

Materialization Strategies in a Column-Oriented DBMS

Seminarvortrag

Tobias Müller

14. Januar 2011

Betreuer: Tom Schreiber
Wilhelm-Schickard-Institut

Hinführung (1)

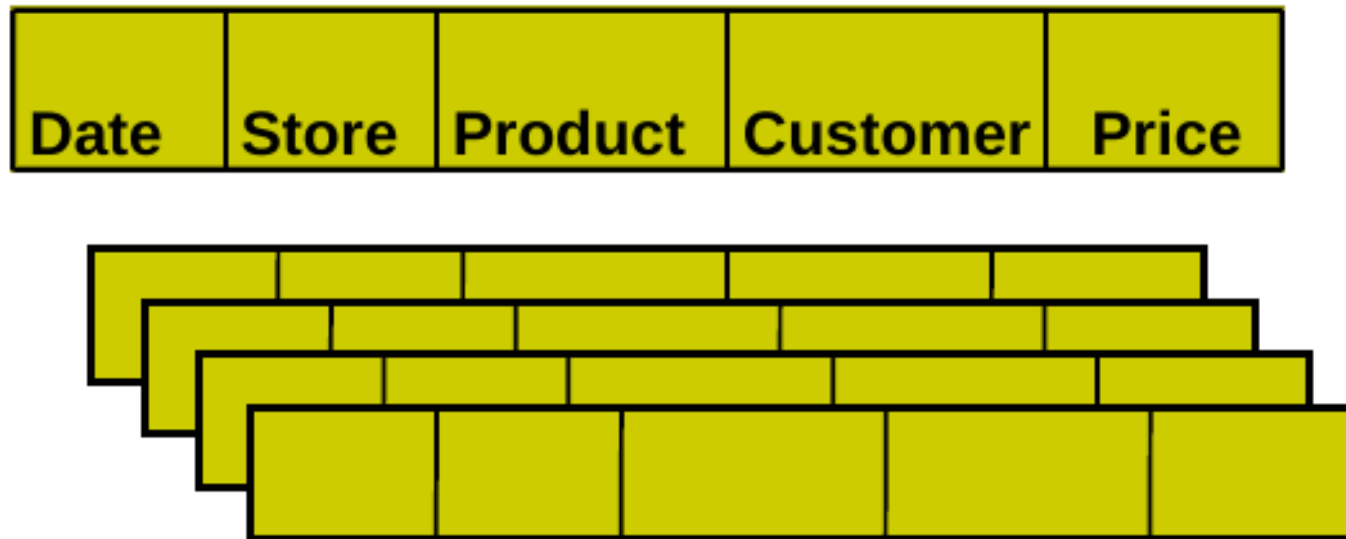


Abbildung: Bsp. Row Store

lese alle Spalten ✗

projiziere unerwünschte Spalten wieder weg ✗

Hinführung (2)

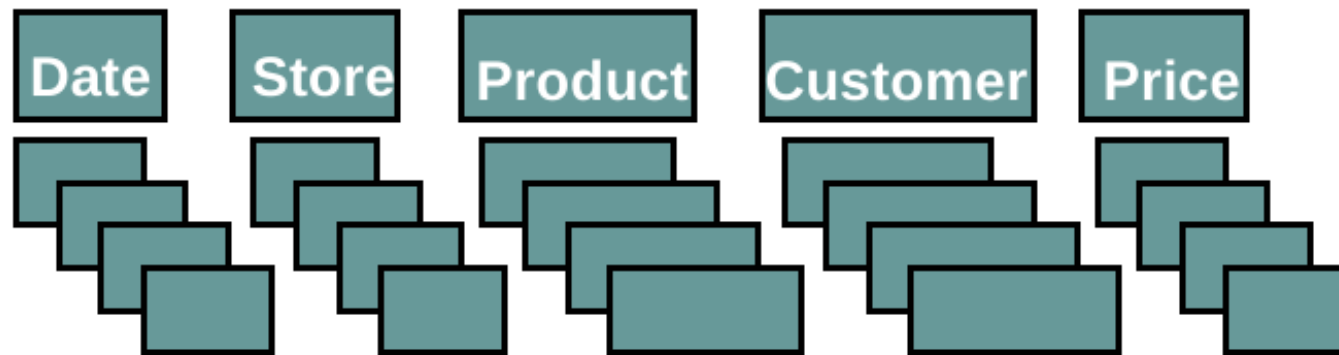


Abbildung: Bsp. Column Store

lese nur gewünschte Spalten ✓

erzeuge Tupel aus den gewünschten Spalten ✗

Hinführung (3)

