

## Vorlesung Wissensentdeckung in Datenbanken Data Cube

Katharina Morik, Claus Weihs

Informatik LS 8  
Computergestützte Statistik  
Technische Universität Dortmund

21.04.2015

## Gliederung

- 1 Einführung
- 2 Aggregation in SQL, GROUP BY
- 3 Probleme mit GROUP BY
- 4 Der Cube-Operator
- 5 Implementierung des Data Cube
- 6 Zusammenfassung

## Datenbanksysteme

- Entwurf, z.B. mit Entity Relationship Model
- Deklaration
- Speichern der Daten
  - Hauptspeicher, Cache, virtueller Speicher, Platte
  - Indexierung, z.B. mit B-Bäumen
- Anfragen
  - Syntax in SQL, Semantik
  - Pläne zur Ausführung
  - Optimierung
- Änderungen (Transaktionen)

## Relationen

| Titel         | Jahr | Dauer | Foto   |
|---------------|------|-------|--------|
| Star Wars     | 1977 | 124   | farbig |
| Mighty Ducks  | 1991 | 104   | farbig |
| Wayne's World | 1992 | 95    | farbig |

- Attribute (Spaltennamen): X=Titel, Jahr, Dauer, Foto
- Tupel (Zeilen):  $t : X \rightarrow C$ ;  $C$  ist die unendliche Menge der Konstanten
- Relation  $r$ : Menge von Tupeln mit gleichem Definitionsbereich

## Schema

| Titel         | Jahr | Dauer | Foto   |
|---------------|------|-------|--------|
| Star Wars     | 1977 | 124   | farbig |
| Mighty Ducks  | 1991 | 104   | farbig |
| Wayne's World | 1992 | 95    | farbig |

- Name der Relation und Menge von Attributen mit Wertebereichen, hier: Filme(Titel:string, Jahr:integer, Dauer:integer, Foto:{farbig, sw})
- Wertebereiche sind einfache Datentypen: integer, string, date, Aufzählung

## Ein Schema erzeugen – DDL

Data Definition Language, Teilmenge von SQL

CREATE TABLE FilmeTest  
(titel VARCHAR2(18), jahr NUMBER);  
DESCRIBE FilmeTest liefert:

| Name  | Type         |
|-------|--------------|
| Titel | VARCHAR2(18) |
| Jahr  | NUMBER       |

**Instanz eines Schemas**

- Relationenschema der Form  $R(A_1 : D_1, \dots, A_p : D_p)$
- $(d, r)$  ist eine Instanz von  $R(A_1 : D_1, \dots, A_p : D_p)$  gdw.
  - $d \subset C$ , alle konkreten Werte sind Teilmengen der unendlichen Menge an Konstanten
  - $dom(r) = \{A_1, \dots, A_p\}$ , die Instanz hat die selben Attribute
  - $t(A_i) \in D_i$ , die Attributwerte der Instanz entsprechen dem Wertebereich des jeweiligen Attributs

Eine Instanz heißt auch *Zustand*. Das Schema ändert sich (fast) nie, der Zustand sehr häufig.

**Eine Instanz erzeugen – DML**

Zeilenweise werden Tupel eingefügt.

- INSERT INTO FilmeTest (Titel, Jahr, Star) VALUES ('Star Wars', 1988, 'C. Fisher');

Wenn man die Daten aus einer anderen Tabelle bekommen kann, darf statt VALUES ein SELECT-Ausdruck stehen, der Tupel aus der gegebenen Tabelle holt.

**Datenbank und Datenbankschema**

- Eine Menge von Relationenschemata ist ein Datenbankschema. *Es gibt nicht nur EINE Tabelle!*
- Eine Menge von Instanzen von Relationenschemata ist ein Datenbankzustand (kurz: Datenbank). *Also lauter gefüllte Tabellen.*
- Metadaten beschreiben Daten. Ein Datenbankschema beschreibt eine Datenbank.

**Datenbankanfragen**

**Algebra** Aus Operationen und Variablen oder Konstanten werden Ausdrücke gebildet, z.B. (Un-)Gleichungen.

**Relationenalgebra** Operationen bilden neue Relationen aus gegebenen Relationen.

- Variable und Konstante bezeichnen Relationen.
- Operationen sind
  - Vereinigung, Schnittmenge, Differenz
  - Projektion
  - Selektion
  - Kartesisches Produkt, natürlicher Verbund
- Die Ausdrücke heißen *Anfragen*.

