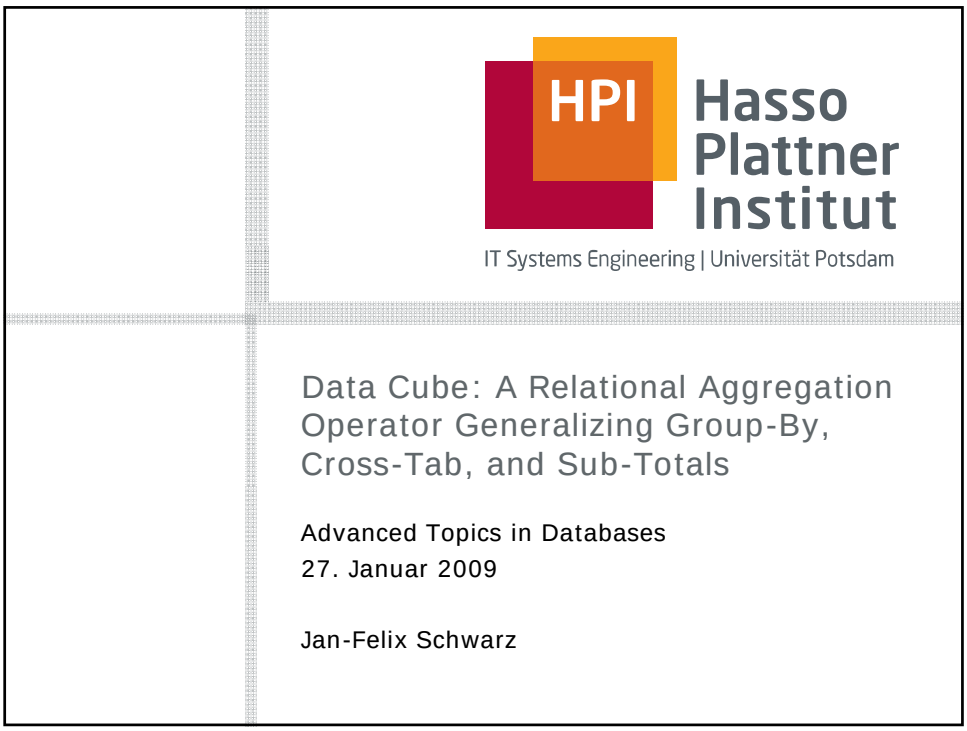
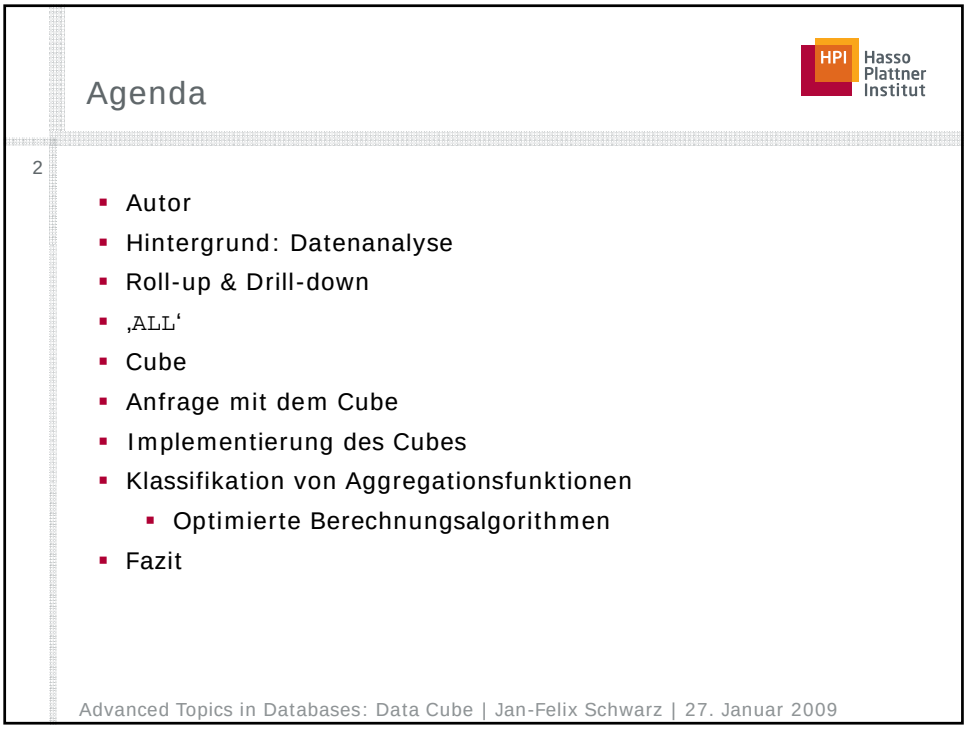
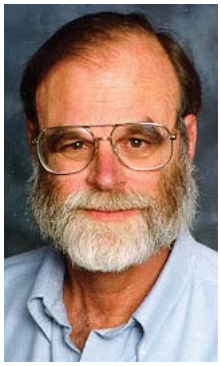

