Anfrageoptimierung in objektrelationalen Datenbanken mit bedingten Termersetzungen

Mazeyar E. Makoui, Udo W. Lipeck
{mem, ul}@dbs.uni-hannover.de
Universität Hannover

Zusammenfassung

Die bisherige Optimierung von Anfragebäumen mit Hilfe von Termersetzungen in der Relationenalgebra liefert durch bekannte Heuristiken (z. B. frühzeitiges Ausführen von Projektionen und Selektionen) die Grundlage eines Termersetzungssystems, welches aber kaum eine Möglichkeit für weitere Optimierungsstrategien zur Anfrageoptimierung in objektrelationalen Datenbanken bietet. In diesem Artikel werden algebraische Termersetzungen um kostenbasierte Bedingungen erweitert. Als Basis dafür dienen sowohl Metadaten der vorhandenen Relationen, als auch der einzusetzenden Operatoren und Zugriffsmethoden. Diese werden durch ein erweitertes Kostenmodell während der Optimierung berücksichtigt und verbessern damit den Entscheidungsprozess verschiedener Anfrageoptimierungsalgorithmen. Als Beispiel hierfür wird der von Vance/Meier [Van96] vorgeschlagene Join-Ordering-Algorithmus in seinen Suchmöglichkeiten stark reduziert und dadurch beschleunigt.

1 Einleitung

In objektrelationalen Datenbanken sind die althergebrachten Heuristiken zur Anfrageoptimierung hinsichtlich der frühen Ausführung von Selektionen und Projektionen in den meisten Fällen nicht mehr sinnvoll. Vielmehr sollten kostspielige Operationen wie z. B. Selektionen auf räumlichen Daten in einem Zugriffsplan später ausgeführt werden, also nicht nur auf Reduzierung der zu betrachtenden Tupel geachtet werden. Demzufolge sollte ein Optimierer Operatoren und deren Implementierungskosten auch in früheren Phasen mit in Betracht ziehen können. Folgende drei Aspekte wären ideal:

1. Alle möglichen Operatoren mit ihren verschiedenen Implementierungen sollten mit einbezogen werden.
2. Alle möglichen Reihenfolgen einzelner Operatoren müssen betrachtet werden können.
3. Es muss eine Basis dafür geliefert werden, dass beliebige Optimierungsstrategien angewendet werden können.

Dieses wird von dem Konzept der bedingten Termersetzungen ermöglicht, ohne dabei den kompletten Suchraum betrachten zu müssen.

Im folgenden Abschnitt wird das Konzept der bedingten Termersetzung vorgestellt und motiviert. Danach folgt die Herleitung einer bedingten Termersetzungsregel für den Selektionsoperator, um im dritten Abschnitt für den Join-Ordering-Algorithmus verwendet zu werden. Im vierten Teilabschnitt wird ein Beispiel präsentiert. Schließlich befasst sich der fünfte Abschnitt mit Erweiterungen und einem Ausblick der vorgestellten Arbeit.

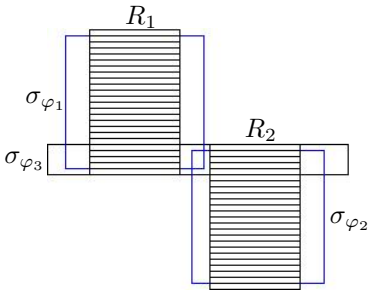
2 Konzept der kostenbedingten Termersetzungen

Die Idee der bedingten Termersetzung besteht darin, Termersetzungen nicht mehr rein heuristisch vorzunehmen, sondern von einer Kostenfunktion abhängig zu machen. Dabei erweitern wir die bisherigen algebraischen Ersetzungsregeln durch Bedingungen, so dass die Regeln auf Basis aller zusätzlichen Informationen, die in ein erweitertes Kostenmodell eingeflossen sind, alternative Ausführungspläne in Beziehung zueinander setzen können.