**Projektion**

- Eine Relation auf eine andere projizieren.  $\pi A_1, \dots, A_m(r)$  liefert nur die Attribute  $A_1, \dots, A_m$  von  $r$ . Beispiel:  $\pi_{\text{Titel, Jahr}}(\text{Filme})$  liefert gerade  $\text{FilmeTest}$ .
- Eine Relation gemäß Bedingungen projizieren.  $\pi_q(r)$ , wobei  $q$  definiert werden muss. Beispiel:  $\pi_q(\text{Filme})$  mit  $q(\text{Foto}) := \text{Farbe}$  benennt das Attribut um;  $\pi_q(\text{Filme})$  mit  $q := \text{farbig}$  ergibt die Relation  $\frac{\text{farbig}}{\text{true}}$ . Es gibt nur ein Tupel, denn alle Filme waren farbig.

**Selektion**

Die Selektion  $\sigma_{\text{Cond}}(r)$  aus einer Relation  $r$  ergibt eine Relation mit der Teilmenge von Tupeln von  $r$ , die der Bedingung  $\text{Cond}$  genügen.  $\text{Cond}$  besteht aus Gleichheit und Vergleichsoperationen.

Die Ergebnisrelation hat dasselbe Schema wie  $r$ .  $\sigma_{\text{Dauer}}(\text{Filme}) \geq 120(\text{Filme})$  liefert:

| Titel     | Jahr | Dauer | Foto   |
|-----------|------|-------|--------|
| Star Wars | 1977 | 124   | farbig |

- $\pi L(\sigma \text{Cond}(r))$
- SELECT L from r WHERE Cond;
- L ist eine Liste von Attributen
- r ist eine Relation
- Cond ist eine Bedingung
- SELECT Titel from Filme WHERE Jahr(Filme)=1977;  
Titel  
Star Wars

Ziel: Auffinden interessanter Muster in großen Datenmengen

- Formulierung einer Anfrage
- Extraktion der Daten
- Visualisierung der Ergebnisse
- Analyse der Ergebnisse und Formulierung einer neuen Anfrage

- Datenmenge wird als  $p$ -dimensionaler Raum aufgefasst
- Identifizierung von „interessanten“ Unterräumen
- In relationalen Datenbanken werden  $p$ -dimensionale Daten als Relationen mit  $p$  Attributen modelliert
- Dimensionsreduktion durch Aggregation der Daten entlang der weggelassenen Dimensionen

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Opel   | 1990 | rot   | 5      |
| Opel   | 1990 | weiß  | 87     |
| Opel   | 1990 | blau  | 62     |
| Opel   | 1991 | rot   | 54     |
| Opel   | 1991 | weiß  | 95     |
| Opel   | 1991 | blau  | 49     |
| Opel   | 1992 | rot   | 31     |
| Opel   | 1992 | weiß  | 54     |
| Opel   | 1992 | blau  | 71     |
| Ford   | 1990 | rot   | 64     |
| Ford   | 1990 | weiß  | 62     |
| Ford   | 1990 | blau  | 63     |
| Ford   | 1991 | rot   | 52     |
| Ford   | 1991 | weiß  | 9      |
| Ford   | 1991 | blau  | 55     |
| Ford   | 1992 | rot   | 27     |
| Ford   | 1992 | weiß  | 62     |
| Ford   | 1992 | blau  | 39     |

- **Aggregatfunktionen:**  
COUNT(), SUM(), MIN(), MAX(), AVG()

Beispiel: SELECT AVG(Anzahl) FROM Autoverkäufe

- Aggregation nur über verschiedene Werte

Beispiel: SELECT COUNT(DISTINCT Modell) FROM Autoverkäufe

- Aggregatfunktionen liefern einen einzelnen Wert
- Aggregation über mehrere Attribute mit **GROUP BY**