HPI
Hasso
Plattner
Institut
IT Systems Engineering | Universität Potsdam
Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals
Advanced Topics in Databases
27. Januar 2009
Jan-Felix Schwarz


Agenda
HPI
Hasso
Plattner
Institut
2
Autor
Hintergrund: Datenanalyse
Roll-up & Drill-down
‚ALL‘
Cube
Anfrage mit dem Cube
Implementierung des Cubes
Klassifikation von Aggregationsfunktionen
Optimierte Berechnungsalgorithmen
Fazit
Advanced Topics in Databases: Data Cube | Jan-Felix Schwarz | 27. Januar 2009

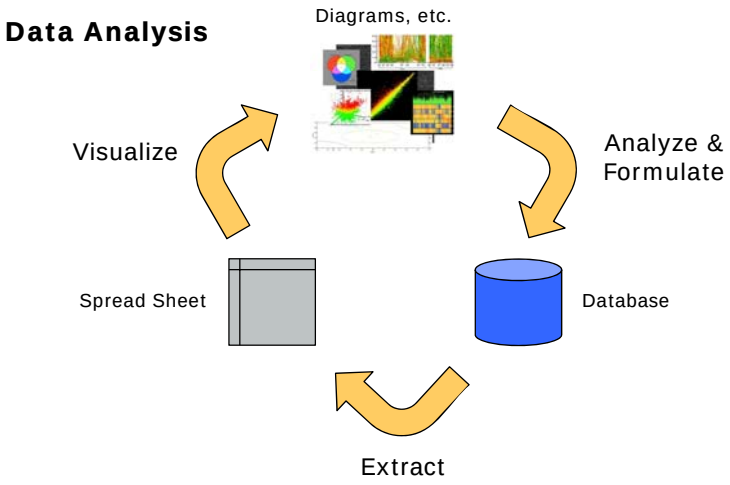
Jim Gray

- Geboren 1944
- Erster PhD in Computer Science an der UC Berkeley (1961)
- IBM Research in San Jose
  - System R, Transaktionen
- Tandem, DEC (80er / 90er Jahre)
  - Innovationen im Grid Computing
- Ab 1995 bei Microsoft Research



- Vermisst auf See seit dem 28. Januar 2007
- Blog zur Suchaktion:
  http://openphi.net/tenacious/

Hintergrund



Roll-up & Drill-down

5

Year	07	08
Sales	123	156

Drill-down

Roll-up

Month	Jan 07	Feb 07	...	Dez 07	Jan 08	...	Dez 08
Sales	12	8	...	16	13	...	19

Mehrdimensionaler Roll-up

6

Sales Roll Up by Model by Year by Color.

