

# Abfrageverarbeitung und Optimierung

Udo Kelter

04.12.2013

## **Zusammenfassung dieses Lehrmoduls**

Ein wesentlicher Vorteil relationaler Datenbanken liegt darin, daß die Abarbeitung von Abfragen automatisch optimiert werden kann. Zunächst wird im Rahmen der algebraischen Optimierung die Abfrage anhand von Heuristiken in eine äquivalente, aber effizienter ausführbare Form umgewandelt. In der anschließenden internen Optimierung wird zwischen ggf. verfügbaren Implementierungen der Elementaroperationen entschieden und damit zusammenhängend über die Ausnutzung von Indexen. Die alternativen Ausführungspläne werden hierzu anhand ihrer geschätzten Ausführungskosten bewertet.

## **Vorausgesetzte Lehrmodule:**

- obligatorisch: - Das relationale Datenbankmodell
- Implementierung relationaler Operationen
- empfohlen: - Architektur von DBMS

**Stoffumfang in Vorlesungsdoppelstunden:** 1.2

# Inhaltsverzeichnis

<b>1 Motivation</b>	<b>3</b>
1.1 Grob Ablauf einer Abfrageverarbeitung . . . . .	3
1.2 Optimierungsansätze . . . . .	4
<b>2 Optimierungskriterien</b>	<b>7</b>
<b>3 Algebraische Optimierung</b>	<b>8</b>
3.1 Äquivalente algebraische Ausdrücke . . . . .	8
3.2 Optimierungsheuristiken . . . . .	12
3.3 Optimierungsalgorithmen . . . . .	13
<b>4 Kostenschätzung</b>	<b>16</b>
Literatur . . . . .	20
Glossar . . . . .	20
Index . . . . .	20

# 1 Motivation

Praktisch benutzte Abfragesprachen und selbst die relationale Algebra sind Hochsprachen im gleichen Sinne wie “höhere” Programmiersprachen: sie entlasten den Nutzer von vielen systemnahen, lästigen Details der Datenverwaltung und erlauben es, Probleme in der Denkwelt abstrakterer Konzepte zu lösen, im Falle von Abfragesprachen in der Denkwelt des Datenbankmodells.

Die Operationen des Datenbankmodells müssen auf systemnahe Funktionen zurückgeführt werden, insb. müssen die Datenobjekte, mit denen die Operationen des Datenbankmodells arbeiten (einzelne Tupel bzw. Tupelmengen) letztlich auf Strukturen der Speichermedien (i.d.R. Magnetplatten) abgebildet werden. Diese Abbildung stellt man sich softwaretechnisch am besten als eine Schichtenarchitektur vor. Die oberste Schicht exportiert Operationen mit Tupelmengen und basiert auf der darunterliegenden Schicht, die Operationen mit einzelnen Tupeln exportiert<sup>1</sup>.

Wir beziehen uns hier begrifflich auf relationale Systeme, bei objektorientierten Systemen gelten diese Betrachtungen analog für Objekte. Navigierende Datenmodelle haben i.d.R. keine mengenwertigen Operationen, so daß die oberste Schicht entfällt.

Wir konzentrieren uns in diesem Lehrmodul auf die Frage, wie Operationen mit Tupelmengen, insb. also Abfragen, auf Operationen mit einzelnen Tupeln zurückgeführt werden können.

## 1.1 Grobablauf einer Abfrageverarbeitung

Den Ablauf der Bearbeitung einer Abfrage kann man wie folgt grob gliedern:

1. Umformung der textuellen Darstellung der Abfrage in eine interne Darstellung (analog zum Parser in einem Compiler) und Korrektheitsüberprüfung
2. Erstellung und Vergleich möglicher Ausführungspläne

---

<sup>1</sup>[DBSA] beschreibt die Schichten detaillierter.

### 3. Ausführung des “optimalen” Plans

Die Korrektheitsüberprüfung im ersten Schritt betrifft

- die Syntax und
- die verwendeten Ressourcen; mit Hilfe des Systemkatalogs wird festgestellt, ob die angegebenen Relationen und Attribute überhaupt existieren und ob entsprechende Zugriffsrechte gegeben sind.

Weiter werden in diesem Schritt virtuelle Relationen durch ihre definierende Abfrage ersetzt (*view resolution*). Im Endeffekt wird die Abfrage in eine interne Baumdarstellung umgeformt, die einen Ausdruck in der relationalen Algebra oder einer Erweiterung derselben repräsentiert.

Der zweite Schritt, die Optimierung, ist dadurch motiviert, daß es meist zu einer Abfrage mehrere Ausführungsvarianten gibt, die alle das gleiche Resultat liefern, aber verschiedene Kosten verursachen.

## 1.2 Optimierungsansätze

Die Frage, wo und wie die Abarbeitung einer Abfrage optimiert werden kann, kann offensichtlich nicht ohne Bezug auf die Abfragesprache diskutiert werden. Wir beschränken uns hier auf die relationale Algebra (s. [RDBM]), da sie einerseits die wichtigsten mengenwertigen Abfrageoperationen umfaßt, andererseits syntaktisch sehr einfach strukturiert ist. Bei komplexeren Sprachen steigt vor allem die Zahl der Abfrageoperationen an, die prinzipielle Vorgehensweise bleibt aber gleich.

Die Abarbeitung einer Abfrage können wir für unsere aktuelle Fragestellung in zwei Themenkomplexe gliedern:

- die Implementierung der Operationen der relationalen Algebra
- die Verarbeitung geschachtelter Ausdrücke.

Die beiden Themenkomplexe sind nicht ganz unabhängig voneinander, aber hilfreich für die Gliederung möglicher Optimierungsmaßnahmen.