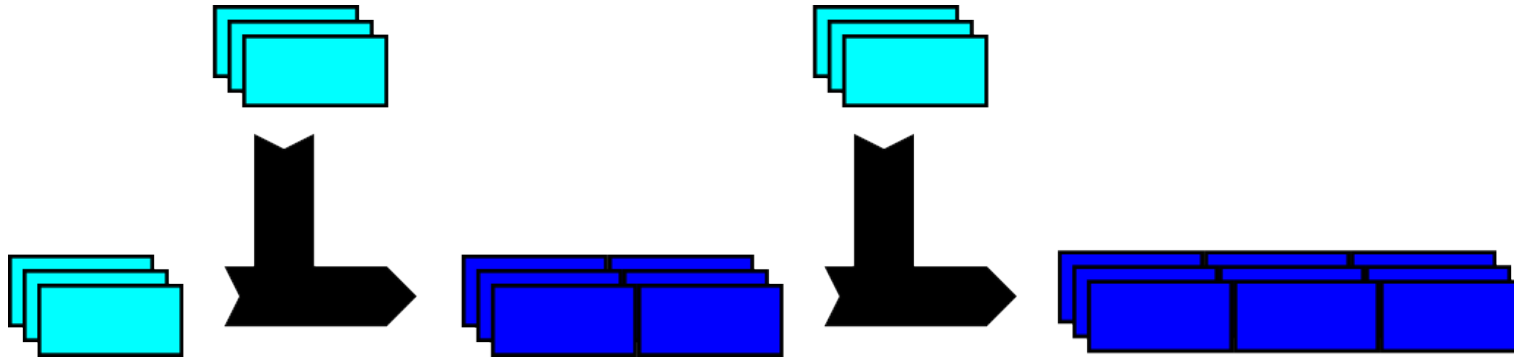


Abbildung: Materialisierung im Column Store

Problem:

Wann ist der beste Zeitpunkt um zu materialisieren?

Hauptstrategien

Early Materialisation (EM)

- Tupel werden von Prädikaten wegselektiert ✗
- Datenkompression verliert sich ✗

Late Materialisation (LM)

- Spalten mehrmals von Platte lesen ✗
- Komplexität ✗

Datenspeicherung

Hier am Bsp. C-Store (2006).

- 1 Spalte = 1 Datei
- 1 Datei = Aneinanderreihung von Disk Blöcken
- Spalten können komprimiert sein
- “Position Ordering”