Model	Year	Color	Sales by Model by Year by Color
Chevy	1994	Black	50
		White	40
	1995	Black	85
		White	115

‚ALL‘

7

Model	Year	Color	Units
Chevy	1994	Black	50
Chevy	1994	White	40
Chevy	1994	ALL	90
Chevy	1995	Black	85
Chevy	1995	White	115
Chevy	1995	ALL	200
Chevy	ALL	ALL	290

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
   FROM    Sales
   WHERE   Model = 'Chevy'
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales)
   FROM    Sales
   WHERE   Model = 'Chevy'
   GROUP BY  Model
UNION
SELECT Model, Year, 'ALL', SUM(Sales)
   FROM    Sales
   WHERE   Model = 'Chevy'
   GROUP BY  Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
   FROM    Sales
   WHERE   Model = 'Chevy'
   GROUP BY  Model, Year, Color;
```


Roll-up ist asymmetrisch!

8

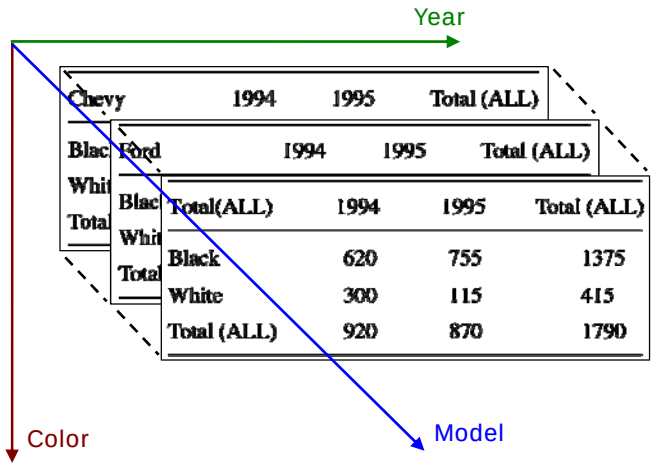
Fehlende Zeilen:

Model	Year	Color	Units
Chevy	ALL	Black	135
Chevy	ALL	White	155

```
UNION
SELECT Model, 'ALL', Color, SUM(Sales)
   FROM    Sales
   WHERE   Model = 'Chevy'
   GROUP BY  Model, Color;
```

Cube

9



Anfrage mit dem Cube Operator

10

Sales

Model	Color	Time	Price
Chevy	Red	2008-06-02	18 000
Chevy	Red	2009-01-03	39 000
Chevy	Black	2008-10-10	23 000
Chevy	Black	2007-08-11	21 000
Ford	Blue	2008-03-23	13 000
Ford	Blue	2007-11-04	25 000
Ford	Black	2008-12-14	12 000
Ford	Black	2009-01-10	10 000
Ford	White	2007-12-22	14 000

```
SELECT  Model, Color, Year, COUNT(*) AS Sales
FROM    Sales
WHERE   Year >= 2008
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```

Ergebnis

Data Cube

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	ALL	2008	2
Chevy	ALL	2009	1
Chevy	ALL	ALL	3
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

...

...

Model	Color	Year	Sales
Ford	ALL	2008	2
Ford	ALL	2009	1
Ford	ALL	ALL	3
ALL	Red	2008	1
ALL	Red	2009	1
ALL	Red	ALL	2
ALL	Blue	2008	1
ALL	Blue	ALL	1
ALL	Black	2008	2
ALL	Black	2009	1
ALL	Black	ALL	3
ALL	ALL	2008	4
ALL	ALL	2009	2
ALL	ALL	ALL	6

Implementierung des Cubes

Implementierung der Aggregationsfunktionen:

- Initialisiere Berechnung und alloziere Handle für Zwischenergebnis:
  - `Init(handle)`
- Aggregiere den nächsten Wert zum Zwischenergebnis:
  - `Iter(handle, value)`
- Resultierendes Aggregat berechnen und Handle deallozieren:
  - `value = Final(handle)`

Implementierung des Cubes

13

Naiver Berechnungsalgorithmus:

1. Für jede Zelle des Cubes:
   - `Init(handle)`
2. Für jedes neue Tupel $(x_1, x_2, ..., x_N, v)$:
   - Für jede Zelle mit ‚ALL' auf mindestens einem der x_i:
     - `Iter(handle, v)`