**Optimierung der Elementaroperationen.** Wie schon in Abschnitt 3.9 in [RDBM] erwähnt, haben Ausdrücke in der relationalen Algebra eine intuitive operationale Semantik, nämlich die Auswertung von innen nach außen. Dies unterstellt, daß für jede Operation der relationalen Algebra genau eine Implementierung vorhanden ist und daß bei der Abarbeitung des Ausdrucks diese Implementierung mit den passenden Zwischenresultaten ausgeführt wird. Diese Annahme trifft aber in doppelter Hinsicht nicht zu:

- Für eine Operation kann es mehrere Implementierungen geben, die verschieden effizient sind und von denen eine auszuwählen ist, ggf. abhängig von den benutzten Datenstrukturen.
- Es kann Implementierungen kompletter Teilausdrücke geben. Als Beispiel betrachten wir eine Selektion gefolgt von einer Projektion ( $\pi_A(\sigma_B(\dots))$ ): bei der intuitiven Abarbeitung würde man bei der Selektion ganze Tupel in eine temporäre Relation schreiben und diese dann projizieren. Stattdessen erzeugt man besser gleich die projizierten Tupel und spart so den Platz und Aufwand für die temporäre Relation komplett ein.

Alternative Implementierungen einzelner Operationen und Implementierungen kompletter Teilausdrücke zählen wir hier zu den Optimierungsmaßnahmen im Bereich der Implementierung von Operationen.

**Optimierung der Abarbeitung von Ausdrücken.** Die Optimierung der Abarbeitung von geschachtelten Ausdrücken sei durch folgendes Beispiel motiviert: wir unterstellen wieder die Relationen **kunden** und **lieferungen** und suchen die Namen der Kunden, die am 24.12.2000 eine Lieferung bekommen haben. Diese Frage können wir auf zwei verschiedene Arten beantworten:

$$\pi_{\text{Kundenname}}(\sigma_{\text{Datum}=24.12.2000}(\text{lieferungen} \bowtie \text{kunden}))$$

und

$$\pi_{\text{Kundenname}}(\sigma_{\text{Datum}=24.12.2000}(\text{lieferungen}) \bowtie \text{kunden})$$

Beide Ausdrücke sind **äquivalent**, berechnen also das gleiche Ergebnis, der zweite ist aber effizienter, weil der Verbund mit einer kleineren

Menge gebildet wird. Hierbei unterstellen wir wieder die intuitive operationale Semantik von Ausdrücken; wenn wir von einem **effizienteren** Ausdruck reden, sehen wir den Ausdruck als einen (vereinfachten, abstrakten) Ausführungsplan an.

Daß es zu einer Aufgabe mehrere verschiedene, aber äquivalente Lösungen gibt, trifft generell auf alle Hochsprachen zu und auf SQL sogar in deutlich stärkerem Ausmaß als auf die relationale Algebra.

Ein Gegenstand der Optimierung ist somit die Auswahl einer möglichst kostengünstigen Lösung. Hierbei wird in zwei Schritten vorgegangen:

1. Bei der **algebraischen Optimierung** werden zu dem vom Nutzer vorgegebenen Ausdruck ein oder mehrere äquivalente Ausdrücke gebildet, die aufgrund von Heuristiken effizienter ausführbar erscheinen. Ein Beispiel hierfür wurde oben gegeben.

Die algebraische Optimierung ist im Gegensatz zu den weiteren Maßnahmen unabhängig vom aktuellen Inhalt der Datenbank.

2. Bei der anschließenden **internen Optimierung** werden für jeden Ausdruck mehrere alternative Ausführungspläne gebildet. In einem **Ausführungsplan** wird über die reine Operationsschachtelung hinaus festgelegt,
  - welche der alternativ verfügbaren Implementierungen einer Operation gewählt wird; da für manche Implementierungen ein Index vorhanden sein muß, wird auch entschieden, ob Indexe ausgenutzt werden;
  - welche Zwischenergebnisse auf Platte oder im Hauptspeicher angelegt werden.

Für jeden Ausführungsplan werden die Ausführungskosten geschätzt. Bei dieser Schätzung spielen eine Rolle:

- Merkmale des aktuellen Datenbestands, insb. die Größe von Relationen und die Verteilung von Attributwerten
- interne Speicherungsstrukturen, z.B. vorhandene Indexe oder Sortierungen

Ferner werden gemeinsame Unterausdrücke berücksichtigt, die nur einmal berechnet werden müssen.

Anzumerken ist noch, daß die Bezeichnung Optimierung insofern mißverständlich (oder gar falsch) ist, als i.d.R. nicht etwa eine optimale, also minimal aufwendige Bearbeitung der Abfrage gesucht wird, sondern nur eine möglichst gute bzw. bessere als die intuitiv naheliegende. Zum einen muß natürlich der Aufwand für die Optimierung kleiner bleiben als der Gewinn durch die bessere Ausführungsvariante. Bei sehr kleinen Datenbanken lohnt sich deshalb die Optimierung oft nicht. Sodann gibt es i.a. zu viele Ausführungsvarianten, um alle einzeln betrachten zu können, und man kann i.a. die Kosten einer Variante nicht exakt prognostizieren, man muß also mit Heuristiken arbeiten.

## 2 Optimierungskriterien

Bisher haben wir die Frage offengelassen, anhand welcher Kriterien wir bestimmen, ob eine Abfrageverarbeitung gut oder weniger gut ist. Im Endeffekt geht es um Performance-Maße wie Antwortzeit oder Durchsatz, die aus Benutzersicht erkennbar sind. Diese Größen können als solche aber nicht direkt geschätzt werden, man muß also auf andere Meßgrößen bzw. Optimierungskriterien zurückgreifen. Aufwand entsteht insb. in folgender Hinsicht:

- Plattenzugriffe zum Lesen der Ausgangsdaten sowie zum Schreiben und Lesen von Zwischenergebnissen
- Plattenplatzbedarf für Zwischenergebnisse
- Hauptspeicherbedarf
- bei verteilten Systemen: Anzahl der Kommunikationen, Datenübertragungsvolumen
- Rechenaufwand (CPU-Belastung)

Ferner gibt es speziellere, ggf. sogar anwendungsspezifische Aufwandskriterien.

Das wichtigste Kriterium ist in der Regel die Zahl der Plattenzugriffe. Daher stellen viele Optimierungsverfahren vor allem auf die Reduktion

der Plattenzugriffe ab. Die CPU-Belastung ist aber auch keineswegs unwichtig; bei den heute üblichen großen Hauptspeichern können oft alle relevanten Daten gepuffert werden, so daß nach dem unvermeidlichen initialen Lesen und ggf. Schreiben der Endergebnisse keine weiteren Plattenzugriffe anfallen.

