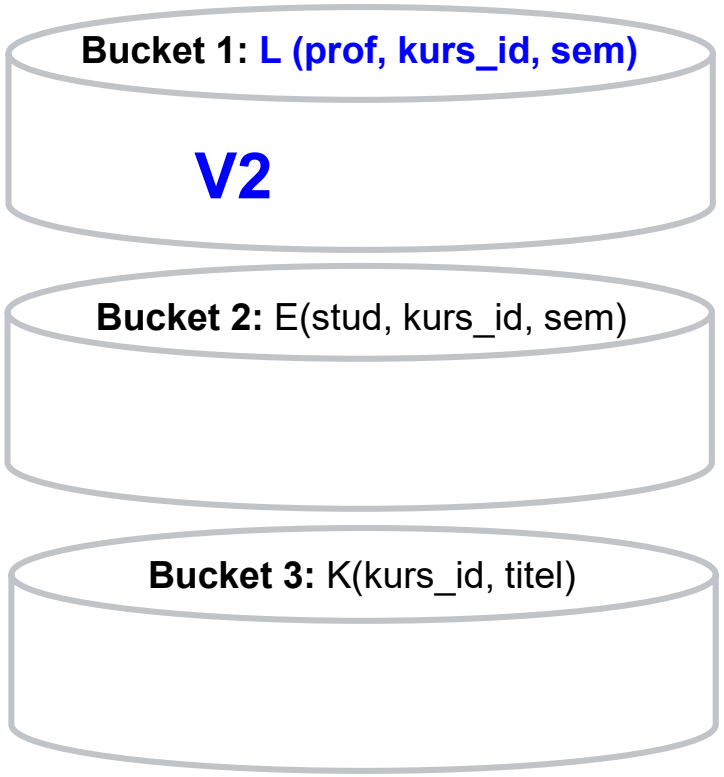
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V2 (stud, prof, sem, kurs_id) :-
E(stud, kurs_id, sem),
L(prof, kurs_id, sem)



BA – en detail

Für jede Sicht: Betrachte deren Teilziele.

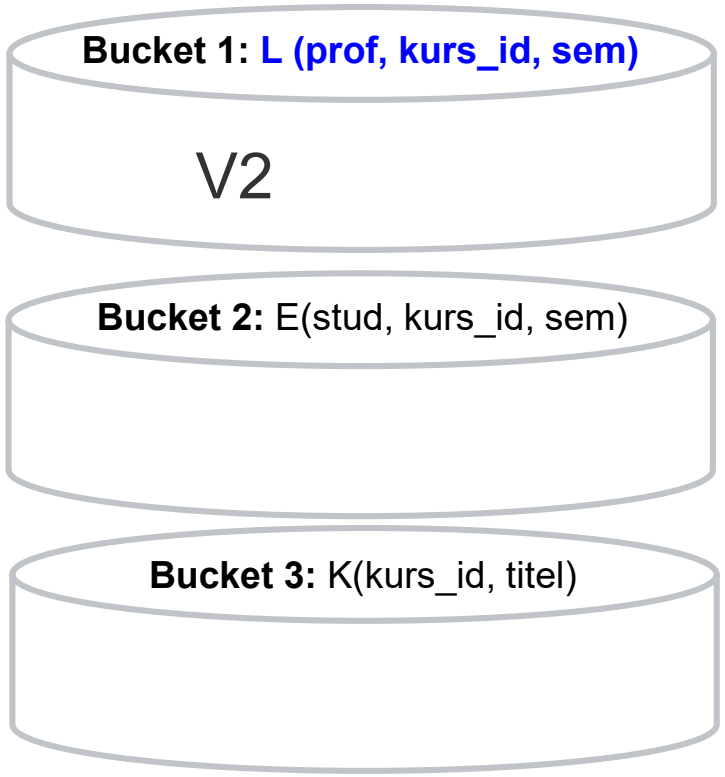
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V3 (stud, kurs_id) :-
E(stud, kurs_id, sem),
sem ≤ WS94

prof wird nicht exportiert und L nicht abgedeckt!



