Optimierte Speicherzuteilung an Operatoren in Datenbanksystemen

Jan Kristof Nidzwetzki
FernUniversität in Hagen

Inhalt

- Die Idee stammt aus der Bachelorarbeit *Operator Kostenmodelle für Fortschrittsschätzung und Optimierung in Datenbanksystemen*, welche im Jahr 2012 am Lehrgebiet *Datenbanksysteme für neue Anwendungen* an der *FernUniversität in Hagen* geschrieben wurde.

Motivation

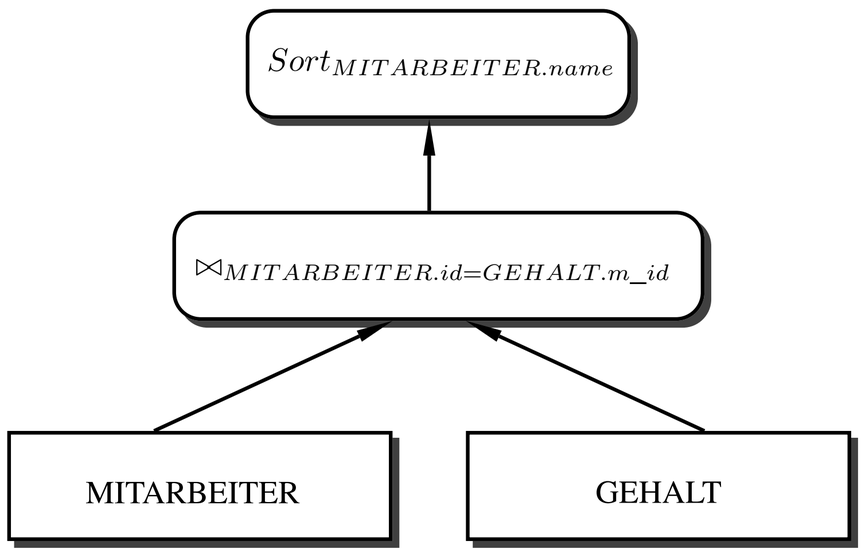
Um Anfragen in DBMS effizient auszuführen, ist es entscheidend, vorhandenen Arbeitsspeicher zwischen Operatoren bestmöglich aufzuteilen.

- Erhält ein Operator zu viel Speicher, ist dieser vergeudet.
- Erhält ein Operator zu wenig Speicher, arbeitet er langsamer als möglich.
- Nicht jeder Operator profitiert gleich stark von zusätzlichem Speicher.
- Ziel: Den Arbeitsspeicher den Operatoren so zuteilen, dass die Laufzeit der Anfrage minimal wird.

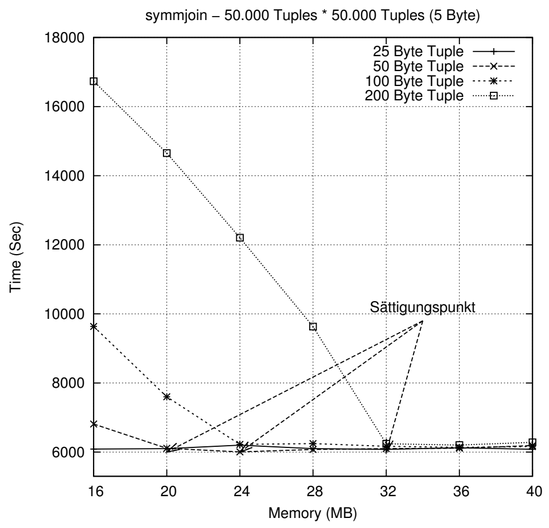
Wie kann dem DBMS bei der Aufteilung des Arbeitssspeichers geholfen werden?

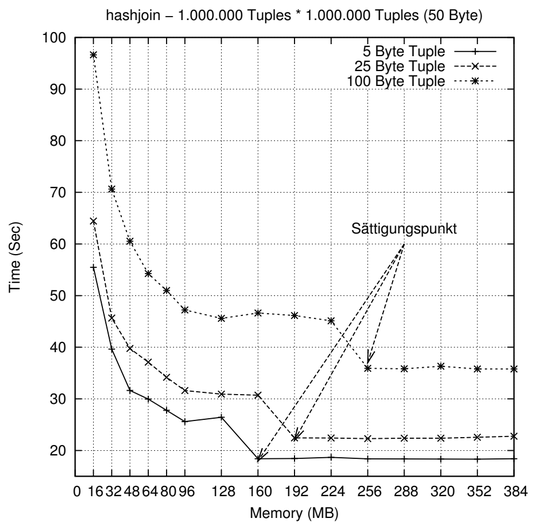
Umwandlung von SQL in einen Operatorbaum

```
SELECT * FROM mitarbeiter, gehalt WHERE mitarbeiter.id =
gehalt.m_id ORDER BY mitarbeiter.name;
```