2.1 Motivation einer kostenbedingten Termersetzung

Gegeben seien zwei Relationen R_1 und R_2 und drei darauf agierende Selektionen ($\overline{\mathsf{R}}_1 := \sigma_{\varphi_1}(\mathsf{R}_1), \overline{\mathsf{R}}_2 := \sigma_{\varphi_2}(\mathsf{R}_2)$ und $\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2 = \sigma_{\varphi_3}(\mathsf{R}_1 \times \mathsf{R}_2)$). Die Joinbedingung ($\sigma_{\varphi_3}$), wie im folgenden Beispiel gezeigt, kann so stark selektierend sein, dass es sich nicht lohnt, erst die Ausgangsrelationen durch die zugehörigen Selektionen einzuschränken und dann zu joinen. Dabei soll vorausgesetzt werden, dass keine Indexe oder Sortierungen existieren, die man zusätzlich dazu berücksichtigen müsste (siehe nachfolgende Abbildung):

Beispiel für einen stark selektierenden Verbund:



Dazu seien folgende Statistiken gegeben[1]:

Selektivität		Kardinalität
$\mathsf{sel}(\varphi_1, \mathsf{R}_1)$	$= 0,90$	$\lvert\mathsf{R}_1\rvert = 30$
$\mathsf{sel}(\varphi_2, \mathsf{R}_2)$	$= 0,90$	$\lvert\mathsf{R}_2\rvert = 30$
$\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2)$	$= 0,01$	
$\mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2)$	$= 0,01$	

Die aus der Standardliteratur (siehe [Kem04, S.233]) bekannte Selektionsverschiebungsregel für Verbunde, hier L-Rule genannt:

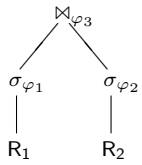
L-Rule $\tau_1 := \sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)) \rightarrow \tau_2 := \sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2)$,

falls $\mathsf{attr}(\varphi_1) \subseteq \mathsf{sch}(\mathsf{R}_1)$ **und** $\mathsf{attr}(\varphi_2) \subseteq \mathsf{sch}(\mathsf{R}_2)$

liefert laut Heuristik immer den Ausführungsplan, der zuerst die Basisrelationen selektiert und dann erst den Verbund berechnet. Genau diese Regel scheint aber hier nicht zu passen.

Nachfolgend betrachten wir beide allgemein möglichen Ausführungspläne τ_1 und τ_2. Dabei benutzen wir ein Kostenmodell (siehe dazu [Mak03]), das nur die Anzahl der Tupel, die in den Join „hineingehen“ und ihn wieder „verlassen“, in Beziehung zueinander setzt. Vorausgesetzt, dass die zu selektierenden Tupel gleichverteilt in den Selektionen vorkommen und somit die Selektivitäten von σ_{φ_1} und σ_{φ_2} konstant bleiben, folgt:

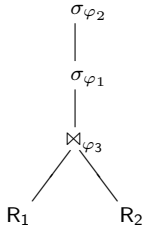
- 1. Heuristisch optimierter Ausführungsplan τ_1:



1. $\lvert\overline{\mathsf{R}}_1\rvert = \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert = 0,9 \cdot 30 = 27$
2. $\lvert\overline{\mathsf{R}}_2\rvert = \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert = 0,9 \cdot 30 = 27$
3. $\lvert\overline{\mathsf{R}}_1 \bowtie_{\varphi_3} \overline{\mathsf{R}}_2\rvert = \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot \lvert\overline{\mathsf{R}}_1\rvert \cdot \lvert\overline{\mathsf{R}}_2\rvert$
   $= 0,01 \cdot (27 \cdot 27) = 7,29$

$\Rightarrow \mathsf{cost}(\tau_1) = 27 + 27 + 7,29 = 61,29$

- 2. Nicht heuristisch optimierter Ausführungsplan τ_2:



1. $\lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert = \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert$
   $= 0,01 \cdot (30 \cdot 30) = 9$
2. $\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert = 0,9 \cdot 9 = 8,1$
3. $\mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert\rvert = 0,9 \cdot 8,1 = 7,29$

$\Rightarrow \mathsf{cost}(\tau_2) = 9 + 8,1 + 7,29 = 24,39$

Unter den vorangehenden Einschränkungen muss τ_2 - also ohne Anwendung der Standardheuristik - weniger als $39,79\,\%$ der Tupel von τ_1 berücksichtigen.