Es liegt daher nahe, mehrere dieser Meßgrößen mit geeigneten Faktoren zu gewichten und daraus ein **Kostenmaß** für Abfrageverarbeitungen zu bilden. In diesem Zusammenhang spricht man auch von **kostenbasierter** Optimierung.

## 3 Algebraische Optimierung

### 3.1 Äquivalente algebraische Ausdrücke

Ziel der algebraischen Optimierung ist es, zu einem vorgegebenen Ausdruck einen äquivalenten Ausdruck zu finden, der effizienter ausführbar ist.

Zwei relationale Ausdrücke  $A_1$  und  $A_2$  sind **äquivalent**, wenn bei einem beliebigen Datenbankinhalt beide Ausdrücke das gleiche Ergebnis liefern. Für die algebraische Optimierung ist daher der Begriff Äquivalenz von zentraler Bedeutung.

Im folgenden listen wir eine Reihe von Äquivalenzen auf, die mehr oder minder unmittelbar durch Einsetzen der Definitionen beweisbar sind und die auch als Gesetze bezeichnet werden.

$r$ ,  $s$  und  $t$  sind i.f. beliebige Relationen; bei den Mengenoperationen müssen die beteiligten Relationen den gleichen Typ haben.  $B$ ,  $B_1$  und  $B_2$  sind Selektionsbedingungen, die syntaktisch zu den Typen der Argumentrelationen passen,  $Q$ ,  $Q_1$  und  $Q_2$  sind Verbundbedingungen.

**Kommutativgesetze.** Vereinigung, Schnitt, Verbund und Kreuzprodukt sind kommutativ:

$$\begin{aligned} r \cup s &= s \cup r \\ r \cap s &= s \cap r \\ r \bowtie s &= s \bowtie r \\ r \bowtie_Q s &= s \bowtie_Q r \end{aligned}$$

$$\mathbf{r} \times \mathbf{s} = \mathbf{s} \times \mathbf{r}$$

**Assoziativgesetze.** Vereinigung, Schnitt, Verbund und Kreuzprodukt sind assoziativ:

$$\begin{aligned} \mathbf{r} \cup (\mathbf{s} \cup \mathbf{t}) &= (\mathbf{r} \cup \mathbf{s}) \cup \mathbf{t} \\ \mathbf{r} \cap (\mathbf{s} \cap \mathbf{t}) &= (\mathbf{r} \cap \mathbf{s}) \cap \mathbf{t} \\ \mathbf{r} \bowtie (\mathbf{s} \bowtie \mathbf{t}) &= (\mathbf{r} \bowtie \mathbf{s}) \bowtie \mathbf{t} \\ \mathbf{r} \bowtie_{Q_1} (\mathbf{s} \bowtie_{Q_2} \mathbf{t}) &= (\mathbf{r} \bowtie_{Q_1} \mathbf{s}) \bowtie_{Q_2} \mathbf{t} \\ \mathbf{r} \times (\mathbf{s} \times \mathbf{t}) &= (\mathbf{r} \times \mathbf{s}) \times \mathbf{t} \end{aligned}$$

**Auflösung komplexer Selektionsbedingungen.** Die Booleschen Operatoren  $\wedge$  und  $\vee$  zwischen Bedingungen  $B_1$  und  $B_2$  können wie folgt in Schachtelungen von Selektionen oder Mengenoperationen umgeformt werden:

$$\begin{aligned} \sigma_{B_1 \wedge B_2}(\mathbf{r}) &= \sigma_{B_1}(\sigma_{B_2}(\mathbf{r})) \\ \sigma_{B_1 \wedge B_2}(\mathbf{r}) &= \sigma_{B_2}(\sigma_{B_1}(\mathbf{r})) \\ \sigma_{B_1 \wedge B_2}(\mathbf{r}) &= \sigma_{B_1}(\mathbf{r}) \cap \sigma_{B_2}(\mathbf{r}) \\ \sigma_{B_1 \vee B_2}(\mathbf{r}) &= \sigma_{B_1}(\mathbf{r}) \cup \sigma_{B_2}(\mathbf{r}) \end{aligned}$$

**Distributivgesetze zwischen Selektion und Mengenoperatoren.**

$$\begin{aligned} \sigma_B(\mathbf{r} \cup \mathbf{s}) &= \sigma_B(\mathbf{r}) \cup \sigma_B(\mathbf{s}) \\ \sigma_B(\mathbf{r} \cap \mathbf{s}) &= \sigma_B(\mathbf{r}) \cap \sigma_B(\mathbf{s}) \\ \sigma_B(\mathbf{r} - \mathbf{s}) &= \sigma_B(\mathbf{r}) - \sigma_B(\mathbf{s}) = \sigma_B(\mathbf{r}) - \mathbf{s} \end{aligned}$$

**Gesetze für die Projektion.**

Seien  $U \subseteq V \subseteq R$ . Dann gilt

$$\pi_U(\pi_V(\mathbf{r})) = \pi_{U \cap V}(\mathbf{r}).$$

Eine Projektion kann mit einer Selektion vertauscht werden, wenn die Selektionsbedingung  $B$  nur Attribute aus  $V$  (die Menge der Attribute, auf die projiziert wird) enthält:

$$\sigma_B(\pi_V(\mathbf{r})) = \pi_V(\sigma_B(\mathbf{r}))$$

Eine Projektion kann mit dem Vereinigungsoperator vertauscht werden<sup>2</sup>:

$$\pi_Y(\mathbf{r} \cup \mathbf{s}) = \pi_Y(\mathbf{r}) \cup \pi_Y(\mathbf{s})$$

**Verbund und Selektion.** Seien  $\mathbf{r1}$  bzw.  $\mathbf{r2}$  Relationen des Typs  $\mathbf{R1}$  bzw.  $\mathbf{R2}$ ,  $B$  eine Selektionsbedingung, die nur Attribute aus  $\mathbf{R1}$  beinhaltet,  $Q$  eine Verbundbedingung. Dann gilt:

$$\begin{aligned}\sigma_B(\mathbf{r1} \bowtie \mathbf{r2}) &= \sigma_B(\mathbf{r1}) \bowtie \mathbf{r2} \\ \sigma_B(\mathbf{r1} \bowtie_Q \mathbf{r2}) &= \sigma_B(\mathbf{r1}) \bowtie_Q \mathbf{r2}\end{aligned}$$