BA – en detail

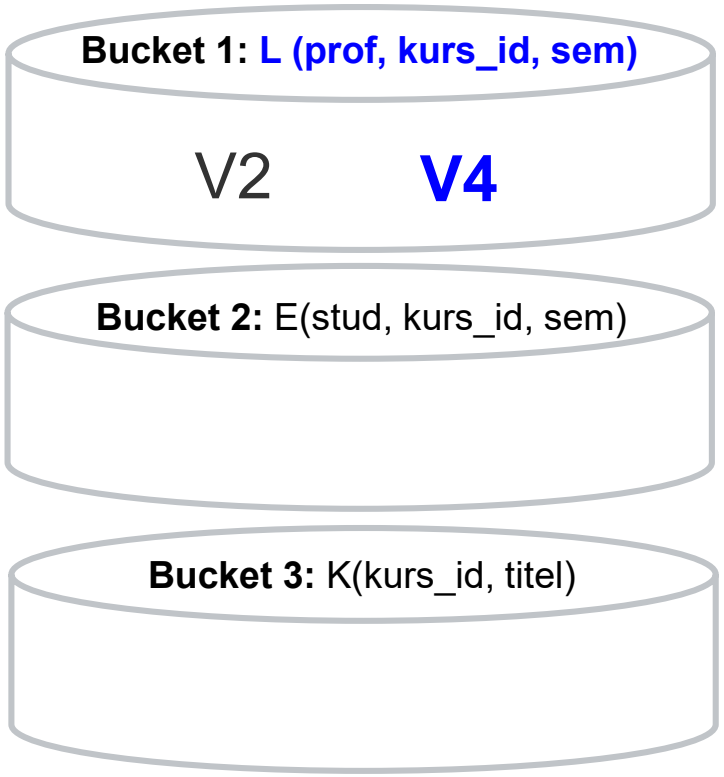
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V4 (prof, kurs_id, titel, sem) :-
L(prof, kurs_id, sem),
K(kurs_id, titel),
E(stud, kurs_id, sem),
sem ≤ WS97



BA – en detail

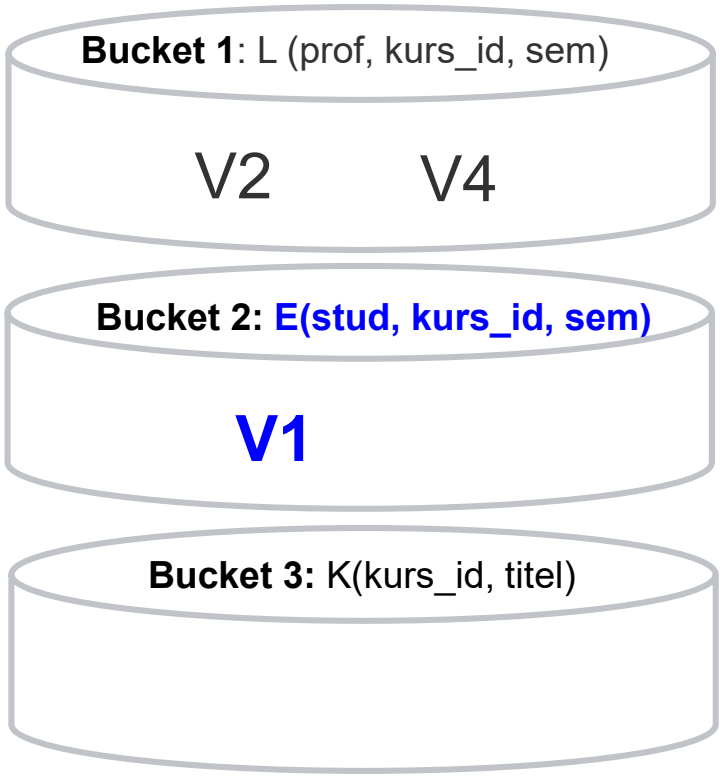
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V1 (stud, titel, sem, kurs_id) :-
E(stud,kurs_id,sem),
K(kurs_id,titel),
kurs_id ≥ 500,
sem ≥ WS98