Query Plans EM

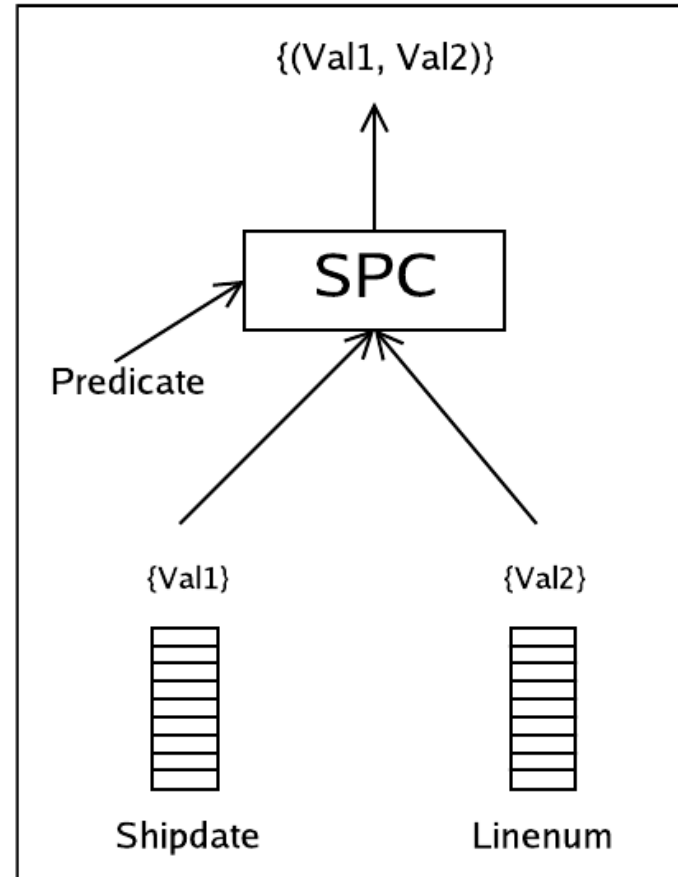
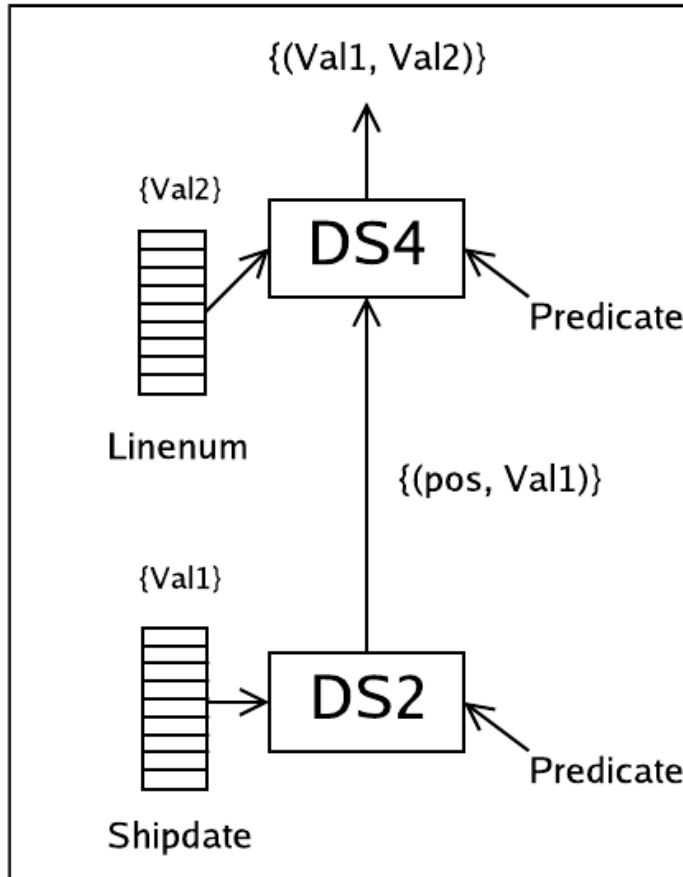


Abbildung: pipelined (links), parallel (rechts)

Experiment Setup

Hardware:

3,8GHz P4

3.5GB RAM (gesamt 4GB)

250GB Western Digital

Daten:

TPC-H (Benchmark)

ca. 10GB

ca. 60 Mio Tupel in Lineitem

Select Query

```
SELECT shipdate, linenum  
FROM Lineitem  
WHERE shipdate < X  
AND linenum < Y
```

