

Anmerkungen zum verfeinerten Join-Algorithmus

Jan Stohner
stohner@cs.uni-bonn.de

Jürgen Kalinski
cully@cs.uni-bonn.de

27. November 1998

Zur Realisierung der Verbund-Operation wird in relationalen Datenbankmanagementsystemen häufig der *Nested Loop Algorithmus* benutzt. In verschachtelten Schleifen werden alle Tupel einer Relation (äußere Schleife) jeweils einzeln mit allen Tupeln der zweiten Relation (innere Schleife) verglichen.

Won Kim hat 1980 ein verfeinertes Verfahren vorgestellt mit dem sich Einsparungen erzielen lassen [4]. Wir wollen im folgenden unterschiedliche, in der Literatur zu findende Aussagen zum verfeinerten Algorithmus gegenüberstellen und den Aufwand zur optimalen Durchführung des Algorithmus kritisch beleuchten.

1 Einführung

Der verfeinerte Algorithmus für Nested Loops Joins liest die beteiligten Relationen nicht tupel- sondern blockweise und durchläuft die innere Relation im „Zick-Zack-Verfahren“ (siehe Kemper [3], Seite 218). Wir nehmen an, daß im Hauptspeicher m Seiten für die Berechnung der Verbundoperation zur Verfügung stehen, von denen k von der inneren Relation verbraucht werden dürfen. Die äußere Relation R besteht aus insgesamt b_R Seiten, die innere Relation S aus b_S Seiten.

Zunächst werden $m - k$ Seiten der äußeren Relation R eingelesen. Um den Verbund für die darin enthaltenen Tupel zu berechnen, werden alle Seiten von S sequentiell gelesen. Danach werden die nächsten $m - k$ Seiten von R in den Hauptspeicher geholt. Anstatt aber für diese die innere Relation wieder von Beginn an vollständig zu durchlaufen, wird ausgenutzt, daß deren k letzten Seiten noch im Hauptspeicher stehen. Folglich werden erst diese Seiten verarbeitet, bevor die restlichen $b_S - k$ Seiten eingelesen werden. Die Relation R wird dabei also „von hinten nach vorn“ durchlaufen (engl.: *Rocking-Technik*).

Zur Berechnung des Verbunds werden auf diese Weise

$$b_R + k + \left\lceil \frac{b_R}{m - k} \right\rceil (b_S - k) \quad (1)$$

Seitenleseoperationen ausgeführt. Man kann sich nun fragen, wie der verfügbare Hauptspeicher für R und S aufgeteilt werden sollte, um die Verbundberechnung möglichst effizient durchzuführen. Mit anderen Worten lautet die Aufgabe, für gegebene Werte von b_R , b_S und m ein k zu bestimmen, bei dem die Kosten in Ausdruck (1) minimal werden. Hierbei ist insbesondere auch zu klären, welche der beiden Relationen im Nested Loops Join der äußeren und welche der inneren Schleife zugeordnet werden sollte.

Erstaunlicherweise äußert sich die Literatur zu diesen Punkten recht uneinheitlich.

Dieser Ausdruck wird minimal, wenn R die kleinere der beiden Relationen ist und für die innere Schleife nur ein Puffer von einer Seite ($k = 1$) verwendet wird ...

Kemper [3], Seite 234

Thus, the total cost for scanning the inner input repeatedly is $\lceil R/(M - K) \rceil \cdot (S - K) + K$. This expression is minimized if $K = 1$ and $R \geq S$, i.e., the larger input should be the outer.

Graefe [1], Seite 106

The block-oriented implementation of the nested loops-join optimizes on I/O overhead in the following way. Since the inner relation is read once for each tuple in the outer relation, the operation is most efficient when the relation with the lower cardinality is chosen as the outer relation. ... An analysis of buffer management for the nested loops method with rocking shows that buffering an equal number of pages for both relations is the best strategy [Hagmann 1986].

Mishra und Eich [5], Seite 72 (basierend auf „I/O-Operationen“ statt der Anzahl zu lesender Seiten als Kostenmaß)

The Problem of computing the product or join of n relations by the nested-block method has now been reduced to the problem of determining an optimal allocation of available main-memory buffer to n blocks for holding the n relations. This reduces to a nonlinear integer-programming problem under a set of linear constraints. . . . The problem of minimizing Formula 1 does not appear to be amenable to an analytic solution.

Kim [4], Seite 180

Wir wollen deshalb im folgenden diskutieren und visuell darstellen, wie der Aufwand des verfeinerten Algorithmus von unterschiedlichen k -Werten beeinflusst wird.