Als Erläuterung hierzu betrachten wir folgende Beispiele; sei  $\mathbf{R1} = \{A, B, C, D\}$ ,  $\mathbf{R2} = \{C, D, E, F\}$ :

$$\begin{aligned}\sigma_{A=B}(\mathbf{r1} \bowtie \mathbf{r2}) &= \sigma_{A=B}(\mathbf{r1}) \bowtie \mathbf{r2} \\ \sigma_{A=E}(\mathbf{r1} \bowtie \mathbf{r2}) &\neq \sigma_{A=E}(\mathbf{r1}) \bowtie \mathbf{r2} \\ \sigma_{A=a}(\mathbf{r1} \bowtie_{A=F} \mathbf{r2}) &= \sigma_{A=a}(\mathbf{r1}) \bowtie_{A=F} \mathbf{r2} \\ \sigma_{A=F}(\mathbf{r1} \bowtie_{B=E} \mathbf{r2}) &\neq \sigma_{A=F}(\mathbf{r1}) \bowtie_{B=E} \mathbf{r2}\end{aligned}$$

Wenn die Attribute, die in einer Selektionsbedingung  $sel$  auftreten, in beiden Relationenschemata enthalten sind, dann gilt:

$$\sigma_{sel}(\mathbf{r1} \bowtie \mathbf{r2}) = \sigma_{sel}(\mathbf{r1}) \bowtie \sigma_{sel}(\mathbf{r2})$$

**Kreuzprodukt und Selektion.** Seien  $\mathbf{r1}$  bzw.  $\mathbf{r2}$  Relationen des Typs  $\mathbf{R1}$  bzw.  $\mathbf{R2}$ ,  $B$  eine Selektionsbedingung, die nur Attribute aus  $\mathbf{R1}$  beinhaltet. Dann gilt:

$$\sigma_B(\mathbf{r1} \times \mathbf{r2}) = \sigma_B(\mathbf{r1}) \times \mathbf{r2}$$

Ist  $B$  eine Verbundbedingung, so gilt:

$$\sigma_B(\mathbf{r1} \times \mathbf{r2}) = \mathbf{r1} \bowtie_B \mathbf{r2}$$

**Verbund und Projektion.** Nach einem Verbund wird meist sofort oder später (z.B. nach einer Selektion) projiziert, d.h. manche der

---

<sup>2</sup>Für den Durchschnitt und die Differenz gilt dies nicht. Übungsaufgabe: Finden Sie ein Gegenbeispiel.

zunächst aufwendig erzeugten Attribute werden später unbenutzt gelöscht. Die Frage ist, ob man dies nicht vermeiden kann und ob man vorher projizieren kann. Wie das nächste Beispiel ( $R1$  und  $R2$  seien wie vorstehend definiert) zeigt, ist die Vertauschung aber i.a. nicht zulässig:

$$\pi_A(\mathbf{r1} \bowtie \mathbf{r2}) \neq (\pi_A(\mathbf{r1})) \bowtie \mathbf{r2}$$

Der Fehler liegt darin, daß  $\{C, D\} = R1 \cap R2$  die Verbundattribute von  $\mathbf{r1}$  und  $\mathbf{r2}$  sind und daß diese Verbundattribute im rechten Teil der Formel bei  $\mathbf{r1}$  wegprojiziert worden sind, der dortige Verbund also zum Kreuzprodukt degeneriert; auf der rechten Seite werden also i.a. mehr Tupel entstehen. Wenn bei der Projektion die Verbundattribute erhalten bleiben, kann das Problem nicht auftreten. Anders gesagt müssen dann, wenn man eine Projektion an einem Verbund vorbei nach innen ziehen will, die Verbundattribute innen erhalten bleiben. Nach dem Verbund müssen sie dann separat entfernt werden.

Diese Vorüberlegungen motivieren die folgende Äquivalenz. Seien  $\mathbf{r1}$  bzw.  $\mathbf{r2}$  Relationen des Typs  $R1$  bzw.  $R2$ ,  $U \subseteq R1 \cup R2$ ,  $Q$  eine Verbundbedingung,  $V$  die in  $Q$  auftretenden Verbundattribute. Seien  $U1 = (U \cup V) \cap R1$  und  $U2 = (U \cup V) \cap R2$ . Dann ist

$$\pi_U(\mathbf{r1} \bowtie_Q \mathbf{r2}) = \pi_U(\pi_{U1}(\mathbf{r1}) \bowtie_Q \pi_{U2}(\mathbf{r2}))$$

Beim natürlichen Verbund ist analog  $V = R1 \cap R2$ .

Man kann also vor der Verbundbildung alle Attribute entfernen, die nicht für die Verbundbildung oder die außenstehende Projektion benötigt werden.

Anzumerken ist hier, daß, sofern  $\mathbf{r1}$  und  $\mathbf{r2}$  nicht im Hauptspeicher gepuffert werden können, der längere Ausdruck effizienter ausführbar sein wird, obwohl er mehr Operationen enthält.

**Kreuzprodukt und Projektion.** Das Kreuzprodukt können wir hier als Sonderfall des Verbunds mit leerer Verbundbedingung behandeln; die Menge der Verbundattribute ist somit leer und die allgemeine Formel für den Verbund vereinfacht sich zu:

$$\begin{aligned} \pi_U(\mathbf{r1} \times \mathbf{r2}) &= \pi_U(\pi_{R1 \cap U}(\mathbf{r1}) \times \pi_{R2 \cap U}(\mathbf{r2})) \\ &= \pi_{R1 \cap U}(\mathbf{r1}) \times \pi_{R2 \cap U}(\mathbf{r2}) \end{aligned}$$

## 3.2 Optimierungsheuristiken

Im vorigen Abschnitt ist eine größere Anzahl Äquivalenzen vorgestellt worden. Im Rahmen der algebraischen Optimierung können diese jetzt ausgenutzt werden, um zu einem vorgegebenen Ausdruck einen äquivalenten, aber effizienter ausführbaren zu finden.

Bei einem gegebenen Ausdruck werden normalerweise mehrere der Äquivalenzen anwendbar sein. Nach einem Umformungsschritt wird dies erneut der Fall sein, wir erhalten somit i.a. viele denkbare Sequenzen von Umformungsschritten. Selbst dann, wenn wir unsinnige Sequenzen – z.B. solche, in denen direkt nach einer Umformung die inverse Umformung auftritt – aussortieren, bleibt die Zahl der Sequenzen i.a. viel zu hoch, um alle parallel verfolgen zu können.