BA – en detail

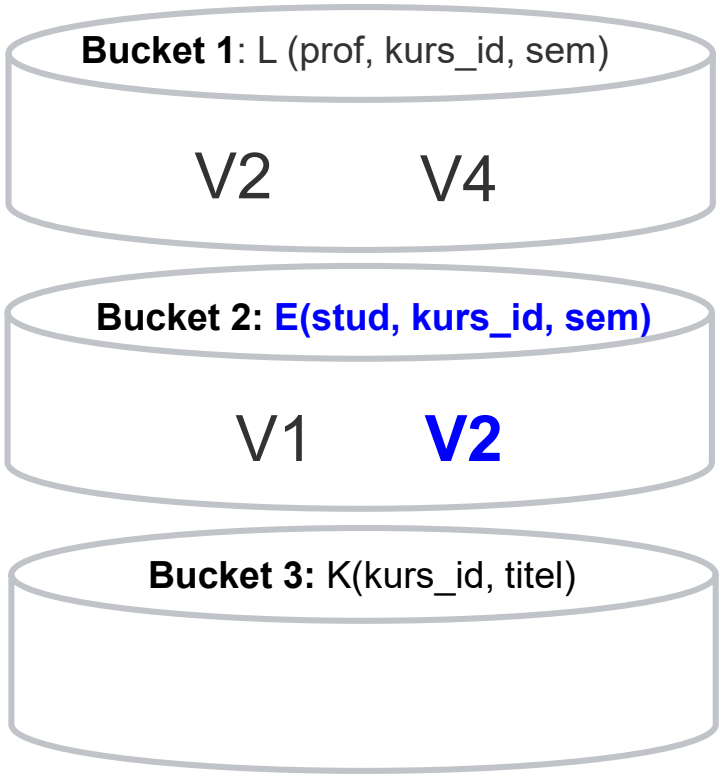
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V2 (stud, prof, sem, kurs_id) :-
E(stud, kurs_id, sem),
L(prof, kurs_id, sem)



BA – en detail

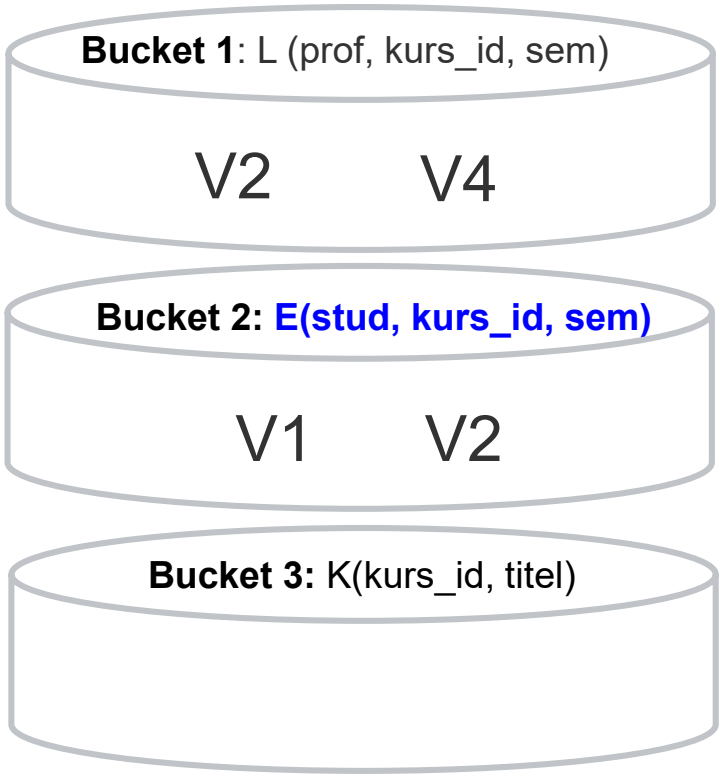
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V3 (stud, kurs_id) :-
E(stud, kurs_id, sem),
sem ≤ WS94



sem ≥ WS95 vs. sem ≤ WS94

BA – en detail

Für jede Sicht: Betrachte deren Teilziele.