2 Ist $k = 1$ optimal?

Wenn die Rundungsoperation $\lceil \cdot \rceil$ vernachlässigt wird, ergibt sich mathematisch tatsächlich ein optimaler Wert von $k = 1$. Desweiteren gilt dann, daß die kleinere Relation im Verbund die äußere sein sollte.

Diese Beobachtungen können allerdings nicht in jedem Fall auf den ursprünglichen Ausdruck in (1) übertragen werden, wie das folgende Beispiel zeigt. Für $m = 100$, $b_R = 110$ und $b_S = 120$ beispielsweise liefert $k = 1$ den Wert

$$b_R + k + \left\lceil \frac{b_R}{m - k} \right\rceil (b_S - k) = 110 + 1 + 2 \cdot 119 = 349, \quad (2)$$

während für $k = 2$ nur

$$110 + 2 + 2 \cdot 118 = 348,$$

Seitenleseoperationen ausgeführt werden. Wenn wir die größere Relation S im Nested Loops Join außen verarbeiten, sind für $k = 40$ sogar nur noch

$$120 + 40 + 2 \cdot 70 = 300 \quad (3)$$

Seitenleseoperationen erforderlich.

Betrachten wir die Herleitung von Gleichung (2) im Detail. Die kleinere Relation R steht außen.

1. Im ersten Schritt werden in die $m - k = 99$ Seiten des Hauptspeichers, der für R reserviert ist, 99 Seiten von R eingelesen.

In der inneren Schleife muß die gesamte Relation S durchlaufen werden, wofür $b_S = 120$ Seiten eingelesen werden müssen.

2. Im zweiten Schritt werden die verbleibenden 11 Seiten von R eingelesen.

Wegen $k = 1$ steht nur eine Seite als Zwischenspeicher für die inneren Schleifen bereit. Dort befindet sich beim Start dieses Schrittes die letzte Seite von S aus dem vorangegangenen Durchlauf. Diese bereits eingelesene Seite kann schon benutzt werden, ohne daß sie neu eingelesen werden müßte. Demnach müssen jetzt noch die 119 anderen Seiten von S eingelesen werden.

Die äußere Schleife wurde zweimal durchlaufen, wobei beim zweiten Durchlauf jedoch nur 11 der verfügbaren 99 Seiten genutzt wurden. Stattdessen könnte man auch 55 Seiten für R reservieren und diese dann in beiden Durchläufen vollständig füllen.

1. Im ersten Schritt wird ein Teil von R die dafür reservierten 55 Seiten des Hauptspeichers eingelesen.

In der inneren Schleife muß die gesamte Relation S durchlaufen werden, wofür $b_S = 120$ Seiten eingelesen werden müssen.

2. Im zweiten Schritt werden die verbleibenden 55 Seiten von R eingelesen.

In der inneren Schleife können zunächst die beim ersten Schritt schon eingelesenen 45 letzten Seiten von S benutzt werden. Danach müssen nur noch die $120 - 45 = 75$ restlichen Seiten von S eingelesen werden.

Insgesamt ergibt sich ein Aufwand von 305 Seitenleseoperationen — offensichtlich eine bessere Ausführung als für $k = 1$ — aber schlechter als die Realisierung bei Ausdruck (3), bei der die größere Relation außen steht.

3 Ein allgemeines Gegenbeispiel

Die Beobachtung, daß $k = 1$ nicht den optimalen Fall darstellt, ist kein Einzelfall für geschickt gewählte Werte von m , b_R und b_S . Vielmehr tritt für jedes $m > 3$ der Fall auf, daß $k = 1$ *nicht optimal* ist. Man wähle beispielsweise $b_R = (m - 2)^2$ und b_S beliebig. Mit $C : \mathbb{N}^4 \rightarrow \mathbb{N}$ als Kostenfunktion

$$C : (m, k, b_R, b_S) \mapsto b_R + k + \left\lceil \frac{b_R}{m - k} \right\rceil (b_S - k)$$

gilt dann:

$$C(m, 1, b_R, b_S) > C(m, 2, b_R, b_S).$$

Ein Beweis findet sich in Anhang A. Das Gegenbeispiel in Abschnitt 2 nutzt den Rest der Division $\lceil \frac{b_R}{m-k} \rceil$ aus. Der letzte Durchlauf der äußeren Schleife belegt — in Abhängigkeit von m und b_R — die zur Verfügung stehenden $m - k$ Seiten eventuell nicht vollständig. Man kann nun k so weit erhöhen, daß die Gesamtzahl der Durchläufe der äußeren Schleife konstant bleibt, der „Verschnitt“ aber minimal wird. Im Beispiel wäre das für $k = 45$ der Fall.