2.2 Herleitung einer kostenbedingten Termersetzungsregel

Angestrebt wird jetzt eine Ersetzungsregel $\tau_1 \rightarrow \tau_2$ mit einer Bedingung, die $\mathsf{cost}(\tau_1) \geq \mathsf{cost}(\tau_2)$ gewährleistet. Um diese Bedingung vorab auszurechnen und angemessen zu vereinfachen, vergleichen wir die Kostenformeln der beiden Ausführungspläne:

$$\begin{aligned}\mathsf{cost}(\tau_1) &= \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert) \\ &= \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert\end{aligned}$$

$$\begin{aligned}\mathsf{cost}(\tau_2) &= \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert) + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \\ &(\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert)) = (\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert) \cdot (1 + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (1 + \mathsf{sel}(\varphi_2, \mathsf{R}_2)))\end{aligned}$$

[1] Das Bild liefert zur besseren Visualisierung nicht maßstabsgetreu die Selektivitäten, die in der nachfolgenden Statistik angenommen werden.

Da wir die Situation suchen, in der beide Ausführungspläne die gleichen Kosten besitzen, setzen wir die beiden Kostenfunktionen gleich:

$$\mathsf{cost}(\tau_1) = \mathsf{cost}(\tau_2)$$

Aufgelöst nach $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2)$ ergibt sich die Grenzselektivität $\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2)$, bei der beide Ausführungspläne die gleichen Kosten verursachen:

$$\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) := \frac{|\mathsf{R}_1| \cdot (|\mathsf{R}_2| \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) + 1) \cdot \mathsf{sel}(\varphi_1, \mathsf{R}_1) + |\mathsf{R}_2| \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2)}{|\mathsf{R}_1| \cdot |\mathsf{R}_2| \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (\mathsf{sel}(\varphi_2, \mathsf{R}_2) + 1) + 1)}$$

Wir haben jetzt eine Ersetzungsregel, die bedingt angewandt werden kann:

L2-Rule

$$\sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)) \rightarrow \sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2), \quad \text{wenn } \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \geq \mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2)$$

$$\sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2) \rightarrow \sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)), \quad \text{wenn } \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \leq \mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2)$$

Bislang sind wir davon ausgegangen, dass in jedem Zeitpunkt alle Selektivitäten bekannt sind. Das ist aber in der Realität nicht so. Vielmehr besitzen wir in unserem konkreten Fall, wenn eine Selektion an einem Join vorbei geführt werden soll, die Selektivität des eigentlichen Joins nicht mehr, da sie sich in diesem Fall verändern kann. Aus diesem Grund müssen wir eine Unabhängigkeit der einzelnen Selektionen annehmen und können dazu die alte Selektivität des Joins für die Berechnung benutzen. Somit wird im Folgenden $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) = \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2)$ gesetzt. In unserem Beispiel ergibt sich für den rechten Kostenterm:

$$\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) = \frac{30 \cdot (30 \cdot \frac{9}{10} \cdot \frac{1}{100} + 1) \cdot \frac{9}{10} + 30 \cdot \frac{9}{10}}{30 \cdot 30 \cdot (\frac{9}{10} \cdot (\frac{9}{10} + 1) + 1)} = \frac{681}{27100} \approx 0,025129$$

Mit $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) = \frac{1}{100}$ folgt, da $0,01 = \frac{1}{100} < \frac{681}{27100} \approx 0,025129$, der richtige Ausführungsplan, da die bedingte Regel dazu rät, die Selektionen erst nach dem Join anzuwenden.

3 Erweiterung des Join-Ordering-Algorithmus um Selektionen

Im Folgenden zeigen wir wie der Join-Ordering-Algorithmus von Vance/Meier mit der L2-Rule beschleunigt werden kann. Dazu beschreiben wir zunächst die Arbeitsumgebung:

Zu optimieren ist ein Term bestehend aus natürlichen Joins und Selektionen über einer Menge von n verschiedenen Basisrelationen $\mathcal{R} = \{\mathsf{R}_\mathsf{i} \mid \mathsf{i} = 1, ..., \mathsf{n}\}$. Die Selektionen lassen sich darstellen als eine Menge von (konjunktiv verknüpfbaren, möglichst atomaren) Selektionsbedingungen $\mathcal{P} = \{\varphi_\mathsf{j} \mid \mathsf{j} = 1, ..., \mathsf{m}\}$, von denen jede sich auf eine oder mehrere Basisrelationen bezieht. Wenn eine Bedingung φ nur Attribute einer Teilmenge $\widetilde{\mathsf{R}}$ von $\mathcal{R}$ enthält, schreiben wir $(\widetilde{\mathsf{R}}, \varphi) \in \mathcal{P}$.