Select Ergebnisse

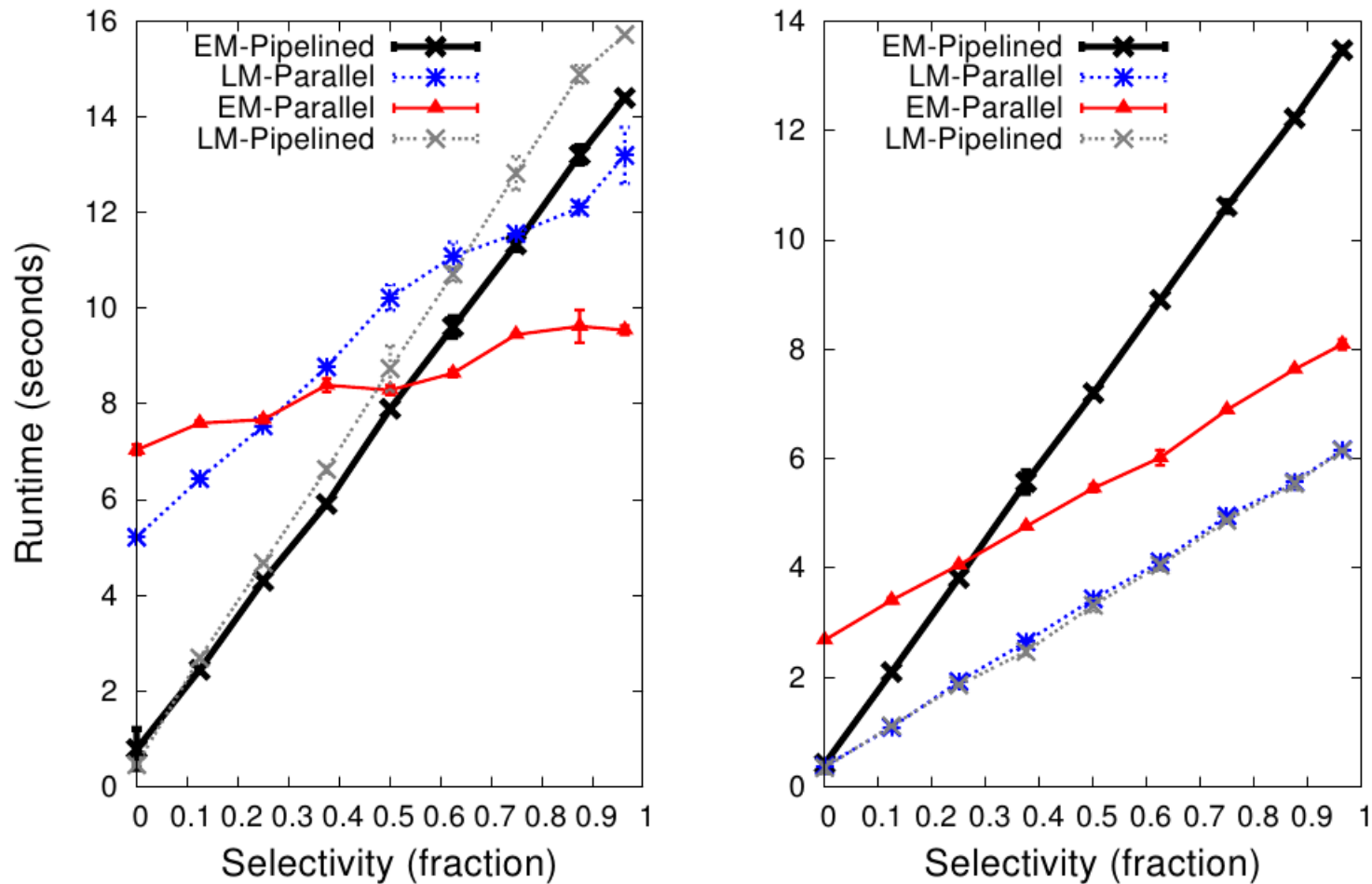


Abbildung: unkomprimiert (links), komprimiert (rechts)

Aggregation Query

```
SELECT shipdate, SUM(linenum)
FROM Lineitem
WHERE shipdate < X
AND linenum < Y
GROUP BY shipdate
```