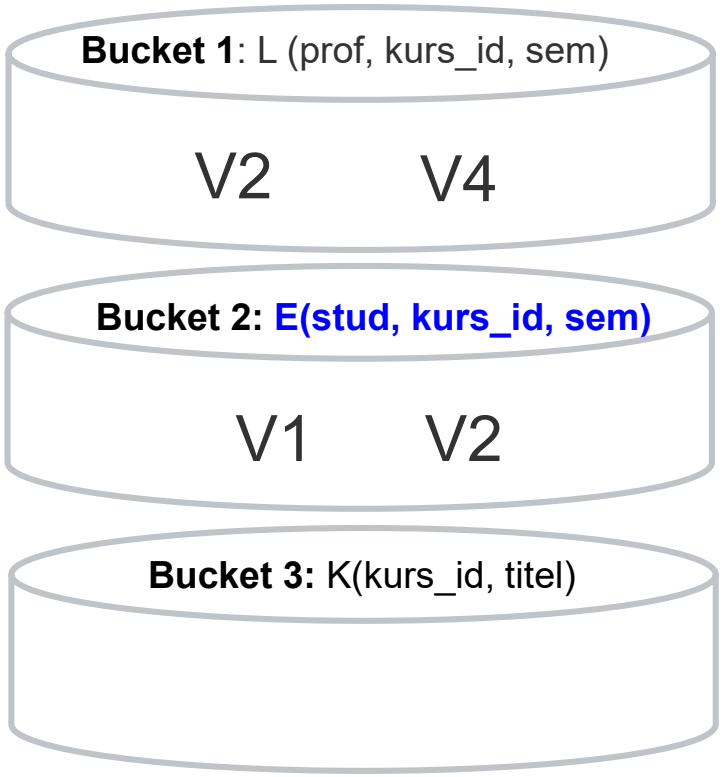
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V4 (prof, kurs_id, titel, sem) :-
L(prof, kurs_id, sem),
K(kurs_id, titel),
E(stud, kurs_id, sem),
sem ≤ WS97

stud wird nicht exportiert!



BA – en detail

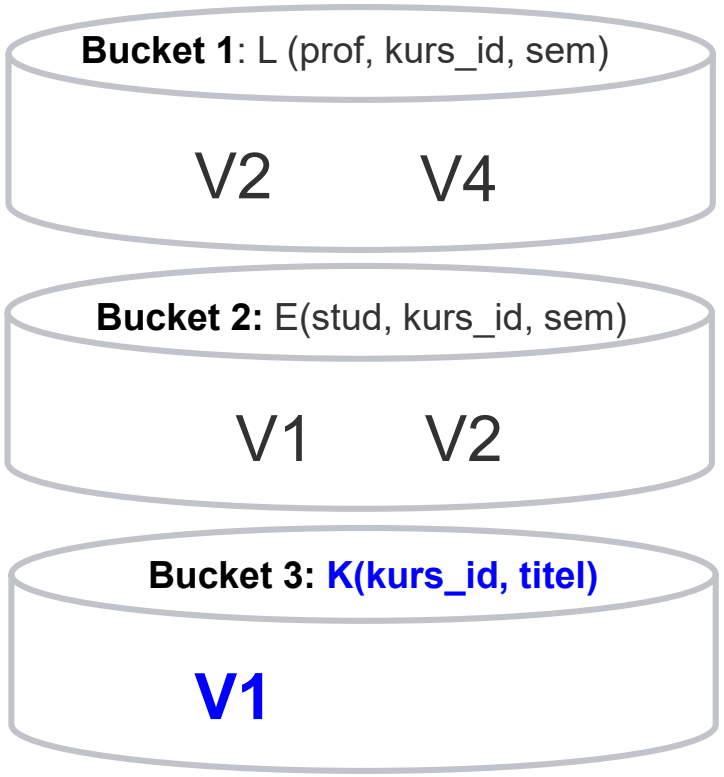
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V1 (stud, titel, sem, kurs_id) :-
E(stud,kurs_id,sem),
K(kurs_id,titel),
kurs_id ≥ 500,
sem ≥ WS98



BA – en detail

Für jede Sicht: Betrachte deren Teilziele.

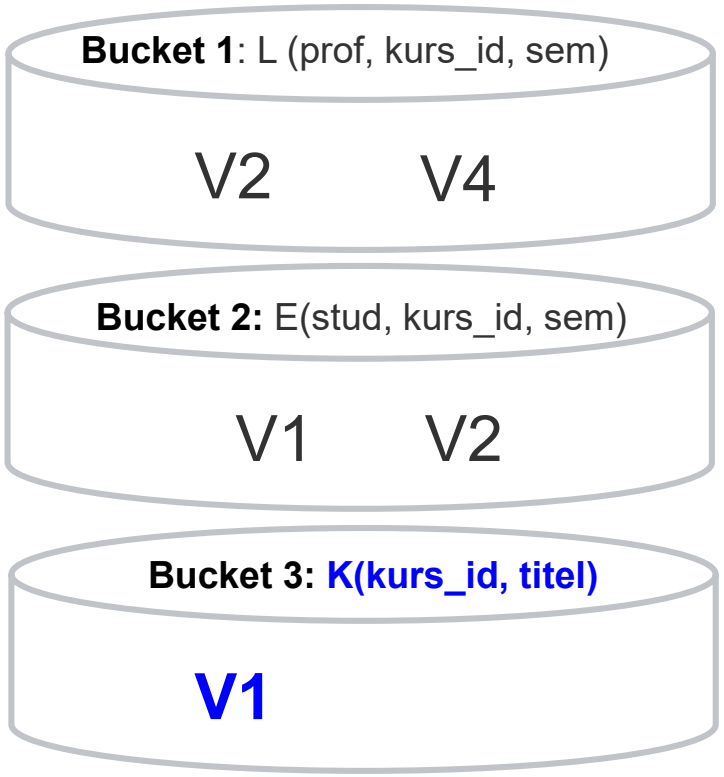
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V2 (stud, prof, sem, kurs_id) :-
E(stud, kurs_id, sem),
L(prof, kurs_id, sem)

