Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Informationsintegration

Der Bucket-Algorithmus

12.6.2007

Felix Naumann

Überblick

- Nutzbarkeit und Nützlichkeit von Views
- Bucket Algorithmus am Beispiel
- Bucket Algorithmus en detail
- (Pruning im Bucket Algorithmus)

Anfrageumschreibung

3

- 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).
  - Geschickte Vorauswahl der Sichten: Nutzbarkeit

LaV – Nutzen der Sichten

4

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

Beispiel – Globale Anfrage

5

- Ausschnitt Globales Schema
  - **Lehrt(prof,kurs_id, sem, eval, univ) L**
  - **Eingeschrieben(stud, kurs_id, sem) E**
  - **Betreut(prof, stud) B**
- Anfrage
  - **SELECT B.prof, B.stud, E.sem**
    FROM Eingeschrieben E, Lehrt L, Betreut B
    WHERE E.kurs_id = L.kurs_id
    AND E.sem = L.sem
    AND B.prof = L.prof
    AND B.stud = E.stud
    AND E.sem ≥ „WS98“
- In Worten
  - Finde Professoren mit betreuten Studenten und Semester, die seit WS98 auch einen Kurs dieses Professors besucht haben.

LaV – Beispiel

6

Globale Anfrage
SELECT B.prof, B.stud, E.sem
FROM Eingeschrieben E, Lehrt L, Betreut B
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
AND E.sem ≥ „WS98“

Quelle 7:
CREATE VIEW StudProf AS
SELECT L.prof, E.stud, E.sem
FROM Eingeschrieben E, Lehrt L
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND E.sem ≥ „WS98“

Frage:
• nutzbar?
• nützlich?

Umschreibung
SELECT B.prof, B.stud, S.sem
FROM StudProf S, Betreut B
WHERE B.prof = S.prof
AND B.stud = S.stud

Vorsicht: Umschreibung nutzt noch lokale Relation B!
Aber Annahme: Triviale Sicht

LaV – Beispiel

7

Globale Anfrage

```
SELECT B.prof, B.stud, E.sem
FROM Eingeschrieben E, Lehrt L, Betreut B
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
AND E.sem ≥ „WS98“
```

Frage:
- nutzbar?
- nützlich?

Quelle 8:

```
CREATE VIEW StudProf2 AS
SELECT L.prof, E.stud, E.sem
FROM Eingeschrieben E, Lehrt L
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND E.sem ≥ „WS97“
```

Schwächere Bedingung

Umschreibung

```
SELECT B.prof, B.stud, S.sem
FROM StudProf2 S, Betreut B
WHERE B.prof = S.prof
AND B.stud = S.stud
AND S.sem ≥ „WS98“
```

Umschreibung ist contained und äquivalent.

LaV – Beispiel

8

Globale Anfrage

```
SELECT B.prof, B.stud, E.sem
FROM Eingeschrieben E, Lehrt L, Betreut B
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
AND E.sem ≥ „WS98“
```

Frage:
- nutzbar?
- nützlich?

Quelle 9:

```
CREATE VIEW StudProf3 AS
SELECT L.prof, E.stud, E.sem
FROM Eingeschrieben E, Lehrt L
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND E.sem ≥ „WS99“
```

Stärkere Bedingung

Umschreibung

```
SELECT B.prof, B.stud, S.sem
FROM StudProf3 S, Betreut B
WHERE B.prof = S.prof
AND B.stud = S.stud
```

Umschreibung ist contained aber nicht äquivalent
⇒ OK für Datenintegration, aber ungünstig für MVs

LaV – Beispiel

9

Globale Anfrage

```
SELECT B.prof, B.stud, E.sem
FROM Eingeschrieben E, Lehrt L, Betreut B
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
AND E.sem ≥ „WS98“
```

Frage:
- nutzbar?
- nützlich?

L.prof fehlt

Quelle 10:

```
CREATE VIEW StudProf4 AS
SELECT E.stud, E.sem
FROM Eingeschrieben E, Lehrt L
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND E.sem ≥ „WS98“
```

Umschreibung

```
SELECT B.prof, B.stud, S.sem
FROM StudProf4 S, Betreut B, Lehrt L
WHERE B.prof = L.prof
AND S.stud = L.stud
AND B.stud = S.stud
```

Mühselig: L muss erneut gejoint werden

LaV – Beispiel

10

Globale Anfrage

```
SELECT B.prof, B.stud, E.sem
FROM Eingeschrieben E, Lehrt L, Betreut B
WHERE E.kurs_id = L.kurs_id
AND E.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
AND E.sem ≥ „WS98“
```

Frage:
- nutzbar?
- nützlich?