Gegenstand der Optimierung ist einerseits die Anordnung der Joins und andererseits die Positionierung der Selektionen, die nach den Überlegungen im Abschnitt 2 eben nicht grundsätzlich wie bei der üblichen Heuristik so früh wie möglich ausgeführt werden sollten.

Die entstehenden *Optimierungsterme* können dann rekursiv dargestellt werden als $\sigma_\Phi(\tau_{\mathsf{left}} \bowtie \tau_{\mathsf{right}})$, wobei Φ eine Menge äußerer Selektionsbedingungen (und-verknüpft) ist, welche leer sein kann (entspricht „true“), und τ_{left}, τ_{right} wieder Terme aus den oben genannten Operationen sind, die alle anderen Selektionsbedingungen enthalten. Basisterme sind von der Form $\sigma_\varphi(\mathsf{R}_\mathsf{i})$, falls $(\varphi, \mathsf{R}_\mathsf{i}) \in \mathcal{P}$, oder $\sigma_{\mathsf{true}}(\mathsf{R}_\mathsf{i})$ wobei R_i eine Basisrelation ist.

Wir orientieren uns an dem in [Van96] vorgeschlagenen Vorgehen der Optimierung von Join-Reihenfolgen durch dynamische Programmierung, bei dem aus den Optimierungstermen für immer größer werdende Teilmengen von $\mathcal{R}$ schließlich ein optimaler Term konstruiert wird. Wir arbeiten allerdings mit durch Selektionen geschachtelten Mengen von Relationen, um auch die Positionierung der Selektionen zu optimieren.[2]

Dazu wird eine Tabelle `plantable` aufgebaut, deren Einträge durch solche eventuell geschachtelten Mengen von Relationen indiziert sind; ein Eintrag liefert den zugehörigen Optimierungsterm durch Angabe der äußeren Selektionsbedingungen `selections` (oben: Φ) und des Eintrags `leftSide` für den linken Teilterm (oben: τ_{left}, aber in Mengendarstellung); der Eintrag für den rechten Teilterm (oben: τ_{right}) ist daraus rekonstruierbar.

Zusätzlich werden zu jedem Eintrag - zur schrittweisen Berechnung - folgende Attribute festgehalten:

[2] Alle Erweiterungen gegenüber dem Original-Algorithmus werden nachfolgend blau dargestellt.

- `cardinality` die Kardinalität der durch den Optimierungsterm dargestellten Relation
- `selectivity` die Selektivität äußerer Joinbedingungen des Optimierungsterms[3]
- `cost` die Gesamtkosten des Optimierungsterms