Dieser Trick läßt sich nicht beliebig ausnutzen, denn mit zunehmender Größe von R im Verhältnis zur Größe des Hauptspeichers m wird die verbleibende Lücke beim letzten Durchlauf der äußeren Schleife immer kleiner und kann nicht mehr zur Erhöhung von k genutzt werden. Grob abgeschätzt erhält man für $b_R > m^2$ eine Zahl von $\lceil \frac{m^2}{m-k} \rceil > \frac{m^2}{m} = m$ Durchläufen der äußeren Schleife. Die Zahl der Läufe ist damit größer als die Zahl der Seiten im Hauptspeicher und eine Einsparung durch Ausnutzung des Restes nicht mehr möglich.

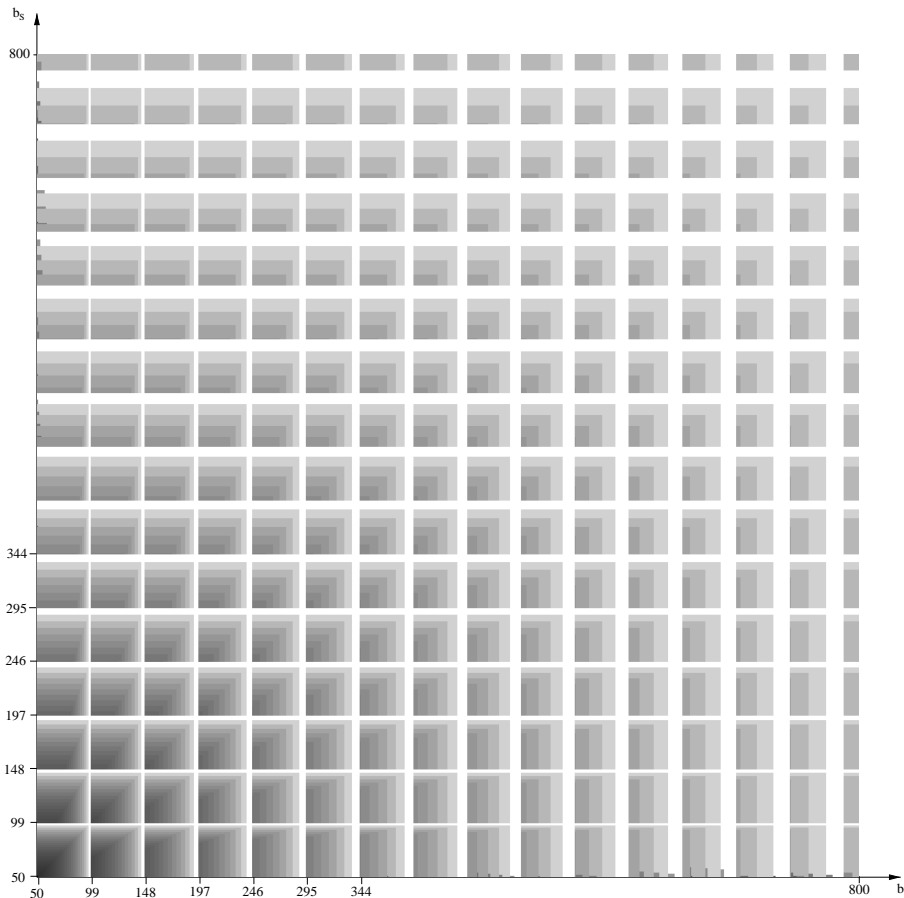


Abbildung 1: Graphische Darstellung am Beispiel $m = 50$. In Graustufen dargestellt ist für Paare b_R und b_S jeweils das optimale k .

4 Graphische Darstellung

Abbildung 1 zeigt deutlich, daß die Frage, welches k optimal ist, nicht so leicht beantwortet werden kann. Die Grafik bezieht sich auf den Fall $m = 50$, also einen Hauptspeicher mit 50 Seiten. Auf den Achsen ist b_R gegen b_S aufgetragen, und zwar jeweils im Bereich von 50 bis 800. Das optimale k erkennt man aus der Farbgebung der Punkte: Weiße Flächen stehen für $k = 1$, danach erhöht sich das k mit jeder Graustufe.

Für jedes Paar (b_R, b_S) wird dasjenige k angezeigt, mit dem der geringstmögliche Aufwand verbunden ist. Es wird nicht gefordert, daß die kleinere Relation außen steht, sondern die tatsächlich effizienteste Konfiguration ermittelt. Deshalb ist das Bild symmetrisch zur Winkelhalbierenden.