Quelle 11:

```
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“
```

E.sem = L.sem fehlt

Umschreibung

```
SELECT B.prof, B.stud, E.sem
FROM StudProf5 S, Eingeschrieben E,
     Lehrt L, Betreut B
WHERE S.sem = L.sem
AND B.prof = L.prof
AND B.stud = E.stud
```

Mühselig: L und E müssen erneut gejoint werden.
L, weil L.sem von Quelle 11 nicht exportiert wird.

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

- Nutzbarkeit und Nützlichkeit von Views
- Bucket Algorithmus am Beispiel
- Bucket Algorithmus en detail
- (Pruning im Bucket Algorithmus)

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 (Eimer).
  - 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

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 ≥ 500
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 3:

```
CREATE VIEW V3 AS
SELECT E.stud, E.kurs_id
FROM   Eingeschrieben E
WHERE E.sem ≤  WS94
```

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
```

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

15

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:

```
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
```

Anfrage Q:

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

16

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

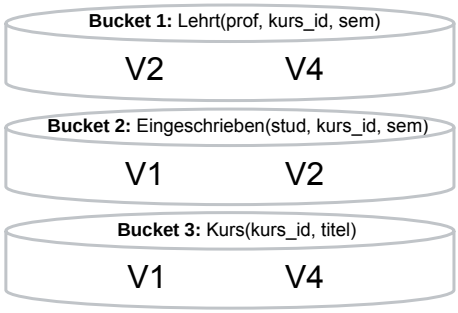
V2 (stud, prof, sem, kurs_id) :- E(stud, kurs_id, sem), L(prof, kurs_id, sem)

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:

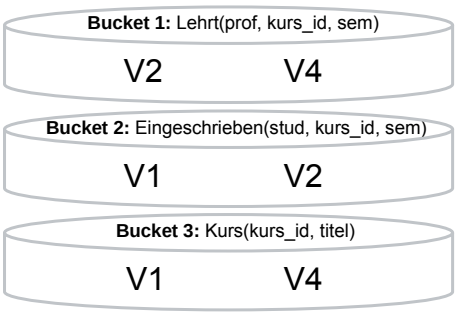
Q (stud, kurs_id, prof) :-
L(prof, kurs_id, sem),
E(stud,kurs_id,sem),
K(kurs_id, titel),
sem ≥ WS95,
kurs_id ≥ 300



LaV – BA – Beispiel

17

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

18

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

19

Ergebnis des Algorithmus: V1,V2 ∪ V2,V4

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
```

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‘:

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

UNION ...
```


LaV – BA – Beispiel

20

Ergebnis des Algorithmus: V1,V2 ∪ V2,V4

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‘:

```
...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
```

LaV – BA – Beispiel

21

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‘:

```
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
```

Jetzt: BA genauer

- Q‘ verwendet ausschließlich Sichten.
- Q‘ ⊆ Q
- Q‘ ist maximal contained in Q.

Überblick

22

- Nutzbarkeit und Nützlichkeit von Views
- Bucket Algorithmus am Beispiel
- Bucket Algorithmus en detail
- (Pruning im Bucket Algorithmus)

BA – en detail

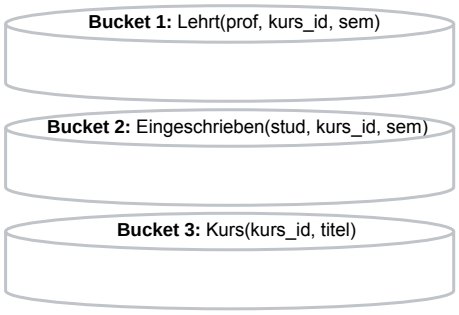
23

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*.



BA – en detail

24

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 sind passend.

BA – en detail

25