**SELECT** Modell, Jahr, **SUM**(Anzahl)  
**FROM** Autoverkäufe  
**GROUP BY** Modell, Jahr

- Die Tabelle wird gemäß den Kombinationen der ausgewählten Attributmenge in Gruppen unterteilt
- Jede Gruppe wird über eine Funktion aggregiert
- Das Resultat ist eine Tabelle mit aggregierten Werten, indiziert durch die ausgewählte Attributmenge

## Beispiel: GROUP BY

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Opel   | 1990 | rot   | 5      |
| Opel   | 1990 | weiß  | 87     |
| Opel   | 1990 | blau  | 62     |
| Opel   | 1991 | rot   | 54     |
| Opel   | 1991 | weiß  | 95     |
| Opel   | 1991 | blau  | 49     |
| Opel   | 1992 | rot   | 31     |
| Opel   | 1992 | weiß  | 54     |
| Opel   | 1992 | blau  | 71     |
| Ford   | 1990 | rot   | 64     |
| Ford   | 1990 | weiß  | 62     |
| Ford   | 1990 | blau  | 63     |
| Ford   | 1991 | rot   | 52     |
| Ford   | 1991 | weiß  | 9      |
| Ford   | 1991 | blau  | 55     |
| Ford   | 1992 | rot   | 27     |
| Ford   | 1992 | weiß  | 62     |
| Ford   | 1992 | blau  | 39     |

```
SELECT Modell, Jahr, SUM(Anzahl)
FROM Autoverkäufe
GROUP BY Modell, Jahr
```

| Modell | Jahr | SUM(Anzahl) |
|--------|------|-------------|
| Opel   | 1990 | 154         |
| Opel   | 1991 | 198         |
| Opel   | 1992 | 156         |
| Ford   | 1990 | 189         |
| Ford   | 1991 | 116         |
| Ford   | 1992 | 128         |

19 von 40

## Roll Up

Gleiche Anfrage in unterschiedlichen Detailierungsgraden

- Verminderung des Detailierungsgrades = **Roll Up**
- Erhöhung des Detailierungsgrades = **Drill Down**

### Beispiel: Autoverkäufe

- Roll Up über drei Ebenen
- Daten werden nach Modell, dann nach Jahr, dann nach Farbe aggregiert
- Die Verkaufszahlen werden zuerst für jedes Modell aus jedem Jahr in jeder Farbe aufgelistet, dann werden alle Verkaufszahlen des gleichen Modells und Jahres aufsummiert und daraus die Verkaufszahlen der Modelle berechnet

20 von 40

## GROUP BY: Roll Up

| Modell | Jahr | Farbe | Anzahl nach Modell, Jahr, Farbe | Anzahl nach Modell, Jahr | Anzahl nach Modell |  |
|--------|------|-------|---------------------------------|--------------------------|--------------------|--|
| Opel   | 1990 | rot   | 5                               | 154                      |                    |  |
|        |      | weiß  | 87                              |                          |                    |  |
|        |      | blau  | 62                              |                          |                    |  |
| 1991   |      | rot   | 54                              | 198                      |                    |  |
|        |      | weiß  | 95                              |                          |                    |  |
|        |      | blau  | 49                              |                          |                    |  |
| 1992   |      | rot   | 31                              | 156                      |                    |  |
|        |      | weiß  | 54                              |                          |                    |  |
|        |      | blau  | 71                              |                          |                    |  |
|        |      |       | 156                             |                          | 508                |  |

21 von 40

## Probleme mit GROUP BY: Roll Up

- Tabelle ist nicht relational, da man wegen der leeren Felder (Null-Werte) keinen Schlüssel festlegen kann.
- Die Zahl der Spalten wächst mit der Zahl der aggregierten Attribute
- Um das exponentielle Anwachsen der Spaltenanzahl zu vermeiden, wird der ALL-Wert eingeführt.
- Der ALL-Wert repräsentiert die Menge, über die die Aggregation berechnet wird.