Laufzeit von Operatoren in Abhängigkeit vom Arbeitsspeicher





- **Idee:** **Zusammenhang von Laufzeit und Arbeitsspeicher durch eine Funktion approximieren.**

Kostenmodelle für Operatoren

Beispiel: Zusammenführen zweier sortierter Tupelströme durch den Operator *mergejoin*.

- Kosten für das Durchlaufen der Tupelströme
- Kosten für das Erzeugen der Ergebnistupel

$$T_{mergejoin} = (C_1 + C_2) \cdot uMergejoin + C_{mergejoin} \cdot (SN_1 + SN_2) \cdot yMergeJoin$$

- C_1 und C_2 - Kardinalität der Eingabe
- **uMergeJoin** - Zeit um ein Tupel der Eingabe zu verarbeiten
- $C_{megejoin}$ - Geschätzte Kardinalität des Ergebnisses
- SN_1 und SN_2 - Anzahl der Attribute in der Eingabe
- **yMergeJoin** - Zeit um ein Attribut in ein neues Tupel einzufügen

Funktionen für die Approximation

Approximation der Laufzeit mit Hilfe von zwei Funktionen:

Lineare Funktion:

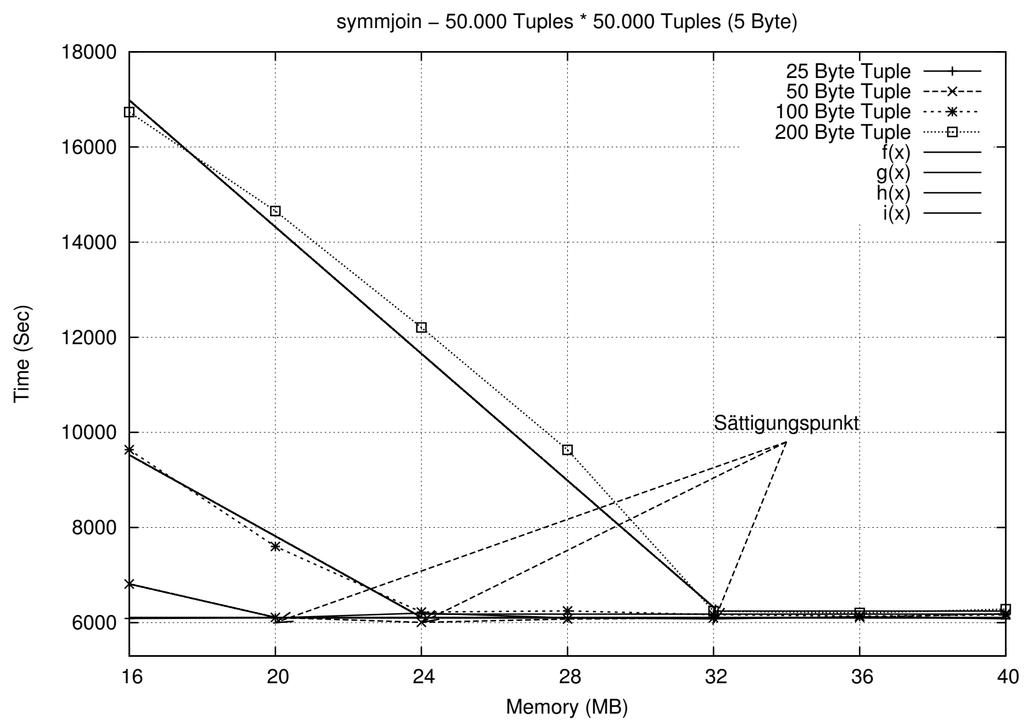
$$\textbf{Laufzeit(x)} = \begin{cases} \textbf{mx} + \textbf{b} & \text{falls x} < \text{Sättigungspunkt} \\ \textbf{c} & \text{sonst} \end{cases}$$