```
procedure generate-relevant-sources(I,Q)
/* I is a set of information sources, and Q is a conjunctive
query of the form q(X̄) : (∃X̄)g1(X̄1) ∧ ... gm(X̄m) ∧ Cq,
where Cq is the conjunction of order atoms in Q. */

for every subgoal gi(X̄i), 1 ≤ i ≤ m do:
  relevantSourcesi = ∅
  for every non-order conjunct u(Ȳ) in a formula
    rv in the pair (v, rv) in a description of a source in
    I do:
    if gi = u or gi and u are non-disjoint classes then:
      Let ψ be the mapping defined on the variables of
      rv as follows:
        if Y is the j'th variable in Ȳ and is not
          existentially quantified in rv
          then ψ(Y) = Xj, where Xj is the j'th
          variable in X̄i.
          else ψ(Y) is a new variable that does not
          appear in Q or rv.
      Let C(Q) and C(v) be the conjunction of
        constraint subgoals in Q and ψ(rv), respectively.
      if C(Q) ∧ C(v) is satisfiable, then add ψ(rv)
        to relevantSourcesi.
return {relevantSources1, ..., relevantSourcesm}.
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

Aus [LRO96a]

BA – en detail Beispiel

26

Für jede Sicht: Betrachte deren Teilziele.

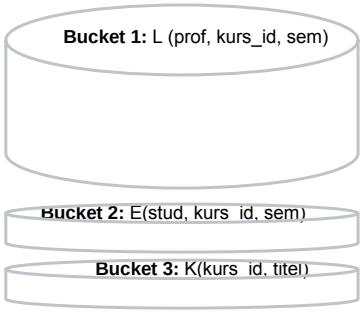
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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 Beispiel

27

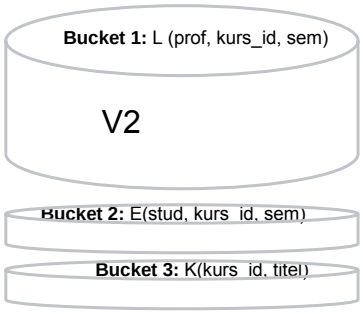
Für jede Sicht: Betrachte deren Teilziele.

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 Beispiel

28

Für jede Sicht: Betrachte deren Teilziele.

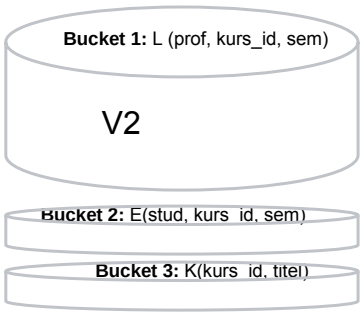
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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!



BA – en detail Beispiel

29

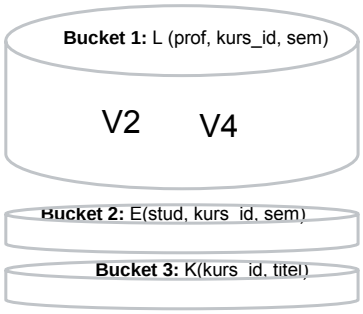
Für jede Sicht: Betrachte deren Teilziele.

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 Beispiel

30

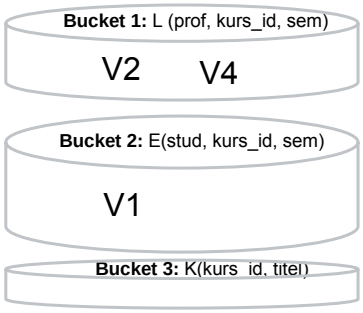
Für jede Sicht: Betrachte deren Teilziele.

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 Beispiel

31

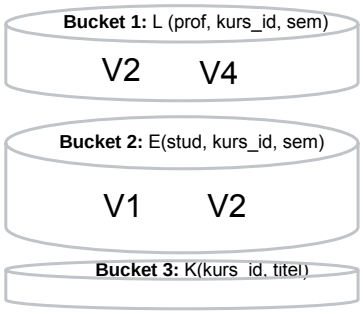
Für jede Sicht: Betrachte deren Teilziele.

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 Beispiel

32

Für jede Sicht: Betrachte deren Teilziele.