*Beispiel: Ein ALL in der Spalte Farbe bedeutet, dass in der Anzahl dieser Zeile die Verkaufszahlen der roten, weißen und blauen Autos zusammengefasst sind.*

22 von 40

## GROUP BY: Roll Up mit ALL

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Opel   | 1990 | rot   | 5      |
| Opel   | 1990 | weiß  | 87     |
| Opel   | 1990 | blau  | 62     |
| Opel   | 1990 | ALL   | 154    |
| Opel   | 1991 | rot   | 54     |
| Opel   | 1991 | weiß  | 95     |
| Opel   | 1991 | blau  | 49     |
| Opel   | 1991 | ALL   | 198    |
| Opel   | 1992 | rot   | 31     |
| Opel   | 1992 | weiß  | 54     |
| Opel   | 1992 | blau  | 71     |
| Opel   | 1992 | ALL   | 156    |
| Opel   | ALL  | ALL   | 506    |

Erzeugung der Tabelle mit SQL:

```
SELECT Modell, 'ALL', 'ALL', SUM(Anzahl)
FROM Autoverkäufe
WHERE Modell = 'Opel'
GROUP BY Modell
UNION
SELECT Modell, Jahr, 'ALL', SUM(Anzahl)
FROM Autoverkäufe
WHERE Modell = 'Opel'
GROUP BY Modell, Jahr
UNION
SELECT Modell, Jahr, Farbe, SUM(Anzahl)
FROM Autoverkäufe
WHERE Modell = 'Opel'
GROUP BY Modell, Jahr, Farbe
```

23 von 40

## Probleme mit GROUP BY: Roll Up

- Beispiel war ein einfaches dreidimensionales Roll Up
- Eine Aggregation über  $p$  Dimensionen erfordert  $p$  Unions
- Roll Up ist asymmetrisch:  
Verkäufe sind nach Jahr, aber nicht nach Farbe aggregiert

24 von 40

## Kreuztabellen

Symmetrische Darstellung mehrdimensionaler Daten und Aggregationen

| Opel        | 1990 | 1991 | 1992 | Total (ALL) |
|-------------|------|------|------|-------------|
| rot         | 5    | 54   | 31   | 90          |
| weiß        | 87   | 95   | 54   | 236         |
| blau        | 62   | 49   | 71   | 182         |
| Total (ALL) | 154  | 198  | 156  | 508         |

Diese Kreuztabelle ist eine zweidimensionale Aggregation  
Nimmt man noch andere Automodelle hinzu, kommt für jedes Modell eine weitere Ebene hinzu  
Man erhält eine dreidimensionale Aggregation

## Der CUBE-Operator

$p$ -dimensionale Generalisierung der bisher genannten Konzepte

Der 0D Data Cube ist ein Punkt

Der 1D Data Cube ist eine Linie mit einem Punkt

Der 2D Data Cube ist eine Kreuztabelle

Der 3D Data Cube ist ein Würfel mit drei sich überschneidenden Kreuztabellen

(Gray, Chaudhuri, Bosworth, Layman 1997)



## Der CUBE-Operator

- Beispiel: **SELECT Modell, Jahr, Farbe, SUM(Anzahl)**  
**FROM** Autoverkäufe  
**GROUP BY CUBE** Modell, Jahr, Farbe

(Der CUBE-Operator ist nicht für jede Datenbank direkt verfügbar)

- Der Cube-Operator erzeugt eine Tabelle, die sämtliche Aggregationen enthält
- Es werden GROUP BYs für alle möglichen Kombinationen der Attribute berechnet
- Die Erzeugung der Tabelle erfordert die Generierung der Potenzmenge der zu aggregierenden Spalten.
- Bei  $p$  Attributen werden  $2^p$  GROUP BYs berechnet
- Sei  $C_1, C_2, \dots, C_p$  die Kardinalität der  $p$  Attribute, dann ist die Kardinalität der resultierenden Data Cube-Relation  $\prod(C_i + 1)$