Input: Relationenmenge $\mathcal{R}$, Menge aller Selektionsbedingungen $\mathcal{P}$
Output: Liefert den optimierten Anfrageplan

```
optimize(Relationenmenge R, Bedingungsmenge P) {
  // Folgender Abschnitt ist die Eintragung der Basisrelationen
  // in die planTable:
  planTable := ∅;
  for each R_i ∈ R do {
    planTable[R_i].cardinality := |R_i|;
    planTable[R_i].leftSide := ∅;
    // κ(τ) liefert die Kosten eines Operators, wobei
    // Indexe und Sortierungen berücksichtigt werden.
    planTable[R_i].cost := κ(σ_true(R_i));
    planTable[R_i].selectivity := 1;
    planTable[R_i].selections := ∅;
    if (S ≠ ∅) {
    S := {φ | (R_i, φ) ∈ P} alle Bedingungen, die auf R_i agieren;
      ψ sei deren Konjunktion
      planTable[σ_ψ{R_i}].cardinality := |σ_ψ(R_i)|;
      planTable[σ_ψ{R_i}].leftSide := ∅;
      planTable[σ_ψ{R_i}].cost := κ(σ_ψ(R_i));
      planTable[σ_ψ{R_i}].selectivity := 1;
      planTable[σ_ψ{R_i}].selections := S;
    }
  }
  // Im folgenden Abschnitt findet die eigentliche Opt. statt:
  for (jede Teilmenge R̃ := R̃_1 ∪ R̃_2, R̃_1 ∩ R̃_2 = ∅, für die
    zu R̃_1 und R̃_2 Einträge in der planTable[R̃] existieren{
    planTable[R̃].cardinality := |R̃|;
    planTable[R̃].leftSide := find_best_split(R̃);
    planTable[R̃].cost := planTable[τ_left].cost +
                         planTable[τ_right].cost + κ(R̃_1 ⋈_ψ R̃_2);
    P̃ := natürliche Verbundbedingungen für R̃
    planTable[R̃].selectivity := compute_selectivity(P̃);
    planTable[R̃].selections := ∅;
    S := {φ | (R̃, φ) ∈ P und φ kommt in R̃ noch nicht vor}
    alle weiteren Bed., die auf R̃ agieren; ψ sei deren Konj.
    if (Eintrag R̃ ist optimierbar mit L2-Regel) {
      remove(R̃, planTable);
    }
    // Entfernt Einträge, bei denen die Selektion zu "tief"
    // im Baum ist; z. B. wird R̃ = {R, σ_ρ{S, T}} daraufhin ge-
    // prüft, ob der Term R ⋈ σ_ρ(S ⋈ T) nach der L2-Rule zu
    // σ_ρ(R ⋈ (S ⋈ T)) optimiert wurde; dann kann der Eintrag
    // gelöscht werden, weil zu σ_ρ{R, S, T) an anderer Stelle ein
    // Eintrag eingeführt wird.
    // Hier folgt die Betrachtung von aufsteigenden Selektionen:
    if (S ≠ ∅) {
      planTable[σ_ψ{R̃}].cardinality := |σ_ψ{R̃}|;
      planTable[σ_ψ{R̃}].leftSide := planTable[R̃].leftSide;
      planTable[σ_ψ{R̃}].cost := κ(σ_ψ{R̃}) + planTable[R̃].cost;
      planTable[σ_ψ{R̃}].selectivity := 1;
      planTable[σ_ψ{R̃}].selections := S;
      // Entfernt Einträge, bei denen die Selektion zu "hoch"
      // im Baum ist. Hier wird geprüft, ob im Term
      // σ_ψ(R ⋈ σ_ρ(S ⋈ T)) die Selektion absinken
      // kann; dann gäbe es andere Einträge.
      if (Eintrag σ_ψ{R̃} ist optimierbar mit L2-Regel) {
        remove(σ_ψ{R̃}, planTable);
      }
    }
  }
  return get_result(R, planTable);
}
```

```
compute_selectivity(Verbundbedingungen P̃) {
  s := 1;
  for each φ_1 ∈ P do {
    s = s · sel(φ_1);
  }
  return s;
}
```

Input: Relationenmenge $\tilde{R}$
Output: Relationenmenge $\tilde{R}_{best}$ (Beste linke Seite des Vebundes leftSide)

```
// Kombiniert alle möglichen nicht leeren Teilmengen der überge-
// benen Relationenmenge und liefert die Kombination zurück, die
// insgesamt die geringsten Kosten verursacht.
find_best_split(Relationenmenge R̃) {
  R̃_best := ∅;
  best_cost := ∞;
  for each R̃_1, ∅ ⊂ R̃_1 ⊂ R̃ do {
    R̃_2 := R̃ \ R̃_1;
    cost = compute_cost(R̃_1, R̃_2);
    if (cost ≤ best_cost) {
      best_cost = cost;
      R̃_best = R̃;
    }
  }
  return R̃_best;
}
```

Durch die Anwendung der L2-Rule lassen sich Einträge in der Optimierungstabelle löschen, die dann beim Erzeugen von weiteren Einträgen nicht berücksichtigt werden müssen. Somit ist dieser Algorithmus auch schneller als die von [Ste97], [Sch97] und [Sch98] vorgeschlagene Optimierung mit Hilfe von Ranking-Verfahren, da unser Algorithmus schon bei der Erzeugung weniger Teilbäume betrachten muss.