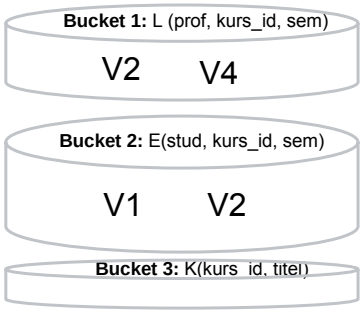
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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 Beispiel

33

Für jede Sicht: Betrachte deren Teilziele.

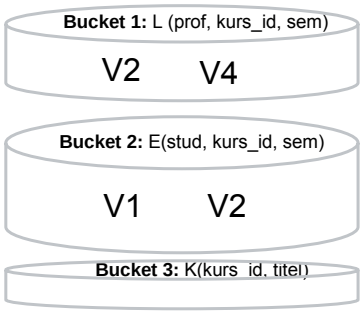
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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 Beispiel

34

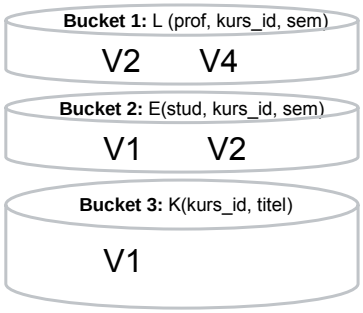
Für jede Sicht: Betrachte deren Teilziele.

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 Beispiel

35

Für jede Sicht: Betrachte deren Teilziele.

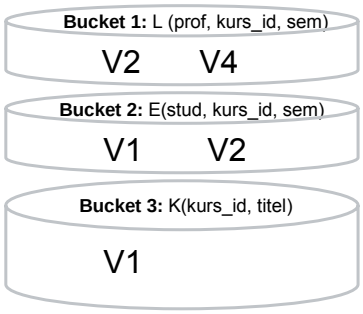
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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!



BA – en detail Beispiel

36

Für jede Sicht: Betrachte deren Teilziele.

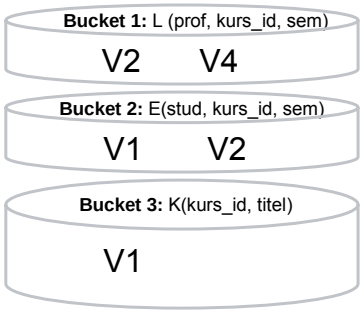
Falls mindestens ein Teilziel geeignet ist, füge Sicht in bucket ein.

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 Beispiel

37

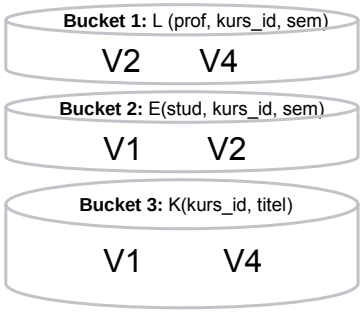
Für jede Sicht: Betrachte deren Teilziele.

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



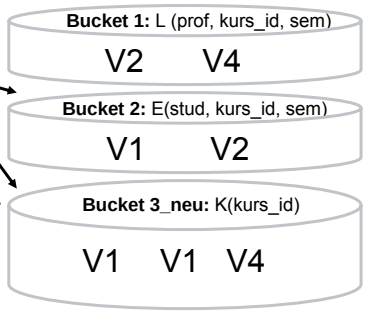
BA – Besonderheiten

38

- 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

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



LaV – BA – Beispiel

39

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

Bucket 1: Lehrt(prof, kurs_id, sem)

V2 V4

Bucket 2: Eingeschrieben(stud, kurs_id, sem)

V1 V2

Bucket 3: Kurs(kurs_id, titel)

V1 V4

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

LaV – BA – Beispiel

40

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, sem) :- 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 zulässig?

LaV – BA – Kombinationen

41

- Wieviele Kombinationen?
  - $|B_1| \times |B_2| \times ... \times |B_n|$ ($n = |Q|$)
  - Falls m = Anzahl Sichten: m^n
  - Wichtig: Jede der exponentiell vielen Kombinationen liefert potentiell einen Teil des Ergebnisses.
- Eine Kombination Q` = V1,...,Vk ist eine Anfrageumschreibung von Q, falls
  - $Q` \subseteq Q$
  - (oder Q`, {Prädikate} $\subseteq$ Q)