## Data Cube des Beispiels

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Opel   | 1990 | rot   | 5      |
| Opel   | 1990 | weiß  | 87     |
| Opel   | 1990 | blau  | 62     |
| Opel   | 1991 | rot   | 54     |
| Opel   | 1991 | weiß  | 95     |
| Opel   | 1991 | blau  | 49     |
| Opel   | 1991 | ALL   | 154    |
| Opel   | 1992 | rot   | 31     |
| Opel   | 1992 | weiß  | 54     |
| Opel   | 1992 | blau  | 71     |
| Opel   | ALL  | ALL   | 156    |
| Ford   | 1990 | rot   | 64     |
| Ford   | 1990 | weiß  | 189    |
| Ford   | 1990 | blau  | 63     |
| Ford   | 1991 | rot   | 52     |
| Ford   | 1991 | weiß  | 9      |
| Ford   | 1991 | blau  | 55     |
| Ford   | 1991 | ALL   | 116    |
| Ford   | 1992 | rot   | 236    |
| Ford   | 1992 | weiß  | 182    |
| Ford   | 1992 | blau  | 104    |
| Ford   | ALL  | ALL   | 508    |
| ALL    | 1990 | rot   | 27     |
| ALL    | 1990 | weiß  | 62     |
| ALL    | 1990 | blau  | 39     |
| ALL    | 1991 | rot   | 128    |
| ALL    | 1991 | weiß  | 143    |
| ALL    | 1991 | blau  | 133    |
| ALL    | 1992 | rot   | 157    |
| ALL    | 1992 | weiß  | 433    |
| ALL    | 1992 | blau  | 104    |
| ALL    | ALL  | ALL   | 314    |
| ALL    | ALL  | rot   | 58     |
| ALL    | ALL  | weiß  | 116    |
| ALL    | ALL  | blau  | 110    |
| ALL    | ALL  | ALL   | 284    |
| ALL    | ALL  | rot   | 233    |
| ALL    | ALL  | weiß  | 369    |
| ALL    | ALL  | blau  | 339    |
| ALL    | ALL  | ALL   | 941    |

## Data Cube des Beispiel

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Opel   | 1990 | rot   | 5      |
| Opel   | 1990 | weiß  | 87     |
| Opel   | 1990 | blau  | 62     |
| Opel   | 1990 | ALL   | 154    |
| Opel   | 1991 | rot   | 54     |
| Opel   | 1991 | weiß  | 95     |
| Opel   | 1991 | blau  | 49     |
| Opel   | 1991 | ALL   | 198    |
| Opel   | 1992 | rot   | 31     |
| Opel   | 1992 | weiß  | 54     |
| Opel   | 1992 | blau  | 71     |
| Opel   | ALL  | ALL   | 156    |
| Opel   | ALL  | rot   | 90     |
| Opel   | ALL  | weiß  | 236    |
| Opel   | ALL  | blau  | 182    |
| Opel   | ALL  | ALL   | 508    |
| Ford   | 1990 | rot   | 64     |
| Ford   | 1990 | weiß  | 189    |
| Ford   | 1990 | blau  | 63     |
| Ford   | 1991 | rot   | 52     |
| Ford   | 1991 | weiß  | 9      |
| Ford   | 1991 | blau  | 55     |
| Ford   | 1991 | ALL   | 116    |

| Modell | Jahr | Farbe | Anzahl |
|--------|------|-------|--------|
| Ford   | 1992 | rot   | 27     |
| Ford   | 1992 | weiß  | 62     |
| Ford   | 1992 | blau  | 39     |
| Ford   | 1992 | ALL   | 128    |
| Ford   | ALL  | rot   | 143    |
| Ford   | ALL  | weiß  | 133    |
| Ford   | ALL  | blau  | 157    |
| Ford   | ALL  | ALL   | 433    |
| ALL    | 1990 | rot   | 69     |
| ALL    | 1990 | weiß  | 149    |
| ALL    | 1990 | blau  | 125    |
| ALL    | 1990 | ALL   | 343    |
| ALL    | 1991 | rot   | 106    |
| ALL    | 1991 | weiß  | 104    |
| ALL    | 1991 | blau  | 104    |
| ALL    | 1991 | ALL   | 314    |
| ALL    | 1992 | rot   | 58     |
| ALL    | 1992 | weiß  | 116    |
| ALL    | 1992 | blau  | 110    |
| ALL    | 1992 | ALL   | 284    |
| ALL    | ALL  | rot   | 233    |
| ALL    | ALL  | weiß  | 369    |
| ALL    | ALL  | blau  | 339    |
| ALL    | ALL  | ALL   | 941    |