4 Beispiel

Gegeben sei folgende Anfrage

$$\pi_{\mathsf{P.Name}}\left(\sigma_{\mathsf{S.Semester}='5'}\left(\mathsf{H} \bowtie_{\mathsf{H.MatrNr}=\mathsf{S.MatrNr}} \mathsf{S} \bowtie_{\mathsf{V.VorlNr}=\mathsf{H.VorlNr}} \mathsf{P} \bowtie_{\mathsf{P.ID}=\mathsf{V.gelesenVon}} \mathsf{V}\right)\right)$$

für die nachfolgenden Relationen und Metadaten[4]

H(<u>VorlNr</u>, MatrNr),

Metadaten	Größe
|H|	10.000
#(MatrNr, H)	2000
#(VorlNr, H)	300
|I(MatrNr, H)|	3
|I(VorlNr, H)|	3

S(<u>MatrNr</u>, Semester),

Metadaten	Größe
|S|	2000
#(MatrNr, S)	2000
#(Semester, S)	20
|I(MatrNr, S)|	3
|I(Semester, S)|	3

P(<u>ID</u>, Name),

Metadaten	Größe
|P|	100
#(ID, P)	100
#(Name, P)	75
|I(ID, P)|	2

V(<u>VorlNr</u>, gelesenVon)

Metadaten	Größe
|V|	300
#(VorlNr, V)	300
#(gelesenVon, V)	100
|I(VorlNr, V)|	2

Unser erweiterter Join-Ordering-Algorithmus erzeugt folgende Einträge:

[3] Dabei wird davon ausgegangen, dass sie entweder gegeben wird oder berechnet werden kann.

[4] Die Selektivität der gegebenen Selektionsbedingung berechnet sich durch den Kehrwert der Kardinalität des dazugehörigen Attributes; also $\mathsf{sel}(\mathsf{S.Semester} = '5') = \frac{1}{\#(\mathsf{Semester,S})} = \frac{1}{20}$.

Relation Set	cardinality	leftSide	cost	selectivity	jointype	selections
{H}	10.000	∅	10	1	Basisrelation	-
{P}	100	∅	10	1	Basisrelation	-
{S}	2.000	∅	200	1	Basisrelation	S.Semester =′ 5′
{V}	300	∅	30	1	Basisrelation	-
{σ(S)}	100	∅	106	1	Index-Selektion	-
{H, P}	1.000.000	{P}	21.012	1	Nested-Loop-Join	-
{H, V}	10.000	{V}	10.736	0.00333	Index-Index-Join	-
{H, S}	pruned	-	-	-	-	-
{H, σ(S)}	500	{σ(S)}	639	0.0005	Rel-Index-Join	-
{P, V}	300	{P}	349	0.01	Rel-Index-Join	-
{P, σ(S)}	10.000	{σ(S)}	1.318	1	Nested-Loop-Join	-
{P, S}	pruned	-	-	-	-	-
{S, V}	pruned	-	-	-	-	-
{V, σ(S)}	30.000	{σ(S)}	3.742	1	Nested-Loop-Join	-
{σ(H, S)}	pruned	-	-	-	-	-
{σ(P, S)}	pruned	-	-	-	-	-
{σ(S, V)}	pruned	-	-	-	-	-
{H, P, S}	pruned		-	-	-	-
{H, P, V}	10.000	{P, V}	10.780	0.00003	Rel-Index-Join	-
{H, P, σ(S)}	50.000	{H, σ(S)}	6.649	0.0005	Nested-Loop-Join	-
{H, S, V}	pruned		-	-	-	-
{H, V, σ(S)}	500	{H, σ(S)}	1.170	0	Rel-Index-Join	-
{H, σ(P, S)}	pruned	-	-	-	-	-
{H, σ(S, V)}	pruned	-	-	-	-	-
{P, S, V}	pruned	-	-	-	-	-
{P, σ(H, S)}	pruned	-	-	-	-	-
{P, σ(S, V)}	pruned	-	-	-	-	-
{P, V, σ(S)}	30.000	{P, V}	4.061	0.01	Nested-Loop-Join	-
{V, σ(H, S)}	pruned	-	-	-	-	-
{V, σ(P, S)}	pruned	-	-	-	-	-
{σ(H, P, S)}	pruned	-	-	-	-	-
{σ(H, S, V)}	pruned	-	-	-	-	-
{σ(P, S, V)}	pruned	-	-	-	-	-
{H, P, V, σ(S)}	500	{H, V, σ(S)}	1.701	0	Rel-Index-Join	-
{H, P, σ(S, V)}	pruned	-	-	-	-	-
{H, V, σ(P, S)}	pruned	-	-	-	-	-
{H, σ(P, S, V)}	pruned	-	-	-	-	-
{P, V, σ(H, S)}	pruned	-	-	-	-	-
{P, σ(H, S, V)}	pruned	-	-	-	-	-
{V, σ(H, P, S)}	pruned	-	-	-	-	-
{σ(H, P, S, V)}	pruned	-	-	-	-	-

