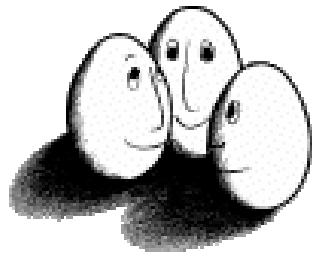


# Data Cubes

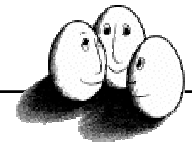
**PG 402 - Wissensmanagement  
Seminarphase**

**23.10.2001 - 25.10.2001**

Hanna Köpcke

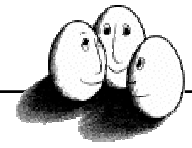


Lehrstuhl für Künstliche Intelligenz  
Universität Dortmund



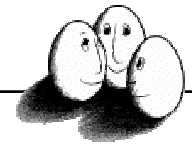
# Übersicht

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



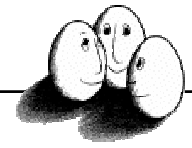
# 1. Datenanalyse

- Ziel: Auffinden interessanter Muster in großen Datenmengen
- Formulierung einer Anfrage
- Extraktion der Daten
- Visualisierung der Ergebnisse
- Analyse der Ergebnisse und Formulierung einer neuen Anfrage



# 1.1 Datenanalysetools

- Darstellung von Tendenzen, Cluster, Anomalien
- Datenmenge wird als n-dimensionaler Raum aufgefasst
- Identifizierung von „interessanten“ Unterräumen
- In relationalen Systemen werden n-dimensionale Daten als Relationen mit n-Attributen modelliert
- Dimensionsreduktion durch Aggregation der Daten entlang der weggelassenen Dimensionen



# 1.2 Beispiel: Autoverkäufe

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



## 2. Aggregation in SQL

- Aggregatfunktionen:

COUNT(), SUM(), MIN(), MAX(), AVG()

Beispiel: SELECT AVG(Anzahl)

FROM Autoverkäufe

- Aggregation über verschiedene Werte

Beispiel: SELECT COUNT(DISTINCT Modell)