Antiproportionale Funktion:

$$\textbf{Laufzeit(x)} = \begin{cases} \frac{\textbf{h}}{\textbf{x}} + \textbf{b} & \text{falls x} < \text{Sättigungspunkt} \\ \textbf{c} & \text{sonst} \end{cases}$$

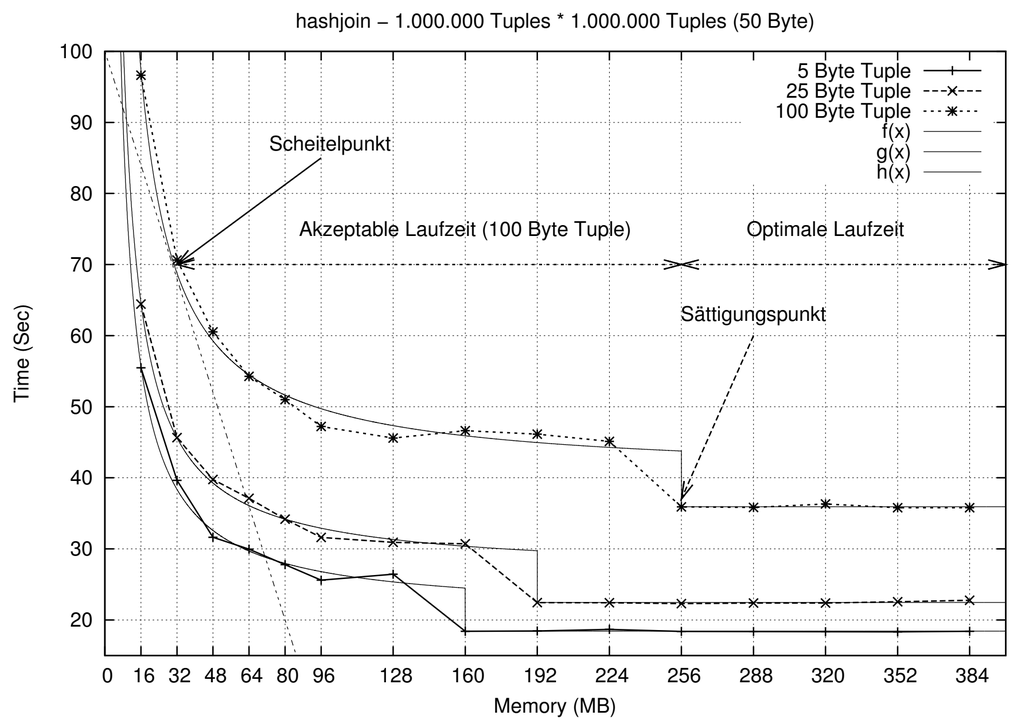
b, **c**, **h** und **m** sind die Lageparameter der Funktionen. **x** ist der zugeteilte Arbeitsspeicher.

Approximation von linearen Laufzeiten



Tupel	Funktion	Sättigungspunkt	Parameter			Abweichung	
			m	b	c	Var(x)	σ_x
25 Byte	f(x)	16 MB	0	6111	6111,00	1822,91	42,696
50 Byte	g(x)	20 MB	-175,802	9627	6097,27	4335,51	65,845
100 Byte	h(x)	24 MB	-426,56	16351	6178,68	19482,00	139,578
200 Byte	i(x)	32 MB	-666,23	27644	6243,72	92579,60	304,269
$\bar{\sigma}_x$	-	-	-	-	-	-	**138,096**

Approximation von antiproportionalen Laufzeiten



Tupel	Funktion	Sättigungspunkt	Parameter			Abweichung	
			h	b	c	Var(x)	σ_x
5 Byte	f(x)	160 MB	555,761	21,0096	18,4181	0,450218	0,670983
25 Byte	g(x)	192 MB	609,459	26,5603	22,4604	0,292114	0,540476
100 Byte	h(x)	256 MB	915,327	40,1894	35,9282	1,515780	1,231170
$\bar{\sigma}_x$	-	-	-	-	-	-	**0,81421**

Fazit

- Betrachtete Laufzeiten lassen sich gut durch Funktionen approximieren.
- Diese bilden eine Grundlage für die Entscheidungsfindung im Optimierer, den Arbeitsspeicher effizient aufzuteilen.
- Ein präzises Kostenmodell der Operatoren ist notwendig.
- Da andere *DBMS* ähnlich aufgebaut sind, ist zu erwarten, dass eine Approximimation auch dort möglich ist.