titel wird nicht exportiert und K nicht abgedeckt!



BA – en detail

Für jede Sicht: Betrachte deren Teilziele.

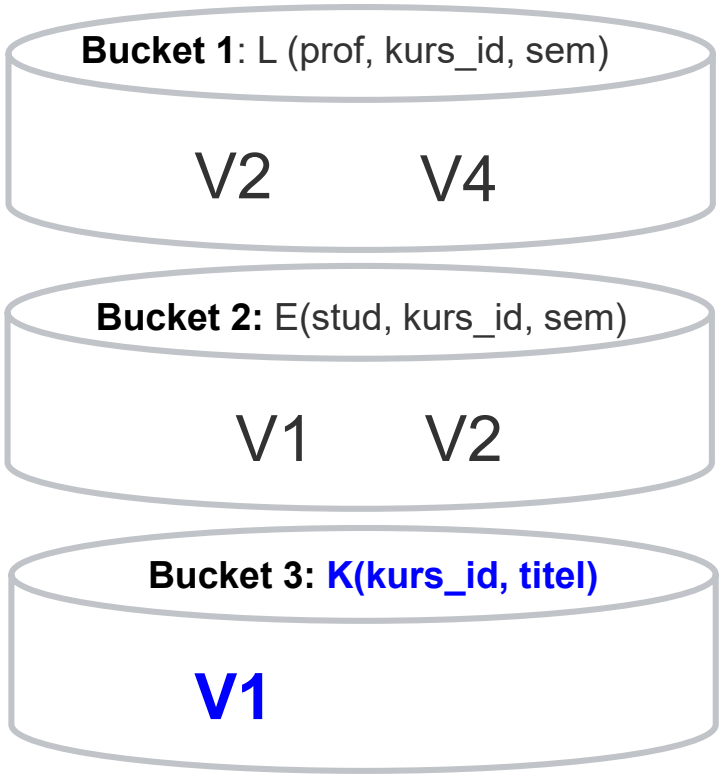
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

V3 (stud, kurs_id) :-
E(stud, kurs_id, sem),
sem ≤ WS94

sem ≤ WS94



BA – en detail

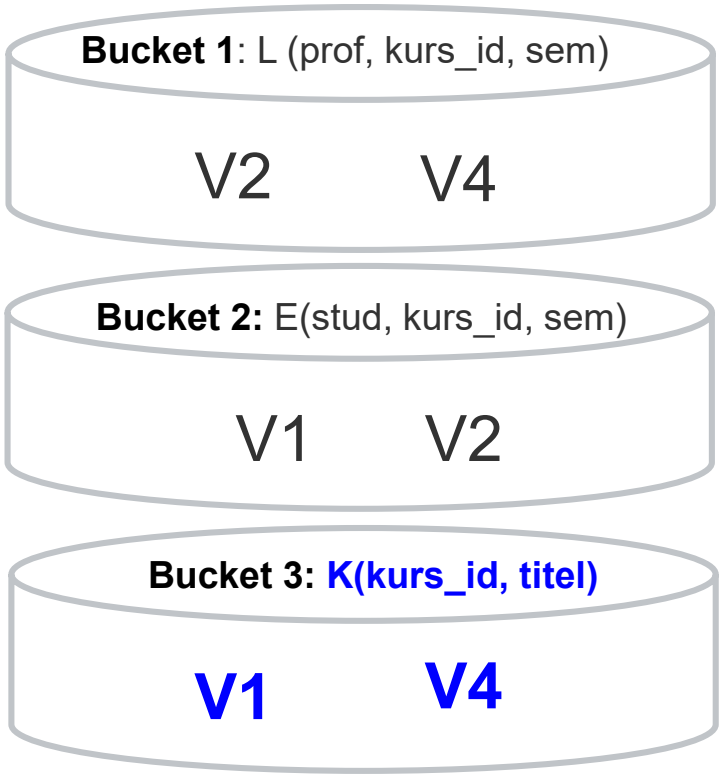
Für jede Sicht: Betrachte deren Teilziele.

Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

Prüfung auf „Eignung“:
1. Unifier: Alle Anfrageattribute müssen vorkommen.
2. Kompatibilität: Prädikate sind passend.

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

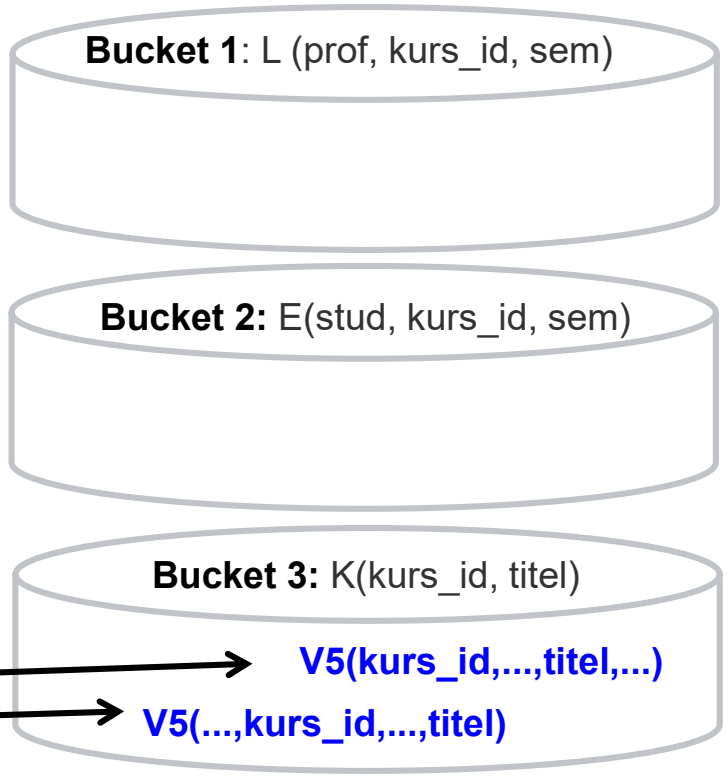
V4 (prof, kurs_id, titel, sem) :-
L(prof, kurs_id, sem),
K(kurs_id, titel),
E(stud, kurs_id, sem),
sem ≤ WS97



- Eine Sicht kann in verschiedenen Buckets auftauchen:
  Verschiedene Rollen
- Eine Sicht kann mehrmals in einem Bucket auftauchen:
  Wenn mehrere Teilziele passen

Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300

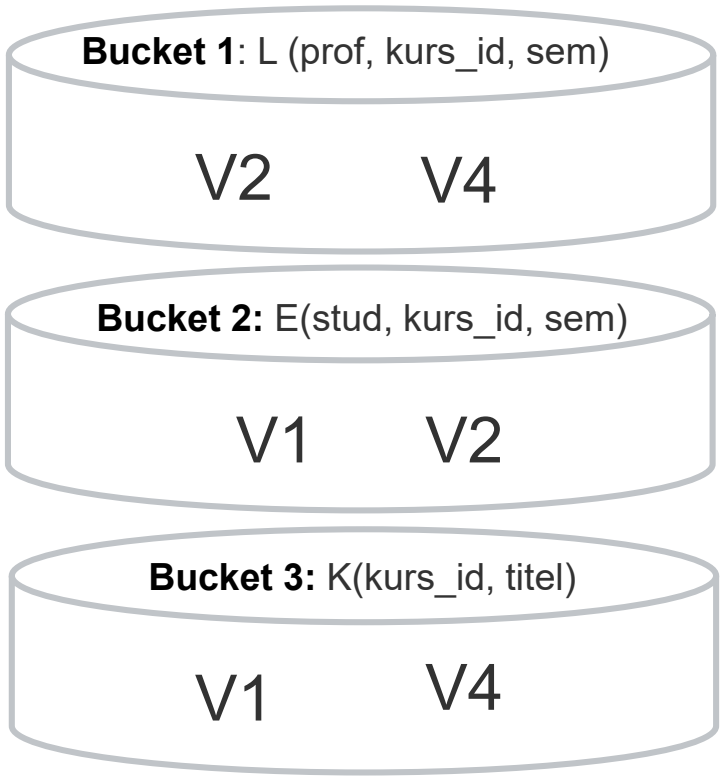
Join zwischen Grund- (bas) und Aufbaukurs (adv):
V5 (bas_kurs_id, adv_kurs_id, bas_titel, adv_titel) :-
K(bas_kurs_id,bas_titel),
K(adv_kurs_id,adv_titel),
adv_kurs_id = bas_kurs_id + 200