   } 2^N
3. Für jedes Handle:
   - `Final(handle)`

Aggregationsfunktionen

14

Distributive Aggregationsfunktionen:

$$F(\{X_{i,j}\}) = G(\{F(\{X_{i,j} \mid i = 1,..., I\}) \mid j = 1,..., J\})$$

Beispiele:

- `COUNT()` $G = SUM()$
- `MIN()`
- `MAX()`
- `SUM()`

} $F = G$

Aggregationsfunktionen

15

Algebraische Aggregationsfunktionen:

$F(\{X_{i,j}\}) = H(\{G(\{X_{i,j} \mid i = 1,\ldots, I\}) \mid j = 1,\ldots, J\})$

UND: Zwischenergebnisse passen in ein Tupel **fester** Größe

Beispiel:

- `AVERAGE()`

G: Speichert die Summe und zählt die Werte

H: Summe / Anzahl

Aggregationsfunktionen

16

Holistische Aggregationsfunktionen:

$F(\{X_{i,j}\}) = H(\{G(\{X_{i,j} \mid i = 1,\ldots, I\}) \mid j = 1,\ldots, J\})$

UND: Keine obere Grenze für Größe des Zwischenergebnis

Beispiele:

- `MEDIAN()`
- `MOSTFREQUENT()`

Optimierter Berechnungsalgorithmus

17

Für distributive Aggregationsfunktionen:

1. Alle N-1 dimensionalen Aggregationen berechnen
   → `(..., ALL, ...)`

2. Nächst-niedrigere Aggregationen berechnen, ausgehend von `(..., ALL, ..., *, ...)` oder `(..., *, ..., ALL, ...)`
   → `(..., ALL, ..., ALL, ...)`

u.s.w.

Optimierter Berechnungsalgorithmus

18

Beispiel für distributive Aggregationsfunktionen:

Sales

Model	Color	Time	Price
Chevy	Red	2008-06-02	18 000
Chevy	Red	2009-01-03	39 000
Chevy	Black	2008-10-10	23 000
Chevy	Black	2007-08-11	21 000
Ford	Blue	2008-03-23	13 000
Ford	Blue	2007-11-04	25 000
Ford	Black	2008-12-14	12 000
Ford	Black	2009-01-10	10 000
Ford	White	2007-12-22	14 000

```
SELECT  Model, Color, Year, COUNT(*) AS Sales
FROM    Sales
WHERE   Year >= 2008
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```

Optimierter Berechnungsalgorithmus

19

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Black	2008	1
Chevy	Black	2007	1
Ford	Blue	2008	1
Ford	Blue	2007	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	White	2007	1

```
SELECT  Model, Color, Year, COUNT(*) AS Sales
FROM    Sales
WHERE   Year >= 2008
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```


Optimierter Berechnungsalgorithmus

20

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Black	2008	1
Chevy	Black	2007	1
Ford	Blue	2008	1
Ford	Blue	2007	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	White	2007	1

```
SELECT  Model, Color, Year, COUNT(*) AS Sales
FROM    Sales
WHERE   Year >= 2008
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```

Optimierter Berechnungsalgorithmus

21

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Black	2008	1
Ford	Blue	2008	1
Ford	Black	2008	1
Ford	Black	2009	1

```
SELECT  Model, Color, Year, COUNT(*) AS Sales
FROM    Sales
WHERE   Year >= 2008
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```


Optimierter Berechnungsalgorithmus

22

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	**Red**	**ALL**	**2**
Chevy	Black	2008	1
Chevy	**Black**	**ALL**	**1**
Ford	Blue	2008	1
Ford	**Blue**	**ALL**	**1**
Ford	Black	2008	1
Ford	Black	2009	1
Ford	**Black**	**ALL**	**2**

```
...     [...]
        GROUP BY CUBE
                Model, Color, Year(Time) AS Year;
```

Optimierter Berechnungsalgorithmus

23

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	**ALL**	**2008**	**2**
Chevy	**ALL**	**2009**	**1**
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