Die mit „pruned“ bezeichneten Kardinalitäten geben an, welche Teilmengen nicht mehr weiter in die Berechnung einfließen. Natürlich liefert das bisherige Join-Ordering dasselbe Ergebnis. Dabei muss es aber 41 Einträge (entsprechend obiger geschachtelten Mengen) anstatt 16 erzeugen, um das Optimum zu finden. Das ist eine Ersparnis von $\frac{25}{41} \approx 61\%$ gegenüber dem Original-Algorithmus.

5 Erweiterungen und Ausblick

Ebenso wie die bedingten Termersetzungsregel für Selektionen, wird die entsprechende Ersetzungsregel für Projektionen ohne Duplikateliminierung zur Korrektur der Kosten eingesetzt. Dabei muss allerdings ein Kostenmodell eingesetzt werden, welches nicht nur die Anzahl von Tupeln, sondern auch deren Länge bzw. deren Speicherbedarf in Seiten betrachtet.

Aus diesem Grund genügt das einfache Kostenmodell aus [Van96] nicht mehr und muss durch ein anderes Modell ersetzt werden, welches Lesen und Schreiben von Seiten betrachtet (siehe dazu [War04]). Danach müssen drei Fälle untersucht werden:[5]

1. Das Einfügen einer Projektion verringert lokal sofort die Verbundkosten, da nicht mehr alle Attribute betrachtet werden müssen und somit mehr Tupel auf eine Seite passen.

2. Das frühe Einfügen einer Projektion senkt zwar nicht lokal die Kosten eines Teilbaumes des Ausführungsplanes (es kann ihn sogar verteuern), aber global werden die Kosten für weitere Verbunde so weit verringert, dass es sich insgesamt rentiert.

[5] Aus Platzgründen wird auf die komplette Herleitung einer bedingten Termersetzungsregel für die Projektion verzichtet. Vielmehr soll hier eine skizzenhafte Einführung geboten werden.

3. Das Einfügen ist in jedem Fall mit höheren Kosten verbunden und sollte nicht in Betracht gezogen werden.

Aus diesen Grund muss man das Einfügen einer Projektion in Beziehung zu den noch weiter im Ausführungsplan benötigten Attributen setzen:

τ	:=	originaler Ausführungsplan
$\overline{\tau}$	:=	optimierter Ausführungsplan
$\overline{\mathsf{attr}}(\tau)$	:=	noch benötigte Attribute von τ
$\mathsf{neededattr}(\mathsf{R})$	:=	liefert die im darüber liegenden Ausführungsplan noch ben. Attr.

M2-Rule

$$\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2 \rightarrow \pi_{\mathsf{l}_1}(\mathsf{R}_1) \bowtie_\varphi \pi_{\mathsf{l}_2}(\mathsf{R}_2),$$
$$\textbf{wenn } \mathsf{cost}(\tau) > \mathsf{cost}(\overline{\tau}) \textbf{ und}$$
$$\mathsf{l_i} := \mathsf{sch}(\mathsf{R_i}) \cap \mathsf{neededattr}(\mathsf{R_i}) \textbf{ mit } \mathsf{i} = 1, 2$$

Verglichen mit der L2-Rule kann diese Regel erst angewandt werden, wenn der gesamte Ausführungsplan generiert worden ist. Somit können Teilbäume nur optimiert werden, wenn man parallel zum Aufbau des Ausführungsplanes auch eine Menge von noch benötigten Attributen mitführt ($\mathsf{neededattr}(\mathsf{R})$).