Die nachfolgenden, als Regeln formulierten Optimierungsheuristiken beschreiben solche Umformungsschritte, die mit sehr hoher Wahrscheinlichkeit und meist auch in intuitiv unmittelbar einsichtiger Weise die Effizienz verbessern. Dabei wird zum einen die “Richtung”, in der die Äquivalenzen ausgenutzt werden sollten, vorgegeben, ferner werden teilweise Bedingungen für einen Umformungsschritt angegeben.

**Regel 1:** Selektionen sollten so früh wie möglich ausgeführt werden.

Selektionen werden also möglichst weit nach innen in geschachtelte Ausdrücke verschoben. Hierdurch werden Zwischenergebnisse kleiner. Selbst wenn diese nicht gespeichert werden, sinkt immer noch der Verarbeitungsaufwand.

**Regel 2:** Eine äußere Selektion, deren Bedingung eine Konjunktion ist, sollte in eine Schachtelung von Selektionen aufgebrochen werden.

Der Sinn dieser Regel liegt darin, daß die Regel 1 dann öfter anwendbar ist und möglichst viele elementare Selektionen unmittelbar bei den Basisrelationen angewandt werden.

Sofern mehrere Einzelselektionen in der Form  $\sigma_{B_1}(\sigma_{B_2}(X))$  unmittelbar aufeinanderfolgen, sollte die Selektion mit der höheren Selektivität nach innen verschoben werden. Sofern  $X$  bereits eine Basisrelation ist, sollten vorhandene Indexe ausgenutzt werden (vgl. Abschnitt 4 in [IRO]). In beiden Fällen beziehen wir uns

auf Merkmale der aktuellen Datenbank, derartige Entscheidungen gehören also nicht mehr strikt zur algebraischen Optimierung, sondern schon zur internen Optimierung.

**Regel 3:** Selektionen und Kreuzprodukte sollten zu Verbunden zusammengefaßt werden, d.h. Tupel, die die Verbundbedingung nicht erfüllen, werden gar nicht erst erzeugt.

**Regel 4:** Bei einem Verbund von 3 und mehr Relationen sollten Verbunde zuerst berechnet werden, die voraussichtlich kleine Ergebnisse erzeugen. Kleine Ergebnisse sind wahrscheinlich, wenn

- die Verbundattribute Identifizierungsschlüssel in einer der beteiligten Relationen sind
- mehrere Verbundattribute statt nur eines vorhanden sind
- die beteiligten Relationen klein sind<sup>3</sup>

**Regel 5:** Sofern sinnvoll (z.B. wenn Verbundergebnisse zwischengespeichert werden müssen) sollten zusätzliche Projektionen eingefügt werden.

Die neuen Projektionen sollten natürlich mit einer ggf. vorhandenen vorhergehenden (also inneren) Operation zusammengefaßt werden (vgl. generelle Bemerkungen zu Projektionen in Abschnitt 6 in [IRO]).

**Regel 6:** Sofern an irgendeiner Stelle im Syntaxbaum Zwischenergebnisse gespeichert werden müssen, sollten alle äußeren Projektionen, soweit möglich, bis an diese Stelle verschoben werden.

### 3.3 Optimierungsalgorithmen

Die vorstehenden Heuristiken sind noch kein Algorithmus – sie geben immerhin an, in welcher “Richtung” die Äquivalenzen aus Abschnitt 3.1 ausgenutzt werden sollten. Nach wie vor können in einer bestimmten

---

<sup>3</sup>Auch dieses Argument basiert auf Merkmalen der aktuellen Datenbank, gehört also nicht mehr strikt zur algebraischen, sondern zur internen Optimierung.

Situation mehrere Äquivalenzen anwendbar sein. Ein **Optimierungsalgorithmus** legt fest,

- welche Äquivalenz an welcher Stelle als nächstes ausgenutzt werden soll;
- wann das Verfahren abgebrochen wird; hierbei ist natürlich die algebraische und interne Optimierung insgesamt zu betrachten. Der zusätzliche Grenznutzen weiterer Optimierungsmaßnahmen kann nach den ersten, offensichtlichen Maßnahmen unsicher sein.

Wegen der kombinatorischen Explosion der möglichen Einzelmaßnahmen kann der Aufwand für die Optimierung selbst ausufern; auch hier müssen Heuristiken eingesetzt werden, um die Untersuchung wenig aussichtsreicher Maßnahmen früh abbrechen zu können.

Ein einfacher Optimierungsalgorithmus könnte folgendermaßen vorgehen:

1. zerlegen von Selektionen, deren Bedingung eine Konjunktion ist, in geschachtelte Selektionen (Regel 2)
2. verschieben aller Selektionen so weit wie möglich nach innen (Regel 1)
3. zusammenfassen von Selektionen und Kreuzprodukten zu Verbunden (Regel 3)
4. vertauschen der Reihenfolge von Verbunden gem. Größe der Relationen (Regel 4)
5. ggf. einfügen von zusätzlichen Projektionen bei Verbunden (Regel 5)
6. verkleinern von zu puffernden Zwischenergebnissen, indem Projektionen nach innen verschoben werden (Regel 6)

Wenn eine Regel an mehreren Stellen innerhalb des Ausdrucks anwendbar ist, wird sie jeweils an der ersten gefundenen Stelle zuerst angewandt.