LaV – BA – Beispiel

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)
V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94
V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage: Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel),
sem ≥ WS95, kurs_id ≥ 300



Kombinationen

V2, V1, V1
V4, V1, V4
V2, V2, V4
V2, V2, V1
V2, V1, V4
V4, V1, V1
V4, V2, V4
V4, V2, V1

Frage: Welche Kombinationen (Pläne) sind contained? Welche sind nützlich?

- Wieviele Kombinationen?
  - $|B_1| \times |B_2| \times \ldots \times |B_n|$ ($n = |Q|$)
  - Falls m = Anzahl Sichten: $O(m^n)$
  - Wichtig: Jede der exponentiell vielen Kombinationen liefert potentiell einen Teil des Ergebnisses.
- Eine Kombination $Q' = V_1,\ldots,V_k$ ist eine Anfrageumschreibung von Q, falls
  - $Q' \subseteq Q$
  - oder Q', {zusätzliche Prädikate} $\subseteq Q$

LaV – BA – Kombinationen

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)
V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94
V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage: Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel),
sem ≥ WS95, kurs_id ≥ 300

Kombinationen

V2, V1, V1
V4, V1, V4
V2, V2, V4
V2, V2, V1
V2, V1, V4
V4, V1, V1
V4, V2, V4
V4, V2, V1

Einzelne Sichten nützlich (in Bezug auf Anfrage), aber zusammen Widerspruch: sem ≥ WS98 vs. sem ≤ WS97

LaV – BA – Kombinationen

V1 (stud, titel, sem, kurs_id) :- E(stud,kurs_id,sem), K(kurs_id,titel), kurs_id ≥ 500, sem ≥ WS98
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)
V3 (stud, kurs_id) :- E(stud, kurs_id, sem), sem ≤ WS94
V4 (prof, kurs_id, titel, sem) :- L(prof, kurs_id, sem), K(kurs_id, titel), E(stud, kurs_id, sem), sem ≤ WS97

Anfrage: Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem), K(kurs_id, titel),
sem ≥ WS95, kurs_id ≥ 300

Kombinationen

V2, V1, V1
~~V4, V1, V4~~
V2, V2, V4
V2, V2, V1
~~V2, V1, V4~~
~~V4, V1, V1~~
V4, V2, V4
~~V4, V2, V1~~

Q‘(stud, kurs_id, prof):-
V2(stud‘, prof, sem, kurs_id),
V1 (stud, titel‘, sem, kurs_id),
V1 (stud‘, titel, sem‘, kurs_id)

x‘ markiert nicht gebrauchte Attribute.

LaV – BA – Kombinationen

- Prüfe Q ⊇ Q‘(stud, kurs_id, prof):-
  V2(stud‘, prof, sem, kurs_id),
  V1 (stud, titel‘, sem, kurs_id),
  V1 (stud‘, titel, sem‘, kurs_id)
- Wie?
  Q verwendet globale Relationen:
  Q (stud, kurs_id, prof) :- L(prof, kurs_id, sem), E(stud,kurs_id,sem),
  K(kurs_id, titel), sem ≥ WS95, kurs_id ≥ 300
  Q‘ verwendet Sichten
  Durch „unfolding“!
  Q‘ = Q‘‘(stud, kurs_id, prof):-
  E(stud‘, kurs_id, sem), L(prof, kurs_id, sem), sem ≥ WS98,
  E(stud,kurs_id,sem), K(kurs_id,titel‘), kurs_id ≥ 500,
- und Finden eines containment mappings.

LaV – BA – Kombinationen

Endgültiges Ergebnis:
Kombination aller Kombinationen durch UNION:

V1,V2 ∪ V2,V4

Umgeschriebene Anfrage Q‘:

```
SELECT V1.stud, V1.kurs_id, V2.prof
  FROM    V1, V2
  WHERE V1.sem = V2.sem
  AND      V1.kurs_id = V2.kurs_id

  UNION

  SELECT V2.stud, V2.kurs_id, V2.prof
  FROM    V2, V4
  WHERE V2.sem = V4.sem
  AND      V2.kurs_id = V4.kurs_id
  AND      V2.prof = V4.prof
```

- Maximally contained:
  BA produziert nach [LMSS95] maximally contained Umschreibungen.