Die Funktion $\mathsf{neededattr}(\mathsf{R})$ wird mit Hilfe einer simulierten Projektion berechnet, die man anfangs auf den kompletten Teilbaum anwendet. Dabei werden alle Attribute sowohl von noch ausstehenden Selektionen und Joins, als auch der abschließenden Projektion in einer Korrekturmenge zusammengefasst, welche die Projektionen auf die Teilbäume einschränkt bzw. erweitert ($\pi_{\mathsf{sch}(\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2) \cap \overline{\mathsf{attr}}(\tau) - \mathsf{sch}(\varphi)}(\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2)$).

Danach wird top-down das Schema genutzt, um die Rückgabewerte der Funktion $\mathsf{neededattr}(\mathsf{R})$ zu berechnen. Daraufhin wird die Anfangsprojektion herausgenommen und man erhält die einzelnen lokalen Projektionen. Abschließend berechnet man den Kostenvorteil bzw. Nachteil jeder einzelnen Projektion relativ zum gesamten Ausführungsplan. Hierfür muss man das Join-Ordering mit Hilfe der L2-Rule einmal anwenden, um daraus einen Kostenwert bzw. einen Ausführungsplan zu bekommen, mit dem man vergleichen kann.

Ein klares Ziel unserer Arbeit ist es, weitere Operatoren mit in die Optimierung einfließen zu lassen. Dazu gehören nicht nur reine Implementierungsarten, sondern auch zusammengesetzte relationale Operatoren wie der Semi- bzw. Antisemijoin. Er bietet bei allen Anfragen, bei denen eine Reduzierung einer Menge gefordert ist, ein immenses Optimierungspotential.

Durch das hier vorgestellte Konzept eines Termersetzungssystems zur Anfrageoptimierung sollte man den Antisemi- bzw. Semijoin als weiteren Operator mit eigens dazu gehörenden Kosten und Regeln hinzufügen können. Schwierig zu beantworten ist dabei nur die Frage, wie sich dieser Operator anwenden lässt. Dabei müssen teilweise Projektionen zusätzlich eingefügt und somit betrachtet werden. Klar ist auch, dass die zu betrachtenden Zwischenmengen wieder größer werden und wiederum durch bedingte Termersetzungen reduziert werden müssen.

Die in diesem Artikel vorgestellten bedingten Termersetzungen wurden im Zuge von Diplom- ([Mak03]), BSc- ([War04],[Die04]) und Studienarbeiten ([Zle05]) hinsichtlich eines Simulators für Anfrageoptimierung implementiert. Näheres zum Programm `RELOpt` kann man unter `www.dbs.uni-hannover.de/~makoui/-RELOpt.html` erfahren.

Literatur

[Van96] **Bennet Vance / David Maier:** Rapid Bushy Join-order Optimization with Cartesian Products, p. 35-46, SIGMOD 1996

[Ste97] **Michael Steinbrunn / Guido Moerkotte / Alfons Kemper:** Heuristic and randomized optimization for the join ordering problem, VLDB Journal, p. 191-208, 1997

[Sch97] **Wolfgang Scheufele / Guido Moerkotte:** On the Complexity of Generating Optimal Plans with Cross Products, PODS, p. 238-248, 1997

[Sch98] **Wolfgang Scheufele / Guido Moerkotte:** Efficient Dynamic Programming Algorithms for Ordering Expensive Joins and Selections, EDBT, p. 201-215, 1998

[Mak03] **Mazeyar E. Makoui:** Heuristische Anfrageoptimierung in Relationalen Datenbanken, Diplomarbeit, Institut für Informationssysteme, Universität Hannover, 2003

[War04] **Hendrik Warneke:** Erweiterung eines Simulators für relationale Anfrageoptimierungen, Bachelorarbeit, Institut für Informationssysteme, Universität Hannover, 2003

[Die04] **Moritz Diehle:** Erweiterung eines relationalen Anfragesimulators um eine regelbasierte Steuerung von physischen Optimierungsregeln, Bachelorarbeit, Institut für Informationssysteme, Universität Hannover, 2004

[Kem04] **Alfons Kemper / Andre Eickler:** Datenbankssysteme - Eine Einführung, Oldenbourg Verlag, 5. Auflage, 2004

[Zle05] **Alexander Zlenko:** Implementierung von Termersetzungsregeln zur regelgesteuerten Anfrageoptimierung in relationalen Datenbanken, Studienarbeit, Institut für Informationssysteme, Universität Hannover, 2005