## Implementationsalternativen

- Physische Materialisierung des gesamten Data Cube:
  - beste Antwortzeit
  - hoher Speicherplatzbedarf
- Keine Materialisierung:
  - jede Zelle wird nur bei Bedarf aus den Rohdaten berechnet
  - kein zusätzlicher Speicherplatz
  - schlechte Antwortzeit
- Materialisierung von Teilen des Data Cube:
  - Werte vieler Zellen sind aus Inhalt anderer Zellen berechenbar
  - diese Zellen nennt man „abhängige“ Zellen
  - Zellen, die einen ALL-Wert enthalten, sind abhängig
  - Problem: Welche Zellen des Data Cube materialisieren?
  - Zellen des Data Cube entsprechen SQL Anfragen (Sichten)

Die Abhängigkeitsrelation  $\leq$  zwischen zwei Anfragen  $Q_1$  und  $Q_2$

$Q_1 \leq Q_2$  gdw.  $Q_1$  kann beantwortet werden, indem die Ergebnisse von  $Q_2$  verwendet werden.  $Q_1$  ist abhängig von  $Q_2$

- Anfragen bilden einen Verband unter folgenden Voraussetzungen:
  - ①  $\leq$  ist eine Halbordnung und
  - ② es gibt ein maximales Element (eine oberste Sicht)
- Der Verband wird durch eine Menge von Anfragen (Sichten)  $L$  und der Abhängigkeitsrelation  $\leq$  definiert und mit  $\langle L, \leq \rangle$  bezeichnet
- Ein Verband wird dargestellt durch einen Graphen, in dem die Anfragen die Knoten sind und  $\leq$  die Kanten.

31 von 40

- Optimierungsproblem, das unter folgenden Bedingungen gelöst werden soll:
  - Die durchschnittliche Zeit für die Auswertung der Anfragen soll minimiert werden.
  - Man beschränkt sich auf eine feste Anzahl von Sichten, die materialisiert werden sollen, unabhängig von deren Platzbedarf
- Das Optimierungsproblem ist NP-vollständig.
- Heuristiken für Approximationslösungen:  
Greedy-Algorithmus
- Der Greedy-Algorithmus verhält sich nie zu schlecht: Man kann zeigen, dass die Güte mindestens 63% beträgt (Harinayaran, Rajaraman, Ullman 1996).

32 von 40

- Gegeben ein Verband mit Speicherkosten  $C(v)$  für jede Sicht  $v$
- Annahme: Speicherkosten = Anzahl der Reihen in der Sicht
- Beschränkung auf  $k$  materialisierte Sichten
- Nach Auswahl einer Menge  $S$  von Sichten wird der Nutzen der Sicht  $v$  relativ zu  $S$  mit  $B(v, S)$  bezeichnet und wie folgt definiert:
  - 1 Für jede Sicht  $w \leq v$  wird  $B_w$  berechnet:
    - 1 Sei  $u$  die Sicht mit den geringsten Kosten in  $S$ , so dass  $w \leq u$
    - 2  $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$
  - 2  $B(v, S) = \sum_{w \leq v} B_w$

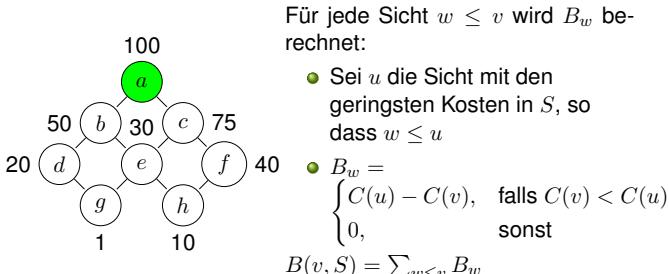
33 von 40

```

1  S = {oberste Sicht}
2  for i = 1 to k do begin
3      Wähle die Sicht v ∉ S, so dass B(v, S) maximal ist;
4      S = S ∪ {v}
5  end;
6  return S;

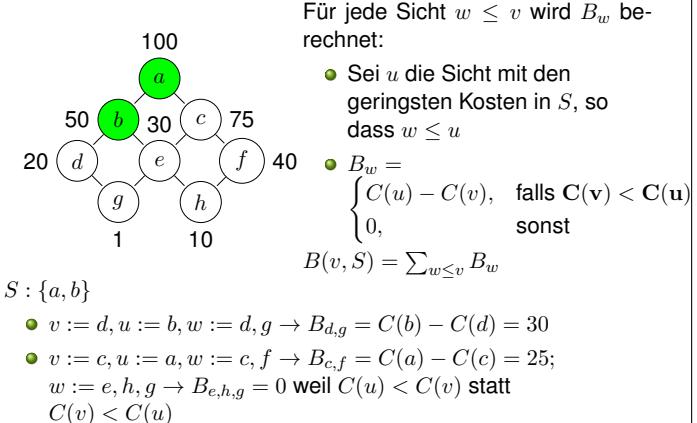
```