LaV – BA – Kombinationen

42

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)~~
- Durch „unfolding“:
  Q‘ = Q‘‘(stud, kurs_id, prof):-
  E(stud‘, kurs_id, sem), L(prof, kurs_id, sem),
  E(stud,kurs_id,sem), K(kurs_id,titel‘), kurs_id ≥ 500, sem ≥ WS98
- 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
```

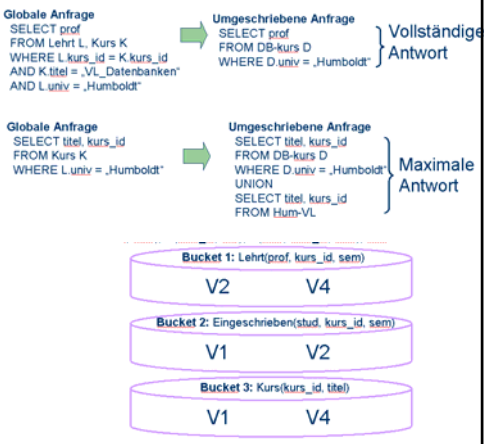

LaV – BA – Analyse

- Vollständigkeit: Gegeben Anfrage Q und Sichten V1,...,Vm, findet BA eine Anfrageumschreibung Q' aus den Sichten, falls eine Umschreibung existiert?
- Beweis:
  - Nach [LMSS95] gibt es nur eine Umschreibung, falls es eine Umschreibung der Länge |Q| gibt.
    - Einschränkung: Die Anfrage darf keine ≥,≤,<,>,≠ Prädikate enthalten.
  - D.h. es reicht, alle Kombinationen zu prüfen.
  - BA verwirft nur Kombinationen, die unzulässig sind.
- Maximally contained
  - BA produziert nach [LMSS95] maximal contained Umschreibungen.

Zusammenfassung bisher

47

- Query Rewriting
- Query Containment
- Naiver Algorithmus
- Bucket Algorithmus
  - Beispiel
  - Analyse



Überblick

48

- Nutzbarkeit und Nützlichkeit von Views
- Bucket Algorithmus am Beispiel
- Bucket Algorithmus en detail
- (Pruning im Bucket Algorithmus)

Problems with the BA

49

Expensive

- Planning: Exponential number of exponential tests
- Execution: Potentially exponential number of conjunctive plans

Question: Who needs all plans ?

Nobody wants all plans !

50

- Avoid sources that
  - have outdated data
  - provide only little data
  - suffer from low reputation
  - ...
- Avoid plans that
  - are combinations of almost disjoint sources
  - offer nothing new
  - take a long time
  - ...

Nobody wants all plans !

51

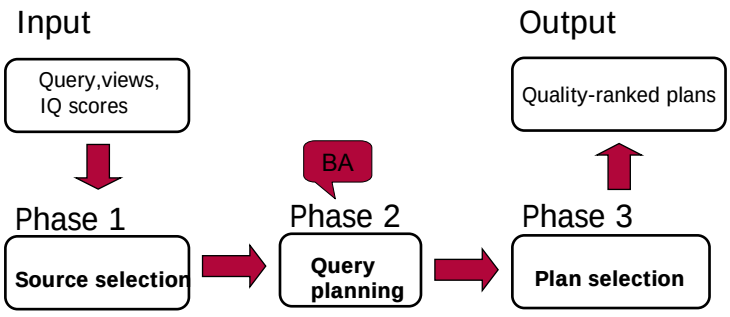
- Avoid sources that