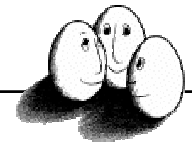
FROM Autoverkäufe

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



## 2.1 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



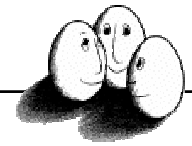
## 2.2 Beispiel: GROUP BY

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
```

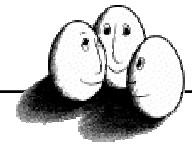
Modell	Jahr	Anzahl
Opel	1990	154
Opel	1991	198
Opel	1992	156
Ford	1990	189
Ford	1991	116
Ford	1992	128





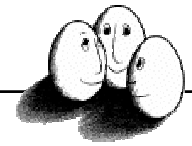
### 3. Probleme mit GROUP BY: Roll Up

- gleiche Anfrage in unterschiedlichen Detailierungsgraden
- Verminderung des Detailierungsgrades = Roll Up
- Erhöhung = 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 wiederum die Verkaufszahlen der Modelle berechnet



## 3.1 Probleme mit GROUP BY: Roll Up

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



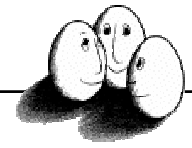
## 3.2 Probleme mit GROUP BY: Roll Up

- Tabelle ist nicht relational, da man wegen der leeren Felder (Null-Werte) keinen Schlüssel festlegen kann
- enorme Anzahl an Spalten:  
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.



## 3.3 Probleme mit GROUP BY: Roll Up

Modell	Jahr	Farbe	Anzahl	Erzeugung der Tabelle mit SQL:
Opel	1990	rot	5	<b>SELECT</b> Modell, <b>ALL</b> , <b>ALL</b> , <b>SUM</b> (Anzahl) <b>FROM</b> Autoverkäufe <b>WHERE</b> Modell = 'Opel' <b>GROUP BY</b> Modell <b>UNION</b>
Opel	1990	weiß	87	
Opel	1990	blau	62	
Opel	1990	ALL	154	
Opel	1991	rot	54	<b>SELECT</b> Modell, Jahr, <b>ALL</b> , <b>SUM</b> (Anzahl) <b>FROM</b> Autoverkäufe <b>WHERE</b> Modell = 'Opel' <b>GROUP BY</b> Modell, Jahr <b>UNION</b>
Opel	1991	weiß	95	
Opel	1991	blau	49	
Opel	1991	ALL	198	
Opel	1992	rot	31	<b>SELECT</b> Modell, Jahr, Farbe, <b>SUM</b> (Anzahl) <b>FROM</b> Autoverkäufe <b>WHERE</b> Modell = 'Opel' <b>GROUP BY</b> Modell, Jahr, Farbe
Opel	1992	weiß	54	
Opel	1992	blau	71	
Opel	1992	ALL	156	
Opel	ALL	ALL	506	



## 3.4 Probleme mit GROUP BY: Roll Up

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



## 3.5 Probleme mit GROUP BY: Kreuztabellen

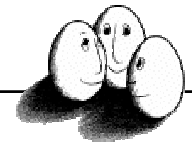
- Symmetrische Darstellung mehrdimensionaler Daten und Aggregationen

---

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

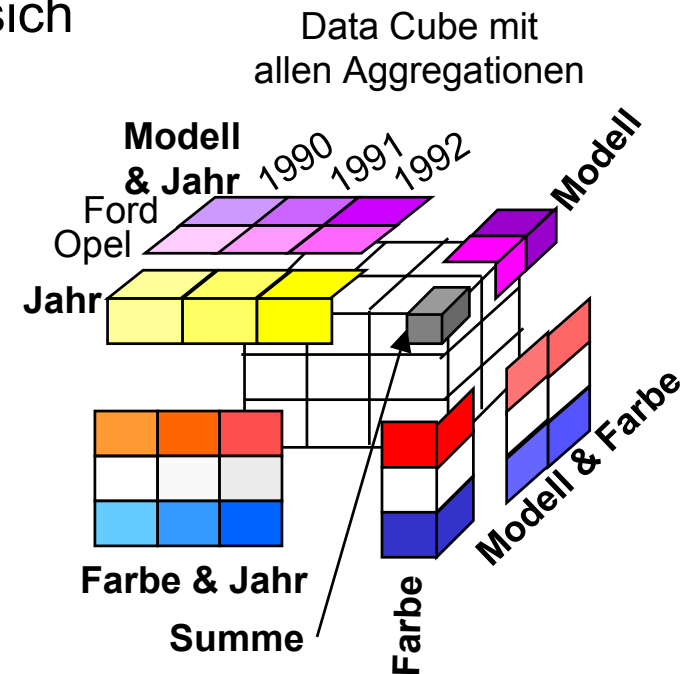
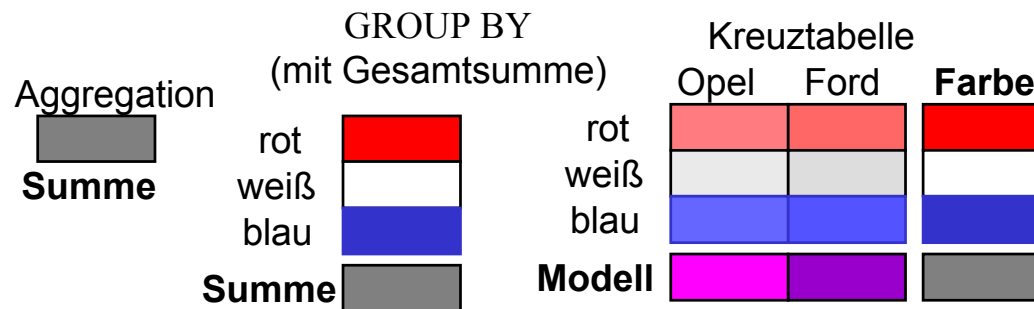
---

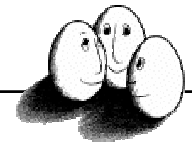
- Diese Kreuztabelle ist eine zweidimensionale Aggregation
- Nimmt man noch andere Automodelle hinzu, kommt für jedes Modell eine weitere Ebene hinzu
- Man erhält eine dreidimensionale Aggregation



# 4. Der CUBE-Operator

- N-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

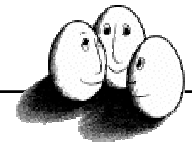




# 4.1 Der CUBE-Operator

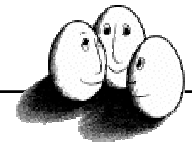
- Beispiel: **SELECT** Modell, Jahr, Farbe, **SUM**(Anzahl)  
**FROM** Autoverkäufe  
**GROUP BY CUBE** Modell, Jahr, Farbe
- Der Cube-Operator erzeugt eine Tabelle, die sämtliche Aggregationen enthält
- Es werden **GROUP BY**s für alle möglichen Kombinationen der Attribute berechnet
- Die Erzeugung der Tabelle erfordert die Generierung der Potenzmenge der zu aggregierenden Spalten.
- Bei  $n$  Attributen werden  $2^n$  **GROUP BY**s berechnet