- Vollständigkeit:
  BA findet eine Anfrageumschreibung Q‘ bestehend aus Sichten V_1,..., V_m für Anfrage Q ohne Prädikate ≥,≤,<,>,≠ (falls sie existiert).
- Beweis:
  Es gibt nur eine (äquivalente) Umschreibung, falls es eine Umschreibung der Länge |Q| gibt [LMSS95].
- ABER: BA findet nicht alle Anfrageumschreibungen [Hal01, LN06]!

BA verpasst Anfrageumschreibung: Extrembeispiel

Globales Schema:

$R(Name, Alter, Stadt)$

Globale Anfrage:

$q(name) :- R(name, alter, \text{stadt}), \text{alter} > 50, \text{stadt} = \text{“}Berlin\text{“}$

Views:

$v_1(name) :- R(name, alter, stadt), alter > 50$

$v_2(name) :- R(name, alter, stadt), \text{stadt} = \text{“}Berlin\text{“}$

Verpasste Umschreibung:

$q(name) :- v_1(name), v_2(name)$

Verpasste Umschreibung hat mehr Relationen als die globale Anfrage (und wird darum vom BA nicht gefunden).

BA verpasst Anfrageumschreibung: Beispiel [LN06]

Globales Schema:

$film(titel, typ, regisseur, länge)$
$spielt(titel, schauspieler\_name, rolle, kritik)$

Globale Anfrage:

$p(r, s, k) :- \ film(t, \_, r, \_), spielt(t, s, \_, k)$

View:

$v(r, s, k) :- film(t, \_, r, \_), spielt(t, s, \_, k)$

Verpasste Umschreibung:

$q(r, s, k) :- v(r, s, k)$

BA kann v aus den Buckets für $film$ und $spielt$ nicht joinen, da das Attribut t (Filmtitel) nicht exportiert wird.

Zusammenfassung

- Nutzbarkeit von Views für eine globale Anfrage
- Bilden von Anfrage-Umschreibungen
- Vorauswahl von Views: Einordnen in Buckets
- Prüfung aller View-Kombinationen
  - Konsistenz Prädikate
  - Containment Check der Kombinationen

Globale Anfrage

```
SELECT prof
FROM Lehrt L, Kurs K
WHERE L.kurs_id = K.kurs_id
AND K.titel = „VL_Datenbanken“
AND L.univ = „Humboldt“
```

Umgeschriebene Anfrage

```
SELECT prof
FROM DB-kurs D
WHERE D.univ = „Humboldt“
```

Vollständige Antwort

Globale Anfrage

```
SELECT titel, kurs_id
FROM Kurs K
WHERE L.univ = „Humboldt“
```

Umgeschriebene Anfrage

```
SELECT titel, kurs_id
FROM DB-kurs D
WHERE D.univ = „Humboldt“
UNION
SELECT titel, kurs_id
FROM Hum-VL
```

Maximale Antwort

Literatur

- Gute Zusammenfassung für LaV und weiterführende Literatur:
  - [Hal01] Alon Y. Halevy: Answering queries using views: A survey, in VLDB Journal 10: 270-294, 2001.
- Erweiterung des BA für Information Quality
  - [NLF99] Felix Naumann, Ulf Leser, and Johann-Christoph Freytag: Quality-driven Integration of Heterogenous Information Systems, VLDB 1999
  - [LN00] Ulf Leser and Felix Naumann: Query Planning with Information Quality Bounds, FQAS 2000.
- Weitere Literatur
  - [LRO96b] Alon Y. Levy , Anand Rajaraman , Joann J. Ordille , Querying Heterogeneous Information Sources Using Source Descriptions Proceedings of the 22nd VLDB Conference, Bombay, India. 1996.
  - [LRO96a] Alon Y. Levy , Anand Rajaraman , Joann J. Ordille , Query answering algorithms for information agents Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence 1996 Click here for abstract.
  - [LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, Divesh Srivastava: Answering Queries Using Views. PODS 1995: 95-104
  - [LN06] Ulf Leser, Felix Naumann, Informationsintegration, 2006.