Aggregation Ergebnisse

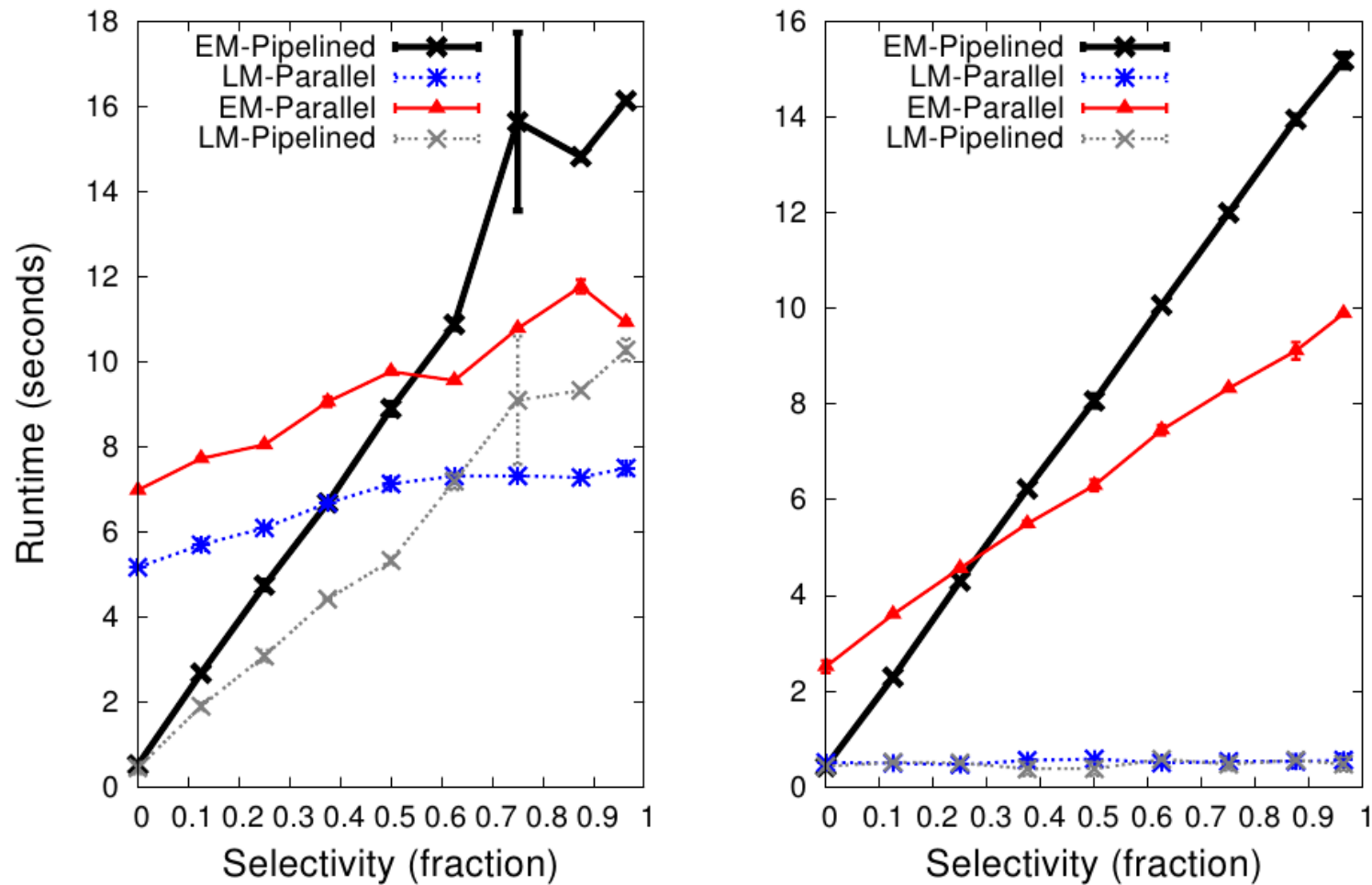


Abbildung: unkomprimiert (links), komprimiert (rechts)

Fazit

stark selektive Prädikate (wenig Tupel)

	select	+aggregation
komprimiert	LM	LM
unkomprimiert	EM+LM	LM

schwach selektive Prädikate (viele Tupel)

	select	+aggregation
komprimiert	LM	LM
unkomprimiert	EM	LM

Vielen Dank für die Aufmerksamkeit.

Multi-Column Optimization

Trick:

- ① Behalte zuletzt gelesene Spalten im Arbeitsspeicher
- ② Invalidiere wegselektierte Datensätze