**Beispiel für eine Optimierung.** Als Beispiel betrachten wir folgende SQL-Abfrage und ihre Optimierung:

```

SELECT DISTINCT r.A, s.B
FROM   R r, S s, T t
WHERE  r.C = t.C AND s.D = t.D AND r.E = e

```

Nach der obligaten Syntaxprüfung wird diese Abfrage nach dem Standardschema zunächst in folgenden Ausdruck übersetzt:

$$\pi_{r.A,s.B}(\sigma_{r.C=t.C \wedge s.D=t.D \wedge r.E=e}(\mathbf{r} \times (\mathbf{s} \times \mathbf{t})))$$

Nach Schritt 1 (Anwendung von Regel 2) erhalten wir:

$$\pi_{r.A,s.B}(\sigma_{r.C=t.C}(\sigma_{s.D=t.D}(\sigma_{r.E=e}(\mathbf{r} \times (\mathbf{s} \times \mathbf{t}))))))$$

Unter Anwendung von Regel 1 verschieben wir die Selektionen  $\sigma_{r.E=e}$  und  $\sigma_{s.D=t.D}$  nach innen in das Kreuzprodukt:

$$\pi_{r.A,s.B}(\sigma_{r.C=t.C}(\sigma_{s.D=t.D}(\sigma_{r.E=e}(\mathbf{r}) \times (\mathbf{s} \times \mathbf{t}))))$$

$$\pi_{r.A,s.B}(\sigma_{r.C=t.C}(\sigma_{r.E=e}(\mathbf{r}) \times \sigma_{s.D=t.D}(\mathbf{s} \times \mathbf{t}))))$$

Unter Anwendung von Regel 3 bilden wir jeweils aus einer Selektion und einem Kreuzprodukt einen Verbund:

$$\pi_{r.A,s.B}(\sigma_{r.E=e}(\mathbf{r}) \bowtie_{r.C=t.C}(\sigma_{s.D=t.D}(\mathbf{s} \times \mathbf{t})))$$

$$\pi_{r.A,s.B}(\sigma_{r.E=e}(\mathbf{r}) \bowtie_{r.C=t.C}(\mathbf{s} \bowtie_{s.D=t.D} \mathbf{t}))$$

Als nächstes stellt sich die Frage, in welcher Reihenfolge die beiden Verbunde ausgeführt werden sollen<sup>4</sup>. Sofern keine weitere Information über die Größe der Relationen verfügbar ist, können wir wegen der Selektion davon ausgehen, daß die Eingaberelation  $\sigma_{r.E=e}(\mathbf{r})$  die kleinste ist; daher sollte zunächst mit ihr ein Verbund berechnet werden. Da sich die Verbundbedingung, in der  $\mathbf{r}$  vorkommt, auf  $\mathbf{t}$  als zweite Relation bezieht, sollte der erste Verbund zwischen  $\mathbf{r}$  und  $\mathbf{t}$  gebildet werden. Wir stellen unseren Ausdruck also wie folgt um:

$$\pi_{r.A,s.B}(((\sigma_{r.E=e}(\mathbf{r}) \bowtie_{r.C=t.C} \mathbf{t}) \bowtie_{s.D=t.D} \mathbf{s}))$$

Zum Schluß fügen wir noch Projektionen ein, um nicht im Endergebnis benötigte Attribute vor den Verbundbildungen zu entfernen:

---

<sup>4</sup>Dies unter der Annahme, daß kein 3-Wege-Verbund gewählt wird, was hier vermutlich die effizienteste Lösung wäre, aber um des Beispiels willen hier nicht weiter verfolgt wird.

$$\pi_{\mathbf{r},\mathbf{A},\mathbf{s},\mathbf{B}} \left( \pi_{\mathbf{r},\mathbf{A},\mathbf{t},\mathbf{D}} \left( \sigma_{\mathbf{r},\mathbf{E}=e} (\mathbf{r}) \bowtie_{\mathbf{r},\mathbf{C}=\mathbf{t},\mathbf{C}} \mathbf{t} \right) \right. \\ \left. \bowtie_{\mathbf{s},\mathbf{D}=\mathbf{t},\mathbf{D}} \right. \\ \left. \pi_{\mathbf{s},\mathbf{B},\mathbf{s},\mathbf{D}} (\mathbf{s}) \right)$$

und

$$\pi_{\mathbf{r},\mathbf{A},\mathbf{s},\mathbf{B}} \left( \pi_{\mathbf{r},\mathbf{A},\mathbf{t},\mathbf{D}} \left( \pi_{\mathbf{r},\mathbf{A},\mathbf{r},\mathbf{C}} \left( \sigma_{\mathbf{r},\mathbf{E}=e} (\mathbf{r}) \right) \right. \right. \\ \left. \left. \bowtie_{\mathbf{r},\mathbf{C}=\mathbf{t},\mathbf{C}} \right. \right. \\ \left. \left. \pi_{\mathbf{t},\mathbf{D},\mathbf{t},\mathbf{C}} (\mathbf{t}) \right) \right. \\ \left. \bowtie_{\mathbf{s},\mathbf{D}=\mathbf{t},\mathbf{D}} \right. \\ \left. \pi_{\mathbf{s},\mathbf{B},\mathbf{s},\mathbf{D}} (\mathbf{s}) \right)$$

## 4 Kostenschätzung

Wir hatten schon bei der algebraischen Optimierung z.B. bei der Berechnung mehrerer Verbunde bemerkt, daß man ggf. Informationen über die Größe der Relationen braucht, um entscheiden zu können, welche Alternative die wahrscheinlich günstigere ist.

Die Kosten einer Abfrage (z.B. die reale Antwortzeit, s. Abschnitt 2) können nicht exakt prognostiziert werden, sondern allenfalls später bei der Ausführung gemessen werden<sup>5</sup>. Die absoluten Kosten sind auch insofern unwichtig, als es primär um den Vergleich verschiedener Ausführungspläne geht, also relative Kostenangaben ausreichen. Anzustreben ist daher allenfalls eine ungefähre Prognose, ferner müssen die Ausgangsdaten leicht beschaffbar sein und die Berechnungsfunktionen dürfen nicht zu aufwendig sein.

Als hinreichend exakter Maßstab hat sich die Zahl der in Zwischenergebnissen oder im Endergebnis erzeugten Tupel bewährt. Um diese zu berechnen oder abzuschätzen, werden folgende Ausgangsdaten (vgl. Abschnitt 6 in [IRO]) verwendet:

- $s_r$             der durchschnittliche Speicherplatzbedarf eines Tupels (incl. Hilfsdaten) der Relation  $r$
- $|\mathbf{r}|$             die Zahl der Tupel in der Relation  $r$

---

<sup>5</sup>Wobei sich herausstellen wird, daß die Kosten nicht exakt reproduzibel sind, insofern eine exakte Prognose weitere Einflußfaktoren berücksichtigen müßte.