Model	Color	Year	Sales
Ford	**ALL**	**2008**	**2**
Ford	**ALL**	**2009**	**1**
ALL	**Red**	**2008**	**1**
ALL	**Red**	**2009**	**1**
ALL	**Blue**	**2008**	**1**
ALL	**Black**	**2008**	**2**
ALL	**Black**	**2009**	**1**

```
...     [...]
        GROUP BY CUBE
                Model, Color, Year(Time) AS Year;
```


Optimierter Berechnungsalgorithmus

24

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	ALL	2008	2
Chevy	ALL	2009	1
Chevy	**ALL**	**ALL**	**3**
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

	Color	Year	Sales
Ford	ALL	2008	2
Ford	ALL	2009	1
Ford	**ALL**	**ALL**	**3**
ALL	Red	2008	1
ALL	Red	2009	1
ALL	Blue	2008	1
ALL	Black	2008	2
ALL	Black	2009	1

```
...     GROUP BY CUBE
                Model, Color, Year(Time) AS Year;
```

Optimierter Berechnungsalgorithmus

25

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	ALL	2008	2
Chevy	ALL	2009	1
Chevy	ALL	ALL	3
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

Model	Color	Year	Sales
Ford	ALL	2008	2
Ford	ALL	2009	1
Ford	ALL	ALL	3
ALL	Red	2008	1
ALL	Red	2009	1
ALL	**Red**	**ALL**	**2**
ALL	Blue	2008	1
ALL	**Blue**	**ALL**	**1**
ALL	Black	2008	2
ALL	Black	2009	1
ALL	**Black**	**ALL**	**3**

...

```
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```


Optimierter Berechnungsalgorithmus

26

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	ALL	2008	2
Chevy	ALL	2009	1
Chevy	ALL	ALL	3
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

Model	Color	Year	Sales
Ford	ALL	2008	2
Ford	ALL	2009	1
Ford	ALL	ALL	3
ALL	Red	2008	1
ALL	Red	2009	1
ALL	Red	ALL	2
ALL	Blue	2008	1
ALL	Blue	ALL	1
ALL	Black	2008	2
ALL	Black	2009	1
ALL	Black	ALL	3
ALL	**ALL**	**2008**	**4**
ALL	**ALL**	**2009**	**2**

...

```
GROUP BY CUBE
        Model, Color, Year(Time) AS Year;
```

Optimierter Berechnungsalgorithmus

27

Beispiel für distributive Aggregationsfunktionen:

Model	Color	Year	Sales
Chevy	Red	2008	1
Chevy	Red	2009	1
Chevy	Red	ALL	2
Chevy	Black	2008	1
Chevy	Black	ALL	1
Chevy	ALL	2008	2
Chevy	ALL	2009	1
Chevy	ALL	ALL	3
Ford	Blue	2008	1
Ford	Blue	ALL	1
Ford	Black	2008	1
Ford	Black	2009	1
Ford	Black	ALL	2

Model	Color	Year	Sales
Ford	ALL	2008	2
Ford	ALL	2009	1
Ford	ALL	ALL	3
ALL	Red	2008	1
ALL	Red	2009	1
ALL	Red	ALL	2
ALL	Blue	2008	1
ALL	Blue	ALL	1
ALL	Black	2008	2
ALL	Black	2009	1
ALL	Black	ALL	3
ALL	ALL	2008	4
ALL	ALL	2009	2
ALL	**ALL**	**ALL**	**6**

```
...        GROUP BY CUI
                Model, Color, Year(Time) AS Year;
```


Optimierter Berechnungsalgorithmus

28

Für algebraische Aggregationsfunktionen:

- Wie für distributive Aggregationsfunktionen
- Aber: Zu jedem Sub-Aggregate muss das vollständige Zwischenergebnis-Tupel gespeichert werden

- Neue Methode für Aggregationsfunktionen, die Sub-Aggregates dem Super-Aggregate hinzufügt:

```
Iter_super(handle, handle)
```

Fazit

29

- Cube generalisiert Group By, Roll-ups und Cross-tabs
- Einführung des ‚ALL' Wertes zur Repräsentation der Menge, über die aggregiert wird
- Cube kann für viele Aggregationsfunktionen effizient berechnet werden

Diskussion

30

?