- Sei  $C_1, C_2, \dots, C_n$  die Kardinalität der  $n$  Attribute, dann ist die Kardinalität der daraus resultierenden Data Cube-Relation  $J$   $(C_i + 1)$





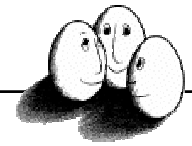
## 4.2 Data Cube des Beispiels

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	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



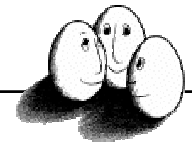
Modell	Jahr	Farbe	Anzahl
Opel	1990	rot	5
Opel	1990	weiß	87
Opel	1990	blau	62
Opel	1990	ALL	154
Opel	1991	rot	54
Opel	1991	weiß	95
Opel	1991	blau	49
Opel	1991	ALL	198
Opel	1992	rot	31
Opel	1992	weiß	54
Opel	1992	blau	71
Opel	1992	ALL	156
Opel	ALL	rot	90
Opel	ALL	weiß	236
Opel	ALL	blau	182
Opel	ALL	ALL	508
Ford	1990	rot	64
Ford	1990	weiß	72
Ford	1990	blau	63
Ford	1990	ALL	189
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



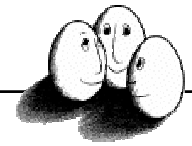
# 5. Implementationsalternativen

- 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)



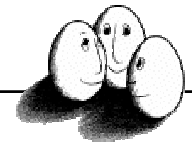
# 5.1 Abhängigkeit von Sichten

- Die Abhängigkeitsrelation:
  - Zwei Anfragen  $Q_1$  und  $Q_2$
  - $Q_1 \sim Q_2 \Leftrightarrow Q_1$  kann beantwortet werden, indem die Ergebnisse von  $Q_2$  verwendet werden.  $Q_1$  ist abhängig von  $Q_2$
- Sichten bilden einen Verband unter folgenden Voraussetzungen:
  1.  $\sim$  ist eine Halbordnung und
  2. es gibt ein maximales Element (eine oberste Sicht)
- Ein Verband wird durch eine Menge von Anfragen (Sichten)  $L$  und der Abhängigkeitsrelation  $\sim$  definiert und mit  $\mathcal{L}, \sim$ , bezeichnet
- Ein Verband wird repräsentiert durch einen Graphen, in dem die Anfragen die Knoten sind



## 5.2 Auswahl von Sichten

- 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
- Naheliegende Wahl: Ein Greedy-Algorithmus
- Der Greedy-Algorithmus verhält sich nie zu schlecht:  
Man kann zeigen, dass die Güte mindestens 63% beträgt.



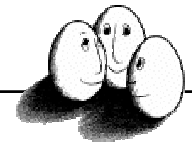
## 5.3 Der Greedy Algorithmus

- Gegeben ein Verband mit Speicherkosten für jede Sicht
- $C(v)$  Kosten der Sicht  $v$
- Annahme: Speicherkosten = Anzahl der Reihen in der Sicht
- Beschränkung auf  $k$  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  $w \sim v$  wird  $B_w$  berechnet:
    - (a) Sei  $u$  die Sicht mit den geringsten Kosten in  $S$ , so dass  $w \sim u$
    - (b) 
$$B_w = \begin{cases} C(v) - C(u), & \text{falls } C(v) < C(u) \\ 0 & \text{ansonsten} \end{cases}$$
  2.  $B(v, S) = \sum_{w \sim v} B_w$

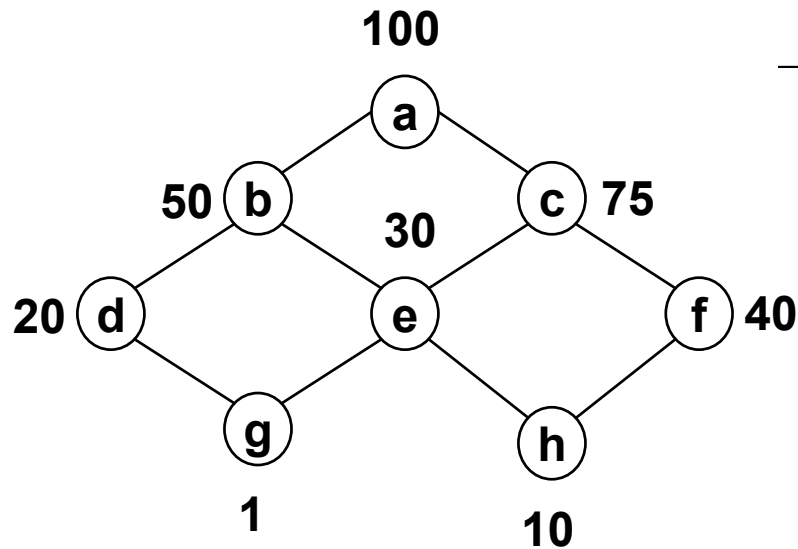


## 5.4 Der Greedy Algorithmus

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



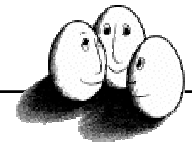
## 5.5 Beispiel



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

Greedy Auswahl: b, d und f





## 6. Zusammenfassung und Ausblick

- Möglichkeiten und Grenzen der Aggregation in SQL
- Einführung von Data Cubes zur Unterstützung von Aggregationen über  $n$  Dimensionen
- Implementationsalternativen von Data Cubes zur effizienten Anfragebearbeitung
- Greedy-Algorithmus zur Auswahl einer festen Anzahl von Sichten, die materialisiert werden
- Ausblick: weitere Algorithmen
  - Basierend auf Sortierung (PipeSort-Algorithmus)
  - Basierend auf Hashverfahren (PipeHash-Algorithmus)