$V(\mathbf{A}, \mathbf{r})$  die Zahl der verschiedenen Werte, die Attribut  $\mathbf{A}$  in Relation  $\mathbf{r}$  annimmt

Diese (und ggf. weitere) Daten über den Datenbestand können i.a. nicht ständig exakt vorgehalten werden; es reicht aber aus, diese Daten z.B. einmal täglich zu betriebsschwachen Zeiten zu aktualisieren. Die laufenden Änderungen in der Datenbank haben auf diese statistischen Werte i.a. nur marginalen Einfluß, d.h. die leicht veralteten statistischen Daten sind weiterhin brauchbar.

**Schätzung der Kosten einer Selektion.** In einigen Sonderfällen kann die Größe des Ergebnisses recht genau vorhergesagt werden:

- Hat die Selektionsbedingung die Form  $\mathbf{A}=\mathbf{a}$  und ist  $\mathbf{A}$  Identifizierungsschlüssel, so gilt  $|\sigma_{\mathbf{A}=\mathbf{a}}(\dots)| \leq 1$ . Analog gilt dies für Identifizierungsschlüssel mit mehreren Attributen.
- Hat die Selektionsbedingung die Form  $\mathbf{A}=\mathbf{a}$  und existiert ein Sekundärindex für  $\mathbf{A}$ , so kann die Zahl der Treffer durch Auslesen eines einzigen Satzes im Sekundärindex bestimmt werden.

Sofern diese Sonderfälle nicht zutreffen, kann mit Hilfe der Zahl  $V(\mathbf{A}, \mathbf{r})$  wie folgt geschätzt werden:

$$|\sigma_{\mathbf{A}=\mathbf{a}}(\mathbf{r})| = \frac{|\mathbf{r}|}{V(\mathbf{A}, \mathbf{r})}$$

Diese Schätzung unterstellt, daß jeder Wert des Attributs  $\mathbf{A}$  in etwa gleichhäufig auftritt, die Varianz der Häufigkeitsverteilung also vernachlässigbar ist.

**Schätzung der Kosten einer Projektion.** Bei Projektionen ohne Duplikateliminierung hat die Ausgabe gleichviele Tupel wie die Eingaberelation. Mißt man die Größe der Relationen in Tupeln, so gilt  $|\pi_{\dots}(\mathbf{r})| = |\mathbf{r}|$ . Für eine detailliertere Analyse des Platzbedarfs des Ergebnisses kann man den Platzbedarf der projizierten Tupel zusätzlich betrachten. Bei Projektionen mit Duplikateliminierung gilt das Vorstehende ebenfalls, sofern die Menge der Attribute, auf die projiziert wird, einen Identifizierungsschlüssel enthält.

**Kosten eines Kreuzprodukts.** Hier gilt ausnahmsweise eine exakte Formel:

$$|\mathbf{r} \times \mathbf{s}| = |\mathbf{r}| \cdot |\mathbf{s}|$$

**Schätzung der Kosten eines Verbunds.** Seien  $\mathbf{r1}$  bzw.  $\mathbf{r2}$  Relationen des Typs  $\mathbf{R1}$  bzw.  $\mathbf{R2}$  und  $\mathbf{V} = \mathbf{R1} \cap \mathbf{R2}$  die Verbundattribute.

Wenn die beiden Relationen keine gemeinsamen Attribute haben, also  $\mathbf{V} = \emptyset$ , ist der Verbund ein Kreuzprodukt und die vorstehende Formel ist anwendbar.

Sofern  $\mathbf{V}$  einen Identifizierungsschlüssel für  $\mathbf{R2}$  enthält, ist für jedes Tupel aus  $\mathbf{r1}$  höchstens ein Verbundpartner vorhanden, also:

$$|\mathbf{r1} \bowtie \mathbf{r2}| \leq |\mathbf{r1}|$$

Sofern zusätzlich  $\mathbf{V}$  in  $\mathbf{r1}$  als Fremdschlüssel auf  $\mathbf{r2}$  deklariert ist, also sichergestellt ist, daß immer ein Verbundpartner vorhanden ist, gilt  $|\mathbf{r1} \bowtie \mathbf{r2}| = |\mathbf{r1}|$ .

Sofern die vorstehenden Voraussetzungen nicht zutreffen, kann es zu einem Tupel von  $\mathbf{r1}$  mehrere Verbundpartner in  $\mathbf{r2}$  geben. Die Größe des Ergebnisses kann dann analog wie bei der Selektion abgeschätzt werden. Hierzu nehmen wir vereinfachend an:

- $\mathbf{V} = \{\mathbf{A}\}$ , also nur ein Verbundattribut
- Gleichverteilung der Werte von  $\mathbf{A}$  in  $\mathbf{r2}$

Für ein Tupel  $t$  in  $\mathbf{r1}$  ist die Menge der Verbundpartner in  $\mathbf{r2}$  gerade  $\sigma_{\mathbf{A}=t[\mathbf{A}]}(\mathbf{r2})$ . Zur Schätzung der Größe dieser Menge setzen wir die bei der Selektion entwickelte Formel ein und erhalten:

$$|\mathbf{r1} \bowtie \mathbf{r2}| = |\mathbf{r1}| \cdot \frac{|\mathbf{r2}|}{V(\mathbf{A}, \mathbf{r2})} = \frac{|\mathbf{r1}| \cdot |\mathbf{r2}|}{V(\mathbf{A}, \mathbf{r2})}$$

In den vorstehenden Überlegungen können wir natürlich die Rollen von  $\mathbf{r1}$  und  $\mathbf{r2}$  vertauschen und erhalten

$$|\mathbf{r2} \bowtie \mathbf{r1}| = \frac{|\mathbf{r2}| \cdot |\mathbf{r1}|}{V(\mathbf{A}, \mathbf{r1})}$$

Wir haben nun zwei unterschiedliche Schätzungen für  $|\mathbf{r2} \bowtie \mathbf{r1}|$ , denn  $V(\mathbf{A}, \mathbf{r1})$  und  $V(\mathbf{A}, \mathbf{r2})$  können signifikant verschieden sein. Wenn

bspw.  $V(A, r_1)$  kleiner als  $V(A, r_2)$  ist, bedeutet das, daß in  $r_2$  mehr unterschiedliche Werte im Attribut  $A$  auftreten als in  $r_1$ . Offensichtlich können dann nicht alle Tupel in  $r_2$  einen Verbundpartner finden, die nur in einer Relation auftretenden Werte sind "hängende Referenzen".