Man erkennt, daß die Bereiche mit $k \neq 1$ mit zunehmenden b_R, b_S weiter ausdünnen, bis schließlich (wenn b_R bzw. b_S größer als m^2 werden) der Fall $k = 1$ als optimale Lösung bleibt.

An den beiden Achsen sind „Verunreinigungen“ im Muster zu beobachten. Tatsächlich verhält sich die Kostenfunktion in Bereichen $m \approx b_R$ bzw. $m \approx b_S$ ganz und gar nicht so regelmäßig, wie ansonsten das Schaubild suggeriert.

A Beweis

Im folgenden wird gezeigt, daß es für jedes beliebige $m > 3$ mindestens ein Paar (b_R, b_S) gibt, für das der Fall $k = 1$ *nicht optimal* ist.

Beweis: Sei m beliebig, wähle $b_R = (m - 2)^2$ und b_S beliebig. Sei C die Kostenfunktion wie oben beschrieben. Die folgende Rechnung beweist, daß für diese Werte das Verfahren mit $k = 2$ besser ist als mit $k = 1$, also:

$$\begin{aligned} C(m, 1, b_R, b_S) &> C(m, 2, b_R, b_S) \\ C(m, 1, (m - 2)^2, b_S) &> C(m, 2, (m - 2)^2, b_S) \\ (m - 2)^2 + 1 + \left\lceil \frac{(m - 2)^2}{m - 1} \right\rceil (b_S - 1) &> (m - 2)^2 + 2 + \left\lceil \frac{(m - 2)^2}{m - 2} \right\rceil (b_S - 2) \\ \left\lceil \frac{(m - 2)^2}{m - 1} \right\rceil (b_S - 1) &> 1 + \left\lceil \frac{(m - 2)^2}{m - 2} \right\rceil (b_S - 2) \end{aligned}$$

Da gilt: $(m - 2)^2 = (m - 2)(m - 1) - (m - 1) + 1$, ersetzt man $(m - 2)^2$ auf der linken Seite:

$$\begin{aligned} \left\lceil \frac{(m - 2)(m - 1) - (m - 1) + 1}{m - 1} \right\rceil (b_S - 1) &> 1 + \left\lceil \frac{(m - 2)^2}{m - 2} \right\rceil (b_S - 2) \\ \left\lceil (m - 2) - 1 + \frac{1}{m - 1} \right\rceil (b_S - 1) &> 1 + \lceil m - 2 \rceil (b_S - 2) \end{aligned}$$

Auf der rechten Seite entfällt das Aufrunden, da $m - 2$ eine ganze Zahl ist.

$$\left\lceil m - 3 + \frac{1}{m - 1} \right\rceil (b_S - 1) > 1 + (m - 2)(b_S - 2)$$

Auf der linken Seite kann das Aufrunden ersetzt werden: Da $0 < \frac{1}{m - 1} < 1$ für alle $m > 2$ gilt, wird zu $m - 3$ eine Zahl kleiner Eins addiert. Da $m - 3$ eine ganze Zahl ist und der Bruchteil kleiner als Eins ist, ist das Ergebnis nach dem Aufrunden $m - 2$.

$$\begin{aligned} (m - 2)(b_S - 1) &> 1 + (m - 2)(b_S - 2) \\ m \cdot b_S - m - 2b_S + 2 &> 1 + m \cdot b_S - 2m - 2b_S + 4 \\ -m + 2 &> -2m + 5 \\ m &> 3 \end{aligned}$$

Diese Bedingung ist erfüllt und damit der Beweis erbracht.

Literatur

- [1] **Graefe, G.:** *Query evaluation techniques for large databases.* ACM Computing Surveys, 25(2):73–169, Juni 1993.

- [2] **Hagman, R. B.:** *An Observation on Database Buffering Performance Metric.* In: W. W. Chu, G. Gardarin, S. Ohsuga und Y. Kambayashi (Herausgeber), *Proceedings of the Twelfth International Conference on Very Large Data Bases (VLDB)*, S. 289–293. Morgan Kaufmann, August 1986. ISBN 0-934613-18-4.
- [3] **Kemper, A. und A. Eickler:** *Datenbanksysteme: Eine Einführung.* Oldenbourg Verlag, München, 2. Auflage, 1997. ISBN 3-486-24136-2.
- [4] **Kim, W.:** *A New Way to Compute the Product and Join of Relations.* In: P. P. Chen und R. C. Sprowls (Herausgeber), *Proceedings of the 1980 ACM SIGMOD International Conference on Management of Data.* ACM Press, Mai 1980.
- [5] **Mishra, P. und M. H. Eich:** *Join processing in relational databases.* ACM Computing Surveys, 24(1):63–113, März 1992.