34 von 40



$S : \{a\}$   
 $v := b, u := a$   
Knoten  $w := b, d, g, e, h$  haben  $C(a) - C(b) = 50$ .  
Greedy Auswahl:  $b$  wird zusätzlich materialisiert

35 von 40



36 von 40

## Beispiel

Für jede Sicht  $w \leq v$  wird  $B_w$  berechnet:

- Sei  $u$  die Sicht mit den **geringsten Kosten** in  $S$ , so dass  $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

$$B(v, S) = \sum_{w \leq v} B_w$$

- $v := e, u := b, w := e, g, h \rightarrow B_{e,g,h} = C(b) - C(e) = 20$
- $v := f, u := a, w := f \rightarrow B_f = C(a) - C(f) = 60, u := b, w := h \rightarrow C(b) - C(f) = 10$

Gewählt wird  $f$ .

## Beispiel

Für jede Sicht  $w \leq v$  wird  $B_w$  berechnet:

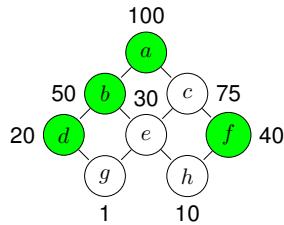
- Sei  $u$  die Sicht mit den **geringsten Kosten** in  $S$ , so dass  $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

$$B(v, S) = \sum_{w \leq v} B_w$$

$$S = \{a, b, f\}$$

- $v := c, u := a, w := c \rightarrow B_c = C(a) - C(c) = 25$
- $w := e, h, g \rightarrow B_{e,h,g} = 0$
- $v := d, u := b, w := d, g \rightarrow B_{d,g} = C(b) - C(d) = 30$
- $v := e, u := b, w := e, g \rightarrow B_{e,g} = C(b) - C(e) = 20$
- $u := f, w := h \rightarrow B_h = C(f) - C(e) = 10$

## Beispiel



|   | Erste Wahl          | Zweite Wahl        | Dritte Wahl         |
|---|---------------------|--------------------|---------------------|
| b | $50 \times 5 = 250$ | $25 \times 2 = 50$ | $25 \times 1 = 25$  |
| c | $25 \times 5 = 125$ | $30 \times 2 = 60$ | $30 \times 2 = 60$  |
| d | $80 \times 2 = 160$ | $20 \times 3 = 60$ | $20 + 20 + 10 = 50$ |
| e | $70 \times 3 = 210$ | $20 \times 3 = 60$ |                     |
| f | $60 \times 2 = 120$ | $60 + 10 = 70$     |                     |
| g | $99 \times 1 = 99$  | $49 \times 1 = 49$ | $49 \times 1 = 49$  |
| h | $90 \times 1 = 90$  | $40 \times 1 = 40$ | $30 \times 1 = 30$  |

$S : \{a\}, S : \{a, b\}, S : \{a, b, f\}, S : \{a, b, d, f\}$

Greedy Auswahl:  $b, d, f$  werden zusätzlich materialisiert

## Was wissen Sie jetzt?

- Eine Menge von Relationenschemata, die Namen von Relationen und Mengen ihrer Attribute mit Wertebereichen angeben, ist ein Datenbankschema. Eine Menge von Instanzen von Relationenschemata ist eine Datenbank.
- Möglichkeiten und Grenzen der Aggregation in SQL
- Einführung von Data Cubes zur Unterstützung von Aggregationen über  $p$  Dimensionen
- Greedy-Algorithmus zur Auswahl einer festen Anzahl von Sichten, die materialisiert werden