Als Beispiel betrachten wir in Bild 1 zwei Relationen  $r_1$  und  $r_2$  mit Verbundattribut  $A$  und  $V(A, r_1) = 3$  und  $V(A, r_2) = 8$ . Zur Vereinfachung treten die Zahlen 1, 2, 3 usw. als Werte in  $A$  auf, ferner nehmen wir an, daß beide Relationen nach  $A$  sortiert sind. Wenn wir nun  $r_1$  durchlaufen, finden wir zu jedem Tupel in  $r_1$   $\frac{|r_2|}{8}$  Tupel in  $r_2$  als Verbundpartner. Insgesamt entstehen also  $\frac{|r_1| \cdot |r_2|}{8}$  Tupel.

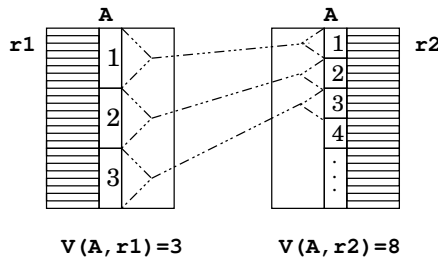


Abbildung 1: Größenschätzung eines Verbunds

Wenn wir stattdessen  $r_2$  durchlaufen, finden wir nur für  $|r_2| \cdot \frac{3}{8}$  Tupel überhaupt einen Verbundpartner in  $r_1$ . Für jedes dieser Tupel finden wir  $\frac{|r_1|}{3}$  Verbundpartner. Insgesamt entstehen also  $|r_2| \cdot \frac{3}{8} \cdot \frac{|r_1|}{3} = \frac{|r_1| \cdot |r_2|}{8}$  Tupel.

Generell gilt also: bei der unterstellten Gleichverteilung der Attributwerte und bei der (optimistischen) Annahme, daß alle in  $r_2$  auftretenden Werte auch in  $r_1$  auftreten, also  $\pi_A(r_2) \subset \pi_A(r_1)$ , müssen wir die höhere Selektivität bzw. den *kleineren Schätzwert* verwenden.

Unsere obige Annahme  $\pi_A(r_2) \subset \pi_A(r_1)$  kann falsch sein. Sofern in beiden Relationen nur wenige Werte in  $A$  gemeinsam auftreten, wird das Ergebnis kleiner werden. Im Extremfall  $\pi_A(r_1) \cap \pi_A(r_2) = \emptyset$  ist das Ergebnis sogar leer. Sofern man die Zahl der in beiden Relationen in  $A$  auftretenden Werte bestimmt, kann hiermit ein weiterer

Korrekturfaktor gebildet werden. Allerdings wird die Schätzung immer unzuverlässiger; an dieser Stelle sei daran erinnert, daß auch die Annahme der Gleichverteilung der Attributwerte i.a. die Realität stark vereinfacht.

## Literatur

[DBSA] Kelter, U.: Lehrmodul “Architektur von DBMS”; 2001

[IRO] Kelter, U.: Lehrmodul “Implementierung relationaler Operationen”; 2002

[RDBM] Kelter, U.: Lehrmodul “Das relationale Datenbankmodell”; 2002

## Glossar

**Äquivalenz relationaler Ausdrücke:** zwei relationale Ausdrücke sind äquivalent, wenn bei einem beliebigen Datenbankinhalt beide Ausdrücke das gleiche Ergebnis liefern

**Ausführungsplan:** konkreter Algorithmus, der zu einer gegebenen Abfrage das Ergebnis berechnet

**Optimierung:** Kollektion von Maßnahmen, die zu einer möglichst effizienten Berechnung eines Abfrageergebnisses führen

**Optimierung, algebraische:** Umformung einer gegebenen Abfrage in eine äquivalente, effizienter ausführbare; unabhängig vom Datenbankinhalt

**Optimierung, interne:** Optimierungsmaßnahmen, bei denen Merkmale des Datenbestands, Indexe, Sortierungen und sonstige interne Merkmale ausgenutzt werden, um einen möglichst guten Ausführungsplan zu konstruieren bzw. auszuwählen

**Sichtenauflösung:** Ersetzung einer virtuellen Relation durch die sie definierende Abfrage bei der Abfrageverarbeitung

# Index

- Äquivalenz, 5, 8, 20
- Abfrage
  - interne Darstellung, 3
  - Korrektheitsüberprüfung, 3, 4
  - Optimierung, 3
  - Verarbeitung, 3
  - Zwischenergebnis, 6, 7, 12, 13
- Abfragesprache, 3
- Assoziativgesetz, 9
- Ausführungsplan, 6, 16, 20
  - Kosten, 6
- Datenmodell
  - navigierendes, 3
- Distributivgesetz, 9
- Duplikateliminierung, 17
- Kommutativgesetz, 8
- Kostenmaß, 8
- Kostenschätzung, 16
  - Kreuzprodukt, 17
  - Projektion, 17
  - Selektion, 17
  - Verbund, 18
- Kreuzprodukt, 8, 9, 10, 11, 13, 14, 17
- Mengenoperationen, 8, 9
- Optimierer, 6
- Optimierung, 4, 20
  - Abarbeitung von Ausdrücken, 5
  - algebraische, 6, 8, 20
  - Algorithmus, 13
  - Ansätze, 4
  - Heuristik, 12
  - interne, 6, 20
  - Kostenmaß, 8
  - Kostenschätzung, *s. Kostenschätzung*
  - Kriterien, 7
  - Meßgrößen, 7
  - relative, 7
  - von Elementaroperationen, 4
  - zusammengefaßte Operationen, 13
- Projektion, 9, 10, 11, 14, 17
- Relation
  - virtuelle, 4
- relationale Algebra, 3
- Schlüssel
  - Fremdschlüssel, 18
  - Identifizierungsschlüssel, 18
- Selektion, 9, 10, 12, 14, 17
  - komplexe  $\sim$ bedingung, 9
- Sichtenauflösung, 4, 20
- $V(\mathbf{A}, \mathbf{r})$ , 18
- Verbund, 8, 9, 10, 13, 14, 15, 18