  - have outdated data
  - take a long time
  - ...

Goal: execute only the best *N* plans

But keep in mind:
quality of a source/plan depends on the query

Naive (3-phase) approach [NLF99]

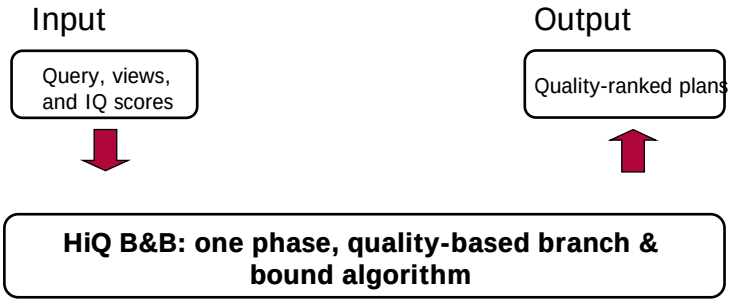
52



Good: executes only best plans

Bad: still needs to compute all plans

Integrated (single-phase) approach [LN00]



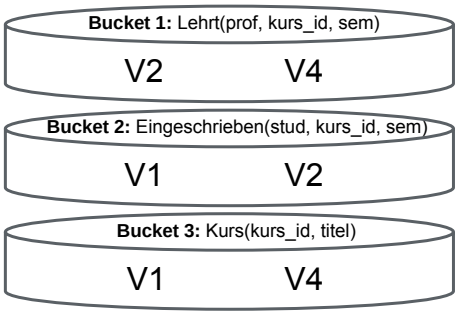
Good: executes only best plans

Good: computes only a fraction of all plans

Idea of HiQ B&B Algorithm

- Incremental plan construction
- Compute IQ scores for subplans as for plans.
- Compute upper IQ bound for subplan s.
  - Maximal score any plan containing s can reach
- Prune subplans…
  - …if partial containment mappings are not compatible. (Old criterion)
  - …if their upper bound is lower than the N 'th best result. (New criterion)

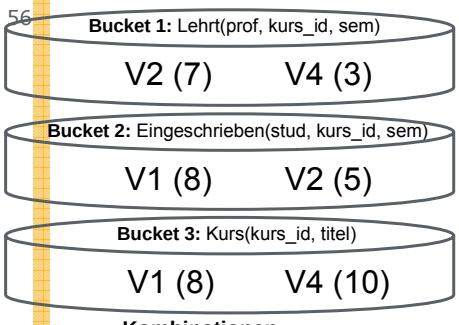
WdH: Bucket Algorithm



Kombinationen

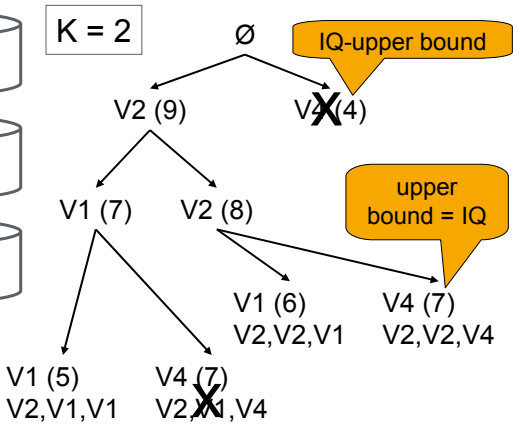
V2, V1, V1 → V1,V2
V4, V1, V4 → Ø
V2, V2, V4 → V2,V4
V2, V2, V1 → V1,V2
V2, V1, V4 → Ø
V4, V1, V1 → Ø
V4, V2, V4 → V2,V4
V4, V2, V1 → Ø

HiQ B&B: Extended Bucket Algorithm



Kombinationen

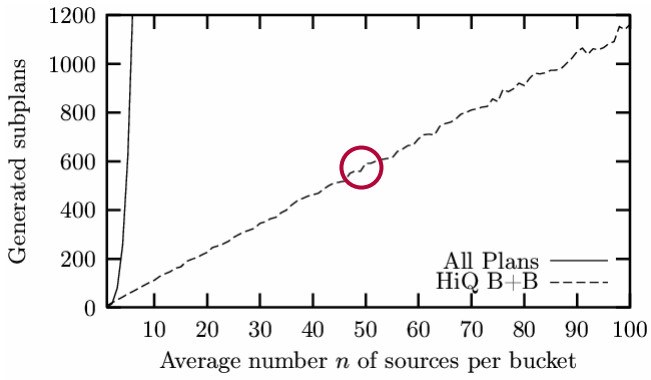
V2, V1, V1 → V1,V2
V4, V1, V4 → Ø
V2, V2, V4 → V2,V4
V2, V2, V1 → V1,V2
V2, V1, V4 → Ø
V4, V1, V1 → Ø
V4, V2, V4 → V2,V4
V4, V2, V1 → Ø



Issues to take care of

- Why not greedy ?
  - Not all plan candidates are correct plans
- No pruning before N plans are found
  - Find complete plans early → intelligent branching
- Upper bound for subplans
  - The tighter, the better
  - Must be efficiently computable

Hi-Q Branch & Bound



Instead of 50^4 = 6.250.000 only 550 Sub-Plans

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